10.48047/jocaaa.2024.33.08.149

Blockchain-driven Encryption and Dual Authentication for Secure Healthcare Data Transmission in the Cloud

A. Hema Ambiha[1], R. Pragaladan[2]

[1,2]*Deparment of Computer Science, Sri Vasavi College, Erode, 636316, India*
E-mail: hemaambiha@gmail.com, https://orcid.org/0009-0000-7495-5107
E-mail: pragaladanr@gmail.com, https://orcid.org/0009-0007-9407-3591

Abstract: ***This paper focuses on enhancing the cloud security for health care data by storing the Hash value in a Hyper Ledger Blockchain. This research work is an extension of our previous work that implemented the Optimal Key-Tuned Rivest Shamir Adelman (OKTRSA), a double authentication technique, which effectively shared the data in cloud. A hybrid approach merged crossover and mutation approaches from Genetic Algorithms (GA) with Monarch Butterfly Optimization (MBO) to enhance key production in the RSA encryption paradigm. This technique optimized overall system performance and increased efficiency. This combination, called MBO-GA, helps to enhance performance. Together, these techniques enhanced the efficiency and effectiveness of generating keys, ultimately increasing the system's performance. In this research work, additional security is proposed to be implemented by storing the hash value in a blockchain, by providing a secure and tamper-proof means of data storage. This ensures the reliability of the information transferred and allows for trouble-free verification. Before the uploaded Electronic Health Record (EHR) files are saved on the cloud server, they are encrypted. This means the files are coded to protect sensitive information, ensuring that only authorized users can access them. The physician uses a hashed access policy in the secure download step and requests files from the cloud. This policy helps ensure that only authorized users can access the data. Once the request is approved, the physician c****an decrypt the files to access the information securely.*

Keywords:

Optimal Key-Tuned Rivest Shamir Adelman (OKTRSA), Secured Hashing Algorithm 512 (SHA512), Monarch Butterfly Optimization (MBO), Genetic Algorithm (GA), Blockchain.

1. INTRODUCTION

Cloud computing has revolutionized the healthcare sector by significantly enhancing operational efficiencies and improving clinical decision-making through the digitalization of healthcare information. It facilitates the sharing and effective storage of healthcare data among various stakeholders. The healthcare sector has experienced a significant transition from maintaining paper records to managing electronic records. To manage the vast amount of data efficiently, the cloud network must be robust enough to store and retrieve data online seamlessly. Cloud Storage(CS) proves invaluable in emergencies, offering timely access to patient history, medications, and allergies, which can be life-saving.

Building on the findings of the previous research, which explored the application of a dual-layer authentication mechanism called the Optimal Key-Tuned Rivest Shamir Adelman technique, that enhances secure data sharing in cloud-based healthcare sector, this article extends the investigation by proposing a blockchain technology based dual authentication for securing electronic healthcare data transmission. This work proposes a novel system that leverages blockchain-based dual authentication alongside an optimized encryption scheme to secure sensitive healthcare data. The system employs the OKTRSA encryption technique for secure data transmission, and it optimizes RSA key generation via a hybrid Monarch Butterfly Optimization–Genetic Algorithm (MBO-GA) approach. Additionally, a secure double hashing algorithm (SDHA512) generates a 512-bit hash (using a 1024-bit block length) for data integrity, with the hash stored immutably on a blockchain. Dual authentication on the receiver side ensures that only verified users can decrypt the data. Lastly, the receiver side performs dual authentication in order to read and download the files from the CS. The results indicated that the system outperforms other leading systems and can safely communicate and download healthcare data in cloud environments.

2. REVIEW OF LITERATURE

Nabeil Eltayieb et al. (2020)[1] proposed a secure cloud data sharing scheme that combines blockchain and attribute-based signcryption. The scheme addresses security concerns like confidentiality and unforgetability, and addresses issues with traditional cloud storage. The smart contract solves issues like erroneous results. Simulation findings and performance comparisons demonstrate the scheme's greater practicality and efficiency. It minimizes communication overhead and enforces user access control.

N. Gupta et al.[2] (2023) proposed a blockchain-based system to securely store medical records on a distributed network, utilizing RSA-Encryption for two-way authentication. This system will prevent tampering or selling patient data without consent.

Nilima et al.[3] (2021) developed a blockchain-based cryptographic method for secure data access and privacy in Electronic Health Records (EHR) programs. The system eliminates central authority and inherent failure, ensuring system protection. Performance evaluations include block size, build time, endorsement policies, and optimization of assessment methods for better results.

In Shahnaz et al.[4] (2019) a blockchain framework was presented for adoption in the healthcare space, which ensures secure storage of electronic records from a privacy perspective and successfully addresses the issues of scalability through off chain storage. It solves the problem of EHR systems though a scalable, secure and integral blockchain based solution.

Aravindakshan et al.[5] (2024) presented a dual authentication system by implementing a Optimal Key-Tuned Rivest Shamir Adelman technique for secure cloud data sharing in hospital data management.The system employs the Caesar cipher

10.48047/jocaaa.2024.33.08.149

and Secure Hashing Algorithm 512 for authentication, alongside the Improved Butterfly Optimization Algorithm for superior key generation. In terms of reliability metrics, encryption time, decryption time, and key breaking time, the system performs better than current techniques[5].

Hoang et al.[6] (2022) work intended to improve the security and privacy of exchanging EHRs using Blockchain technology, used decentralized databases, a permissioned blockchain network, and encryption with owner's attributes to ensure data privacy.

Arunkumar & Kousalya[7] (2020) developed a secure decentralized cloud-based medical block chain to address privacy and security concerns in patient health care data exchange, ensuring interoperability, traceability, and anonymity.

Wang et al [8] (2015) presented the Monarch Butterfly Optimization algorithm, which was motivated by monarch butterflies' migratory patterns.

The algorithm is designed to solve complex optimization problems by simulating the natural migration and genetic variation processes of monarch butterflies. The study demonstrates the effectiveness of MBO through various benchmark functions and comparisons with other optimization algorithms.Wang et al [9] (2019) proposed an enhanced version of the Monarch Butterfly Optimization algorithm, incorporating a global position updating operator to improve its performance on large-scale 0-1 knapsack problems. Their work introduced a dichotomy encoding scheme and a two-stage repair operator to handle the discrete nature and constraints of the knapsack problem effectively. According to the experimental findings, the enhanced MBO algorithm performs better than a number of traditional and cutting-edge algorithms when it comes to resolving large-scale 0-1 knapsack problems.

3. INTRODUCTION TO MONARCH BUTTERFLY OPTIMIZATION–GENETIC ALGORITHM (MBO-GA)

The key ideas of GA, MBO, and greedy strategy are briefly covered in this section.

3.1 GENETIC ALGORITHM

Genetic algorithms (GAs), a technique based on natural selection and evolution theory, were first presented by John Holland. GA starts with solutions, which are collections of chromosomes. The GA operators selection, cross-over, and mutation are then used to produce a new set of solutions. It is anticipated that the recently developed solutions will surpass the previous ones in terms of quality. These steps are repeated until the termination criteria is met.

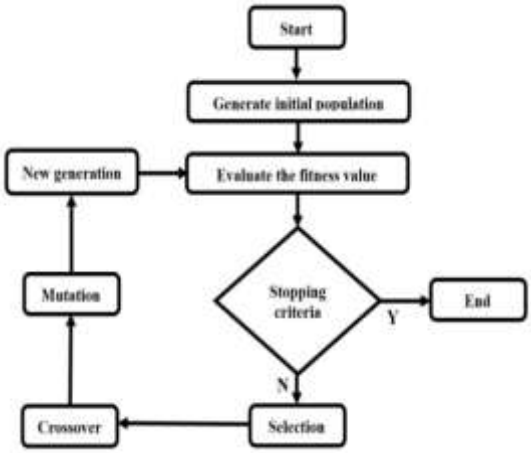

Fig. 1: The flowchart of the GA

3.2 MONARCH BUTTERFLY OPTIMIZATION

The MBO is a metaheuristic optimization methodology that was first introduced in G. G. Wang et al. (2019). MBO addresses optimization challenges by imitating the interactions of monarch butterflies in their natural habitat. Because of this, MBO has acquired a lot of attention and been successfully used to address a variety of optimization problems.

The following lists the main steps of MBO with more specificity:

3.2.1. Division of population

The population count of monarch butterflies in Land 1 (Subpopulation 1) and Land 2 (Subpopulation 2) can be categorized as $NP_1 = ceil(R * NP)$ and $NP_2 = NP - NP_1$, respectively. Population size is denoted by NP, the proportion of monarch butterflies in Land 1 is denoted by R. which is 5/12 according to the migratory period *peri*, and the value *ceil(x)* is rounded to the nearest integer greater than or equal to *x*.

3.2.2 Migration operator

Monarch butterflies migrate from Land 1 to Land 2 between April and September. The monarch butterflies remain in Land 1 for five months, from April to August, by optimizing the migration process. They reside on Land 2 for seven months, from September to March.

The solutions in Land 1 (Subpopulation 1) are updated using Migration operator in the following way:

Let r_1 be calculated as

$$r_1 = rand * peri \qquad (3.1)$$

where the *rand* denotes a random number that is chosen at random from a uniform distribution and the *peri* is the migration time.

When $r_1 \leq R$, $x_{r_2,k}^t$ is calculated using the following equation:

$$x_{i,k}^{t+1} = x_{r_2,k}^t \qquad (3.2)$$

where $x_{i,k}^{t+1}$ indicated the kth component of x_i butterfly in the generation $t+1$ which shows where the monarch butterfly i is

10.48047/jocaaa.2024.33.08.149

positioned, $x_{r_2,k}^t$ denotes the kth component in the generation t of the x_{r2} butterfly, selection of monarch butterfly r_2 at random from Subpopulation 1, and t is the current generation.

If $r_1 > R, x_{i,k}^{t+1}$ is determined by eqn (3.3)

$$x_{i,k}^{t+1} = x_{r_3,k}^t \quad (3.3)$$

where r_3 denotes the order of a monarch butterfly that is chosen at random from Subpopulation 2, and $x_{r_3,k}^t$ indicates the kth component in the generation t of the x_{r_3}butterfly.

The summarized expression for the migration operator is

$$x_{i,k}^{t+1} = \{x_{r_2,k}^t r_1 \leq R \; x_{r_3,k}^t r_1 > R \quad (3.4)$$

Algorithm 1 outlines the procedural phases of the migration operator:

Algorithm 1: Operator of migration

1. Start the migration process for all butterflies in Subpopulation .
- For each monarch butterfly in Subpopulation 1, the following steps will be performed:

2. Perform the migration for each butterfly:
- For each component (from 1 to D) of the butterfly, the following actions will occur:
  - Generate a random number using a uniform distribution.
  - If the generated random number meets the specified condition (e.g., less than 0.5):
- Select a butterfly randomly from Subpopulation 1.
- Use the chosen butterfly's data to generate the kth component of the current butterfly using the corresponding equation (e.g., equation 3.2).
- Otherwise:
  - Select a butterfly randomly from Subpopulation 2.
  - Use the chosen butterfly's data to generate the kth component of the current butterfly using a different equation (e.g., equation 3.3).

3. Repeat the above steps for each butterfly in Subpopulation 1.
4. End the migration process.

3.2.3 Adjusting operator

The solutions in Land 2 (Subpopulation 2) are updated using Butterfly adjusting operator in the following manner:

If $rand \leq R$;where rand indicated random number belonging to $[0, 1]$, $x_{j,k}^{t+1}$ is updated by equation (4.5):

$$x_{j,k}^{t+1} = x_{best,k}^t \quad (3.5)$$

where $x_{j,k}^{t+1}$ represents the kth component of x_j butterfly in the generation $t + 1$ that displays where the monarch butterfly j is positioned, and $x_{best,k}^t$ denotes the kth component of $xbest$ monarch butterfly; where the best monarch butterfly identified so far in the whole population (Land 1 and Land 2) is represented as $xbest$.

If $rand > R$, the value of $x_{j,k}^{t+1}$is updated as in equation (3.6)

$$x_{j,k}^{t+1} = x_{r_4,k}^t \quad (3.6)$$

where $x_{r_4,k}^t$ represents the kth element of the generation t of x_{r_4}butterfly, where r_4 is the order of a monarch butterfly which is randomly chosen from Subpopulation 2 (Land 2). If *rand* > *BAR*, the value of $x_{j,k}^{t+1}$ obtained from equation (3.6) is replaced by equation (3.7):

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \omega \times (dx_k - 0.5) \quad (3.7)$$

Here, Butterfly adjusting rate is referred to as BAR, and the weighting factor is indicated by ω as described in the equation (3.8):

$$\omega = S_{max}/t^2 \quad (3.8)$$

whereS_{max}stands for the maximum walk step a monarch butterfly can take in a single step, and when a monarch butterfly j walks, its step is dx, which is determined using Lévy flight as

$$dx = levy(x_j^t) \quad (3.9)$$

The butterfly adjusting operator steps are displayed in Algorithm 2

Algorithm 2: Adjusting operator

1. Start the adjusting process for all butterflies in Subpopulation 2.

For each monarch butterfly in Subpopulation 2, perform the following steps:

2. Determine the walk step and weighing factor:
  - Use equation (3.9) to calculate the walk step for the butterfly.
  - Use equation (3.8) to calculate the weighing factor for the butterfly.

3. Perform the adjustments for each component (from 1 to D) of the butterfly:
- For each component of the butterfly:
  - Generate a random number (rand) using a uniform distribution.
  - If the random number meets a specific condition (e.g., rand < threshold):
  - Use equation (3.5) to calculate the kth component of the butterfly.
- Otherwise:
  - Choose a butterfly randomly from Subpopulation 2.
  - Use equation (3.6) to calculate the kth component of the current butterfly.
  - If the random number (rand) is greater than a given threshold (e.g., rand > BAR):
  - Use equation (3.7) to calculate the kth component of the butterfly.

4. Repeat the above steps for each butterfly in Subpopulation 2.
5. End the adjusting process.

The MBO algorithm can be formed by idealizing the movement behavior of monarch butterfly individuals. Algorithm 3 shows the preliminary steps of MBO, and Fig. 2 illustrates the flowchart of the MBO algorithm. First, as in Algorithm 3, the MBO parameters are initialized. After that, the population is randomly generated and assessed utilizing the fitness function. Subsequently, each monarch butterfly's location is revised until certain conditions are satisfied. To fix the population and lower the

10.48047/jocaaa.2024.33.08.149

fitness evaluation, the number of monarch butterflies determined by the migratory operator and butterfly adjustment operator is set to NP_1 *and* NP_2, respectively.

Algorithm 3: The basic steps of MBO algorithm.

Step 1: Initialization
- Set the iteration count, t = 1.
- Randomly generate the initial population of monarch butterflies (denoted as NP).
- Calculate the necessary parameters: MaxGen, NP1, NP2, peri, BAR, SMax, and R.

Step 2: Evaluate the Fitness
- Assess the fitness of each monarch butterfly based on its position (location).
- Determine the optimal solution from the evaluation.

Step 3: Loop until optimal solution is found or the maximum generation is reached
- If the optimal solution has not been obtained or t is less than MaxGen:
  - Sort the monarch butterflies based on their fitness levels (fitness ranking).
  - Divide the population into two subpopulations (Land 1 and Land 2) based on their fitness.
- For each monarch butterfly in Land 1 (Subpopulation 1):
  - Apply the Migration Operator (Algorithm 1) to create a new Subpopulation 1.
  - Repeat for the specified number of iterations.
- For each monarch butterfly in Land 2 (Subpopulation 2):
  - Apply the Adjusting Operator (Algorithm 2) to create a new Subpopulation 2.
  - Repeat for the specified number of iterations.
- Combine the two newly created subpopulations (Subpopulation 1 and Subpopulation 2) into a single new population.
- Evaluate the newly formed population and update the best solution found so far.
- Increment the iteration counter: t = t + 1.

Step 4: End the loop
- The loop ends when the optimal solution is found or when the maximum number of generations (MaxGen) has been reached.

Step 5: Present the best solution
- The best solution found during the process is presented as the result.

3.3. GREEDY STRATEGY

The Greedy approach makes a number of decisions, each of which is the best one at the moment (the word "greedy" denotes selecting the best). The greedy approach makes judgments that are suitable for the moment without taking the future into account, and it offers the best options depending on the current circumstances. While a greedy heuristic may swiftly produce locally optimum solutions that are almost optimal globally, such strategies generally do not lead to an optimal solution. (Curtis, 2003).

3.4. HYBRID MBO WITH GA OPERATORS

This section presents the hybrid algorithm that combines GA operators with MBO. It is called MBO-GA. MBO-GA combines the greedy approach with the advantages of the two metaheuristic algorithms, GA and MBO. MBO has outstanding exploration capabilities since it considers every option and determines which is the best. The structure of the solutions can be altered by GA, and its operators offer significant exploitation potential.

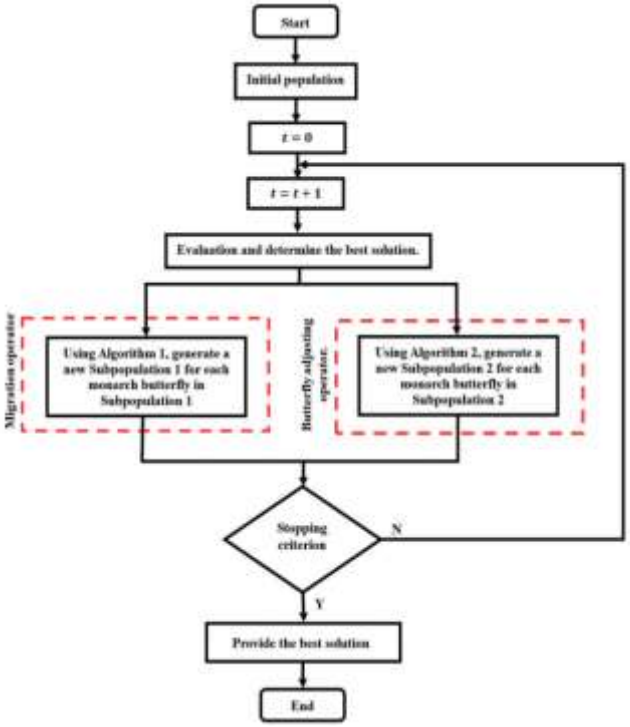

Fig. 2: Flowchart of MBO

As Genetic Algorithm operators (cross-over and mutation) totally change the solutions, the MBO-GA can exit the local optima when needed. To increase the effectiveness of the original MBO approach, three strategics are included in MBO-GA. A SAC operator is utilized to increase population variety (G. G. Wang et al., 2015). SAC is incorporated into both the migration operator and the butterfly adjusting operator. Second, the migration operator and the butterfly adjustment operator both take a greedy approach, only accepting monarch butterfly groups' offspring who are more fit than their parents. The greedy strategy is undoubtedly capable of greatly speeding up convergence (G. G. Wang et al., 2015). Third, to keep solutions from becoming too similar to one another, the mutation operator is applied to some of the monarch butterfly individuals. Thus, convergence to the global optimum may be hampered or even stopped by the mutation operator, which produces a variety of solutions and keeps one from attaining local minima. The suggested MBO-GA algorithm's flowchart is displayed in Fig. 3.

10.48047/jocaaa.2024.33.08.149

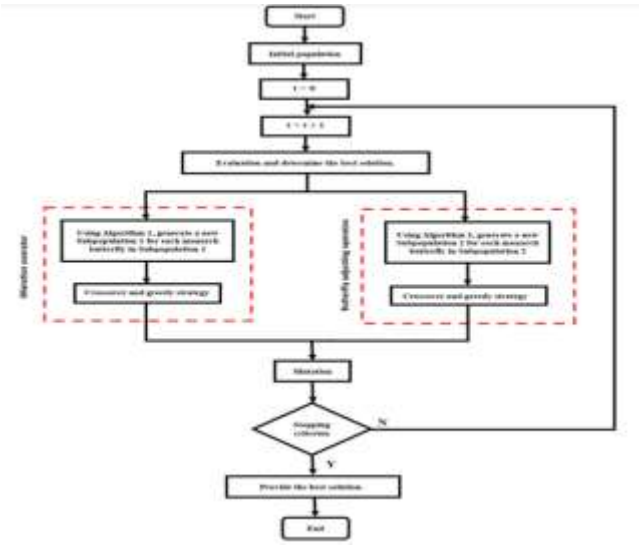

Fig. 3: The flowchart of MBO-GA algorithm

The suggested MBO-GA approach's framework is displayed in Algorithm 4

Algorithm 4: MBO-GAO.
Start
Step 1: Initialization.
· Set t=1 (initial generation).
· Randomly generate the initial population of monarch butterflies, denoted as NP.
· Calculate the parameters:
$MaxGen, NP_1, NP_2, peri, BAR, S_{Max}, and\ R.$
Step 2: Evaluate Fitness.
Based on its location, every monarch butterfly is evaluated and the best solution is determined.
Step 3: Population Division.
· Sort the monarch butterflies based on their fitness level.
· Divide the population into two subpopulations: Land 1 (NP_1) and Land 2 (NP_2).
While: The best solution cannot be discovered or *t* <*MaxGen* Do
Step 4: Updating the solution in Land 1.
- For each butterfly in Land 1 (from 1 to NP1):
  - Apply the Migration Operator (Step 4(1)): Perform migration to explore new regions.
  - Apply the SAC Operator (Step 4(2)): Apply a specific strategy (likely related to local search or adaptation).
  - Apply the Greedy Strategy (Step 4(3)): Select the best solution based on a greedy approach.

Step 5: Updating the solution in Land 2.
- For each butterfly in Land 2 (from 1 to NP2):
  - Apply the Butterfly Adjusting Operator (Step 5(1)): Adjust the solutions in Land 2 to improve them.
  - Apply the SAC Operator (Step 5(2)): Apply another adaptation strategy to enhance the solutions.
  - Apply the Greedy Strategy (Step 5(3)): Use a greedy selection strategy to update the solutions.

Step 6: Mutation Operator
- Apply a Mutation Operator to introduce random changes in the population to maintain diversity and avoid local optima.

Step 7: Evaluation.
- Evaluate the new population after all updates and mutation to assess its fitness.
- Update the best solution found so far.

Step 8: Present the Best Solution

Why Optimize RSA with MBO-GA?

To overcome the traditional RSA key generation issues like high computation time, security risk and inefficient encryption/decryption, the Monarch Butterfly Optimization–Genetic Algorithm (MBO-GA) is used to generate an optimal public key. This optimized RSA is referred to as OKTRSA (Optimized Key Generation in Traditional RSA).

4. PROPOSED METHODOLOGY

The proposed system integrates several advanced techniques to secure healthcare data. This work aims to:

- Develop a secure system for electronic healthcare data transmission using blockchain-based dual authentication.
- Enhance the encryption process by implementing the OKTRSA scheme and optimizing RSA key generation with MBO-GA.
- Ensure data integrity through the SDHA512 algorithm.

4.1. REGISTRATION

When the user establishes their authorization by registering with the cloud service provider (CSP), the implementation process begins. The patient provides the Trusted Center (TC) with their demographic information during this phase, which includes their name, user ID, gender, password, birthdate, current timestamp, address, and medical history. To ensure the confidentiality and authenticity of this sensitive information, the TC processes the input data using two security mechanisms:

First, the TC uses Caesar cipher to generate cipher value for user account data through an user ID(D_{PI}) - password(D_{PW}) combination. A substitution cipher termed the Caesar Cipher alters each letter in the plaintext to a new letter positioned a specified number of places down the alphabet.

It can be mathematically expressed as follows:

$$T^{\leftrightarrow}{}_{l} = \left((D_{PI} \parallel D_{PW}) + a\right) mod\ 26 \quad (4.1)$$

where, a is the shift variable and $T^{\leftrightarrow}{}_{l}$denotes the details of the cipher text provided by the respective user (i.e., user ID and password). The CS keeps records of ciphertext data which serve authentication functions.

The TC generates a hash value through the SHA512 algorithm by combining the user details including their Name (D_N), ID (D_{PI}), and (S_T) timestamp value. SHA512 generates message digest values of size 512 bits and block lengths of 1024 bit. SHA512 hash processing accepts any message input which produces a hash output of length of 512 bits. The hash code of SHA512 is displayed as follows:

$$H^{\leftrightarrow}{}_{c} = \hat{h}_f(D_N \parallel D_{PI} \parallel S_T) \quad (4.2)$$

10.48047/jocaaa.2024.33.08.149

where, $H^{\leftrightarrow}{}_c$ indicates the generated hash code and hash function is referred as $\hat{h}_f$. The hash values generated here is also stored in CS for authentication purpose. The patients must furnish the user ID and password for authentication when they log into the system during the login time. Then, the TC takes the cipher value using Caesar cipher and performs a match against the stored ciphered value. After successful comparison the user starts to upload the files in the cloud.

The generated hash code receives the designation HC within the expression and HF applies to the hash function. The system saves the produced hash values within CS to validate patient authentication. The system requires patients to use their user ID and password for authentication when logging into the system at the login point. When the TC handles the cipher value through Caesar cipher it performs a match against the stored ciphered value.

The SDHA512 algorithm applies the SHA512 hash function twice to the input $H^{\leftrightarrow}{}_c$ for enhanced security:

The First Hashing Stage is

$$H_1 = SHA512(H^{\leftrightarrow}{}_c) \quad (4.3)$$

This produces an intermediate 512-bit digest H_1.

Second Hashing Stage is

$$H = SHA512(H_1) \quad (4.4)$$

The final hash H is also 512 bits in length.

Thus, the complete process of SDHA512 can be expressed as:

$$H = SHA512\left(H^{\leftrightarrow}{}_c, S\right) = SHA512\ (SHA512\left(H^{\leftrightarrow}{}_c\right)) \quad (4.5)$$

This double hashing significantly increases resistance to collision and preimage attacks, ensuring that even if one layer of hashing were compromised, the overall security remains robust.

The final hash H is stored alongside the cipher textC in the secure database (or blockchain ledger) for future authentication purposes.

When a patient logs in, they must provide their user ID and password. The TC then performs the following steps:

1. The user's provided user ID and password are concatenated and processed using the same Caesar cipher with shift k to generate a new cipher text:
2. **Compare with Stored Cipher:** The recomputed cipher text is compared with the stored cipher text C. If they match, it indicates that the user credentials are correct.
3. **Hash Verification (Optional):** Additionally, the system may recompute the SDHA512 hash using the stored demographic details and compare it with the stored hash H to verify the integrity of the registration data.

If both checks are successful, the patient is authenticated and allowed to proceed with uploading or accessing files in the cloud.

4.1.1. Secure Data Transmission

The user can start uploading files to CSP after verifying. To ensure security, sensitive patient data must be encrypted before being moved to the cloud. Due to the frequent and sparse sharing of the collected data, a lightweight encryption strategy needs to be used. In order to increase cloud data security, we encrypt the patients' medical data using OKTRSA.

The predominant public key cryptosystem is RSA, with its crucial component being the generation of private and public keys. The RSA algorithm encrypts the medical data with a sender's public key and decrypts the encrypted data at the receiver end using a receiver's private key. In this key creation process, the public key is randomly selected from the two enormous prime numbers, and the private key is calculated using the public key. The randomly selected public key takes longer time to execute, which slows down the key creation process.

Furthermore, it compromises system security when the data is encrypted and decrypted. Therefore, the RSA key pairs are generated and optimized using a hybrid Monarch Butterfly Optimization–greedy search based Genetic Algorithm (MBO-GA) to maximize throughput and security. As a result, the optimal public key generation method combined with the conventional RSA is called OKTRSA. The three main stages are key generation, encryption, and decryption.

a) Key generation

The public and private keys are used to perform the encryption and decryption procedures. Only messages encrypted with the public key can be decrypted using the private key. The RSA algorithm begins its operation by choosing two separate large random prime numbers $\tilde{x}$ and $\tilde{y}$. Then, it calculates $\tilde{E} = \tilde{x}.\tilde{y}$, where, $\tilde{E}$ serves as the modulus for both the private and public keys.

Next, the Euler totient function (η^*) of $\tilde{E}$ is computed using equation (4.6).

$$\eta^*(\tilde{E}) = (\tilde{x} - 1) * (\tilde{y} - 1) \quad (4.6)$$

Then, the public key is randomly chosen between the ranges of 1 to $\eta^*(\tilde{E})$, but the randomly selected key used through this method increases computation time while producing infinite values when selected improperly. Hence, the current work proposes the use of Monarch Butterfly Optimization–Genetic Algorithm (MBO-GA) for selecting the appropriate public key.

The Monarch Butterfly Optimization (MBO) component of the algorithm mimics the migration behavior of butterflies, helping to refine potential key candidates. The Genetic Algorithm (GA) further enhances this selection through crossover, mutation, and a fitness function that ensures the chosen e value satisfies the greatest common divisor (GCD) condition $gcd\ gcd\left(e, \eta^*(\tilde{E})\right) = 1$. This ensures that e is coprime to $\eta^*(\tilde{E})$, which is necessary for RSA encryption to function correctly. Once an optimal public key is determined, the private key d is computed using the modular inverse formula, ensuring that the key pair is both secure and computationally efficient. Here e is $\tilde{P}_U$ and d is $\tilde{P}_R$.

b) Encryption

Next, encryption takes place following the generation of the keys for both encryption and decryption. Through this process the

10.48047/jocaaa.2024.33.08.149

information is converted into a coded form that can only be decoded by authenticated users to retain its actual meaning. The patient data ($I^{\leftrightarrow}_{PD}$) is encrypted using a public key ($\tilde{P}_U$) that has been generated at key generation process to generate the cipher. It is defined as follows:

$$C^{\leftrightarrow}_{T} = \left(I^{\leftrightarrow}_{PD}\right)^{\tilde{P}_U} mod\ \tilde{E} \quad (4.7)$$

Here, $I^{\leftrightarrow}_{PD}$ indicates the input data of the patient which are to be transmitted to the cloud and $C^{\leftrightarrow}_{T}$ represents the ciphertext of the corresponding patient input data. These encrypted ciphertexts are securely stored in the CS for subsequent processing.

c) Decryption

Decryption is the process of converting encrypted data back into its original form. In most of the cases, decryption is just the reverse process of encryption. The decryption process is mathematically expressed as follows:

$$I^{\leftrightarrow}_{PD} = \left(C^{\leftrightarrow}_{T}\right)^{\tilde{P}_U} mod\ \tilde{E} \quad (4.8)$$

Algorithm OptimalKeyGeneration_MBO_GA()

Start

Step 1: Intialize

1.1 Set t = 1

1.2 Generate a population NP of monarch butterflies randomly.

1.3 Set algorithm parameters: $MaxGen, NP_1, NP_2, peri, BAR, S_{Max}, and\ R.$

1.4 Define the key constraints for RSA key generation:

- Select two large prime numbers $\tilde{x}$ and $\tilde{y}$

-Compute modulus:$\tilde{E} = \tilde{x} \times \tilde{y}$.

-Compute Euler Totient Function:

$$\eta^*(\tilde{E}) = (\tilde{x} - 1) \times (\tilde{y} - 1).$$

- Define the valid range for the public key $\tilde{P}_U$:

$$1 < \tilde{P}_U < \eta^*(\tilde{E})$$

- $gcd\ gcd\ \left(\tilde{P}_U, \eta^*(\tilde{E})\right) = 1\ (ensures\ e\ and\ \eta^*(\tilde{E})\ are\ coprime)$

Step 2: Evaluate Fitness

2.1 Compute fitness for each monarch butterfly using:

$$Fitness = \frac{1}{gcd\ gcd\ \left(\tilde{P}_U, \eta^*(\tilde{E})\right) + 1}$$

2.2 Identify best (x_{best}) and worst (x_{worst}) solutions.

Step 3: Divide Population

3.1 Sort all monarch butterflies based on their fitness.

3.2 Divide population into:

- Land 1 (NP_1) → Migration Operator.

- Land 2 (NP_2) → Adjusting Operator.

Step 4: Updating Solutions in Land 1 (Migration Operator)

4.1 For i = 1 to NP_1 do

- Apply MigrationOperator()

- Apply SelfAdaptiveCrossover()

- Apply GreedyStrategy()

4.2 End for i

Step 5: Updating Solutions in Land 2 (Adjusting Operator)

5.1 For j = 1 to NP_2 do

- Apply ButterflyAdjustingOperator()

- Apply SelfAdaptiveCrossover()

- Apply GreedyStrategy()

5.2 End for j

Step 6: Mutation Operator

6.1 Apply MutationOperator() to 10% of the population.

Step 7: Evaluate Updated Population

7.1 Recompute fitness for all solutions.

7.2 Update the best solution.

Step 8: Termination Criteria

8.1 If termination conditions met (optimal key found or t = MaxGen), stop.

8.2 Otherwise, increment t = t + 1 and return to Step 4.

Step 9: Output the optimal public key $\tilde{P}_U$.

End.

In a cloud-based healthcare system, data security is a major concern due to the sensitivity of medical records. After encrypting data using the Optimized Key Generation in Traditional RSA (OKTRSA), which leverages Monarch Butterfly Optimization–Genetic Algorithm (MBO-GA) for optimized RSA key selection, the system still requires additional measures to ensure data integrity and secure access.

For this reason, Secure Double Hashing Algorithm (SDHA512) is integrated into the system to verify data integrity using blockchain storage. This ensures that medical records remain unchanged from their original state. Additionally, to prevent unauthorized access, dual authentication is implemented, requiring identity verification and key-based access control before users can decrypt, read, and download files. The combination of these mechanisms strengthens security, prevents unauthorized tampering, and ensures that only verified users can access the data.

4.1.2 Secure Data Integrity Using SDHA512 and Blockchain

Even with strong encryption, data integrity verification is essential to ensure data remains unchanged. SDHA512 (Secure Double Hashing Algorithm with SHA-512) is used for this purpose.

10.48047/jocaaa.2024.33.08.149

Step 1: Hash Generation for Encrypted Data

After encrypting medical data MMM using OKTRSA, the resulting ciphertext$C^{\leftrightarrow}{}_T$is hashed using SHA-512:

$$H = SHA - 512(C^{\leftrightarrow}{}_T) \quad (4.9)$$

This 512-bit hash value serves as a unique fingerprint of the encrypted file.

Step 2: Storing Hash on Blockchain

The generated hash H is recorded on a blockchain ledger:

$$Blockchain\ Store\ (H) \quad (4.10)$$

Since blockchains are immutable, once stored, no one can modify or delete this hash.

Step 3: Data Integrity Verification During Retrieval

When a user attempts to access or decrypt a file, the system checks its integrity:

1. The encrypted file $C_T^{\leftrightarrow}$is retrieved from cloud storage.
2. A new hash H' is computed using:

$$H' = SHA - 512\ (C^{\leftrightarrow}{}_T) \quad (4.11)$$

3. The retrieved hash H' is compared with the stored blockchain hash H:

$$H'? = \neq H \quad (4.12)$$

4. If $H' = H$,the data is unchanged and decryption proceeds. **If** $H' \neq H$**,** data has been tampered with, and access is denied.

Even after data integrity is verified, unauthorized users must not access or decrypt medical data. Therefore, dual authentication is implemented to restrict access to only verified users.

5. RESULT AND DISCUSSION

Python is used in the proposed task's implementation on a computer system with an i7 core processor, a 6 GB GPU, 16 GB of RAM, and Windows 10. The proposed work uses the publicly available and accessbile dataset of New York State Department of Health dataset for testing and verification the effectiveness of the proposed work through https://www.nyc.gov/site/doh/data/data-sets/data-sets-and-tables.page. Researchers, public health experts, members of the media, and community-based organizations can all benefit greatly from the data sets. The available dataset features 34 columns together with 2 Million records of data. The dataset is separated into two sections where 80 percent used for training purposes and 20 percent serves for testing purposes. The dataset contained a record range from 1000 to 10,000 records in its input area.

The results of the digital signature algorithm (DSA), advanced encryption standard (AES), and blowfish algorithm are also included in this section along with evaluations comparing the proposed OKTRSA system with RSA. The evaluation includes multiple indicators that measure (EXT) - the encryption time and (DCT) - the decryption time as well as (KBT) - key breaking time for reliability measures. These analyses are shown in the following table and figures.

5.1. ENCRYPTION TIME (EXT)

Encryption Time (EXT) is the duration required to convert plain (unencrypted) data into its encrypted (cipher) form using a given encryption algorithm. It is a critical performance metric that affects system throughput and responsiveness, especially when handling large datasets.

A simple representation of encryption time can be given as:

$$EXT = t_{finish} - t_{start} \quad (5.1)$$

- t_{start}**:** Time when the encryption process begins.

t_{finish}: Time when the encryption process completes.

5.2. DECRYPTION TIME (DCT)

Decryption Time (DCT) is the amount of time required to convert encrypted data back into its original, readable form. It is essential for timely data retrieval, especially in applications like healthcare where quick access to patient data is critical.A similar basic formula applies

$$DCT = t_{finish_{decryption}-} - t_{start\_decryption} \quad (5.2)$$

$t_{start\_decryption}$: Time when the decryption process begins.

$t_{finish\_decryption}$: Time when the decryption process completes.

5.3 KEY BREAKING TIME (KBT)

Key Breaking Time (KBT) refers to the estimated time required for an adversary to compromise or crack the encryption key through methods such as brute-force attacks. A higher KBT indicates a more secure encryption scheme. While KBT is not always expressed by a single formula in practical systems, it can be conceptualized as a function of key size and algorithm complexity:

$$KBT \propto 2^n \quad (5.3)$$

- n: Number of bits in the key.

Algorithm Complexity: More complex encryption algorithms further increase the difficulty of key breaking.

5.4 RELIABILITY METRICS

Reliability metrics assess the consistency, fault tolerance, and robustness of the encryption and decryption processes. In the context of secure data transmission, reliability indicates the system's ability to perform accurately and consistently under various conditions.

Reliability can be measured in several ways:

Success Rate:

$$Reliability\ (\%) = \left(\frac{Number\ Of\ Successful\ Operations}{Total\ Number\ of\ Operations}\right) \times 100 \quad (5.4)$$

(or)

10.48047/jocaaa.2024.33.08.149

MTBF (Mean Time between Failures) and MTTR (Mean Time To Repair):

$$Reliability = \frac{MTBF}{MTBF + MTTR}$$

These formulas help quantify how often the system operates successfully versus the frequency and duration of any failures.

Table 1 presents the outcomes of the current and proposed frameworks for EXT and DCT, varying the data record quantity from 1,000 to 10,000. EXT is the amount of time the encryption model takes to convert plain text into a ciphertext. It is calculated as the difference between the beginning and ending times of the encryption. DCT is calculated as the difference between the beginning and ending times of the decryption process. The process of record encryption and decryption with RSA, DSA, AES, blowfish requires 81.77s and 56.25s and 102.48s, 78.86s, 119.79s and 96.15s, 129.48s and 106.17s respectively. The suggested OKTRSA and blockchain with MBO-GA algorithms take 57.48s, 38.48s, and 33.85s, 13.25s, respectively, to encrypt and decode 1000 identical records. Similarly, the suggested strategy yields superior results for the remaining record range (2500-10000) compared to existing techniques. Therefore, it is determined that the suggested system is superior in speed and efficiency compared to alternative encryption approaches.

Table 1: EXT and DCT analysis

Table 2:Key Breaking Time (KBT) analysis

Metrics	No of Records	Proposed Blockchain with MBO-GA	OKTRSA	RSA	DSA	AES	Blowfish
EXT (s)	1000	38.48	57.48	81.77	102.48	119.79	129.48
	2500	35.31	55.61	79.49	101.77	117.84	127.51
	5000	38.53	58.52	82.64	103.47	120.77	130.51
	7500	40.37	60.57	84.82	105.81	122.54	133.57
	10000	47.57	67.77	91.59	112.57	129.92	138.54
DCT (s)	1000	13.25	33.85	56.25	78.86	96.15	106.17
	2500	11.97	31.97	53.86	80.92	97.95	104.86
	5000	15.12	35.02	57.25	83.32	107.99	115.98
	7500	16.16	36.86	58.86	96.17	117.88	123.89
	10000	23.22	43.92	69.22	108.86	129.33	144.02

No of Records	Key Breaking Time (KBT)					
	Proposed Blockchain with MBO-GA	OKTRSA	RSA	DSA	AES	Blowfish
1000	85	89	95	99	110	125
2500	79	81	88	94	104	114
5000	82	88	96	110	121	132
7500	84	90	97	106	115	123
10000	87	95	111	122	134	140

Table 3:Reliability Metrics

No of Records	Reliability Metrics					
	Blockchain with MBO-GA	OKTRSA	RSA	DSA	AES	Blowfish
1000	97	96	95	90	87	81
2500	95	94	91	88	81	77
5000	97	95	89	82	78	73

10.48047/jocaaa.2024.33.08.149

7500	96	94	90	84	79	71
10000	94	90	85	80	75	70

Table 2 presents findings regarding Key Breaking Time (KBT) evaluation results obtained through the proposed and existing frameworks across different data record levels from 1000 to 10000. As the number of records increases, the KBT values tend to increase for most methods. This suggests that when dealing with larger datasets, the computational effort (or time) required breaking the key becomes higher—implying enhanced security. It demonstrates that improved security in medical image transmission was attained by implementing MBO-GA algorithms within the blockchain framework.

Fig. 4 presents the results of the techniques considering the KBT for 1000-10000 records. It shows that the proposed OKTRSA and blockchain integrated with MBO-GA algorithms have improved security in medical image transfer with regard to KBT and reliability.

The proposed OKTRSA and blockchain with MBO-GA runs a maximum KBT of 10000 records while requiring 89ms and 85ms processing time making it perform 1.83%, 3.81%, 20.19%, and 36.98% more efficiently than RSA, DSA, AES, and blowfish. The proposed methods maintain KBT times lower than those of existing authentication methods. Next, reliability is essential to verify the system's operational efficiency and security level.

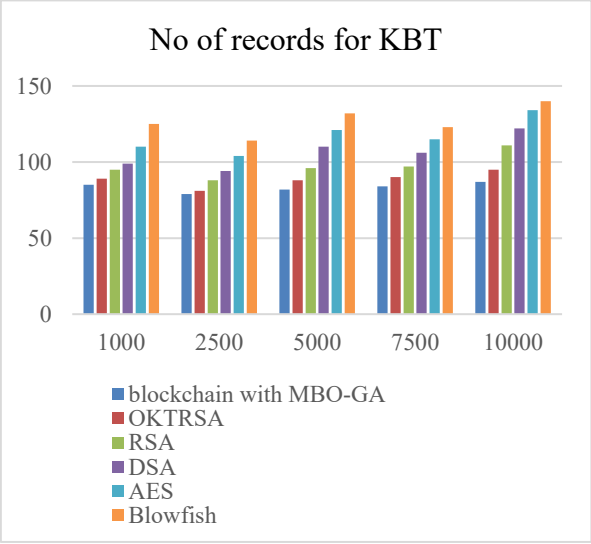

Fig. 4: Analysis based on KBT

Table 3 presents the results of the reliability of the techniques for 1000-10000 records. As the number of records increases (from 1,000 to 10,000), the reliability scores tend to decrease slightly across all methods.

This trend is common in systems where increased load can introduce more opportunities for errors or performance bottlenecks. This suggests that the integration of blockchain technology with the MBO-GA provides a more robust framework, ensuring a high level of success in encryption and decryption operations.

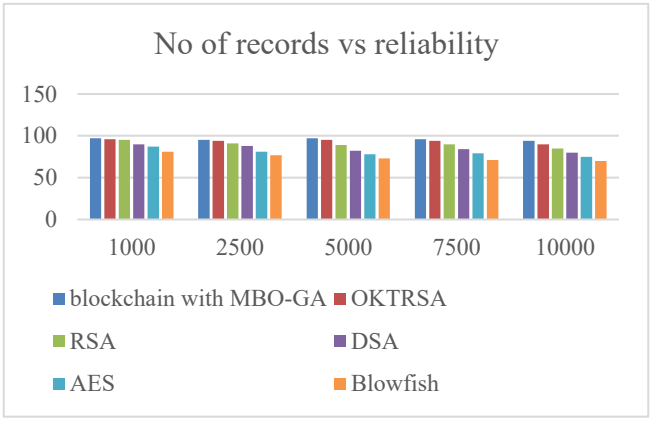

Fig. 5: Analysis based on reliability

Fig. 5 illustrates the results of the methods with respect to reliability for 1000-10000 records. Reliability is concerned with the data integrity, confidentiality, and accessibility of data within systems. The existing RSA, DSA, AES, and Blowfish algorithms exhibit average reliability rates of 95%, 90%, 85%, and 81%, respectively, for datasets ranging from 1,000 to 10,000 records. In contrast, the proposed OKTRSA and blockchain with MBO-GA algorithms demonstrate superior average reliability rates of 98%, 99%, and 96% for the same dataset sizes. So, the overall results indicate that the suggested blockchain with MBO-GA algorithm achieves improved performance compared to the previous techniques.

6. CONCLUSION

In this work, blockchain technology based dual authentication is proposed for securing electronic healthcare data transmission. The Secured Double Hashing technique 512 (SDHA512) technique, which guarantees integrity, and the "hash value," or message digest having 512-bit size and a 1024-bit block length, are the components of the authentication scheme. MBO-GA is exploited to maximize throughput and optimize the key generation. The outcomes of the proposed blockchain with MBO-GA is better than previous existing OKTRSA, RSA, DSA, AES, and Blowfish algorithms in terms of EXT, DCT, KBT, and reliability metrics.

REFERENCES

[1]. Eltayieb N, Elhabob R., Hassan A and Li F(2020)," A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud", Journal of Systems Architecture, 102, p.101653, Elsevier.

[2]. N. Gupta, J. Shah, V. Shah and S. Patil, "Secured Medical Record Sharing Application Using Blockchain

10.48047/jocaaa.2024.33.08.149

Technology," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10306929.

[3]. Nilima, M., Dr, V.P., Deshmukh, V.M., P.R.M.I.T.R, & P.R.M.I.T.R, B. (2021). Secure electronic healthcare (EHC) system using blockchain Technique.

[4]. Shahnaz, A., Qamar, U., & Khalid, A. (2019). Using Blockchain for Electronic Health Records. IEEE Access, 7, 147782-147795.

[5]. Aravindakshan H.A., Rengasamy P. Enhancing healthcare data security in cloud environments with dual authentication and optimal key-tuned encryption. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2024, vol. 24, no. 3, pp. 456–463. doi: 10.17586/2226-1494-2024-24-3-456-463.Mamun, A.A., Azam, S., & Gritti, C. (2022). Blockchain-Based Electronic Health Records Management: A Comprehensive Review and Future Research Direction. IEEE Access, 10, 5768-5789.

[6]. Mamun, A.A., Azam, S., & Gritti, C. (2022). Blockchain-Based Electronic Health Records Management: A Comprehensive Review and Future Research Direction. IEEE Access, 10, 5768-5789.

[7]. H. D. Hoang, P. T. Duy, N. T. Tien, D. T. Thu Hien and V. -H. Pham, "A Blockchain-based approach and Attribute-based Encryption for Healthcare Record Data Exchange," 2022 RIVF International Conference on Computing and Communication Technologies (RIVF), Ho Chi Minh City, Vietnam, 2022, pp. 65-70, doi: 10.1109/RIVF55975.2022.10013886.

[8]. Wang, G.G., Deb, S., & Cui, Z. (2015). Monarch Butterfly Optimization. Neural Computing and Applications, 31(7), 1995–2014.

[9]. .Feng, Y., Yu, X., & Wang, G.G. (2019). A Novel Monarch Butterfly Optimization with Global Position Updating Operator for Large-Scale 0-1 Knapsack Problems. Mathematics, 7(11), 1056.

[10]. Vidhya, S., Kalaivani, V. A blockchain based secure and privacy aware medical data sharing using smart contract and encryption scheme. Peer-to-Peer Netw. Appl. 16, 900–913 (2023). https://doi.org/10.1007/s12083-023-01449 -1

[11]. Afzal, I., Parah, S.A., Hurrah, N.N. et al. Secure patient data transmission on resource constrained platform. Multimed Tools Appl ,83, 15001–15026 (2024). https://doi.org/10.1007/s11042-020-09139-3.

[12]. Ali, M., Dhamotharan, R., Khan, E., Khan, S.U., Vasilakos, A.V., Li, K., & Zomaya, A.Y. (2017). SeDaSC: Secure Data Sharing in Clouds. IEEE Systems Journal, 11, 395- 404.

[13]. Arunkumar, B., & Kousalya, G. (2020). Blockchain-Based Decentralized and Secure Lightweight E-Health System for Electronic Health Records. Intelligent Systems, Technologies and Applications.

[14]. Gharat, A., Aher, P., Chaudhari, P., & Alte, B. (2021). A Framework for Secure Storage and Sharing of Electronic Health Records using Blockchain Technology. ITM Web of Conferences.

[15]. Sri, V., Hema, V., & Kesavan, R. (2019). ECC Based Secure Sharing of Healthcare Data in the Health Cloud Environment. Wireless Personal Communications, 108, 1021 - 1035.

[16]. Ravi, D., Ramachandran, S., Vignesh, R., Falmari, V. R. & Brindha, M. Privacy preserving transparent supply chain management through hyperledger fabric. Blockchain Res. Appl. 3(2), 100072 (2022).

[17]. N. Andola, Raghav, S. Prakash, S. Venkatesan and S. Verma, "SHEMB:A secure approach for healthcare management system using blockchain," 2019 IEEE Conference on Information and Communication Technology, Allahabad, India, 2019, pp. 1-6, doi: 10.1109/ CICT48419. 2019. 9066237.

[18]. P. William, Y. N., S. Vimala, P. Gite and S. K. S, "Blockchain Technology for Data Privacy using Contract Mechanism for 5G Networks," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 461-465, doi: 10.1109/ICIEM54221.2022.9853118.

[19]. R. T.A.V.Y, H. H.N.H, H. H.K.D.W.M.C.B., P. K.L.K.T, A. Senarathne and L. Ruggahakotuwa, "Ensuring Electronic Health Record (EHR) Privacy using Zero Knowledge Proofs (ZKP) and Secure Encryption Schemes on Blockchain," 2023 5th International Conference on Advancements in Computing (ICAC), Colombo, Sri Lanka, 2023, pp. 792-797, doi: 10.1109/ICAC60630.2023.10417417.

[20]. Zhang, L., Kan, H., & Huang, H. (2022). Patient-centered cross-enterprise document sharing and dynamic consent framework using consortium blockchain and ciphertext-policy attribute-based encryption. Proceedings of the 19th ACM International Conference on Computing Frontiers.

[21]. Hoang, H.D., Duy, P.T., Tien, N.T., Hien, D.T., & Pham, V. (2022). A Blockchain-based approach and Attribute-based Encryption for Healthcare Record Data Exchange. 2022 RIVF International Conference on Computing and Communication Technologies (RIVF), 65-70.

[22]. Basarkod, P.I. (2022). A Novel Approach to Create a Secure Robust Blockchain Architecture using Lightweight Encryption Model for e-Healthcare Records.

10.48047/jocaaa.2024.33.08.149

[23]. Naveed, N., Sultan, A., Khan, F., & Tahir, S. (2023). Efficient, Immutable and Privacy Preserving E-Healthcare Systems using Blockchain. 2023 International Conference on Communication Technologies (ComTech), 140-145.

[24]. Gupta, N., Shah, J., Shah, V., & Patil, S. (2023). Secured Medical Record Sharing Application Using Blockchain Technology. 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 1-6.

[25]. Chen, J., Yin, X., & Ning, J. (2022). A fine-grained and secure health data sharing scheme based on blockchain. Transactions on Emerging Telecommunications Technologies, 33.

[26]. Psarra E, Verginadis Y, Patiniotakis I, et al. A Context-aware Security Model for a Combination of Attribute-based Access Control and Attribute-based Encryption in the Healthcare Domain. Caserta, Italy: Springer Nature; 2020: 1133-1142

[27]. Fan K, Wang S, Ren Y, Li H, Yang Y. MedBlock: efficient and secure medical data sharing via blockchain. J Med Syst. 2018; 42(8): 136-148.

[28]. Shen B, Guo J, Yang Y. MedChain: efficient healthcare data sharing via blockchain. Appl Sci. 2019; 9(6): 1207-1216