SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I12P120

Volume 12 Issue 12, 241-257, December 2025
© 2025 Seventh Sense Research Group®

Original Article

End-to-End Security and Privacy Protection for Healthcare Data Using AES-256 and Dynamic Authentication

T. Rathi Devi[1*], S. Nallusamy[2], D. Sobya[3], P. Divya[4], P. S. Chakraborty[5]

[1,2,4,5]*Department of Adult, Continuing Education and Extension, Jadavpur University, Kolkata, India.*
[3]*Department of Computer Science and Engineering, Institute of Engineering and Management, University of Engineering and Management, Kolkata, India.*

[1]*Corresponding Author : revathi.rathi26@gmail.com*

Received: 18 October 2025 Revised: 19 November 2025 Accepted: 18 December 2025 Published: 27 December 2025

Abstract *- Security and Confidentiality of patient information are important in the modern healthcare system. Patient information is often stored on digital platforms through digital health records, telemedicine, and remote monitoring. The proposed work presents a cryptographic authentication framework for healthcare monitoring that uses AES-256 and Virtual Password Authentication(VPF) to protect sensitive data. The Virtual Password Function (VPF) is a little trick that combines a secret function with a code booking technique. This technique prevents unauthorized users from compromising security. It mitigates password-based attacks. Patient data is stored in a completely encrypted way to meet healthcare privacy mandates. The proposed system was developed in Java for encryption and matching authentication of processes. The implementation uses AES-256 encryption to safeguard patient data. It includes custom authentication logic for managing virtual passwords. The cloud uses encrypted end-to-end patient information and stores it in MySQL. The scalable and maintainable front-end web interface and backend control logic are developed using Java JSP Servlet. The framework provides secure, adequate protection of sensitive healthcare data in digital health ecosystems by leveraging strong encryption and adaptive authentication. As shown by experimental results and security analysis, the proposed model is effective for healthcare applications requiring high-level security. It offers relatively low execution, processing, key generation, and encryption/decryption times, alongside enhanced security.*

Keywords *- Patient information, Cryptographic Authentication, AES-256 Encryption, MySQL, Java JSP Servlet.*

1. Introduction

The rapid digitalization of healthcare systems has increased the number of stores that use EHRs, telemedicine platforms, and cloud-based medical data storage [1]. Although such innovations enhance the ease of healthcare delivery, interoperability, and efficiency, they raise significant privacy and security concerns [2]. Healthcare systems are among the targets of cybercriminals due to the increasing volume of sensitive patient data exchanged and stored on open, distributed networks, such as medical history, diagnostic results, and treatment details [3]. Many healthcare facilities still use outdated authentication techniques, such as fixed usernames and passwords, despite the existence of legislative frameworks to protect the confidentiality of patients' medical records [4]. These approaches can be attacked by various processes, including phishing, keylogging, brute-force attacks, malware injection, and shoulder surfing [5]. A breach of access to healthcare infrastructure can lead to data manipulation, identity theft, service disruption, and even fatal consequences, as recent research indicates that such attacks have been on the rise, with alarming rates of credential theft and ransomware. Modes of research and operation should be designed to ensure high levels of end-to-end security for healthcare data [6].

To ensure confidentiality of health data, past studies have presented numerous cryptographic solutions, including RSA, ECC, attribute-based encryption, blockchain-based systems, and hybrid encryption systems [7]. This is because many of these approaches cannot be used in real-time, resource-constrained healthcare environments due to their high computational complexity, longer latency, scalability concerns, or the complexity of managing keys, despite their greater confidentiality. A significant security weakness of end-to-end protection is that authentication processes are not dynamic and are vulnerable, and most past studies have primarily focused on data encryption.

Most healthcare platforms still use plaintext authentication, putting them at risk of cyber threats, though secure methods such as SSL/TLS are widely known to secure data transmission. Phishing attacks, malware and trojans, keyloggers, shoulder surfing, and hidden surveillance are some common threats.

- Malicious attackers masquerade as trusted healthcare providers to illegally collect login credentials and patient information, and call this a phishing scam.
- Malware and trojans are malicious programs responsible for stealing sensitive medical information.
- Keyloggers are programs that silently record your keystrokes to gather passwords and private patient information.
- In public spaces, such as clinics or hospitals, attackers see medical personnel keying credentials.
- Cloaked Surveillance: The assailant can use hidden cameras to track the user's login behaviour and check the photos to capture the password [8].

To protect users' passwords against the mentioned threats, a new security mechanism is developed that allows users to choose their virtual password scheme from less secure to more secure. A security mechanism based on VPF helps safeguard passwords while maintaining an acceptable trade-off between security and complexity. The scheme requires very little computation from the user's end. The system proposes several functions, using a codebook approach, to conduct security analysis. For user-specified functions, secret little functions are used to secure them by hiding important authentication processes and algorithms. This method enhances password security to prevent cyberattacks while remaining user-friendly.

Encryption techniques are important because encryption algorithms enable secure data transmission and exchange—an effective way to reduce loss with ransomware attacks [10]. Making EHRs unreadable and inaccessible to unauthorized users through encryption can protect them from ransomware attacks. Cloud-based apps help send different kinds of data, like sound, images, and even text, despite changes in a network's properties. Healthcare has telemedicine, and entertainment has social media platforms, which are common examples. Various applications interface with text and images, such as medical reports with diagnoses in telemedicine.

In such cases, data transmission should proceed without any hiccups. Asymmetric and symmetric cryptography protect the data stored in the cloud. Here, key generation is based on key size to ensure security [11]. The analysis of asymmetric algorithms is much more intensive, as they are generally more complex than symmetric algorithms, requiring longer key lengths to balance. Besides, for data security, Digital Signature Algorithm (DSA), Rivest-Shamir-Adleman (RSA), Blowfish, and Elliptic Curve Cryptography (ECC) are other cryptographic techniques.

DSA ensures "Document Integrity" and "Document Authenticity" by generating a unique digital signature using the private key and validating it with the public key associated with the document to attest to the document's originality and prevent tampering [12]. RSA utilizes a pair of keys to encrypt any data. It is used for encrypting patients' medical data. RSA encryption creates a complex ciphertext that is hard to crack due to its high computational requirements. Thus, it has been one of the most researched, most reliable, and elegantly designed cryptographic keys [13]. The Blowfish algorithm makes medical data secure by strengthening its F-function to generate very strong round sub-keys. Thus, it is differentially attack-resistant. Blowfish secures both textual and graphical data [14]. ECC, which stands for Elliptic Curve Cryptography, is an encryption system that relies on the algebraic structure of elliptic curves over finite fields. It needs fewer keys than non-EC Encryption methods and offers equivalent security. Digital signatures, key agreements, and other cryptographic functionalities are helpful and can be integrated with symmetric encryption algorithms to enhance security indirectly. Moreover, they are computationally intensive and take longer to process [15].

To reduce security threats, the encryption model was enhanced to make it harder to breach or access without authority. Using an AES-based cryptosystem can reduce security risks and attacks. The mentioned encryption model uses AES encryption to convert medical images into ciphertext, thereby increasing the data's security.

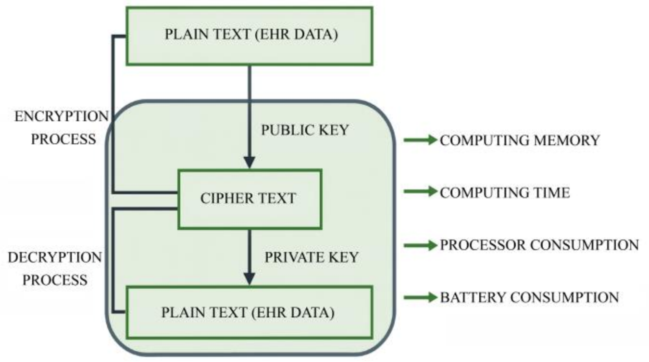


Fig. 1 Schematic representation of cryptographic encryption workflow

To begin with, cryptography was primarily used to safeguard state secrets and other strategic information. Over time, it has found wide application across various fields, primarily in security. The basic idea of cryptographic keys is to encrypt and decrypt data. Plaintext is information that does not require software to read and understand. Encryption is the transformation of plaintext into ciphertext so that it no longer makes sense and appears entirely different from the original text. It ensures that data is not outputable by unauthorized people or anyone who can access the final result. The ciphertext is decrypted to obtain the plaintext. Figure 1 visually represents the encryption and decryption workflow.

A cryptography algorithm, or cipher, is a mathematical function used to encrypt and decrypt data. It operates alongside a keyword, number, or phrase for authenticating plaintext, with different keys generating distinct ciphertexts from the same plaintext. Security of encrypted data depends on both the robustness of the cryptographic method and the confidentiality of the key. Essentially, cryptographic algorithms use either public or private keys to perform their functions.

Research Gap: In current healthcare security solutions, there is a gap in adaptive, attack-resistant authentication processes. These systems ought to be capable of resisting credential-based attacks without augmenting user complexity or computational expense, even though techniques such as AES-based strong encryption can effectively ensure the confidentiality of data. There is limited information on how to secure electronic health record systems stored in the cloud against phishing, replay, and keylogging attacks using lightweight encryption and dynamic authentication.

Problem Statement: An end-to-end healthcare security model is suggested, which combines the AES-256-based encryption algorithm with a dynamic authentication approach, based on Virtual Password Function (VPF). The encrypted data of patients is stored in the cloud as session-specific passwords of the virtual type created due to secret functions and a code booking procedure. The framework, implemented with Java JSP Servlet and MySQL, is compared to available practices and proves to be highly secure, with low computational costs, and appropriate in the context of real-time healthcare settings.

Motivation: The protection of patient data in cloud-based healthcare systems has been a critical research topic supported by the use of encryption techniques. Nonetheless, conventional algorithms with large key sizes occupy a lot of memory space and require a lot of power. To resolve this issue, the proposed AES-256 algorithm is used for the adequate storage and retrieval of patient data.

1.1. Significant Contributions of Proposed Work

- This research presents the advanced AES-256 encryption, which efficiently consolidates various data inputs, optimizing the process while strengthening data integrity and security.
- The proposed methodology follows acknowledged privacy legislation and data security standards. It effectively meets regulatory criteria through guaranteeing transparency and strong security measures.

The paper is structured as follows: Section 2 provides a review of existing literature on securing patient-medical data mechanisms. Section 3 outlines the proposed system description, while Section 4 presents the proposed approach to implementing such a mechanism. Section 5 evaluates the effectiveness of the method compared to previous techniques, highlighting performance measures and discussing the results. Finally, Section 6 summarizes the findings and offers suggestions for further research.

2. Related Works

Cryptography and distributed systems for protecting healthcare information. Several studies have focused on healthcare information. Capraz et al. [16] suggested sharing medical data using the blockchain and a ChaCha20-Poly1305 encryption method, and even divided a file into parts to make it more secure and difficult to compromise. The rate of encryption of this technology is very low and therefore could not be applicable to real-time healthcare, although it improves data protection. In the context of healthcare data analytics, Moheshkumar et al. [17] provided a patient monitoring system that utilized patient data secured by SHA-256. This is a better method of increasing data integrity, and it opens the system to unwanted access as it fails to go to the extent of ensuring privacy.

Hadad et al. [18] suggested that they would make their approach of Modified IDEA more robust to allow transfer of healthcare data through encryption of healthcare information with a 1024-bit key. The key size is highly exaggerated in its strength by the complexity of computation. In order to protect medical records that were stored on cloud servers against cybercriminals, Almalawi et al. [19] recommended a Serpent encryption system that was pegged on Lionized Remora Optimization, but since the system was costly to implement, it could not be applied broadly. In providing security in health care, Rastogi et al. [20] used a blockchain system to carry out Darkie-Hellman Galois-Elliptic Curve Cryptography. This approach has delays in processing in real-time applications. The issue of scalability of cloud systems continues to exist, although Vijaykumar et al. [21] used Diffie-Hellman key exchange to improve cloud data security.

According to these studies, the current methods can improve either encrypted data or access control and are commonly linked with a higher level of complexity, latency, or overhead of implementation. The unexploited requirement of dynamic, attack-resistant authentication processes is the motivation behind the proposed system, which is founded on AES-256 and VPF.

2.1. Evaluating Existing Solutions and Comparing Them to Benchmarks

The existing healthcare security designs utilize Federated Learning (FL), advanced cryptography, and blockchain to safeguard the patient. Although blockchain-based solutions have the following benefits, immutability and decentralization, they cause limitations to scalability, storage overhead, and high transaction latency that are not appropriate to real-time EHR systems.

Despite FL-based models having superior data privacy due to local storage, they are susceptible to poisoning attacks, complicated in nature regarding synchronization, and expensive to communicate. Although they significantly ameliorate security, complex cryptographic constructions like attribute-based encryption, Modified IDEA, Serpent-based optimization plans, and Diffie-Hellman-based schemes have been linked with enormous increases in processing latency, cost of implementation, and computing complexity.

The design offered, nevertheless, has low encryption, execution, and key-generation time and minimizes the impact of credential-based attacks; this is by combining the use of AES-256 encryption with a dynamic authentication scheme utilizing the Virtual Password Function (VPF). The gap analysis shows that the proposed solution is innovative and is not related to other solutions in the healthcare security domain that use blockchain, FL, or cryptography. The solution is scalable, lightweight, and real-time.

2.2. New Development and Regulatory Compliance

Healthcare security research has recently focused on blockchain technology, Federated Learning (FL), and privacy-preserving approaches such as secure multi-party computation and differential privacy. Even though blockchain technology may enhance auditability and integrity, it complicates scalability and latency. Although centralised storage can be eliminated and data privacy enhanced, FL is prone to inference attacks and has high communication overhead. Advanced encryption algorithms are used to increase computational cost and enhance confidentiality. Regulations, such as GDPR and HIPAA, all require data minimization, auditability, encryption, and strict control over access, which are equally essential for compliance. Many existing solutions lack a good balance between compliance and efficiency in security. The suggested framework includes data encryption, access controls, and limited credential exposure, all of which promote practical compliance.

Table 1. Summary of existing security approaches for protecting healthcare data, including their core operations and identified challenges

Ref.	Research Author	Title	Operations	Challenges
[16]	Seval Capraz et al	A secure medical data-sharing framework using a public blockchain to combat pandemics like COVID-19.	ChaCha20-Poly1305 encryption enhances security by dividing files into 80 segments, requiring 29 segments for reconstruction, making it an efficient and reliable choice for safeguarding medical data.	However, it runs more slowly when encrypting data.
[17]	G. Moheshkumar et al	Secure data analytics for patient monitoring in healthcare applications employing the Secure Hash Algorithm (SHA-256).	This study proposes improving healthcare data analytics for secure patient monitoring by applying a security-driven strategy based on SHA-256.	Nonetheless, it raises substantial privacy concerns, increasing the danger of data breaches and illegal access.
[18]	Bilas Haldar et al	A modified IDEA with a 1024-bit key improves the security and efficiency of data transmission in healthcare.	MIDEA ensures safety against attacks by converting the given plaintext to ciphertext by using 64-bit blocking, substitution, and permutation. Furthermore, the overall method is cryptographically strong. Thus, it can be used for encryption and decryption with a 1024-bit key.	Due to the increased bit size, computational complexity is increased.
[19]	Abdulmohsen Almalawi et al	Securing Healthcare Data for a Modern System	The Lionized Remora Optimization-based Serpent (LRO-S) encryption method improves healthcare information protection by encrypting patient data stored in the cloud. Reducing privacy breaches and	Nonetheless, due to the high costs, the implementation process hinders the adoption of this technique.

			cyber-attacks.	
[20]	Parag Rastogi et al	Enhanced blockchain framework for ORAP verification and healthcare data security.	Improving patient care while reducing costs through efficient use of medical resources, initially secured by encrypting the resource provider's IoT data using Diffie-Hellman Galois–Elliptic Curve Cryptography (DHGECC).	Nevertheless, it experiences a processing delay in real-time scenarios.
[21]	VijayKumar et al	A Diffie-Hellman Key Exchange Algorithm: Improving Cloud Data Security: Cloud Data Security	DHKEA enhances cloud security by enabling secure cryptographic key exchange over open networks, addressing threats in decentralized environments.	However, they often struggle to accommodate the dynamic, interconnected nature of cloud infrastructure.

3. Proposed Model Description

In modern healthcare systems, patient information security and confidentiality are of utmost importance. The risks of cyberattacks and data breaches have increased dramatically as people rely more on EHRs, telemedicine, and remote patient monitoring. To securely protect medical data from unauthorized access, this research proposes an advanced AES-256 algorithm for efficient encryption. The novel VPF-based password generation mechanism is employed to resist cyber-attacks, comprising a secret little function and code booking approach. It provides strong security by preventing unauthorized access and reducing the vulnerabilities found in traditional password-based systems. Figure 2 shows the overall block diagram of the proposed model.

For uploading or downloading medical data in the EHR form by the patient or healthcare provider for the first time, they need to register. In the registration phase, the user downloads a random jar file from 11 jar files, each containing a unique expression, which is stored in the database for authentication purposes. After registration, users log in by entering their username and password. After logging in to the system, it generates an access key on the next page to prevent cyberattacks. The associated textbox is left empty instead of auto-filling the details. The user retrieves a session-specific access key, runs a JAR file on a Java-compatible device, and calculates an expression. AES-256 encryption protects patient records by converting them into unintelligible ciphertext, preventing unauthorized access. Only authorized users can decrypt and access EHRs. If it is wrong, access is blocked, and re-authentication is required.

3.1. System Architecture and Protocol Description

The many layers that comprise the proposed system architecture include user devices, authentication systems, encryption systems, and cloud-based electronic health record storage. It has a web or mobile interface through which user entities, including healthcare providers and patients, can interact with the system. The parameters of a codebook and a Virtual Password Function (VPF) are safely registered and stored on the server. The server issues a random challenge during the login process, which the VPF locally processes to produce a custom session credential. Patient-related information is encrypted with AES-256 before transmission and stored in the cloud database after successful authentication. Access to and decryption of data are restricted to verified users. The formal processes in the sequential implementation of the protocol are session termination, authentication, dynamic credential computation, challenge generation, and registration. This structured architecture and protocol flow offer scalability, resistance to attacks, and are helpful in health care systems in real time, allowing unambiguous separation between authentication, encryption, and storage.

3.2. Key Management Processes

The proposed system is based on a systematic key management process to protect cryptographic keys throughout their existence. AES-256 symmetric keys are generated by using a cryptographically safe random number generator when a user logs into the server. There is a reduction in the risk of key compromise when each key is used once and never reused. A key broadcast is never done in plaintext across the network; it is used solely to encrypt and decrypt patient data.

Quite to the contrary, there are server-side secure key derivation and storage techniques. To ensure forward secrecy, session keys are automatically invalidated whenever the user logs out or a session expires. In addition, the Virtual Password Function helps ensure that cryptographic material does not leak through the authentication channels by not relying on encryption keys. It will enhance the system's security by separating authentication credentials and encryption keys. The specified key issuance, use, storage, and revocation processes ensure controlled access, the secrecy, and the integrity of healthcare data.

3.3. Validation of Security Properties

The security features of the proposed framework are not validated using formal cryptographic proofs; rather, analytical reasoning and a specified threat model are used. The adversaries in the model assume they can perform attacks such as phishing, replay attacks, keylogging, and credential theft. Examples of authentication methods that offer these security measures include dynamic generation of virtual passwords, non-credential reuse, and challenge-response-based authentication using sessions. When applied with the generally accepted adversarial assumptions in realistic security studies, these strategies ensure that these attacks are resisted.

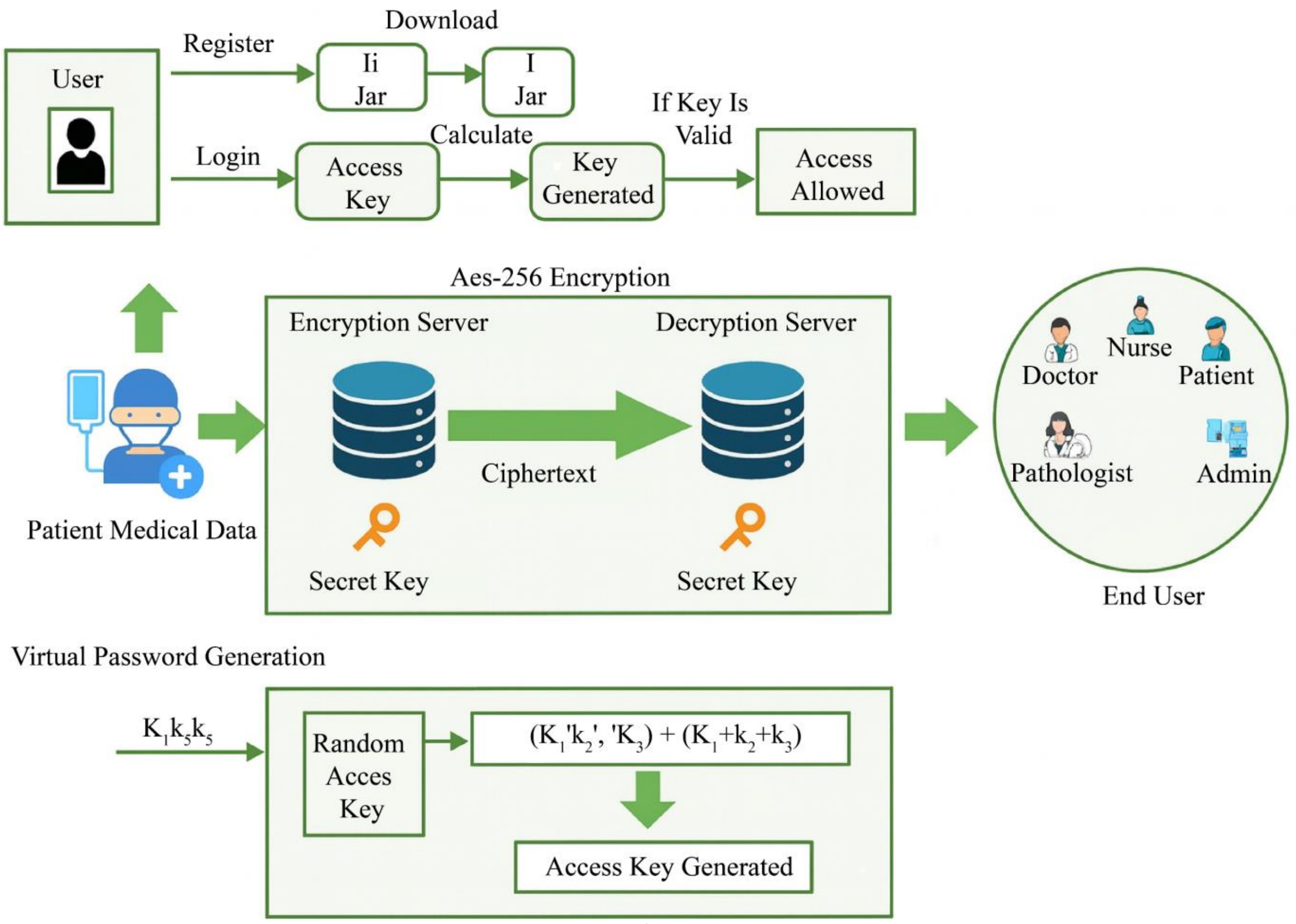


Fig. 2 Block diagram of the proposed model

4. Proposed System Modeling

4.1. AES-256-Based Encryption Algorithm

Encryption is a widely used method for securing sensitive data, transforming plain text into ciphertext composed of random characters. Only those with the designated key can decrypt this encoded information (Figure 3).

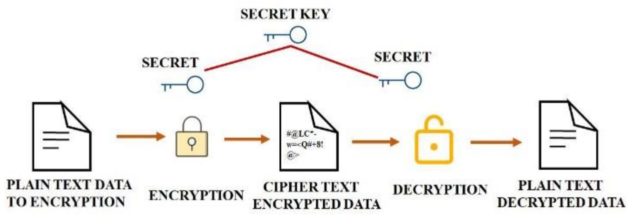


Fig. 3 Symmetric key encryption

4.2. AES Encryption Algorithm Modes

Block ciphers are designed to securely process large data streams without compromising security, and they encrypt data in fixed-size blocks, so identical plaintext produces identical ciphertext, since a deterministic algorithm does that. As a result, attackers may attempt to recover m by noticing repeated message parts. There are approaches for masking the output of ciphers.

The process works in such a way that a block of known plaintext is combined with a block formed from ciphertext. Therefore, the next round of encryption will use a modified input, which improves security. There are five standardized modes of operation:

4.2.1. Electronic Code Book (ECB)

In this mode, each block of plaintext is converted into a single ciphertext using the same key. Typically, this mode is appropriate for messages less than the block length. Longer communications that require encryption are first separated into properly sized blocks, with the final block padded as needed. As a result, the ECB approach is commonly employed to encrypt small amounts of data, providing some resistance to future cyberattacks.

4.2.2. Cipher Block Chaining

This mode requires that identical plaintext blocks produce different ciphertext blocks. To do this, cipher block chaining allows for an XOR operation between each plaintext block and the ciphertext from the previous round, all while using the same encryption key.

4.2.3. Cipher Feed Back (CFB)

Cipher Feedback mode makes it easier to convert a block cipher into a stream cipher, reducing the need for padding to ensure the message adheres to an integral number of blocks.

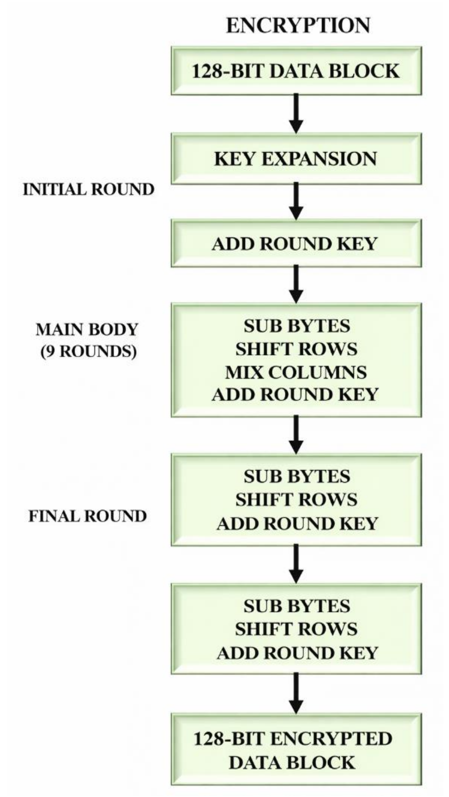

ENCRYPTION
128-BIT DATA BLOCK
KEY EXPANSION
INITIAL ROUND
ADD ROUND KEY
MAIN BODY (9 ROUNDS)
SUB BYTES SHIFT ROWS MIX COLUMNS ADD ROUND KEY
FINAL ROUND
SUB BYTES SHIFT ROWS ADD ROUND KEY
SUB BYTES SHIFT ROWS ADD ROUND KEY
128-BIT ENCRYPTED DATA BLOCK

Fig. 4 AES schematic diagram

4.2.4. Output Feed Back (OFB)

OFB mode is quite similar to CFB mode. However, there is an internal feedback mechanism to ensure that identical plaintext blocks do not generate identical ciphertext blocks. This mechanism operates independently of plaintext and ciphertext bit strings.

4.2.5. Counter (CTR)

Each block of plain text is encrypted with a different counter value. While creating the ciphertext block during the encryption process, the plaintext is XORed with the encrypted counter. Decryption processes in the opposite direction, using the same counter values and XOR to recover the original plaintext. The main benefits of this mode include clear architecture, resource efficiency, and enhanced hardware and software security.

The AES algorithm operates through four key steps in each encryption round, as depicted in Figure 4.

The Key Expansion procedure creates round keys from cipher keys, producing separate 128-bit round key blocks for each encryption round, plus an additional one.

Sub Bytes: Substitutes each byte with a predetermined lookup table (S-box) for increased security, as depicted in Figure 5. Byte substitution is the AES algorithm's only nonlinear operation and is critical to its security. AES employs 16 identical S-boxes, each handling an 8-bit input and producing an 8-bit output in tandem. Byte replacement replaces each byte in the algorithm state with a new byte in a nonlinear fashion.

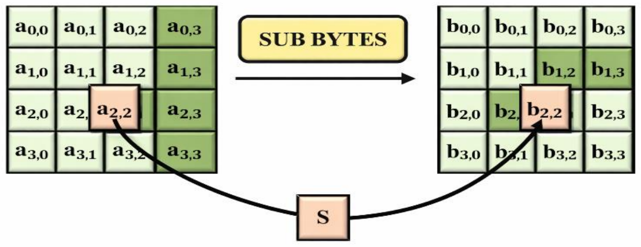

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,3}$
SUB BYTES
S
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,2}$ $b_{3,3}$

Fig. 5 Sub bytes step updates each byte in the array using an 8-bit s-box

The S-box generation in AES follows two steps:

- In GF (2^8), a byte is turned into its multiplicative inverse, except for element 0, which maps to itself.
- The output undergoes an affine transformation, which involves multiplying each byte by a constant matrix and adding an 8-bit hexadecimal constant vector {63}, as specified in (1).

$$\begin{pmatrix} b7 \\ b6 \\ b5 \\ b4 \\ b3 \\ b2 \\ b1 \\ b0 \end{pmatrix} = \begin{pmatrix} 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \\ 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \end{pmatrix} \times \begin{pmatrix} a7 \\ a6 \\ a5 \\ a4 \\ a3 \\ a2 \\ a1 \\ a0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{1}$$

Shift Rows is a transposition step that changes the rows of a data matrix to introduce diffusion: the row-shift procedure cyclically adjusts each row of the algorithm state using various displacement amounts (Figure 6). The initial row is unchanged, while the second and third rows are rotated by 3-byte and 2-byte right shifts, respectively. The transformation substantially improves diffusion in the AES algorithm, thus enhancing security.

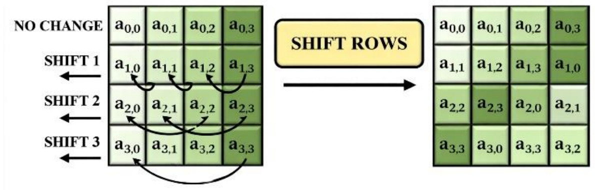


Fig. 6 The function cycles through the state's rows, shifting bytes by a specific offset

Mix Columns: A mixing operation that transforms each column using mathematical functions to strengthen encryption, as given in Figure 7.

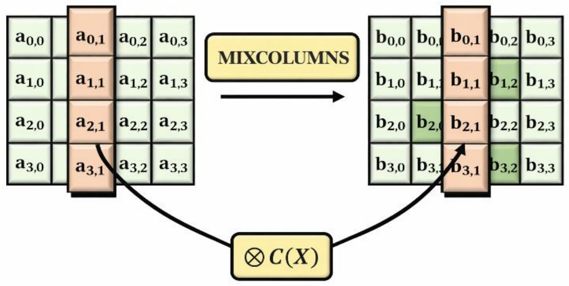


Fig. 7 An invertible linear transformation combines four bytes from each state column

A column-blending transformation is a process that individually applies a blending operation to each column in the algorithm state. Each column is regarded as a polynomial with coefficients in GF (2^8), multiplied by a fixed polynomial $c(x)$, then modularly reduced using the polynomial $(x^4 + 1)$.

$$c(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\} \quad (2)$$

Each byte in the algorithm state is transformed into a new value, determined by its relationship with the four bytes in the column.

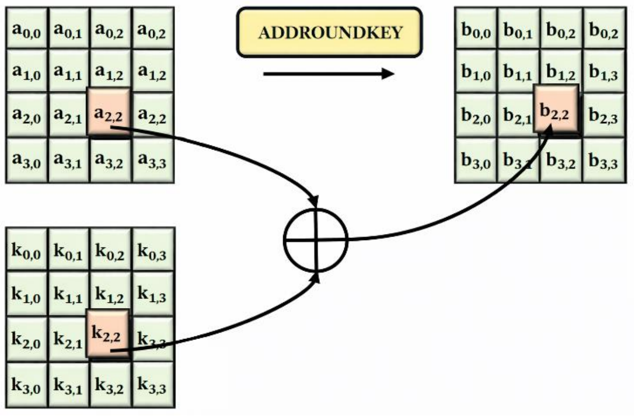


Fig. 8 Round key combined

Add Round Key: A step in which the transformed data is combined with a round key using bitwise XOR for added security, as illustrated in Figure 8.

A round key addition involves a bitwise XOR operation between the 16-byte algorithm state matrix and the 16-byte subkey. Subkey is derived through key expansion from the initial key.

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & ka_{3,3} \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \quad (3)$$

Mix Columns is a mixing step that combines four bytes from each column to spread diffusion across the state matrix. Moreover, in the last round, Mix Columns is skipped, leaving only SubBytes, ShiftRows, and AddRoundKey.

- Generate round keys from the cipher key.
- Set the state array and apply the initial round key.
- Perform standard rounds (1 to 9) executing all four transformations.
- Execute the final round, omitting Mix Columns.
- Output the ciphertext chunk from the last round.

Key expansion algorithm: AES is a symmetric block cipher with a 128-bit block length that supports three different key sizes: 128, 192, and 256 bits. AES operates on a symmetric key encryption model, meaning a single secret key is used for both encryption and decryption. The encryption process varies depending on the key length, as listed in Table 2.

Table 2. AES-256 algorithm block, key size, and number of rounds

AES Algorithm	128-bit	192-bit	256-bit
Key Size	4	6	8
Number of Rounds	10	12	14
Block Size	4	4	4

During encryption, multiple transformations are applied to data blocks in a series of rounds. The number of rounds is based on the key length. Ten rounds for 128-bit keys. 192-bit keys have two rounds, and 256-bit keys have fourteen rounds. These rounds encrypt data by changing data at every level to increase security. The AES algorithm is controlled by rounds, with four key functions in each encryption and decryption round. The encryption rounds are SubByte, ShiftRows, MixColumn, and AddRoundKey.Meanwhile, the decrypting InvShiftRow, InvSubByte, AddRoundKey, InvMixColumn. The last round has a slight variation and requires three functions. This system will allow data to be safely transformed.

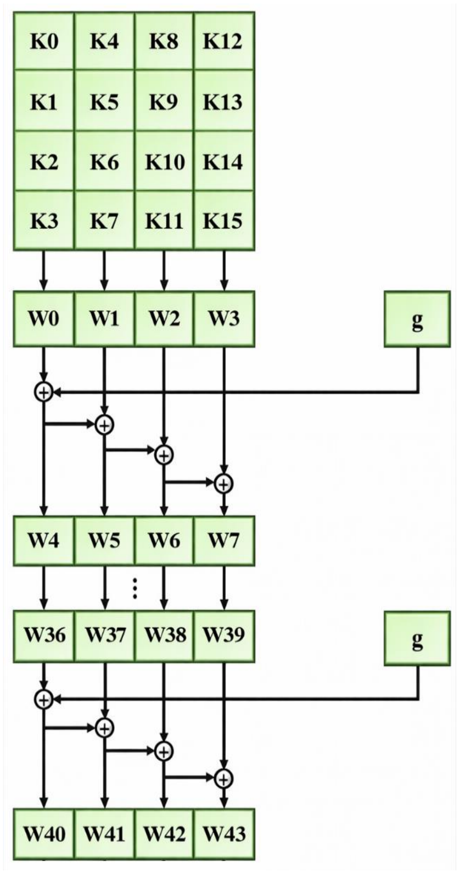


Fig. 9 Key expansion algorithm

In the AES algorithm, subkeys are generated recursively, meaning that to compute subkey w_i, the preceding subkey w_{i-1} must be known. The fundamental unit for key expansion in AES is a 32-bit word. The key expansion algorithm takes 4 words as input and produces a 1D array of 44 words.

Initially, the AES key serves as the first 4 keywords in the extended key array. Subsequently, every newly generated set of 4 keywords is appended to the remaining portion of the extended key array. Within this array, each new keyword w_i is derived from w_{i-1} *and* w_{i-4}. Additionally, a specialized function g is applied to compute keywords whose array index is a multiple of 4.

From Figure 9, wi keyword is computed as,

$$w_i = \begin{cases} w_{i-4} \oplus w_{i-1}; & i \bmod 4 \neq 0 \\ w_{i-4} \oplus g(w_{i-1}); & i \bmod 4 = 0 \end{cases} \tag{4}$$

Function $g\,()$ is a non-linear transformation that processes a 4-byte input to generate a 4-byte output. First, the four input bytes undergo a left cyclic shift of 1 byte. Next, each of the four bytes is substituted using S-box transformations. Finally, the transformed byte is XORed with round constant $Rcon[j]$, which is a 32-bit word where only the leftmost byte holds a nonzero value.

The key keyword undergoes an XOR operation with the round constant, which involves explicitly XORing the first byte positioned to the left of the round constant. Each round has a distinct round constant, defined as $Rcon[j] = (Rc[j], 0,0,0)$, where $Rc[1] = 1$ and subsequent values follow the relation $Rc[j] = Rc[j-1] \times 2$, with multiplication performed in $GF(2^8)$. Table 3 provides the hexadecimal representation of the ten round constants used in the AES algorithm.

Table 3. Hexadecimal representation of rounds

j	$Rc[j]$
1	01
2	02
3	04
4	08
5	10
6	20
7	40
8	80
9	1B
10	36

The round constants used in AES ensure that no two round keys in the AES key expansion are alike and are not susceptible to cryptanalysis. In this way, data is modified at each stage to achieve the required level of encryption for data protection. After using the AES-256 algorithm for effective encryption of the medical data, authentication follows.

4.3. Virtual Password Authentication

To authenticate a user, a system (S) must confirm their identity (U) by verifying the user-provided credentials: a password (X) and user ID (U). This authentication process is denoted as $S \rightarrow U: U, X$, where S verifies U using these fixed credentials. Because both U and X remain unchanged, passwords are designed for easy recall. However, this convenience makes them susceptible to theft, allowing adversaries to gain unauthorized access to a victim's account. While making X a random variable could enhance security, it would make it difficult for users to recall their passwords. To address this challenge, we introduce a virtual password scheme, which dynamically generates passwords while maintaining usability and security.

A virtual password is a dynamically generated authentication credential, uniquely created each time through a virtual password scheme and subsequently sent to the server for validation. The virtual password scheme (P) comprises two elements: a static alphanumeric value (X), referred to as the hidden password, and a function (F) that operates within a predefined letter space (ψ). This function, referred to as the virtual password function, generates a virtual password (V) used for authentication. VPF includes hidden parameters (H), which serve as confidential elements shared between the server and the user. If such parameters exist, the function is denoted as $FH\ (\dots)$. The hidden password X is represented as a vector $x1, x2, \dots, xn$, where $xi\ for\ (i = 1,2, \dots, n)$ is a digit, and n is the length of the password. Similarly, a random number (R), called random salt, is provided by the server and displayed on the login screen as $r1, r2, \dots, rn$. The virtual password (V), expressed as $v1, v2, \dots, vn$ is used for authentication. User submits (U, V), $where\ U$ is the user ID. On the server side, the system computes V using the same function and compares it with the submitted password. This process is represented as $V = FH\ (X, R)\ or\ FH\ (xi\ , ri\ )$, ensuring secure authentication.

The server efficiently verifies a user if F is a bijective function, as it allows straightforward authentication. However, even if F is not bijective, the system still authenticates the user by retrieving their record from the database using their user ID (U), computing V, and comparing it with the provided password. A bijective function simplifies the process by enabling the system to use a reverse function to derive V, and this assumption is not mandatory. To enhance security and flexibility, users should be able to choose their own hidden password. A jar-file security mechanism is introduced, allowing users to choose a VPF that aligns with their preferred security level, as illustrated in Figure 10.

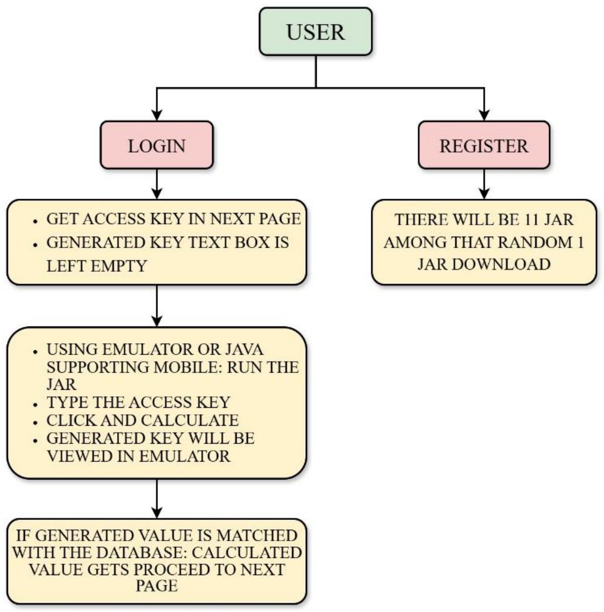

USER
LOGIN
REGISTER
GET ACCESS KEY IN NEXT PAGE
GENERATED KEY TEXT BOX IS LEFT EMPTY
THERE WILL BE 11 JAR AMONG THAT RANDOM 1 JAR DOWNLOAD
USING EMULATOR OR JAVA SUPPORTING MOBILE: RUN THE JAR
TYPE THE ACCESS KEY
CLICK AND CALCULATE
GENERATED KEY WILL BE VIEWED IN EMULATOR
IF GENERATED VALUE IS MATCHED WITH THE DATABASE: CALCULATED VALUE GETS PROCEED TO NEXT PAGE

Fig. 10 User login and registration

4.3.1. Registration Module

In the registration module, users (Patients, Doctors, Nurses, Healthcare Technicians) register their accounts. At first, it offers a combination of 11 unique jar files, each containing a distinct, small encryption function. A jar file is downloaded with a random value, then merged with their account.

4.3.2. Login Module

Once registered, the user enters the login module where security measures are enforced. At first, the user gives a username and password for authentication. The system generates a one-time key for that session only; it is generated elsewhere, not the one created during registration, to prevent replay attacks. The login system includes an appropriate text box for the user to enter a dynamic key.

The value needs to be entered in an empty text box is computed as: the previously installed JAR files are opened by the user on a Java-compatible device. In which the session-specific access key is entered. By using predefined mathematical expressions, the jar file processes the access key based on the “Secret Little Function” generated during the registration module. Based on the encoded logic, the output key is computed and entered in the designated text box for authentication within the jar.

4.3.3. Validation

Validating the Generated Key

The key generated on a Java-compatible device or emulator is entered by the user into the system text box if the entered value matches the server's secret logic and access key. If the value is correct, then access to EHR data (Read patient health records, prescribe medications, publish lab results, and manage hospital staff tasks) is granted. If the provided key value is incorrect, the access is permitted.

If the key is incorrect, access is denied, and the user may be prompted to retry or reauthenticate.

4.4. Secret Little Functions

The most effective security strategies involve allowing users to define custom encryption functions or programs. Since these functions remain private between the user and the server, and the range of possible functions is vast and complex, this method ensures a high level of security, even for simple functions. Historically, classical encryption methods relied on keeping the encryption algorithm itself secret. Today, algorithms are open, and encryption keys are kept secret in modern cryptography.

The key shift happened because secret algorithms prevent communication and compatibility between systems, for instance, in commercial applications and networking protocols. Thus, the modern cryptographic schemes prefer to keep keys secret since they are usually small data elements and make the algorithm public. The proposal enhances security, promotes interoperability, and enhances transparency of encrypted data, as shown in Table 4.

Table 4. First step in user registration

Select your preferred PIN registration method from the following options:
Default Option [Default ( )]: No virtual function is used. Recommended Virtual Function ( ): A suggested encryption function is applied. Custom Functions: Function B = XXX Function B = YYY Function B = ZZZ User-Defined Function: A unique function set by the user, shared securely between the user and the server. System-Defined Function with Adjustable Security Levels: Choose a security level: Low ( ), Medium ( ), High ( ), or Very High ( ). User-Defined Program: A custom encryption program written in Java, shared between the user and the server.

Share-Little Function is a security approach in which users define their own encryption functions rather than relying solely on secret keys. Since only the user and the server know the function, and the range of possible functions is infinite, even simple functions become highly secure. Traditional ciphers often used secret encryption algorithms, and modern ciphers keep algorithms public while protecting the encryption keys. Having algorithms open ensures that other systems, such as Wi-Fi, remain interoperable as they communicate across different manufacturers. Nonetheless, for private user-server interactions, the use of secret encryption algorithms makes attacks hard.

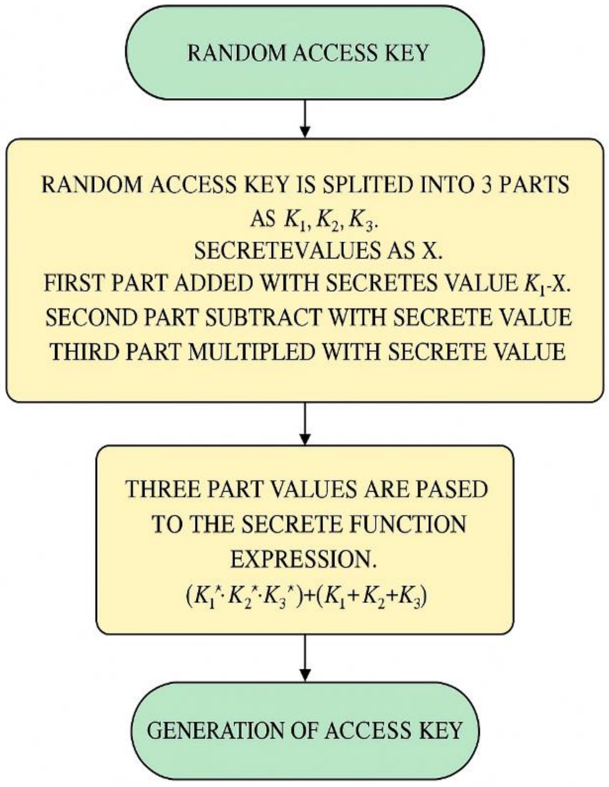

RANDOM ACCESS KEY
RANDOM ACCESS KEY IS SPLITED INTO 3 PARTS AS K_1, K_2, K_3.
SECRETEVALUES AS X.
FIRST PART ADDED WITH SECRETES VALUE K_1-X.
SECOND PART SUBTRACT WITH SECRETE VALUE
THIRD PART MULTIPLED WITH SECRETE VALUE
THREE PART VALUES ARE PASED TO THE SECRETE FUNCTION EXPRESSION.
$(K_1^* \cdot K_2^* \cdot K_3^*)+(K_1+K_2+K_3)$
GENERATION OF ACCESS KEY

Fig. 11 User access code generation

From Figure 11, the random-access key is classified into $K1$, $K2$, $K3$, and a secret value is utilized to influence each part $(K1 + X)$, $(K2 - X)$, $(K3 * X)$. These modified values are given as a secret function $(K1 * K2 * K3) + (K1 + K2 + K3)$. At last, the final access key is generated to encrypt patient information. Password cracking becomes much harder because bridge passwords are generated through secret calculations. It has strong defence against attacks such as phishing, keylogging, brute-force attempts, and more to secure data access.

Safety is the one concern. It could be said that the user will not be able to create any safe function. Even decision functions are secure. The attacker does not know which function was chosen. Hence, the level of safety is still high.

The password modification techniques are:

- Alter a single bit in the password.
- Modify one digit in the password.
- Increase each odd digit by one and decrease each even digit by one.
- Triple the first digit of the password.
- Compute the password as an integer from ASCII codes, then apply the formula: 100x + birthdate, where x represents the transformed password.
- Reverse even bits of the password in its binary representation.
- Additional variations and modifications as needed.

These techniques introduce randomness and complexity to password generation, enhancing security against unauthorized access.

4.5. Code Booking

When the user does not have the helper application, they compute their dynamic password using a virtual function, with a random salt and a manually computed constant virtual password component. The virtual function needs to be simple enough to perform without a mobile device (Manual Computation). Changing your password is done in a similar way to conventional methods: to select a new password, a new virtual function, or both. In any case, the process needs to memorise your updated virtual password. The importance of the virtual function is the assurance of security when a recommended function is used. With the sheer volume of functions available, careful selection will help avert threats. Virtual functions, when designed effectively, can enhance resilience against phishing, keyloggers, and shoulder-surfing attacks.

- The function must receive random input from the server, ensuring different input values for different users and for each user's login. Keyloggers cannot capture and reuse the password since the actual password is never typed directly.
- User-friendly design. As complexity increases, so does

security; too many complicated functions can be complex for users to remember and/or use, and we strive to create virtual functions that are both secure and manageable.

- The password functionality should hide its secrets so no one can discover any information from the user's password. This prevents thieves who use stolen passwords from getting into places they should not be.
- The actual equivalent password must not be computable from the function, even if attackers obtain its potential information.

They provide guidelines for the effective design of virtual functions. Although there are zero-knowledge authentication protocols, they are often computation-intensive and therefore not applicable. Moreover, some functions can appear harmless and are unsafe upon closer review.

Using fake passwords is an effective way to protect against phishing attacks. For this, users need only a small codebook, which can be easily carried as a printed card or stored on a PDA or mobile device. It would be unrealistic to ask users to memorize an entire codebook. A zero-knowledge interactive proving protocol would be the best solution. However, current constraints do not allow for this. As far as security is concerned, the machine that will be used as a server must have sufficient computational power to run a cryptographically secure Random Number Generator (RNG). This safeguard prevents the system from being compromised if the user's book is ever hijacked. In such cases, the user requests a new codebook without altering RNG's parameters. It is important to note that Linear Congruential Generators (LCGs) do not meet the standards of cryptographically secure RNGs.

The initial codebook design is simple. During the setup process, the user selects the desired password length, denoted as n. The server then generates n random numbers, each consisting of 10 digits. For instance, if the system is securing a 4-digit PIN ($i.e.$, $n = 4$), the server provides four random numbers: $X0$, $X1$, $X2$, and $X3$, each containing 10 digits. The digits of Xi is represented as (0), $x(1)$, $x(2)$, … , $x(9)$. The user's codebook is structured accordingly.

$$\begin{array}{l} x(0,0),x(0,1),x(0,2),\ldots,x(0,9) \\ x(1,0),x(1,1),x(1,2),\ldots,x(1,9) \\ x(2,0),x(2,1),x(2,2),\ldots,x(2,9) \\ x(3,0),x(3,1),x(3,2),\ldots,x(3,9) \end{array} \tag{5}$$

It is up to the user to choose whether to save or memorize the codebook. To log in, the system displays a four-digit random number $R = abcd$, with each letter representing a digit. The virtual password for the user to type in is:

$$(x(i,a)x(i,b)x(i,c)x(i,d)) \tag{6}$$

For security evaluation, phishing attacks are the primary focus, as they represent the most aggressive form of attack where the adversary has control over the random number R. In each instance, the attacker provides a fraudulent random number R to the victim. If successful, the attacker gains access to four corresponding digits from the codebook. Consequently, the probability that the attacker correctly guesses a single digit of the password depends on the likelihood that the system requests the same position, combined with the probability that the system asks for any of the other nine positions and that the attacker correctly guesses it.

$$\frac{1}{5} + \frac{9}{10} \times \frac{1}{10} \approx \frac{1}{5} \tag{7}$$

The probability of an attacker successfully breaching a victim's account after a single phishing attempt is calculated as (0.2)2 = 1⁄625. However, since attackers often execute multiple phishing attempts, victims remain unaware during the initial rounds. To maximize the amount of stolen information, the attacker requests different password positions in each attempt.

Let p represent the number of successful phishing attempts targeting the same user, n denote the password length, and s indicate the number of unique symbols available for each digit (in this case, s = 10). The probability of an attacker successfully accessing a victim's account is determined using a specific formula based on these parameters.

$$\left(\frac{p}{s} + \frac{s-p}{s} \times \frac{1}{s}\right)^p = \left(\frac{1+p}{s} - \frac{p}{s^2}\right)^p \tag{8}$$

When the system uses this codebook and gets the following Table 5 outcome.

A conventional PIN code typically consists of four Arabic digits, providing a key space of size 10 in a phishing-free environment. Without virtual password protection, a single successful phishing attack can completely compromise the PIN. The codebook approach significantly reduces the likelihood of a breach, as repeated phishing attempts alert the victim, prompting them to stop engaging with the attacker. After three successful phishing attacks, the probability of unauthorized access increases, and the account remains relatively secure if the system locks access after multiple failed login attempts. However, since longer passwords are impractical, an alternative approach is to expand the symbol set by incorporating letters and special characters. A symbol size of 64 is considered reasonable for improving security while maintaining usability.

Table 6 indicates that, in a phishing-free environment, the security level of four-digit passwords after five successful phishing attempts remains at the level of a traditional four-digit PIN code, even when the symbol size is extended to 64. In actuality, a user is unlikely to appease a phisher more than five times before becoming distrustful. It is noted that cyber-attacks differ greatly from chosen (or known) plaintext attacks in the context of cryptography; the phisher does not have access to a significant amount of plaintext.

Table 5. The number of phishing attacks occurred

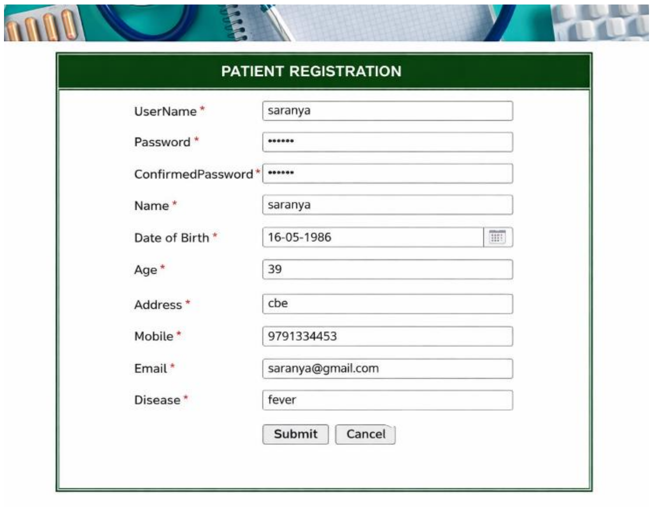
Symbol Size (s = 10)		Number of phishing attacks occurred ($\boldsymbol{p}$)					
		0	1	2	3	4	5
Length of the password ($\boldsymbol{n}$)	4	1.00×10^{-4}	1.30×10^{-3}	$6.1.5 \times 10^{-3}$	1.87×10^{-2}	4.48×10^{-2}	9.15×10^{-2}
	6	1.00×10^{-6}	4.70×10^{-5}	4.82×10^{-4}	2.57×10^{-3}	9.47×10^{-3}	2.77×10^{-2}
	8	1.00×10^{-8}	1.70×10^{-6}	3.78×10^{-5}	3.51×10^{-4}	2.00×10^{-3}	8.37×10^{-3}
	10	1.00×10^{-10}	6.13×10^{-8}	2.96×10^{-6}	4.81×10^{-5}	4.24×10^{-3}	2.53×10^{-3}

Table 6. Phishing attacks-free security levels

Symbol Size ($\boldsymbol{S} = \mathbf{10}$)		Number of phishing attacks occurred ($\boldsymbol{p}$)										
		0	1	2	3	4	5	6	7	8	9	10
Length of the password ($\boldsymbol{n}$)	4	5.96×10^{-8}	9.84×10^{-7}	5.03×10^{-6}	1.60×10^{-5}	3.92×10^{-5}	8.14×10^{-5}	1.51×10^{-4}	2.58×10^{-4}	4.13×10^{-4}	6.30×10^{-4}	9.23×10^{-4}
	6	1.46×10^{-11}	9.76×10^{-10}	1.31×10^{-8}	6.39×10^{-8}	2.45×10^{-7}	7.43×10^{-7}	1.85×10^{-6}	4.14×10^{-6}	8.40×10^{-6}	1.58×10^{-5}	2.81×10^{-5}
	8	3.55×10^{-15}	9.68×10^{-13}	2.53×10^{-11}	2.56×10^{-10}	1.53×10^{-9}	6.62×10^{-9}	2.28×10^{-8}	6.46×10^{-8}	1.71×10^{-7}	3.97×10^{-7}	8.53×10^{-7}
	10	8.67×10^{-19}	9.60×10^{-16}	5.68×10^{-14}	1.02×10^{-12}	9.59×10^{-12}	5.97×10^{-11}	2.80×10^{-10}	1.07×10^{-9}	3.47×10^{-9}	9.97×10^{-9}	2.59×10^{-8}

5. Results and Discussion

In this section, results from the implementation of AES-256 encryption to safeguard patient data, along with enhanced authentication logic for virtual password management, are analysed.

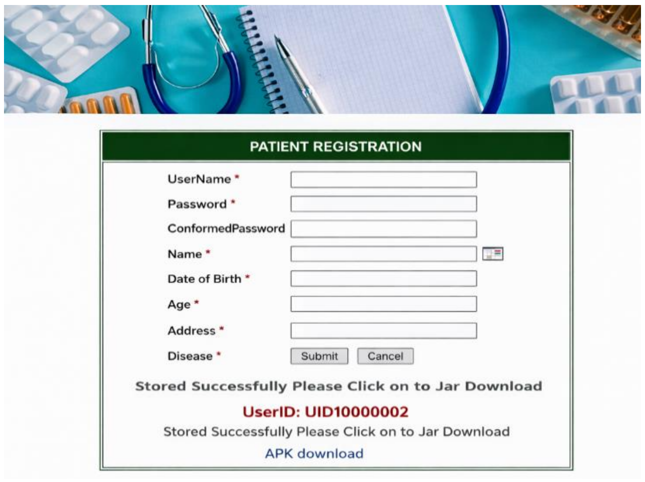


PATIENT REGISTRATION

UserName *	saranya
Password *	••••••
ConfirmedPassword *	••••••
Name *	saranya
Date of Birth *	16-05-1986
Age *	39
Address *	cbe
Mobile *	9791334453
Email *	saranya@gmail.com
Disease *	fever

Fig. 12 Registration form

Figure 12 shows the patient registration form used in the healthcare system to authenticate patient information. The patient registration form includes key fields such as Username, Password, Name, Date of Birth, Age, Address, Mobile number, Email, and Disease conditions. This registration process provides precise data entry and ensures that healthcare authorities store and manage HER data effectively. The AES-256 encryption method is used to create and authenticate sensitive patient records. The healthcare service visitor registration form can enhance patient registration and is EHR-compliant.



PATIENT REGISTRATION

UserName *
Password *
ConformedPassword
Name *
Date of Birth *
Age *
Address *
Disease *

Stored Successfully Please Click on to Jar Download

UserID: UID10000002

Stored Successfully Please Click on to Jar Download

APK download

Fig. 13 Jar file downloaded

Figure 13 shows the download webpage for the jar file. The secure healthcare system uses AES-256 encryption and virtual passwords to protect patient registration, authentication, and data management from unauthorized access. Patients must register by entering the necessary information, which will be encrypted and safely stored on the MySQL cloud database. To stop brute-force and phishing attacks, the solution dynamically validates credentials via a code-reservation system. Similarly, a jar file download enables consumers to access encrypted health data through a mobile application. Backend processes run on Java JSP Servlets are highly scalable and maintainable. So, it is a practical approach to digital healthcare that respects privacy.

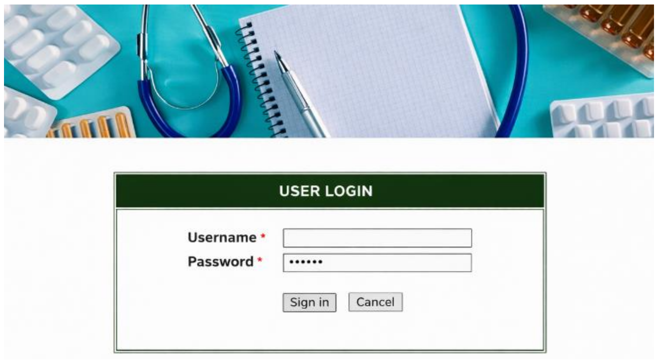
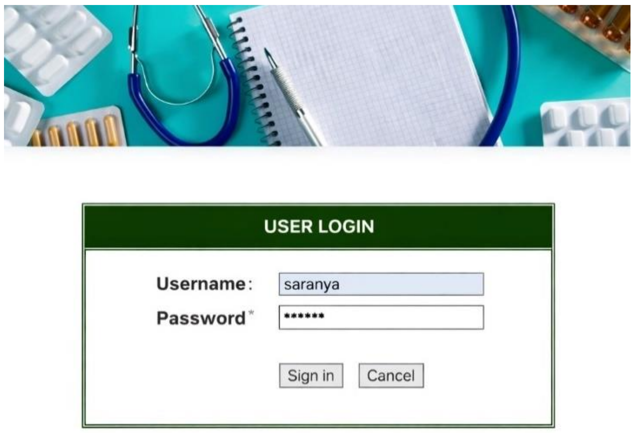


Fig. 14 User login form

Figure 14 represents the user login form. On this webpage, the user's name and a strong password are required for authentication. The patient information is protected by the AES-256 mechanism to prevent unauthorized access. Furthermore, a virtual password mechanism is integrated, utilizing a code-based approach to modify login credentials and enhance security dynamically.


USER LOGIN

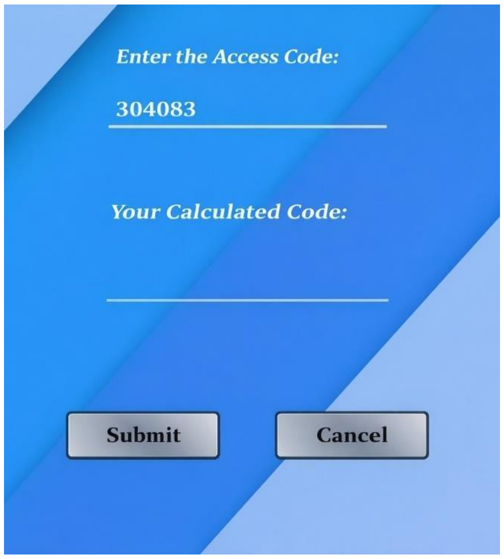
Username: saranya

Password: ••••••

Fig. 15 User login credentials

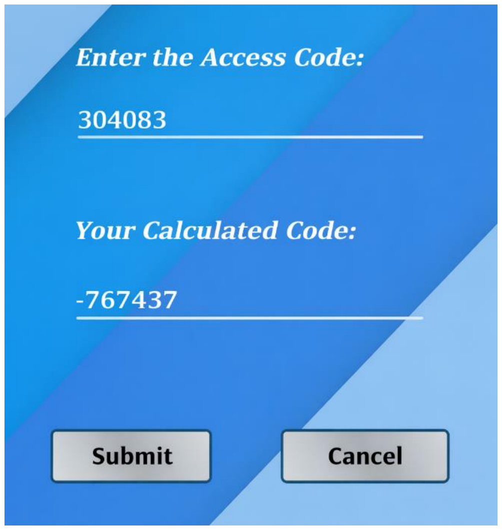
According to Figure 15, it is the login credentials form used by users. When using this website, users enter details such as a username and password for authentication. Moreover, the credentials are sent and stored in MySQL cloud databases, encrypted with AES-256 for extra security by the system. After a successful sign-in, the user is redirected to the next page that contains the access code. This access code serves as an extra layer of protection, enhancing virtual password authentication using a code booking method that prevents phishing and brute-force attacks.

When the jar file is installed on a mobile phone, the following page appears. This webpage contains a secure access code entry form, where users enter an access code to generate a new computed code, as shown in Figure 16.


Enter the Access Code: 304083

Your Calculated Code:

Fig. 16 Access code inserted


Enter the Access Code: 304083

Your Calculated Code: -767437

Fig. 17 Virtual password generation

Figure 17 displays an interface that makes safe authentication for healthcare systems easier by allowing users to enter an access code and generate a calculated code for verification. The interface has separate fields for entering the access code and displaying the dynamically produced computed code. The Submit and Cancel buttons allow users to confirm or reset their actions. The solution includes AES-256 encryption and virtual password authentication, which ensures strong data protection against unauthorized access. The calculated code is dynamically updated using a small, hidden function and code-based booking to deter phishing and brute-force attacks. Once a user is authenticated, they gain access to encrypted patient records that comply with medical privacy laws.

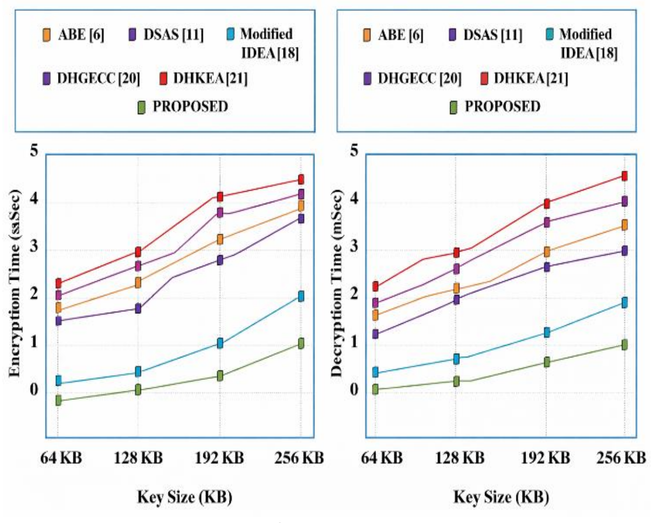


Fig. 18 Comparison of encryption and decryption time

Figure 18 includes a graph showing the time required for encryption and decryption using various algorithms. The researchers analysed encryption algorithms like Attribute-Based Encryption (ABE) [6], Data Sharing and Authorized Search (DSAS) [11], Modified International Data Encryption Algorithm (MIDEA) [18], Diffie-Hellman Galois–Elliptic-Curve Cryptography (DHG-ECC) [20], Diffie-Hellman Galois–Elliptic-Curve Cryptography (DHGECC) [21] with the proposed AES-256 algorithm. The designed AES-256 technique is the fastest encryption and decryption method in all key sizes. Thus, it is highly efficient and promotes feasible healthcare systems.

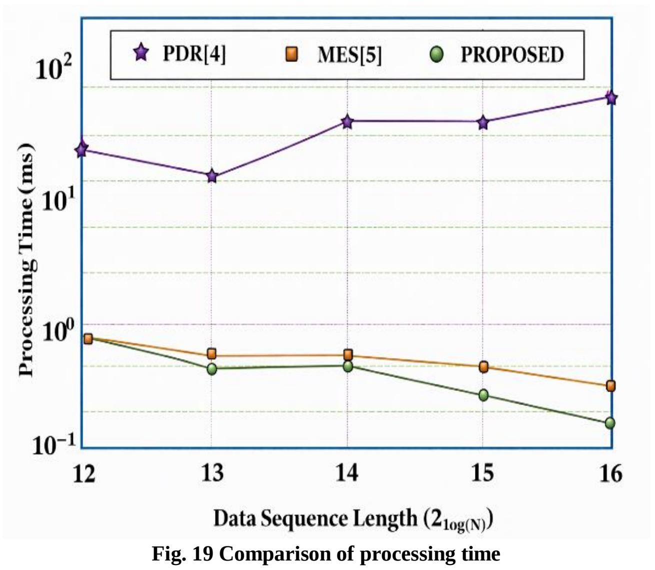


Fig. 19 Comparison of processing time

Figure 19 compares processing time (ms) with data sequence length (2log (n)) for PDR [4], MES [5], and the proposed AES-256 algorithm. Based on the graph, the processing time of the PDR and MES methods is the highest, ranging from 10 ms to almost 100 ms.

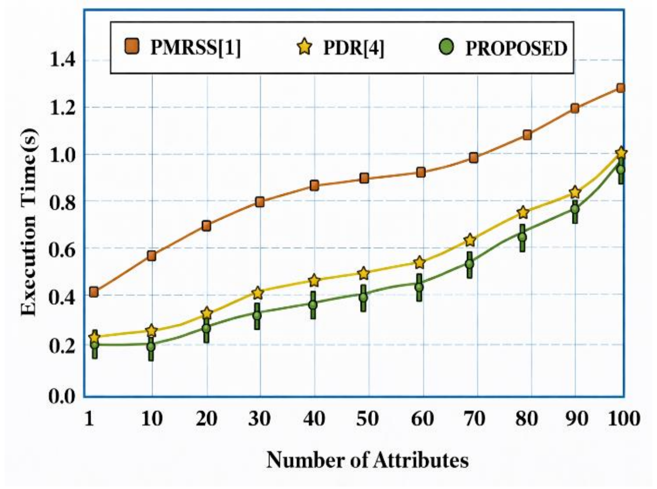


Fig. 20 Comparison of execution time

The chart in Figure 20 illustrates the execution time required for three encryption techniques: PMRSS [1], PDR [4], and the proposed AES-256 algorithm. The processing time range (in milliseconds) is between 10^{-1} and 10^{2} (ms). PDR displays a significant computational burden and the longest execution time, while PMRSS is moderately efficient. The proposed AES-256 algorithm is well-suited for real-time medical applications and for securing healthcare data. This can be gauged from the algorithm's execution time, which is pretty efficient.

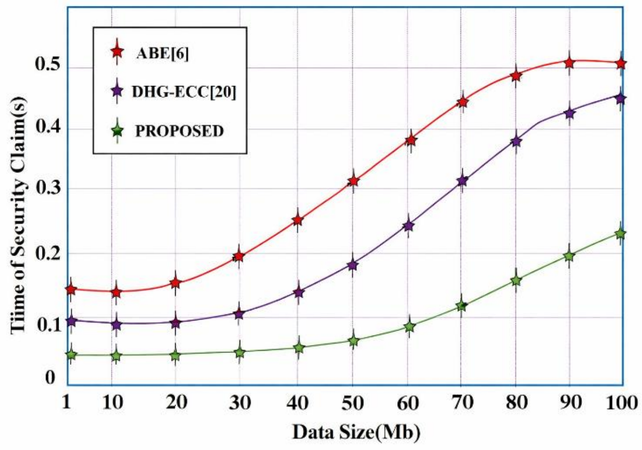


Fig. 21 Security level evaluation

Figure 21 shows how secure different data size security approaches are. The ABE [6] and DHG-ECC [20] took the longest time compared to the suggested approach, AES-256, which exhibited the shortest time suitable for protecting patient information in healthcare.

As shown in Figure 22, the key generation time for several algorithms, including the proposed technique, Modified IDEA [18], DSAS [11], and DHG-ECC [20]. The new algorithm generates keys the fastest, producing excellent results. The Modified IDEA, DES, and DHGECC traditional algorithms take longer to process and are growing faster than the proposed algorithm. As stated before, the paper reveals that the proposed algorithm is effective for VPF-based rapid key generation using a secret little function and code booking approach for the health care system to process patient data securely and efficiently.

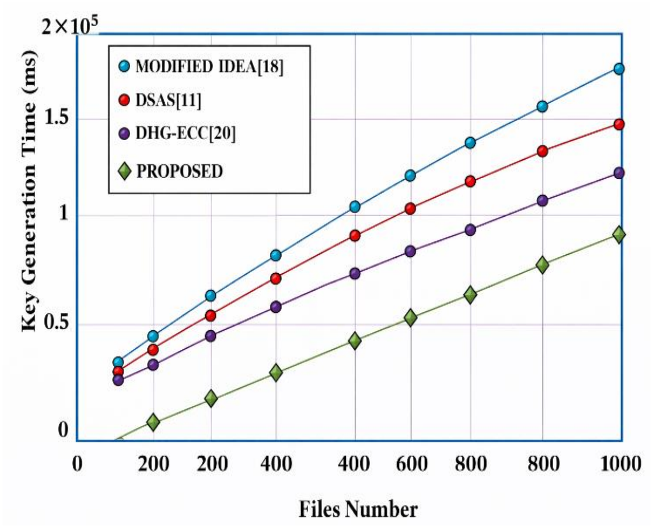


Fig. 22 Performance evaluation of key generation time

6. Conclusion

The proposed system highlights the critical importance of safeguarding patient data to secure the digital healthcare system. The AES-256 encryption and VPF code booking technique framework will prevent security loopholes and hacking vulnerabilities from gaining access. VPF modifies the authentication process by generating a time-sensitive access key, thereby mitigating the risks of password-based attacks. Plus, there's that little secret function in the system's authentication framework. This significantly enhances security, as the password is encoded using the system's logic in accordance with the norms. This will prevent password guessing and brute-force attacks. Patient data is encrypted before storage to prevent unauthorized access and ensure privacy. The system is developed in Java technology, and MySQL is used as a secure database for all data. The Java JSP Servlet-based interface for a system is easy to use and manage. According to security tests, AES-256, VPF, and the code booking mechanism are very effective at preventing cyber-attacks. Thus, it can use this framework robustly and scalably in a cloud-based system for a healthcare facility.

Acknowledgments

The authors sincerely thank the supervisor for providing consistent guidance and strong support throughout the progress of this research.

References

[1] Yi Sun et al., “PMRSS: Privacy-Preserving Medical Record Searching Scheme for Intelligent Diagnosis in IoT Healthcare,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1981-1990, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Rui Zhang, Rui Xue, and Ling Liu, “Security and Privacy for Healthcare Blockchains,” *IEEE Transactions on Services Computing*, vol. 15, no. 6, pp. 3668-3686, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Guangjun Wu et al., “Privacy-Preserved Electronic Medical Record Exchanging and Sharing: A Blockchain-based Smart Healthcare System,” *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1917-1927, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jusak Jusak et al., “A New Approach for Secure Cloud-Based Electronic Health Record and its Experimental Testbed,” *IEEE Access*, vol. 10, pp. 1082-1095, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Maryam Shabbir et al., “Enhancing Security of Health Information Using Modular Encryption Standard in Mobile Cloud Computing,” *IEEE Access*, vol. 9, pp. 8820-8834, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Fengqi Li et al., “EHRChain: A Blockchain-Based EHR System using Attribute-based and Homomorphic Cryptosystem,” *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2755-2765, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Mohammad Kamrul Hasan et al., “Lightweight Encryption Technique to Enhance Medical Image Security on Internet of Medical Things Applications,” *IEEE Access*, vol. 9, pp. 47731-47742, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Sangjukta Das, and Suyel Namasudra, “A Lightweight and Anonymous Mutual Authentication Scheme for Medical Big Data in Distributed Smart Healthcare Systems,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 21, no. 4, pp. 1106-1116, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Mehedi Masud et al., “Lightweight and Anonymity-Preserving User Authentication Scheme for IoT-Based Healthcare,” *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2649-2656, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Leonardo Da Costa et al., “Sec-Health: A Blockchain-Based Protocol for Securing Health Records,” *IEEE Access*, vol. 11, pp. 16605-16620, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Linlin Xue, “DSAS: A Secure Data Sharing and Authorized Searchable Framework for e-Healthcare System,” *IEEE Access*, vol. 10, pp. 30779-30791, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Asep Saepulrohman, and Agus Ismangil, “Data Integrity and Security of Digital Signatures on Electronic Systems using the Digital Signature Algorithm (DSA),” *International Journal of Electronics and Communications System*, vol. 1, no. 1, pp. 11-15, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Osama Fouad Abdel Wahab et al., “Hiding Data Using Efficient Combination of RSA Cryptography and Compression Steganography Techniques,” *IEEE Access*, vol. 9, pp. 31805-31815, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Pahrul Irfan et al., “Application of the Blowfish Algorithm in Securing Patient Data in the Database,” *Matrix: Journal of Technology and Informatics Management*, vol. 12, no. 2, pp. 102-108, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Hailong Yao et al., “ECC-based Lightweight Authentication and Access Control Scheme for IoT E-Healthcare,” *Soft Computing*, vol. 26, no. 9, pp. 4441-4461, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Seval Capraz, and Adnan Ozsoy, “A Secure Medical Data Sharing Framework for Fight against Pandemics like COVID-19 by using Public Blockchain,” *IEEE Access*, vol. 12, pp. 39593-93605, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] G. Moheshkumar et al., *Security-Driven Data Analytics for Secure Patient Monitoring in Healthcare Application using Secure Hash Algorithm (256), Challenges in Information, Communication and Computing Technology*, CRC Press, pp. 167-172, 2025. [Google Scholar] [Publisher Link]

[18] Bilas Haldar, Partha Kumar Mukherjee, and Himadri Nath Saha, “An Approach of Modified IDEA with 1024 Bits Key to Enhance Security and Efficiency of Data Transmission in The Healthcare Sector,” *International Journal of Mathematical, Engineering and Management Sciences*, vol. 9, no. 6, pp. 1453-1482, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Abdulmohsen Almalawi et al., “Managing Security of Healthcare Data for a Modern Healthcare System,” *Sensors*, vol. 23, no. 7, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Parag Rastogi, Devendra Singh, and Sarabjeet Singh Bedi, “An Improved Blockchain Framework for ORAP Verification and Data Security in Healthcare,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 15, pp. 2853-2868, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] Vijaykumar Mamidala, “A Diffie–Hellman Key Exchange Algorithm: Improving Cloud Data Security,” *International Journal of Advanced Research in Information Technology and Management Science*, vol. 1, no. 1, pp. 88-99, 2024. [Google Scholar] [Publisher Link]

[22] M. Natarajan et al., “Quantum Secure Patient Login Credential System using Blockchain for Electronic Health Record Sharing Framework,” *Scientific Reports*, vol. 15, pp. 1-29, 2025. [CrossRef] [Google Scholar] [Publisher Link]