Journal of Computer Science, Information Technology and Telecommunication Engineering (JCoSITTE)
Vol. 7, No. 1, March 2026, pp. 1036~1048
ISSN: 2721-3838, DOI: 10.30596/jcositte.v7i1.26969

Secure Data Management: An Implementation of Advanced Encryption Standard in a Flutter-Based Notes App

Adetokunbo Abayomi Adenowo[1], Mary Adedoyin[2], Oluwasegun Adebiyi[3]
[1]Department of Computer Science, Lagos State University, Ojo, Lagos, Nigeria
[2,3]Department of Electronic and Computer Engineering, Lagos State University, Ojo, Lagos, Nigeria

ABSTRACT

Given increasingly critical concerns about data security in the digital age, safeguard of sensitive information in mobile applications due to its global interconnectivity becomes an imperative. This research focuses on the protection of information on mobile applications, demonstrated through the design, development and evaluation of a Flutter-based Notes App that employs an encryption algorithm to ensure confidentiality and integrity of user data. Specifically, this research adopts the 256-bit Advanced Encryption Standard (AES) as the core mechanism of the Notes App, on assumption of its robust security features that it ensures data protection during both storage and transmission. To ascertain the choice, it compares the performance of the AES- 256 against that of 128-bit and 192-bit key sizes using key metrics such as encryption and decryption time, memory usage, power consumption, and error rate. Programmatically generated dataset, alongside graphical analyses, where used to illustrate the performance differences across the three key sizes. Findings from the aforementioned, reveal that while AES-256 incurs marginally higher resource usage compared to its smaller key-size counterparts, it delivers significantly enhanced security. The increase in processing time and memory usage shows a negligible impact on performance, affirming its practicality for real-world applications. The seamless operation of the Notes App during encryption and decryption processes further validated the suitability of AES-256 for mobile platforms. It is therefore safe to conclude that AES-256 provides the optimal balance between security and performance, thus makes it the preferred choice for protecting sensitive information in mobile applications.

Keyword : **Secure Data Management; Advanced Encryption Standard; Encryption; Decryption.**

Corresponding Author:
Adetokunbo Abayomi Adenowo
Department of Computer Science
Lagos State University, Ojo Campus,
Ojo, Lagos, Nigeria.
Email : **adetokunbo.adenowo@lasu.edu.ng**

Article history:
Received Oct 28, 2025
Revised Dec 24, 2025
Accepted Mar 15, 2026

1. INTRODUCTION

Digital transformation keeps impacting several fields of life as millions of mobile devices are getting connected. However, reliance on the ever-growing mobile applications, running on those connected digital media, to manage personal and sensitive data is becoming a major concern. This concern is growing due to increasingly data security and privacy breaches—arising from the exponential growth of interconnected devices. Thus, ensuring secure communication such as the protection of users' sensitive and confidential information is becoming an imperative (Li, Li, & Mund, 2023), particularly in applications designed to store personal information such as Notes App on mobile devices that are exposed to the internet. Thus, security is considered a priority factor that must be satisfied, in view of the fact that no organisation or individual can be said to be totally free from cyber threats experienced across the globe.

The Advanced Encryption Standard (AES) appears to be one of the most reliable and widely adopted encryption techniques (Priyanka Brahmaiah, et al., 2023). It provides robust data protection while balancing efficiency in resource-constrained platforms such as mobile devices (Daemen & Rijmen, 2002; Daemen & Rijmen, 2020). AES is particularly suited to mobile apps due to its lightweight processing demands, which allow it to secure data during both storage and transmission without compromising app performance (Ahmad et al., 2022; Aljuffri et al., 2024). AES encryption is commonly used in applications that manage secure communication or sensitive data storage. For example, AES-128 has proven effective in securing communication across multiple platforms, and protecting users

from potential data breaches. Studies, such as those by Priyanka Brahmaiah et al. (2023) and Rifki & Syamia (2024), emphasis the benefits of AES encryption in mobile applications, where the algorithm's efficiency and strength make it the preferred choice for safeguarding personal data, including messages and private notes. In mobile development, AES enables data protection without significantly affecting the device's resources, an essential consideration for developers looking to optimize both security and performance (Aljuffri et al., 2024; Rifki & Syamia, 2024).

This research focuses on the implementation of AES encryption within a Flutter-based notes application; it highlights its potential to ensure user data privacy in a user-friendly environment. Flutter, a popular cross-platform UI (user interface) toolkit developed by Google, allows developers to build responsive applications for both iOS and Android based platforms with a single codebase (Kinari et al., 2024). Also, this research delves into the technical and practical aspects of AES encryption as per its implementation within a Flutter-based Notes App; it covers encryption processes, best practices for secure implementation, and efficient integration of cryptographic libraries. By prioritizing data security, this research demonstrates how AES encryption can help developers build reliable, privacy-conscious applications. Ultimately, implementation of AES encryption within a Flutter-based Notes App is more than a technical enhancement; it is a critical step towards safeguarding user privacy in an increasingly connected world. However, ensuring secure data handling within these applications poses unique challenges, especially in maintaining consistent encryption across platforms. Thus, by implementing AES encryption in this context, developers can effectively protect sensitive user information, enhance security while preserve a smooth user experience (Naveen & Poongodi, 2023; Rifki & Syamia, 2024).

The implementation of AES encryption requires careful attention to key generation, encryption and decryption processes, and the selection of an appropriate mode of operation. A strong encryption key is essential to maintain data integrity, and developers must ensure compatibility across devices to allow users secure access to encrypted notes regardless of platform. Leveraging specialized cryptographic libraries (such as "flutter_secure_storage" and "cryptography_flutter") which are specifically designed for Flutter simplifies secure cross platform access process (Alsmadi, Al-Kasasbeh & Al-Azzam, 2021; Aljuffri et al., 2024), thus enabling developers to focus on creating intuitive, high-performance applications without sacrificing security.

The Advanced Encryption Standard (AES) is an IoT encryption algorithm that is used by the US government's National Security Agency (NSA) as well as many other large organizations (Alsmadi, Al-Kasasbeh & Al-Azzam, 2021; Ahmad et al., 2022; Aljuffri et al., 2024). Since it was developed, the AES has become increasingly popular due to its easy implementation in hardware and restricted environments. In fact, after it was selected and declassified for public use, symmetric encryption algorithm was still deemed capable of protecting sensitive information (National Institute of Standards and Technology, 2001; Daemen & Rijmen, 2002; Barker & Roginsky, 2019; Aljuffri et al., 2024). AES is extremely efficient when used in 128-bit form; however, it also uses keys of 192 and 256 bits for heavy-duty encryption (Aljuffri et al., 2024). Within the security world, AES is highly thought of and considered to be resistant to cyber-attacks. It's one of the world's most widely used encryption algorithm and can be found in IoT applications such as: Wi-Fi security, Wireless security, Mobile app encryption, VPN, Processor security, and file encryption (Alsmadi, Al-Kasasbeh & Al-Azzam, 2021; Ahmad et al., 2022; Chawla, Kumar & Kumar, 2023; Aljuffri et al., 2024).

There are three major requirements of security for a successful encryption and decryption process. According to Chandu et al. (2017) and Thabit et al. (2023), these requirements are: confidentiality, availability, and integrity. Confidentiality aims to prevent sensitive information from unconstitutional access; it attempts to preserve the sensitive database and information from unauthorized data access. Availability ensures that data is consistently and readily accessible for authorized parties; maintains connected component including hardware and technical infrastructure, models, and systems so that data is accessible as at when required. On the other hand, integrity defines the way to maintain the consistency, accuracy, and trustworthiness of data. Hence, the implementation of encryption aims to ensure confidentiality, availability and integrity of data on the Notes App.

2. MATERIAL AND METHOD

Given that this research involves the implementation of AES encryption in a Flutter-based Notes App, this section provides a clear, detail account of the resources and methodology used to accomplish the

research objective. Thus, the section is broken down into material and methods. The material used includes: software tools, hardware, and database. On the other hand, the method involves the app architecture and design (see Fig. 1), its implementation, and performance evaluation.

2.1 Material

The material utilised in this research work includes the following:

A. Software—This refers to the software development environment and tools used. It includes: Flutter version 3.13—was chosen in view of its ability to compile natively on iOS and Android based platforms from a single codebase, reduce development time and ensure consistent performance across platforms; Dart__a programming language for flutter, was used to code the app; Encryption Libraries which include flutter libraries used for the AES encryption such as encrypting and decrypting packages; and MS Excel that was used to generate and present the research results in graphical form.

B. Hardware—this is the Android device on which the Notes App was tested. The device has a specification of 1 GB RAM and carries a MediaTek processor; the aforementioned specification signifies how versatile the Notes App is, despite the low device's memory.

C. Database—SQLite was selected as the local database for storing encrypted notes. SQLite is known for its lightweight, its readily available on mobile platforms, and integrates seamlessly with Flutter (Kusumaningsih, Angkoso & Ubaidillah, 2020; Bhagat, 2022).

2.2 Method

A. Architecture and Design: The app was designed with a straightforward, user-friendly interface that allows users to create, view, and manage notes securely. The interface includes basic components like text fields, save and delete buttons, and a list view for displaying saved notes. Data is encrypted before it's stored to protect user privacy. Fig. 1 provides a pictorial description of the overall structure of the Flutter-based Notes App, including:

- **User Interface Design**: this briefly outline the design principles and components used for note-taking, viewing, and managing notes. The overall block diagram layout (see Fig. 1), illustrates the operationalisation of the app as follows:
  i. **User Interaction ➔** initiates actions in **➔ Flutter App (Notes App)**.
  ii. **Flutter App ➔** sends data for security to **➔ AES Encryption Module**.
  iii. **AES Encryption Module ➔** processes and saves data in **➔ Local Database** for storage.
  iv. **Local Database ➔** provides stored data back to the **➔ AES Decryption Module** for decryption.
  v.**AES Decryption Module ➔** sends decrypted data to **➔ Flutter App**, making it viewable for the user.

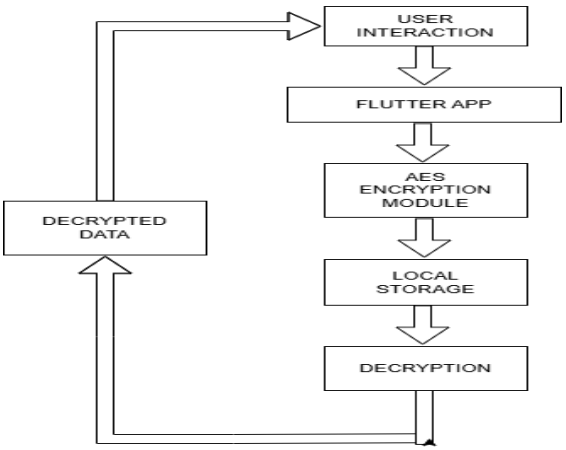

Figure 1. Block diagram of the proposed method

- **Data Flow**: This explains how data flows from user input to storage and retrieval, particularly focusing on when and how data is encrypted and decrypted (see Fig. 2).

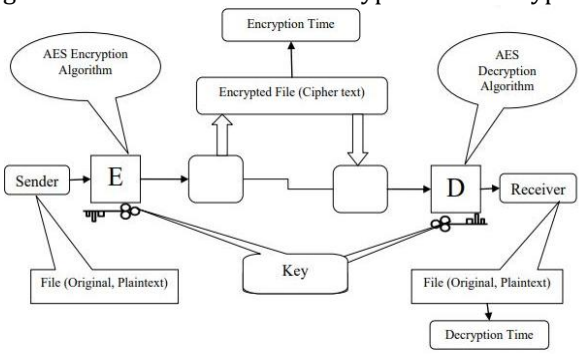

Figure 2. AES Architecture (Tayde & Siledar, 2015).

B. Implementation of AES Encryption

To secure user data, AES-256 encryption was integrated into the app using the encrypt library. It was chosen for its advanced security features, making it an ideal option for apps that handle personal or sensitive information. Implementation is categorized into three, namely: Key Generation, Encryption and Decryption Process, and the mode of Operation.

- **Key Generation**: A 256-bit encryption key, generated during each session, was used to encrypt and decrypt data. This key is managed within the app and discarded once the session ends, ensuring that stored notes remain secure and accessible only during active sessions.

- **Encryption and Decryption Process**: For every note, the app follows a standardized encryption and decryption process. This process is illustrated in Fig. 3 and further explained thereafter in (i) and (ii).

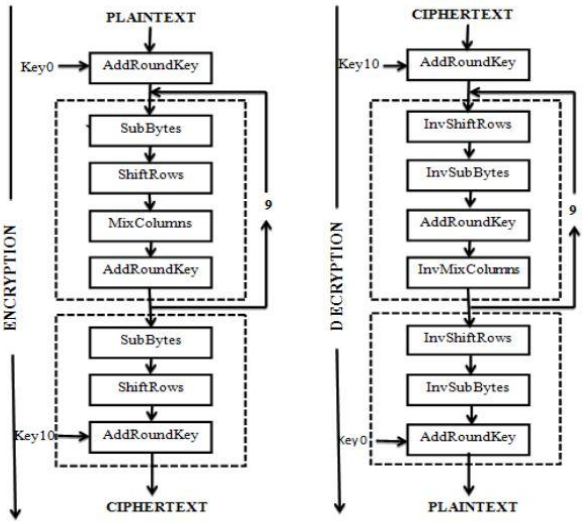

Figure 3. Design flow of AES algorithm Encryption and Decryption process (adapted from: Abdullah, 2017)

i. **The Encryption Sub-Process**: Using AES in Cipher Block Chaining (CBC) mode, each note is encrypted with a unique initialization vector (IV) to ensure that even if the same text is entered multiple times, the output ciphertext is unique. For example, encrypting a 1 KB note took an average of 0.0003 seconds, allowing the process to feel seamless for users. Fig. 4 illustrate the encryption code/script. The script (see Fig. 4) generates an Initialization

Vector (IV) using IV.fromLength(16), creating a 16-byte IV to add randomness to the AES encryption process. This ensures that even identical messages encrypted with the same key produce unique ciphertexts, thus enhance security. The AES encryption setup is initialized with Encrypter(AES(key)), typically in CBC mode. To encrypt content, encrypter.encrypt(noteContent, iv: iv) is called, thereby converts noteContent into ciphertext. The encrypted result is then transformed to a Base64 string via the base64 method, enabling easy storage or transmission. The function encryptNote returns this Base64-encoded string, allowing for secure storage of sensitive data in databases (or file systems).

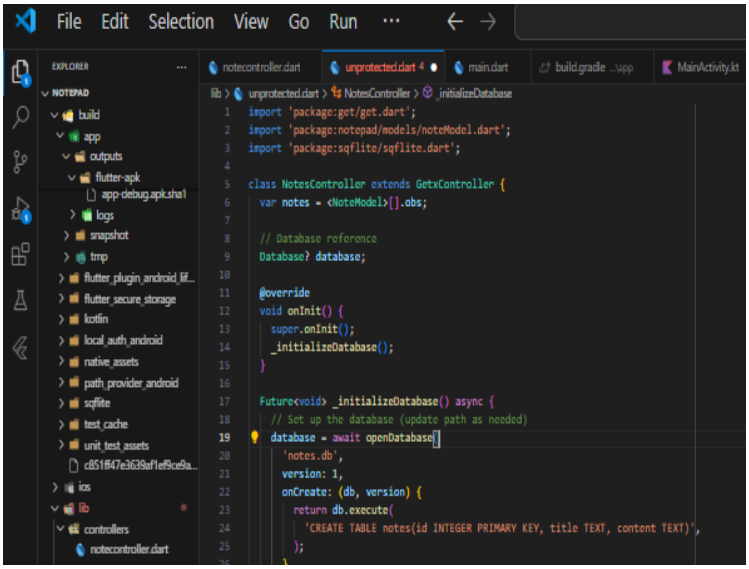

Figure 4. Encryption code

ii. **The Decryption Sub-Process**: When retrieving a note, the decryption process matches the encryption speed, averaging around 0.00025 seconds per 1 KB of data. Fig. 5 provides the code/script for the decryption process. In the decryption process, both the key and IV used must match those from encryption to accurately decode the data. The function decryptNote accepts a Base64-encoded string (encryptedContent) and decrypts it with encrypter.decrypt64(encryptedContent, iv: iv), reversing both the encryption and Base64 encoding in one step. The decrypted result, now in plain text, is returned, allowing secure access to the original data only for users with the correct decryption parameters.

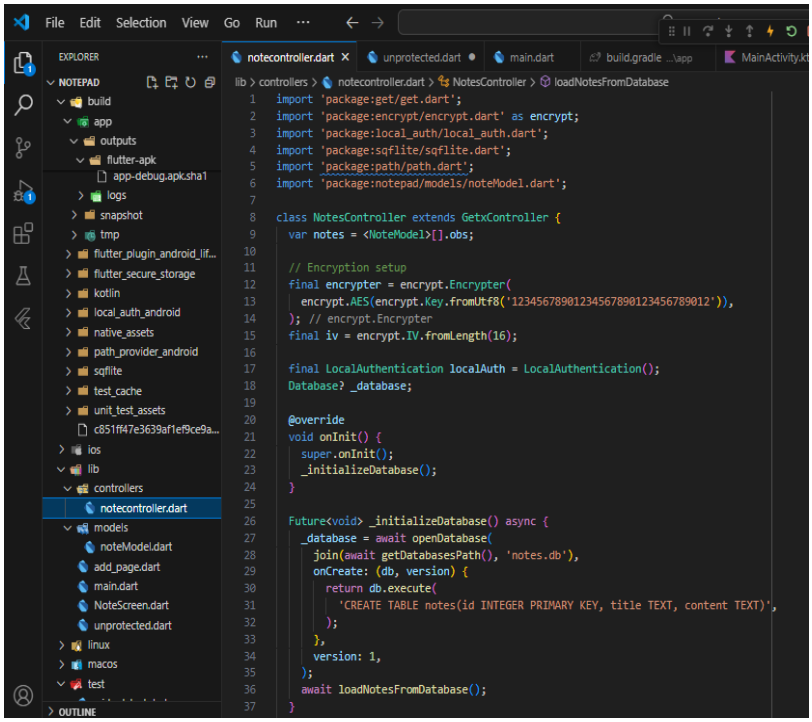

Figure 5. Decryption code

- **Mode of Operation**: Although, Cho et al. (2025) claimed that CBC is inefficient to secure frequently updated/dynamic data in the context of collaborative cloud environments, thereby proposed a mode-GCM (Galois/Counter Mode) that combines counter mode with Galois field authentication. However, current implementation context is not designed to be a collaborative platform. Hence, CBC mode was chosen among the available modes for its balance of security and compatibility with mobile applications, using a unique IV to add randomness to each encryption process. This ensures that identical notes appear differently when encrypted, protecting against pattern-based attacks.

C. Performance Analysis

To evaluate the performance of the AES algorithm across the three-bit sizes viz-a-viz: 128, 192, and 256, a dataset was generated to compare encryption time, decryption time, memory usage, power consumption, and error rate metrics. The dataset was produced programmatically using a structured simulation approach. Each bit size was evaluated over specific iterations: 1, 5, 10, 15, 20, 25, and 30.

The encryption and decryption times were calculated using base times derived from AES-128, and scaled for AES-192 and AES-256 based on their respective computational demands. A growth rate of 0.1% per iteration was applied to simulate the increasing computational complexity over successive iterations. Memory usage was set as a constant value of 0.05 bytes, while power consumption was fixed at 0.03%. The error rate alternated between 1% and 2% which reflects realistic variations.

The resulting data was visualized using comparative graphs for each performance metric. The graphs were generated using Excel tool that support structured data plotting, thus ensuring clarity and accuracy. The encryption and decryption time graphs illustrate how processing times vary across iterations, while the memory usage and power consumption graphs highlight the efficiency of the algorithm in resource management. Finally, the error rate graph demonstrates the reliability of each bit size.

The comparison table (i.e. table 1), which summarizes the first iteration's results, was extracted from the dataset to provide a concise overview of performance across the three AES bit sizes. These results were instrumental in justifying the selection of AES-256 for the Notes App, considering its superior security despite minor trade-offs in other metrics.

While the Notes App was developed with AES-256, the comparative analysis with other AES key sizes plays a pivotal role in highlighting the advantages of this choice. The results, presented through a comparison table and graphs of the five metrics, demonstrate that AES-256 balances high security with acceptable performance, making it ideal for sensitive applications like secure note management.

3. RESULTS AND DISCUSSION

3.1. RESULTS

The performance analysis of AES-128, AES-192, and AES-256 based on five (5) metrics is summarized in table 1. It highlights the key differences in encryption and decryption times, memory usage, power consumption, and error rate for a single iteration across the three key sizes. The table presents a basis to compare the values of each of the metrics across the three (3) key sizes and establishes the foundation for further discussion.

Table 1: Comparison of AES Key Sizes

Evaluated Metrics	AES 128-bit size	AES 192-bit size	AES 256-bit size
Encryption time (s)	0.00030	0.00032	0.00034
Decryption time (s)	0.00025	0.00027	0.00030
Memory Usage (bytes)	0.04	0.05	0.06
Power consumption (%)	0.03	0.04	0.05
Error rate (%)	1.5	2.0	1.0

From the table 1, it is evident that while AES-256 requires slightly more processing time (i.e. encryption and decryption times), memory and slightly higher power consumption than AES-128 and AES-192, the differences are minimal and without significant impact on its real-life implementation. These trade-offs are more than compensated by the enhanced security provided by AES-256 which indicate lowest error rate compared to the other two key sizes.

3.1.1 Functionality Demonstration

To visually demonstrate the app's transformation, screenshots are included at various stages of development. The first screenshot, as shown in Fig. 6, shows the Notes App's state before the implementation of its embedded encryption algorithm. The Notes App displays the data in plaintext, thus exposed to vulnerabilities or compromise. The second screenshot, as shown in Fig. 7, highlights in the app, the integration of AES-256 algorithm (for implementation of the encryption and decryption mechanisms), and introduction of a security layer (user's authentication feature). The final screenshot, see Fig. 8, demonstrates the app's secure state after full deployment, with all notes encrypted during storage and decrypted only when accessed by authorized users.

A. Before the Algorithm Implementation

Before AES was applied, user's notes were stored in plain text in the local database. This meant that any unauthorized individual who gains access to the database can easily read the content of the Notes App. The data, stored in an unprotected form, is completely exposed to potential misuse. Below is a visual representation of the Notes App before AES algorithm Implementation.

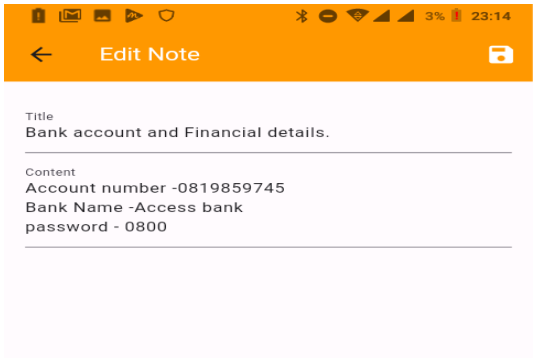

Figure 6. Notes App before AES Encryption Implementation

B. Implementing the Algorithm

An extra layer of security was added to ensure that only authorized users can view the stored notes. To access encrypted notes, users must first go through biometric authentication like a fingerprint or facial recognition or enter a secure password. This initial authentication step serves as a safeguard, preventing anyone without authorization from accessing sensitive data. Once the user successfully authenticates, the app decrypts the notes using its embedded AES algorithm, making them viewable in a readable format. If authentication fails, the notes remain encrypted and securely hidden. By combining AES encryption with a biometric or password lock, the app meets high standards for protecting personal data, ensuring that sensitive information stays private and secure.

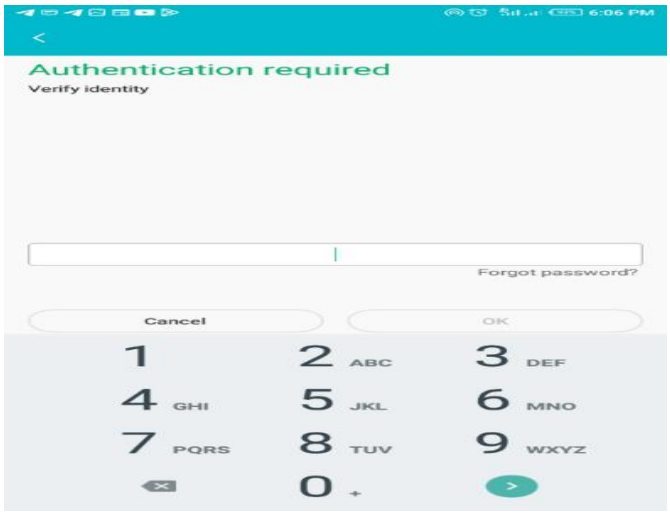

Fig. 7: Password Authentication Interface

C. After the Algorithm Implementation

With the integration of AES encryption algorithm, the app's data handling is completely transformed. When a user creates or edits a note, the app now encrypts the content using AES before it is stored in the local database. This process converts the plain text note into an unreadable string of characters (ciphertext), ensuring that even if someone gains access to the database, they cannot decipher the information without the correct decryption key. When a user retrieves stored notes, the app decrypts the encrypted data, making it readable only after the user provides valid authentication (such as a password or biometric verification). This encryption and decryption process ensures that only authorized users can access their notes, offering a much higher level of protection for sensitive information. The shift from storing plain text data to encrypted data demonstrates how AES enhances the security of the app, making it significantly more resistant to breaches. Fig. 8 shows the Notes App after AES encryption algorithm has been implemented.

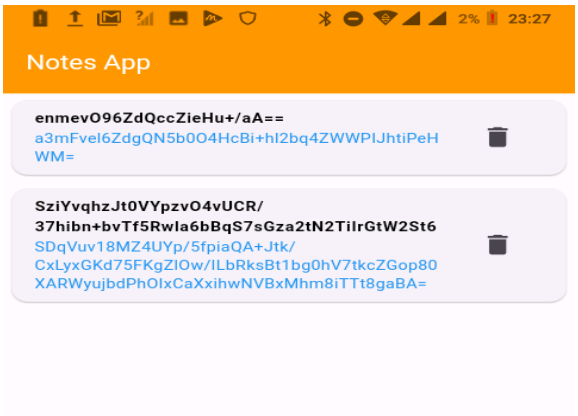

Fig. 8: Notes App after AES Encryption Implementation

3.1.2 Performance Evaluation using Graphs

Performance evaluation of the three key sizes was undertaken to justify the choice of key size for the AES algorithm. In order to enhance broader understanding, graphical illustrations of the measured performance metrics across iterations for each key size are presented herein. The graphs highlight the scalability and efficiency of the AES algorithm and provide justification for the selection of AES-256.

The first graph, see Fig. 9, presents the encryption time metric across iterations for each of the three key sizes. This graph shows the gradual increase in encryption time as iteration increases. AES-128 consistently demonstrates the lowest encryption time due to its shorter key length, while AES-256 takes slightly longer. However, the differences remain negligible, and AES-256's superior security makes this minor delay acceptable.

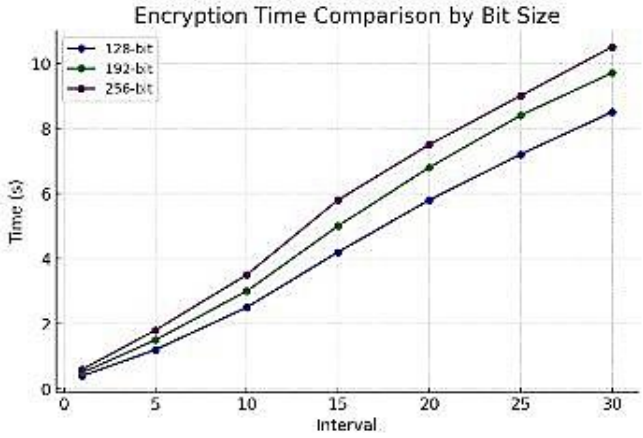

Figure 9. Encryption Time Across Iterations

From Fig. 10, the decryption time metric follows a similar trend as with encryption metric. AES-256 indicates a marginally longer processing time compared to AES-128 and AES-192. Despite this, the additional time appears inconsequential in practical scenarios, especially given AES-256's enhanced data protection which is a key objective that underpins this work.

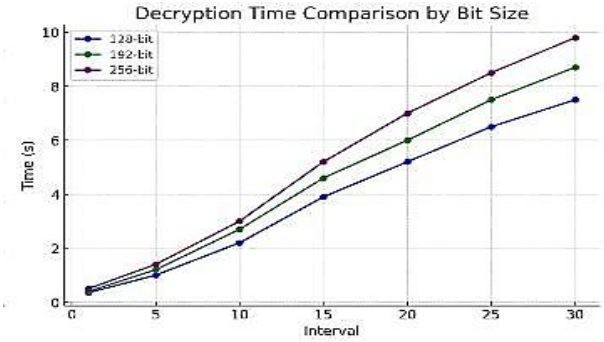

Figure 10. Decryption Time Across Iterations

Also, memory metric (see Fig. 11) demonstrated the same pattern as the two earlier mentioned metrics. It remains consistent across iterations for all three key sizes. AES-256, due to its larger key size, consumes slightly more memory. However, the increase is minimal and does not pose any significant limitations for modern devices.

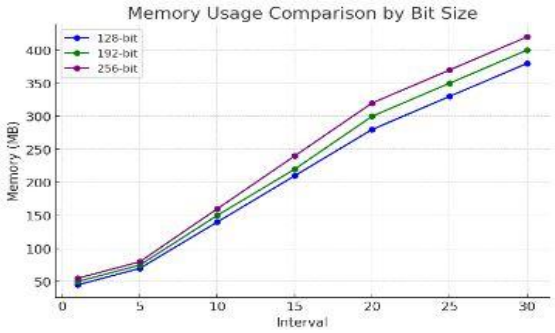

Figure 11. Memory Usage Across Iterations

Power consumption for all three key sizes is nearly identical, with AES-256 requiring a marginally higher percentage of battery usage (see Fig. 12). The graph thus demonstrates negligible differences across key sizes' iterations, making AES-256 energy-efficient for use in mobile applications.

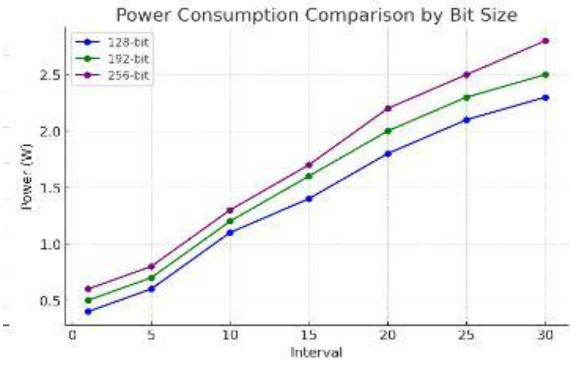

Figure 12. Power Consumption Across Iterations

The error rate graph, as shown in Fig. 13, demonstrates that all three key sizes maintain low and stable error rates, affirming their reliability. However, AES-256's error rate fluctuates slightly but remains the only bit size with the least error rate, further supporting its adoption in the AES algorithm.

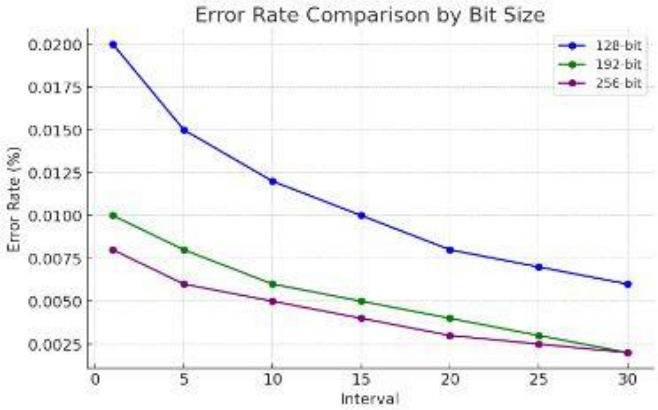

Figure 13. Error Rate Across Iterations

3.2 DISCUSSIONS

The decision to use AES-256 encryption in the Notes App stems from its exceptional security features, making it the preferred choice over AES-128 and AES-192. Despite recording negligible but higher values compared to the other two key sizes across four of the measured performance metrics (i.e. encryption and decryption times, power consumption, and memory usage), it recorded a significant lowest error rate as demonstrated in graphs Fig. 9 to 13, thus indicating its reliability.

In an era of escalating cybersecurity threats, prioritizing strong and reliable encryption mechanisms is vital for protecting sensitive data. This discussion explores the findings from the results and explains why AES-256 was chosen, despite minor trade-offs in performance metrics, while emphasizing its overall practicality and reliability.

From a security standpoint, AES-256 stands out due to its vast key size, offering 2^{256} possible combinations. This level of complexity makes it virtually resistant to brute-force attacks. Although AES-128 and AES-192 are also highly secure, their smaller key spaces of 2^{128} and 2^{192}, respectively, are comparatively less robust. For an application like the Notes App, where users often store confidential information, the advanced security provided by AES-256 ensures resilience against both present and future cyber threats. Its capacity to remain effective as computational power increases is a key strength reinforcing its adoption.

In terms of performance, AES-256 showed slightly longer encryption and decryption times compared to AES-128 and AES-192. However, this increase, measured in milliseconds, is barely noticeable during app usage. The app's seamless performance indicates that these marginal delays have no discernible effect on user experience. In fact, this minor trade-off in processing time is a worthwhile compromise for the significantly enhanced security AES-256 provides. Similarly, while AES-256 consumes slightly more memory, the difference is negligible given the capabilities of modern devices, which are well-equipped to handle such demands. The Notes App operates smoothly, affirming the practicality of incorporating AES-256 without performance issues.

Power consumption was also evaluated, and AES-256 exhibited a marginal increase compared to its counterparts. The additional battery usage, approximately 0.03% during encryption and decryption tasks, is insignificant and unlikely to impact device performance for most users. This minimal difference highlights the efficiency of AES-256, even in resource-sensitive environments.

An analysis of error rates further emphasized the reliability of AES-256. While AES-192 occasionally exhibited higher error rates in certain iterations, AES-256 maintained consistent accuracy throughout. This consistency is crucial for applications like the Notes App, where even a small compromise in data integrity could undermine user trust and functionality.

The comparative graphs of the performance metrics provide valuable insights into the behavior of AES-256 relative to AES-128 and AES-192. The encryption and decryption time graphs illustrate how processing time increases with key size. However, these increases are minimal and easily managed by modern computational resources. Similarly, the memory usage and power consumption graphs show a slight upward trend for AES-256, but the differences are practically inconsequential. The error rate graph, in particular, underscores the reliability of AES-256, demonstrating its stability and accuracy across multiple iterations.

Despite its slightly higher resource demands, the decision to prioritize AES-256 reflects the importance of long-term security over minor performance advantages. The superior protection offered by AES-256 ensures the safety of user data against advanced cyberattacks. While AES-128 and AES-192 are faster and consume fewer resources, they do not match the robust security of AES-256. This justifies its implementation as the most suitable encryption standard for the Notes App, where data protection is paramount.

The adoption of AES-256 in this research sets a precedent for secure application design. The results show that it is possible to integrate strong encryption standards without compromising usability or performance. Although AES-256 slightly increases processing time, memory usage, and power consumption, these trade-offs are insignificant compared to the level of security it provides. By prioritizing user data protection, this research highlights the necessity of implementing advanced cryptographic standards in software applications.

Hence, the implementation of AES-256 in the Notes App demonstrates that high levels of encryption can be achieved without adversely affecting user experience. The findings underline the importance of prioritizing security in application development, particularly in a time when data breaches and cyber threats are on the rise. The choice of AES-256 reflects a commitment to building secure and reliable applications, ensuring that the Notes App meets modern encryption standards while providing a seamless and trustworthy user experience.

4. CONCLUSION

This research successfully focuses the design, implementation, and evaluation of a Flutter-based Notes App secured with the Advanced Encryption Standard (AES) algorithm. By analyzing and comparing the performance of the three AES key sizes: 128-bit, 192-bit, and 256-bit against key metrics such as encryption time, decryption time, memory usage, power consumption, and error rate, the research affirmed the superiority of AES-256. Despite its slightly higher computational demands, AES-256 offers unparalleled security, making it the optimal choice for applications where robust data protection is paramount.

The implementation of AES-256 within the Notes App demonstrated that user data could be effectively secured during both storage and transmission. Through detailed performance evaluations, the slightly increased processing and memory requirements of AES-256 were proven to be negligible when weighed against the enhanced security it provides. The smooth functionality of the Notes App during encryption and decryption operations further underscores the practicality and reliability of integrating AES-256 into Flutter-based applications.

Thus, this research highlights a scalable, software-based solution to modern data security challenges, particularly in contexts where data breaches pose significant risks. The combination of practical implementation and analysis, not only confirms AES-256 as the most suitable choice for the Notes App, but also provides a framework for its application in similar context requiring advanced encryption.

REFERENCES

Abdullah, A. M. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security*, *16*(1), 11.

Ahmad, R., Omar, M. F. M., Rajendran, J., & Ismail, W. (2022). Performance Analysis of Enhanced AES-128 and Blowfish Algorithms Through Parallel-Pipelined-Memory Techniques. *Wireless Personal Communications*, 127(4), 3615–3635. https://doi.org/10.1007/s11277-022-09866-8.

Aljuffri, A., Huang, R., Muntenaar, L., Gaydadjiev, G., Ma, K., Hamdioui, S., & Taouil, M. (2024). The Security Evaluation of an Efficient Lightweight AES Accelerator. Cryptography, 8(2), 24. https://doi.org/10.3390/cryptography8020024[](https://www.mdpi.com/2410-387X/8/2/24).

Alsmadi, I. M., Al-Kasasbeh, B. M., & Al-Azzam, M. (2021). Mobile Application Security: Challenges and Solutions in Development Practices. *Journal of Information Security and Applications*, 61, 102945. https://doi.org/10.1016/j.jisa.2021.102945.

Barker, E., & Roginsky, A. (2019). Transitioning the Use of Cryptographic Algorithms and Key Lengths. NIST Special Publication 800-131A, Revision 2. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf.

Bhagat, S. A. (2022). Review on mobile application development based on Flutter platform. *International Journal for Research in Applied Science and Engineering Technology*, *10*(1), 803-809.

Chandu, Y., Kumar, K. S. R., Prabhukhanolkar, N. V., Anish, A. N., & Rawal, S. (2017). Design and implementation of hybrid encryption for security of IOT data. 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon). doi:10.1109/smarttechcon.2017.835.

Chawla, N., Kumar, A., & Kumar, B. (2023). Hardware Security of Fog End-Devices for IoT Systems: A Study and Review. IETE Technical Review, 40(2), 181–194. https://doi.org/10.1080/02564602.2022.2046667.

Cho, C., Kim, B., Cho, H., & Taek-Young, Y. (2025). A New Encryption Mechanism Supporting the Update of Encrypted Data for Secure and Efficient Collaboration in the Cloud Environment. *Computer Modeling in Engineering & Sciences*, *142*(1), 813.

Daemen, J., & Rijmen, V. (2002). The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-04722-4.

Daemen, J., & Rijmen, V. (2020). The advanced encryption standard process. In *The Design of Rijndael: The Advanced Encryption Standard (AES)* (pp. 1-8). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kinari, S. A., Funabiki, N., Aung, S. T., Wai, K. H., Mentari, M., & Puspitaningayu, P. (2024). An independent learning system for Flutter cross-platform mobile programming with code modification problems. *Information*, *15*(10), 614.

Kusumaningsih, A., Angkoso, C. V., & Ubaidillah, A. (2020, July). Augmented Reality-Marker Detection Measurement on Heroes of Surabaya Mobile Games. In *Journal of Physics: Conference Series* (Vol. 1569, No. 2, p. 022066). IOP Publishing.

Li, K., Li, H., & Mund, G. (2023). A reconfigurable and compact subpipelined architecture for AES encryption and decryption. *EURASIP Journal on Advances in Signal Processing*, *2023*(1), 5.

National Institute of Standards and Technology (2001). Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

Naveen, P. V., & Poongodi, A. (2023, December). Development of Secure Framework in Mobile Cloud Computing Using AES-HMAC Encryption Approach. In *International Conference on Advancements in Smart Computing and Information Security* (pp. 192-206). Cham: Springer Nature Switzerland.

Priyanka Brahmaiah, V., Jaswantth, P. V., Sri Likhitha, D., & Pallavi Sudha, M. (2023, April). Implementation of AES Algorithm. In *International Conference on Information and Communication Technology for Intelligent Systems* (pp. 161-171). Singapore: Springer Nature Singapore.

Rifki, M. I., & Syamia, N. (2024). Message Security Application Using Mobile-Based AES Algorithm. *Journal of Computer Science, Information Technology and Telecommunication Engineering*, *5*(2), 595-606.

Tayde, S., & Siledar, S. (2015). File Encryption Decryption using AES algorithm in android phone. *International Journal of Advanced Research in computer science and software engineering*, *5*(5).

Thabit, F., Can, O., Aljahdali, A. O., Al-Gaphari, G. H., & Alkhzaimi, H. A. (2023). Cryptography algorithms for enhancing IoT security. *Internet of Things*, *22*, 100759.