A New Encryption Scheme Using Blockchain for Secured Accessing of Sensitive Health Care Records

Sonya Ansar [1], Prakash Natarajan [1], Liliana Guran [2,3], and Rozalia Veronica Rus [2,*]

[1] Department of Information Technology, B. S. Abdur Rahman Crescent Institute of Science and Technology, Vandalur, India
[2] Department of Hospitality Services, Babeș-Bolyai University, 7 Horia Street, Cluj-Napoca, Romania
[3] Department of Computers, Technical University of Cluj-Napoca, 28 G. Barițiu Street, Cluj-Napoca, Romania
Email: sonya@crescent.education(S.A.); prakash@crescent.education(P.N.); liliana.guran@ubbcluj.ro(L.G.); veronica.rus@ubbcluj.ro (R.V.R.)
*Corresponding author

***Abstract*—Safe storage and retrieval of private patient information has become crucial as healthcare facilities rapidly move to digital. Electronic Health Records (EHRs) have transformed healthcare by streamlining management and enabling seamless sharing among stakeholders, including doctors, laboratory technicians, and insurance agents. However, this digital shift has also exposed EHRs to cyber threats, such as hacking and data breaches. Once compromised, the integrity of the health records is permanently affected. Therefore, it is crucial that patients have complete control over who can access their EHRs, when they can access them, and under what circumstances. This is why it is necessary to design a cloud-based system that helps to ensure authentication and keeps the health records integrity. We propose a novel methodology called Effective Key-length Tri iterative Elliptic-Integrated Data Encryption Standard (EKT-EDES) combined with a blockchain-driven safety mechanism to protect healthcare records in the cloud. This includes encrypting healthcare insurance records before saving them on the Interplanetary File System (IPFS). Enhanced Role-Based Access Control (RBAC) is implemented via Smart Contracts (SC) on the blockchain, which also stores an encoded keyword index for a safe Electronic Medical Records (EMRs) search. Our performance analysis shows key generation time (3 ms), encryption (1628 ms), decryption (1087 ms), execution (3295 ms), latency (107 ms), and throughput (120 ms). We conclude by comparing SC's total price and throughput, demonstrating the superiority of our method.**

***Keywords*—healthcare, blockchain, Electronic Medical Records (EMR), Smart Contract (SC), access control, Effective Key-length Tri-iterative Elliptic-integrated Data Encryption Standard (EKT-EDES)**

I. Introduction

The healthcare sector faced a significant transition in the rapidly developing digital age, as Electronic Health Records (EHRs) are increasingly used to store and handle sensitive patient data. Although the change has many advantages, such as increased efficiency and accessibility, it also creates serious problems with data privacy and safety. Protecting private medical records from hackers and other outside parties has become a top priority for hospitals and other healthcare organizations [1]. Integrating a unique encryption method and blockchain-based security for the safekeeping and retrieval of private healthcare records was one of the innovative approaches explored in response to the Cipher-Text Policy-Attribute Based Encryption (CP-ABE) problem. In addition to improving overall data management efficiency, the ground-breaking project seeks to meet the urgent need for robust safety precautions in the healthcare industry [2]. The unique nature of healthcare data makes such an advanced security system necessary for gathering confidential and sensitive data, such as medical records, diagnosis, course of treatment, and more can be found in healthcare records. Safeguarding the data was a moral and legal requirement and a question of patient confidentiality and trust [3].

Blockchain technology offers several critical advantages for managing healthcare records. First, its robust encryption ensures enhanced security, protecting sensitive patient data from unauthorized access, and making it difficult for hackers to tamper with or steal information. Blockchain's immutable nature guarantees data integrity by preventing egrity by preventing any alterations once the records are added, thus preserving the accuracy of healthcare data. Decentralization further strengthens the system by distributing data across multiple nodes, reducing reliance on a single point of failure and ensuring availability. Privacy and anonymity are safeguarded through encryption methods that allow only authorized individuals to access specific data based on their roles [4]. Furthermore, blockchain enables seamless interoperability between healthcare systems, facilitating secure exchange of patient information and improving treatment coordination. Technology also supports

Manuscript received November 6, 2024; revised December 12, 2024; accepted December 11, 2024; published April 16, 2025.

auditability by providing a traceable and time-stamped history of medical records, ensuring transparency [5]. Lastly, smart contracts automate role-based access control, enabling secure, permissioned access to records, and adding a layer of governance over the data management process. Together, these features make the blockchain a powerful solution for handling healthcare records. A smart contract for healthcare records uses blockchain to securely manage Electronic Health Records (EHRs) with privacy and transparency. The contract also allows patients to preauthorize the sharing of data with specific entities, such as laboratories or insurers, for defined periods. Procedures for access control will be essential in determining who has access to and can alter medical records [6, 7]. The guidelines will be customized to satisfy the unique requirements and laws of healthcare facilities, guaranteeing that only authorized staff members, such as physicians, nurses, and administrators, can access patient data. Further boosting data security was the fine-grained control offered by access control policies, which guarantee that people only have access to the data required for their tasks [8]. The study's objective was to create and implement the novel health data security, (Effective Key-length Tri-iterative Elliptic-integrated Data Encryption Standard) EKT-EDES. The purpose of using blockchain for healthcare records is to improve security, ensure data integrity, and enable seamless data exchange while protecting privacy. The encryption, immutability, decentralization and smart contracts of the blockchain provide secure, traceable, and role-based access to healthcare data, improving transparency and coordination across healthcare systems [9].

The main goal of our paper is to propose a novel methodology called Effective Key-length Tri iterative Elliptic-Integrated Data Encryption Standard (EKT-EDES) combined with a blockchain-driven safety mechanism to protect healthcare records in the cloud. In this methodology, the healthcare insurance records will be encrypted before saving them on the Interplanetary File System (IPFS). Enhanced Role-Based Access Control (RBAC) will be implemented via Smart Contracts (SC) on the blockchain, which also stores an encoded keyword index for a safe Electronic Medical Records (EMRs) search. Also, we will conduct a performance analysis of the proposed model to demonstrate the superiority of our method.

Different aspects of our research are covered in the following sections of the paper. An extensive selection of relevant works on the topic was reviewed in Section II. We explain the structure of our suggested approach in Section III. After comparing our strategy with other systems, Section IV provides the results and leads to a complete discussion. Ultimately, we make inferences from our data in Section V.

II. Literature Review

Chen *et al.* [10] recommended blockchain technology to implement patient-centric access control for medical applications, including EMR. According to their strategy, patients only allowed their trusted doctor access to their encrypted EMRs that would be maintained on a blockchain. Through simulations, they examined the adaptability of their method. Its outcomes demonstrated good scalability, with a linear rise in the stored chain quantities when the number of nodes in the network increased. Amanat *et al.* [11] presented a Post - Quantum Public-key Searchable Encryption on Blockchain (PPSEB) for digital healthcare records. They used lattice-based encryption to encrypt search results to avoid the unauthorized release of sensitive medical information. Blockchain technology solved the problem of not being able to trust search engines. In eHealthcare situations, the accuracy and forward security were verified by a security analysis. The strategy was shown to be efficient compared to other options through a thorough effectiveness assessment [12].

Chen *et al.* [13] developed a blockchain and cloud-based system to manage individual patients' medical records. Additionally, a service framework for the exchange of health records was described. Further, they compared the properties of the medical blockchain with those of more conventional systems. Gong and Zhao [14] proposed a blockchain-based architecture for securing Electronic Health Record sharing among several EHRs, with user identities confirmed with a Proof of Stake (POS) cryptographic consensus method and Secure Hash Algorithm (SHA256). The EHR sensors were validated using an Elliptic Curve Digital Signature Algorithm (ECDSA) prior to data collection and transmission to the cloud. According to the findings, the proposed solution performed effectively compared to other possibilities. Ali *et al*. [15], Chandrasekaran *et al*. [16] established an encrypted, distributed, Cloud-based Medical Blockchain (CMBC) to address concerns about the safety of patient data during its transfer between healthcare providers. Data from EHRs were encrypted using the lightweight authentication encryption method AES_256_GCM before being uploaded to the blockchain in the cloud. It was done to improve healthcare efficiency. Ethereum's Solidity smart code was used to limit the utilization of electronic health records in the cloud using the encrypted information on the blockchain [17, 18]. Górski developed a private blockchain-based searchable data encryption system for EHRs and a decentralized cloud-based blockchain for medical purposes [19].

III. Methodology

This section discusses the encryption scheme and blockchain-based security for storing and retrieving sensitive healthcare records using access control policies. To protect patient confidentiality and ensure information security, the highest level of precautions must be taken to secure the storage and retrieval of private medical records.

The working principle of the system model is shown in Fig. 1. Each phase of this method is described below:

- Step 1: Interaction with the doctor

To ensure security during healthcare interactions, encrypted communication channels are used for remote consultations, while private rooms protect in-person interactions. Role-based access control ensures only

authorized professionals access patient data, with encryption safeguarding it in transit and at rest. Authentication and authorization methods secure access to records, and compliance with regulations like Health Insurance Portability and Accountability Act (HIPAA) and General Data Protection Regulation (GDPR) ensures data privacy. These measures maintain the confidentiality and integrity of sensitive patient information.

- Step 2: Smart contract deployment

It handles data processing and interaction logic securely and autonomously. Connects multiple entities, including the patient, doctor, and the blockchain network.

- Step 3: Encrypt medical record

Data are secured using powerful encryption methods before being uploaded with any medical information. Using encryption, it is made sure that the data is safe and unreadable even in case of illegal access.

- Step 4: Uploads to IPFS

The decentralized IPFS or some comparable system is then used to store the encrypted medical records. Data may be stored in a distributed and reliable manner using IPFS.

- Step 5: Location retrieval

The blockchain stores the IPFS hash or address of where the encrypted health information is found on the IPFS network. As a result, the data may be found and retrieved at any time by authorized parties.

- Step 6: Encryption and transaction broadcast

A requester for data encrypts and broadcasts their request to the blockchain system whenever they make a request to obtain a medical record. The smart contract receives this request in a secure manner.

- Step 7: Confirmation: Address recording

The smart contract accepts the request, confirms the requester's legitimacy and identity, and logs the address of the requester for future use.

- Step 8: Encryption, indexing, storage

From the blockchain, the smart contract finds the precise location of the required medical record. The data are then decrypted, downloaded from IPFS, and indexed for quick retrieval.

- Step 9: Authorization

The smart contract verifies that the person making the request has the proper authorization to view the requested medical record. Validates the identity and authorization status of the requester.

- Step 10: Key Generation for Data Requester

A unique decryption key is generated for the requester by the smart contract if they are allowed. This key is sent to the requester in encrypted form.

- Step 11: Requester invokes with search token

With a search token and the decryption key, the requester invokes the smart contract and specifies the precise record they wish to view.

- Step 12: Authorization results

The search token is examined by the smart contract to ensure that it corresponds to the permitted request that was made earlier. If everything checks out, access is granted.

- Step 13: Transaction information calculation

For auditing and accountability, transaction details, such as access time stamps and requestor data, are stored on the blockchain.

- Step 14: Encrypted medical data, decryption

Using the produced key to decrypt the requested medical record, the authorized requester can access it. The original record's confidentiality and integrity are kept while the decrypted medical information is sent to the authorized user.

The danger of unauthorized access and data theft is reduced to these measures, which provide a safe and regulated method for keeping and retrieving critical healthcare records. Privacy and security of data in the healthcare industry are improved by using blockchain, encryption, and distributed storage technologies.

The process flow and interaction in a blockchain-based EHR system is illustrated in Fig. 2.

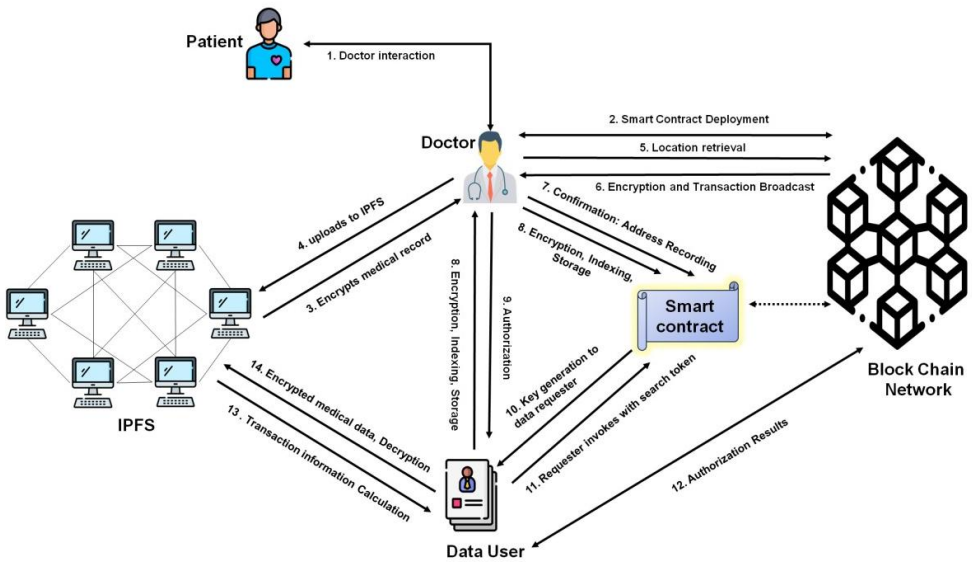

Fig. 1. Working principle of a system model.

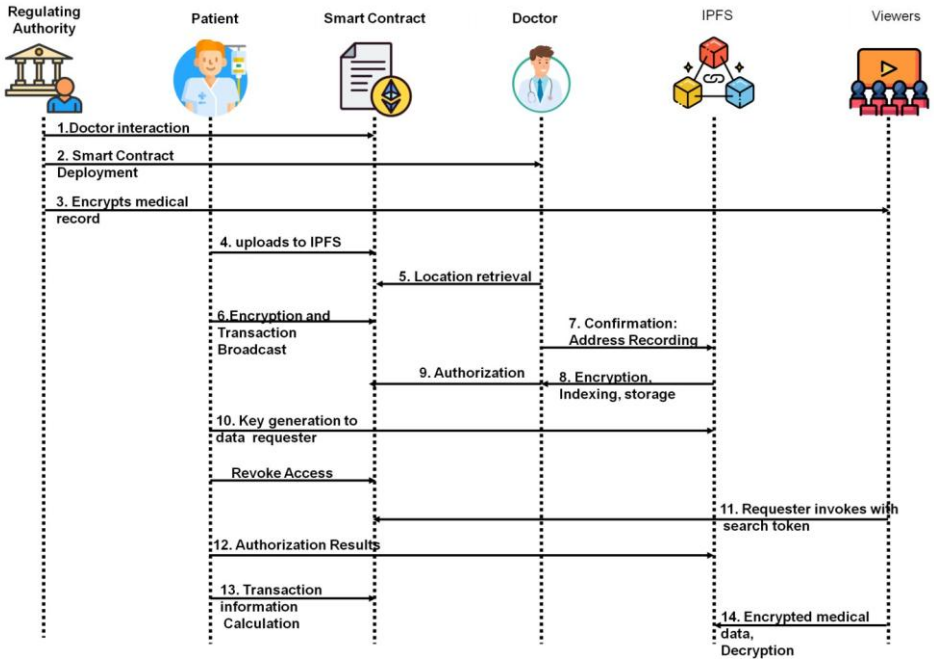

Fig. 2. Process flow and interaction in a blockchain-based EHR system.

A. *Effective Key-length Tri-iterative Elliptic-integrated Data Encryption Standard (EKT-EDES)*

Cloud computing services can be obtained in the form of EKT, which encrypts and decrypts data and is used to provide efficient sorts of cloud services. Elliptic Curve Cryptography is an advanced and highly efficient cryptographic method for protecting data during transmission and cloud storage. Because it can offer robust security with comparatively modest key sizes, making it proper for resource-constrained contexts, it is beneficial for healthcare records and other sensitive data. Elliptic curve-based mathematical techniques are used in Elliptic Curve Cryptography (ECC) to store and retrieve sensitive medical records. These algorithms are used for key exchange, digital signatures, and encryption, among other security features. Fig. 3 depicts the user and data kept on a cloud server.

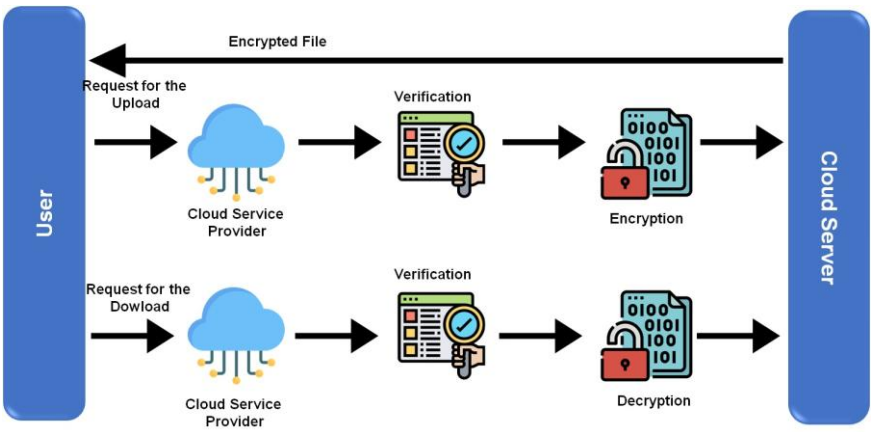

Fig. 3. User and data kept on a cloud server.

For safe cloud storage of healthcare record cloud storage, vital encryption keys must be generated for Elliptic Curve Cryptography (ECC). To ensure robust data protection, this method creates encryption keys and private and public ECC keys.

$$ECC\ Key\ Pair = (Private\ Key, Public\ Key) \quad (1)$$

Sensitive medical records are encrypted using the ECC approach, the generated keys. Decryption is carried out in reverse order to maintain data confidentiality and integrity during retrieval, first utilizing ECC.

$$Encrypted\ Data = (ECC)_{Encrypt}(Healthcare\ Record, Recipient's\ Public\ Key) \quad (2)$$

$$Decrypted\ Data = (ECC)_{Decrypt}(Encrypted\ Data, Recipient's\ Private\ Key) \quad (3)$$

A "Private Key" must be kept secret to decrypt the data. The "Public Key" is shared with the public to encrypt the data. The following Eqs. (4) and (5) can be used to

comprehend how ECC is used to secure the cloud storage and retrieval of medical records.

The set of points *(a, b)* over an elliptic curve E over the finite space *GS(p).*

$$b^2=x^3+xa+y \quad (4)$$

where *x, y* are integers *modulo p* that meet the following equation and p are referred to as a prime integer modular.

$$4x^3+27y^2\neq 0 \bmod p \quad (5)$$

Thus, in understanding of elliptic curves over the finite field, *EGS(p)* is composed of the set of values *(a, b)*, in the point format of affine coordinates, there is an extra element representing the location of *E* at indefinitely, represented by *G*.

The equation of the elliptic curve equation $b2 = x^3 + xa + y$ is often used to describe a fundamental domain elliptic curve in cryptography. There are six basic parameters in this formula: *x, y, p, n, G, h*, which is the order of the fundamental point of E; All the points on an elliptic curve are multiplied by their order, or *h, n*, and *x, y, p* determine the elliptic curve.

In theory, a cryptosystem with a greater parameter p will be more secure but will have a slower processing time. This article describes the basic elliptic curve operations of point multiplication and value adding, which are frequently used in both encryption and decryption processes. The best geometric explanation of the addition rule may be found in Fig. 4, workflow from public key encryption to private key decryption.

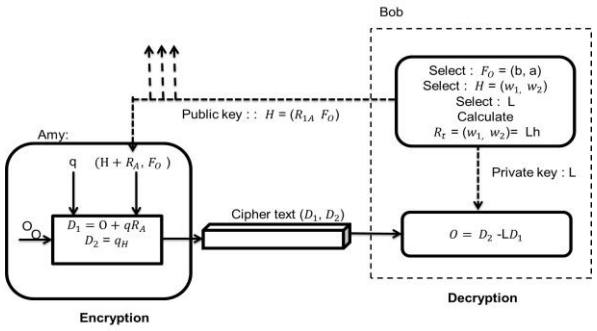

Fig. 4. Workflow from public key encryption to private key decryption.

Let $b^2 = x^3 + xa + y$ create an elliptic curve *E*, and let $P = (a_1, b_1)$ and $Q = (a_2, b_2)$ be two unique points on the curve. The following geometric method is used to implement the sum of *P* and *Q*, which is represented by $R = (a_3, b_3)$. When the line is drawn through *P* and *Q* first it crosses the elliptic curve at a third point, *R*. Reflection *R* across the *x* axis confirms the final point, $P + Q = (a_3, b_3)$. Explain a_3 and b_3 using Eqs. (6)–(9).

$$a_3 = (\lambda^2 - a_1 - b_2) \bmod p \quad (6)$$

$$b_3 = \lambda(a_1 - b_2)\, y_1 \bmod p \quad (7)$$

where,

$$\lambda = \frac{b_2-b_1}{a_2-a_1} (if\ P \neq Q) \quad (8)$$

$$\lambda = \frac{3a_1^2+x}{2b_1} (if\ P = Q) \quad (9)$$

Adding *P* to itself *k* times is known as scalar point multiplication, given an integer *k* and a point *P* over the elliptical curve. The addition of the ellipse curve addition can be repeated k times using the same guidelines and a generator point to achieve the scalar multiplication kp. Scalar point multiplication, which establishes an elliptic curve cryptosystem's speed and serves as the foundation for all elliptic curve arithmetic, is the predominant cost operation in ECC.

B. Tri-iterative Elliptic-integrated Data Encryption Standard

Block cipher encryption techniques like EDES are well known for their ability to securely encode data using just one encryption key and one decryption key. However, some performance operations, such as statistical analysis and findings within cloud storage systems—may present challenges when it comes to cloud storage. The EDES technique is chosen for this research article because it is simple to use and works well with the time-sensitive requirements of retrieving data stored in environments in the cloud.

3EDES is designed to overcome the problems in DES without creating an entirely novel cryptosystem. DES must be more sufficient to protect confidential information since it employs a 56-bit key. By constantly performing the DES algorithm with three separate keys, 3-EDES effectively increases the critical size by a factor of three. Therefore, the total number of data in the key is 168 bits (3×56).

The TDEA utilizes three 64-bit DEA keys, denoted as H1, H2, and H3, in an Encrypt-Decrypt-Encrypt (EDE) mode. In this manner, the original plaintext undergoes H1, followed by decryption using H2, and finally encryption again using H3. Fig. 5 demonstrates the Process of iterative EDES. The 3EDES algorithm possesses an efficient key length of 168 bits and is formally described Eq. (10) as follows.

$$D = F(H3,F(H2,E(H1,O))) \quad (10)$$

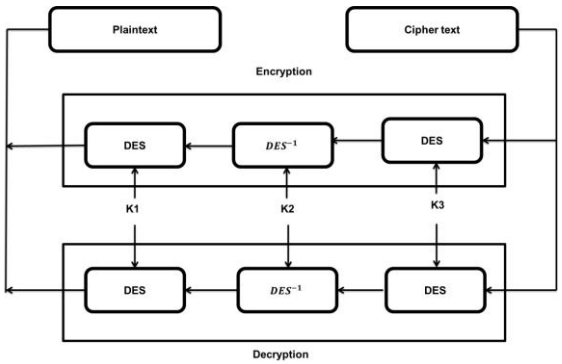

Fig. 5. Process of iterative EDES.

The norms specify three different keying schemes: the recommended approach utilizes three distinct and unrelated keys (H1 ≠, H2 ≠, H3 ≠, H1). The critical space is determined by multiplying three by 56, resulting in 168 bits.

The second possibility utilizes a system consisting of two distinct keys that are not equal to each other (H1 ≠ H2)

and a third key that is identical to the first key (H3 = H1). The key space is determined by multiplying two by 56, resulting in 112 bits.

The third possibility involves the utilization of a critical bundle consisting of three identical keys, denoted as H1, H2, and H3. This alternative is comparable to the DES algorithm. The 3-EDES method uses three iterations to improve average processing time and encryption efficacy. The fact that the 3EDES block cipher technology performs worse than other methods is well known.

Approaches for 3EDES Encryption and Decryption:

- *Encryption*
  voidDES_Encrpt (N,H1)
  voidDES_Encrpt (N,H2);
  voidDES_Encrpt(N,H3)
- *Decryption*
  voidDES_Encrpt (N,H1)
  voidDES_Encrpt (N,H2);
  voidDES_Encrpt(N,H3)

C. EKT-EDES

A hybrid encryption method that combines the concept of effective key length with the robust security of the tri-iterative Elliptic-integrated Data Encryption Standard (EKT-EDES) is used to increase the security of cloud storing and retrieving sensitive medical information. This technique combines a tri-iterative Data Encryption Standard (3DES) algorithm with Elliptic Curve Cryptography (ECC) to leverage the cryptographic power of EKT-EDES. The suggested algorithm combines the best features of EDES with EKT to produce a sophisticated and effective cloud storage cryptography method. The hybrid approach, EKT-EDES, has the benefit of a smaller key size and improved security for data protection, but pure EDES can be a little slower because of its larger key size. The primary characteristic of EKT is its tiny key size, which improves performance when used with EDES for encryption. EKT is a perfect addition to EDES for data security since it uses both decryption and encryption key standards to produce a small and safe key system. For information encryption and decoding, ciphertext is created when the key size is established. EDES then uses the key produced by EKT, combining the two to create a powerful combination that is ideal for protecting data stored in cloud storage. This method offers a strong answer for data protection by improving security and lowering cloud storage needs. Fig. 6 shows the suggested block diagram of the proposed method.

When it comes to accessing data stored in the cloud, the user must first make the request. The cloud server checks the access policy which serves to restrict and regulate who can access the cloud data first to guarantee secure and regulated access. Keys are created for the user when access rights are validated. Message encoding is used in this technique to encode information as Elliptic curve locations. These points are then encrypted using the EKT-EDES technique to further increase security. The encrypted points are then subjected to a series of EDES procedures, and the resultant data is sent to the required user. An EDES operation and the EKT decryption method are used in a reverse procedure to recover the original data. The Elliptic curve values are finally transformed back into a clear message format via message decoding. This all-encompassing strategy guarantees both data security and restricted access, making it an essential part of managing data in the cloud and security.

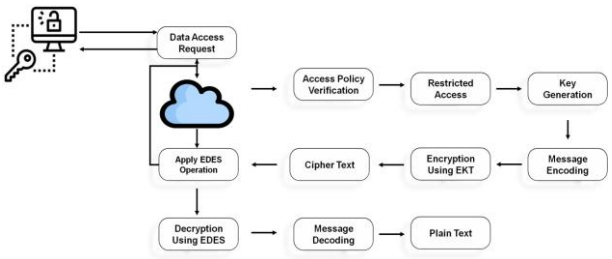

Fig. 6. Block diagram of the proposed method.

1) Key generation

The procedure for creating an EKT key combination is described in the accompanying Algorithm 1. In the beginning, it creates a random integer "*d*" that falls within the range "[1, *n*]", where "*n*" is the numerical value of the generating point "*G*". After that, it multiplies the generator point "*G*" by the private key "*d*" to compute the public key "*Q*". Finally, a safe EKT key combination for encryption operations is returned, consisting of the private key "*d*" and the associated public key "*Q*", in the accompanying Algorithm 1. In the beginning, it creates a random integer "*d*" that falls within the range "[*1*,*n*)", where "*n*" is the numerical value of the generating point "*G*". After that, it multiplies the generator point "*G*" by the private key "*d*" to compute the public key "*Q*". Finally, a safe EKT key combination for encryption operations is returned, consisting of the private key "*d*" and the associated public key "*Q*".

Algorithm 1: Healthcare Records Key Generation

Input:

- Sensitive healthcare records
- An elliptic curve E with parameters, including p (the prime field)
- A generator point G on the curve E
- A random integer d (private key) such that $1 \leq d < n$ where n is the order of G
- Point multiplication function multiply (d, G) which computes d, G on the curve
- Point addition function add (P, Q) which computes $P+Q$ on the curve

Output:

1. Private key d
2. Secured healthcare records (encrypted using the public key).

Procedure: Generate Healthcare Key Pair

1. **Step 1**: Generate a random integer d such that $1 \leq d < n$, where n is the order of the generator point G.
2. **Step 2**: Compute the public key Q:
   $$Q=d\times G$$
   Q is a point on the elliptic curve.
3. **Step 3**: Encrypt the sensitive healthcare records using the public key Q to obtain secured records:
   Secured Records=encrypt with public key Q

Return (d,Q)as the healthcare key pair.

2) Hash function using SHA-256 for data hashing

The process of developing a safe and effective algorithm is required to create a cryptographic hash function for this particular use case. A condensed version of a hash function designed for the given situation is shown below (Algorithm 2). The security-critical applications in real life, we should use well-known cryptographic hash methods like SHA-256.

Algorithm 2: Hash function

Function custom_hash(data):
Input:
data: The input data to be hashed (string)
Operations:
Initialize a hash object using SHA-256 (or another secure hash function)
Hasher = hashlib.sha256()
Update the hash object with the data (encoded as bytes)
Hasher.Update(data.encode('utf-8'))
Get the hexadecimal representation of the hash
Hash_result = hasher. Hexdigest()
Output:
Hash_ Result: The hexadecimal hash value of the input data (string)
Example Usage:
Plaintext = "Your sensitive data here."
Hash_ Value = Custom_ Hash(plaintext)
Print("Hash Value:", hash_value)

3) Encryption phase

Algorithm 3: Encryption phase

Input:
- Sensitive healthcare record (**healthcare_record)** to be encrypted
- Private key (**d_medical)** associated with the ECC key pair for healthcare records
- Public key (**Q_medical**) associated with the ECC key pair for healthcare records
- Random number generator function (**get Random Numbers For Healthcare**()) with special considerations for sensitive data

Output:
- Encrypted sensitive healthcare record (**encrypted_healthcare_record)**

Encryption Procedure:
Step 1: Generate a random integer **k_medical** such that **1 <= k_medical < n_medical** (where **n_medical** is the order of the **generator point G_medical**)
k_medical = get Random Numbers For Healthcare() % n_medical
Step 2: Compute the temporary point **T1_medical** as the result of point multiplication of the **generator point G_medical** by **k_medical**
T1_medical = multiply(G_medical, k_medical)
Step 3: Compute the **x**-coordinate of **T1_medical**
x_coordinate_T1_medical = get X Coordinate (T1_medical)
Step 4: Calculate the shared secret **S_medical** by multiplying the x-coordinate of **T1_medical** with the private key **d_medical**
S_medical = multiply(x_coordinate_T1_medical, d_medical)
Step 5: Compute the encryption point **T2_medical** by adding the public key **Q_medical** to the shared secret **S_medical**
T2_medical = add(Q_medical, S_medical)
Step 6: Encode the sensitive healthcare record (**healthcare_record**) into an elliptic curve point **encoded_healthcare_record_point** = *Encode Record To ECC Point (**healthcare_record**)*
Step 7: Encrypt the healthcare record point using the temporary point **T2_medical**
encrypted_healthcare_record_point = *add(**encoded_healthcare_record_point, T2_medical**)*
Step 8: Output the encrypted sensitive healthcare record point
encrypted_healthcare_record= encrypted_healthcare_record_point
Return ***encrypted_healthcare_record***

4) Decryption phase

Algorithm 4: Decryption phase

Input:
- Private Key (***d_priv)*** for sensitive health care records
- Public key (***Q_pub***) for sensitive health care records
- Encrypted point (***E***) representing sensitive health care records

Output:
- Decrypted point (***D***) representing sensitive health care records

Procedure
ECC_Secure_Decrypt(d_priv, Q_pub,E)*:*
Step 1: Compute the inverse of the private key ***'d_inv'*** in the field of the elliptic curve:
d_inv = modular_inverse(d_priv, n) # *n* is the curve's order
Step 2: Decrypt the encrypted point '***E***'using the public key ***'Q_pub'*** and ***'d_inv'***:
D = multiply(E,d_inv) # 'multiply' is an ECC point multiplication function
Return ***D*** as the decrypted sensitive health care record.

Algorithm 5: EKT-EDES

Encryption Process:
Input: (plaintext (data to be encrypted), recipient_public_key (ECC public key of the recipient), random_key (168-bit random key))
Output:(ciphertext (encrypted data))
Step 1: Encrypt the plaintext using **ECC** (Elliptic Curve Cryptography) with the recipient's public key.
Step 2: Split the **168-bit random key** into three parts (each 56 bits): h1, h2, and h3.
Step 3: Encrypt and decrypt the random key using **3DES** encryption:
- Encrypt the random key using the first part h1.
- Decrypt the result using the second part h2.
- Encrypt the decrypted result again using the third part h3

Step 4: Encrypt the ECC-encrypted data (ecc_encrypted_data) using the final **3DES-encrypted key** (h3_encrypted_key).
Return the final ciphertext.

Decryption Process:
Input:(ciphertext (encrypted data), recipient_private_key (ECC private key of recipient), h3 (third part of the random key))

Output:(plaintext (decrypted data))
Step 1: Decrypt the ciphertext using **3DES** with the third part of the key (h3).
Step 2: Decrypt the **ECC-encrypted data** using the **3DES-decrypted key** (h3_decrypted_key).
Step 3: Decrypt the ECC-decrypted data using the recipient's private ECC key.
Step 4: Finally, ecc_decrypted_data is decrypted using the recipient's ECC private key to obtain the original plaintext.

The safe safeguarding and decryption of private patient healthcare records is guaranteed by the EKT-EDES Algorithm 5. What it does is as follows: the patient's data is first encrypted using the recipient's ECC public key using ECC. The next step is to divide the 168-bit key into its three 56-bit components (H1, H2, and H3). Both data encryption and decryption employ random keys. The key is encrypted by H1, decrypted by H2, and encrypted once again by H3. The H3-encrypted key is then used to encrypt the ECC-encrypted data a second time. The procedure is reversed during decryption: H3 decrypts the key, which is then used for decoding the ECC-encrypted data. The original plain text information about the patient is then obtained by decrypting the ECC-decrypted data with the recipient's private ECC key.

D. Enhanced Role-Based Access Control

Enhanced Role-Based Access Control (ERBAC) is an encryption method for medical records that improves data privacy and security by assigning access permissions based on user responsibilities. This ensures that only those with permission can access, store, or receive sensitive medical records. Fig. 7 depicts the roles of enhanced access control.

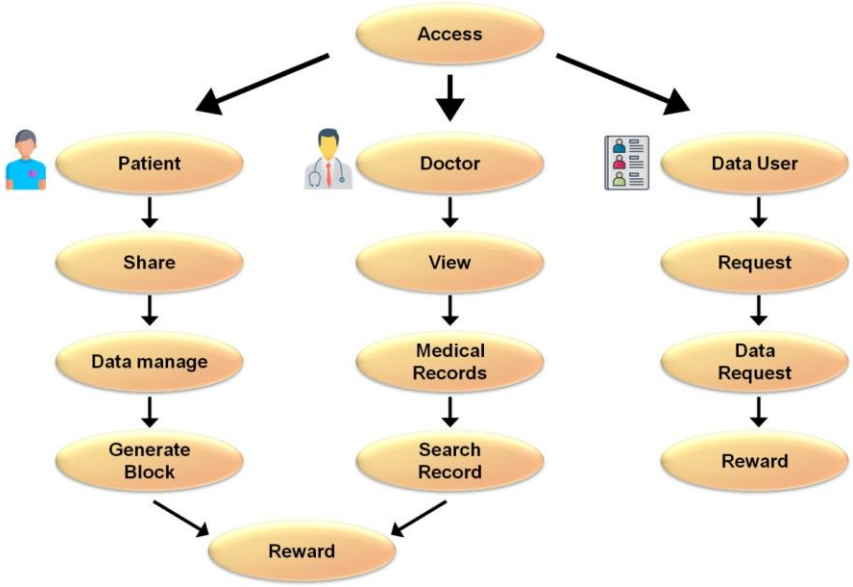

Fig. 7. Roles of enhanced access control.

Patient: The patient uses data as well, imagine a patient $j \in V$ becomes a hospital patient $l \in G$ to see a doctor $i \in T$. A certificate is generated by the hospital's server φj for the patient and gives it back to them. Concurrently, the server keeps $\mu j = G(\varphi j)$ on the list of doctors' services registered. Here φj is comparable to a license plate in terms of how the physician may create medical records for patient j. Upon consultation, the patient will provide the doctor with a certificate. Personal health information and prior medical records must be obtained.

Doctor: With the patient's consent, the physician oversees creating the patient's medical record and extracting a set of keywords from it. The plan assumes that the doctor will be truthful and will not conspire with the data requester to get patient medical information for unlawful gain. Additionally, the patient and the physician discuss a policy of access for encryption of the ciphertext, which is created using a key and medical information, to safely save relevant content. The IPFS is where the client's health-related encryption key is kept, and the smart contract is where the keyword encryption key is kept. The ciphertext consists of these two components.

Data User: This person makes data requests, such as academic organizations, health insurance providers, and relatives of patients, who may acquire pertinent information if their characteristics align with the applicable access policy.

IPFS: The doctor encrypts the patient's medical data and the IPFS saves the ciphertext of that information. The physician is then sent the matching hash value comparable to a file address. The approved requester of data may get the ciphertext pursuing the location after the acquired lesson.

Blockchain: Our research focuses on the system known as blockchain, which is essential for guaranteeing the secure transmission and cloud storage of digital medical data since it enables distributed smart contract execution without the need for centralized authorities. The Ethereal blockchain can help us improve system performance. Right now, our system does not take blockchain miners into account.

Three main stages make up this framework: "step 1-System Establishment," "step 2-Medical Data Generation and cloud Storage," and "step 3-Data Search and Access."

Step 1: The doctor uses the *GlobalSetup*(λ) function to generate a system public *key(PK)*, where λ is the safety parameter. The *KeyGen(PK)* function is used by both patients and physicians to produce their private and public keys *GlobalSetup(* λ *), KeyGeni(PK)* for recipients, and *KeyGenj(PK)* for physicians are the three sub-algorithms included in this. Recording the address and Application

Binary Interface (ABI) of the contract, the physician implements a smart contract on the block-chain of Ethereum to store encrypted keyword index and provide search services for data requesters. The user-specific characteristics or passwords connected with a certain user are represented by the variable "*sku*" in the algorithms for security. These characteristics are essential to access control measures because they guarantee that only authorized individuals with the required characteristics may access private medical information. By allowing for fine-grained access control, the inclusion of "*sku*" in the code makes it easier to generate specific to user's keys and access policies, improving the safety and confidentiality of healthcare data. As a crucial part of the cryptographic infrastructure, the variable "*PK*" is utilized in the algorithm for safety in the setting of storing and retrieving sensitive medical information. It provides the public key system, which contains the keys and cryptographic settings required for safe data access. To create unique user keys (*yi* and *xi*) for safe access control, the *KeyGen* function uses *PK*. To protect patient confidentiality and information integrity in medical facilities, this infrastructure makes sure that only authorized individuals may access and access sensitive medical information.

Algorithm 6: System Establishment

Input:
λ: Security parameter
Function GlobalSetup(λ):
U,g0,e,H = Define attributes ()
a,b = RandomNumbers ()
A=e(g0,g0)$^{\alpha}$
B=g0$^{\beta}$
F = PseudoRandomFunction ()
PK =*(G0,G1,g0,e,H,A,B,F)*
MSC=(α,β)
x = RandomNumber()
sku={ξ}
DeploySmartContract(PK)
RecordContractAddress(ContractAddress)
RecordContractABI(ContractABI)
return *PK,MSK,sku*
Output:
Function *KeyGen(PK)*
yi,xi=GenerateRandom Keys()
pki=g^{yi}
pkj=g^{xi}

Step 2: Patient registration with a hospital is a prerequisite for the generation and cloud storage of medical data. Each patient's φi is chosen by the hospital's server, and both the patient and the server compute μi. During patient interactions, the physician gathers keywords, works with the patient to establish an access policy, and creates data (Data) to be stored on the blockchain. This data consists of an authentication value, the patient's and doctor's signatures, and a hash of the medical records. In accordance with the negotiated access policy, the physician also encodes the patient's initial records *(m)* and keywords set *(Wm)*. IPFS is used to store encrypted information and file location, with the location being encrypted and included into a transaction on the blockchain. The variables and procedures are utilized in the given algorithm to protect the confidentiality and privacy of private medical information. To create secure access tokens that guarantee only authorized users may access the data, φi and μi are used. yi adds an extra degree of security by generating random keys. Keywords and their associated functionalities enable effective retrieval while *AccessPolicy* and *AVij* assist in regulating and authenticating data access. Security and immutability are improved by using the blockchain to store data. Overall, these factors and capabilities work together to protect patient health information, uphold confidentiality, and ensure safe storage and retrieval of healthcare data.

Algorithm 7: Medical Data Generation and cloud Storage

Input: *m* (medical data), *Wm* (set of keywords)
Output:$\varphi i, \mu i, yi$ (from PatientRegistration)
function PatientRegistrationAndDoctorActions(*m,Wm*):
φi = GenerateRandomNumber()
$\mu i = H(\varphi i)$
Send φiToPatient()
yi = GenerateRandomKey()
keywords = ExtractKeywords(*m*)
AccessPolicy = NegotiateAccessPolicy(keywords)
Data = GenerateData(*m,AccessPolicy*)
AVij = GenerateAuthenticationValue(*pki,pkj,AccessPolicy*)
StoreDataOnBlockchain*(Data,AVij)*
RecordFileLocation(hlocation)
EcryptAndStoreRecord (*m,keywords,AccessPolicy*)
for each keyword *wi* in *Wm*:
$hi = F(sku, 2^{kw\sim i})$
$ni = F(sku, 2^{kw\sim i})$
$txidi = ni * txid$
$Ki = ni * K\sim$
StoreKeywordIndexInSmartContract((*hi,txidi,Ki*))
Return $\varphi i, \mu i, yi$
Usage:
m = "Medical data..."
Wm = set(["keyword1", "keyword2", "keyword3"])
φi,μi,yi = PatientRegistrationAndDoctorActions(*m,Wm*)

Step 3: Health Information Medical data can be accessed by authorized data requesters through Search and Access. After sending the doctor an access request, they are granted the necessary *characteristics(S)* and *secret keys(SK)*.

Search *tokens(Tw~)* is generated by data requesters for particular keywords. After determining whether the requester is permitted, the smart contract uses the token to return the found result (*Sk*,*Stxid*). After that, data requesters read the data from the blockchain, decrypt it, and use their secret keys to calculate the decryption keys needed to see if their attributes comply with the access policy. In that case, they decrypt the encrypted data and retrieve the medical record *(m).S, SK, Sk, Tw,* and *Stxid* are only a few of the variables utilized in the given algorithm to improve safety precautions for the preservation and access of sensitive medical information. To provide safe access, secret key creation, and distribution are done using *S* and *SK*. While *Sk* and *Stxid* are employed to securely access search results, *Tw* is produced to securely trigger the smart contract. These variables provide strong security in the

healthcare data ecosystem by assisting in the authentication of data requesters, enforcing access rules, and maintaining the privacy and security of medical information throughout the process.

Algorithm 8: Data Search and Access

Input: *w~,sk* (for TokenGeneration)
Input: *Tw~* (for SmartContractCheck)
Input: *Sk,Stxid* (for DecryptAndAccessData)
Output: *S,SK*, ContractAddress (from KeyGeneration)
Function DataRequesterWorkflow():
SendAccessRequestAndEthereumAddress()
AuthenticateDataRequester()
S = DistributeAttributes()
SK = GenerateSecretKeys(S)
SendSecretKeysAndSmartContractAddress(*SK,Contract Address*)
w~ = "Your search keyword"
sk = "Your search secret key"
$Tw\sim = F(sk, 2^{kw\sim})$
InvokeSmartContractWithToken(*Tw~*)
if DataRequesterIsAuthorized():
Sk,Stxid= RetrieveSearchResult(*Tw~*)
DecryptAndRetrieveData(*Sk,Stxid*)
$n=F(sk,2^{kw\sim})$
for each txi~dj in Stxid:
$Kj=n*K\sim j$
$txidj=n*txi\sim dj$
H location = DecryptLocation(*txidj*)
DownloadEncryptedRecord(hlocation)
if AttributesSatisfyAccessPolicy():
DecryptAndAccessMedicalRecord()
Usage:
DataRequesterWorkflow()

For the purpose of controlling access policies and permissions in a healthcare or health data management application, the provided Algorithm 9 defines a role-based access management system. Different roles are established, including "patient", "doctor", and "data_user" each with distinct actions or permissions. To specify which roles, have access to certain resources, including patient records, access policies are developed. The function "has_permission" which is part of the code, determines whether a user, depending on the role they play and the policy of access, has the appropriate permissions to carry out a certain action on a resource. A more complete medical data management system may incorporate this algorithmic piece, giving it the ability to control and secure the possession of private medical information. It helps to ensure that only people who have appropriate permission are allowed to carry out specific operations on medical records.

Algorithm 9's "has_permission" permission-checking function determines if a user has the necessary action permission for a certain resource. This judgment is based on resource ownership, access regulations, and user roles. It returns true if the user possesses the necessary role and the action is permitted for that role; otherwise, it returns False. The use example shows how to use this method to determine if "user1" can "view_own_records" for "record1" and outputs the result appropriately.

Algorithm 9: Enhanced Role based access control for access policy

Input: user (user ID), resource (resource ID), action (action to perform)
Output: True (if user has permission), False (if user does not have permission)

```
roles = {
  "patient":
["view_own_records","share_own_records","request_access
"],
 "doctor":
["view_patient_records","create_medical_records","search_r
ecords"],
   "data_user": ["request_data"]
}
access_policy = {
   "patient": {
"own_records":
["view_own_records" ,"share_own_records" ],
"doctor_records": ["request_access"]
},
   "doctor": {
"all_records":
"view_patient_records","create_medical_records","search_re
cords"]
 },
"data_user": {
"request_data": ["request_data"]
   }
}
user_roles = {
 "user1": ["patient"],
 "user2": ["doctor"],
 "user3": ["data_user"]
}
patient_records = {
   "record1": {"owner": "user1","data":
"encrypted_data_here"},
 "record2": {"owner": "user1","data":
"encrypted_data_here"},
   "record3": {"owner": "user2","data":
"encrypted_data_here"}
}
function has_permission(user, resource, action):
if user in user_roles:
user_role = user_roles[user][0]
if resource in patient_records:
resource_owner = patient_records[resource]["owner"]
if user_role in access_policy and action in
access_policy[user_role]:
if user_role == "patient":
if user == resource_owner:
return action in access_policy[user_role]["own_records"]
else:
return action in access_policy[user_role]["doctor_records"]
else:
return action in access_policy[user_role]["all_records"]
return False
Usage:
user = "user1"
resource = "record1"
action = "view_own_records"
if has_permission(user,resource,action):
print(f"{user} has permission to {action} {resource}")
else:
print(f"{user} does not have permission to {action}
{resource}")
```

E. Smart Contract

A unique report with pertinent code and data is known as a smart contract (Algorithm 10). The code, uploaded through a report to a worldwide database called the blockchain, is in a binary format unique to Ethereum. The blockchain has fascinating applications, one of which is smart contracts. When two groups conduct a transaction transfer on the Blockchain resource, the smart contract on the Blockchain may be considered a single line of code (contract) that starts to run. In this work, the physician develops a smart contract that ensures a fair search while keeping the patient's keyword index health registers and specific connected data. When there is no reliable outside source, the mortgage search data requester costs to intelligent contracts, facilitating user data retrieval. The service price is only refundable if the correct search results are obtained.

Algorithm 10: Smart contract

```
contract EMRsSharing {
   address public Doctor;
mapping(address => bool) public authorizeRequesters;
mapping(bytes =>IndexSet[]) public Index;
   event Searchevent(address indexed key,bytes[]
result,bytes[] result1);
   modifier onlyDoctor() {
require(msg.sender == Doctor, "Only the doctor can perform
this operation.");
   }
   struct IndexSet {
      bytes txid;
      bytes key;
   }
constructor() {
Doctor = msg.sender;
   }
Input: newRequester (address)
Output: success (bool)
   function addRequester(address newRequester) public
onlyDoctor() returns (bool) {
if (!authorizeRequesters[newRequester]) {
authorizeRequesters[newRequester] = true;
      }
return authorizeRequesters[newRequester];
   }
Input: keywordIndex (bytes),txid (bytes),key (bytes)
function addIndex(bytes memory keywordIndex, bytes
memory txid, bytes memory key) public {
require(msg.sender == Doctor, "Only the doctor can add an
index.");
Index set memory newIndexSet = IndexSet({
txid: txid,
         key: key
      });
Index[keywordIndex].push(newIndexSet);
   }
Input: Token (bytes)
function search(bytes memory Token) public {
require(msg.sender == Doctor, "Only the doctor can perform
a search.");
require(authorizeRequesters[tx.origin], "Requester not
authorized.");
uint256 length = Index[Token].length;
bytes[] memory result = new bytes[](length);
bytes[] memory result1 = new bytes[](length);
for (uint256 i = 0; i< length; i++) {
result[i] = Index[Token][i].key;
result1[i] = Index[Token][i].txid;
      }
emit Searchevent(tx.origin, result, result1);
   }
}
```

IV. Results Analysis

This section details the encryption scheme and blockchain-based security for Storing and Retrieving sensitive healthcare records using access control policies. This platform name refers to the usage of the Python programming language on a machine running the Windows operating system, with Python applications being executed using an Intel Core i5-2450M CPU and 4 GB of RAM. The hybrid EKT-EDES improved the system's uniqueness and efficiency in a unique way by securing many cloud-stored data and establishing secure links for data description and encryption. The proposed methods compared to the existing techniques are:

- Hierarchical Multi-Authority Ciphertext Policy ABE (HM-CP-ABE), Multi-Authority Attribute-Based Encryption (MA-ABE), Multi-Authority Cipher text-Policy Attribute-Based Encryption (MA-CP-ABE) [16].
- Attribute-Based Encryption (ABE), Enhanced Cipher-text Policy-based Attribute Encryption (CP-ABE) and CP-ABE with Rivest Shamir Adleman (RSA) Algorithm [17].
- Hybrid Encryption, TABU and MA-ABE [18].
- Improved Rivest Cipher 4 (IRC4), Fractal Video Sequence Compression Algorithm (FVSCA) and Field Programmable Gate Arrays (FPGA) [19].

The evaluation metrics are execution time, encryption time, decryption time, key generation time, throughput, latency, data loss and running time.

The Enhanced CP-ABE, ABE, and CP-ABE with RSA algorithm are some of the existing techniques that are compared effectively with the suggested EKT-EDES method. With support for keyword search and re-encryption, EKT-EDES guarantees data integrity and secrecy. It allows for decentralized storage and offers several choices for access control. EKT-EDES is additionally capable of fending off security risks like man-in-the-middle, collusion, and brute force attacks. ABE and Enhanced CP-ABE, for example, might not provide as many complete security features as EKT-EDES, especially when it comes to re-encryption, keyword search, and resistance to particular attack routes. As a result, EKT-EDES is a strong option for protecting sensitive data. The suggested approach for massive healthcare data cloud storage in a blockchain setting is assessed using several parameters. These are outlined below to enhance the blockchain environment. The secrecy of the data is dependent on these cryptographic keys. Because it has an effect on the system's overall efficiency and operational effectiveness, key generation time is a crucial component to take into account in healthcare record security. As the

decryption and encryption of data procedures move more quickly, it makes it possible to retrieve patient records more quickly when necessary. An overview of the security analysis is provided in Table I.

TABLE I. SECURITY ANALYSIS

Security Aspect	EKT-EDES (proposed)	ABE	Enhanced CP-ABE	CP-ABE with RSA	HM-CP-ABE	MA-ABE	MA-CP-ABE	Hybrid Encryption	TABU	IRC4	FVSCA	FPGA
Confidentiality of Data	High	High	High	High	High	High	High	High	High	High	High	Low
Data Integrity	High	High	High	High	High	High	High	High	High	High	High	Low
Re-Encryption	High	Low	High	High	High	Low	High	High	High	Low	Low	Low
Keyword Search	High	Low	High	Low	High	Low	Low	Low	Low	High	Low	Low
Control of Multiple Access	High	Low	High	Low	Low	Low	Low	Low	Low	Low	Low	Low
Dispersed Storage	High	Low	Low	Low	Low	Low	Low	Low	Low	Low	Low	Low
Overall Security	High	Low	Low	Low	Low	Low	Low	Low	Low	Low	Low	Low
Brutal Force Offensive	Low	High	High	High	High	High	High	High	High	Low	Low	Low
Collisional Assault	Low	High	High	High	High	High	High	High	High	Low	Low	Low
Man-in-the-Middle Operation	Low	High	High	High	High	High	High	High	High	Low	Low	Low

For this reason, a shorter key time for generation is frequently preferred. Fig. 8 depicts the outcome of key generation time. Due to the speed in key creation, which greatly cuts the time needed to construct cryptographic keys, the suggested technique is preferred for safely conserving and accessing sensitive patient information in healthcare. By using an expedited key generation procedure, strong security protections are maintained while allowing authorized staff quick access to patient data.

The term "execution time security" describes how sensitive healthcare records are safeguarded during the whole execution lifecycle, from the time they are accessed or retrieved until the task is completed (when storing and retrieving them via access control regulations).

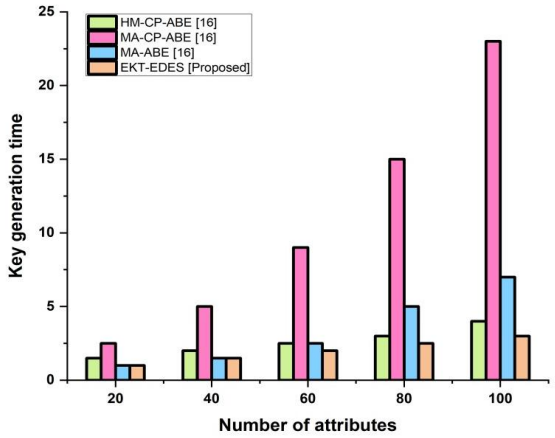

Fig. 8. Outcome of key generation time.

Throughout the execution process, we will take several security precautions to guarantee the records' privacy, availability, and integrity. The mathematical Eq. (11) is used for calculating the execution time.

$$Execution\ time = P \times CPP \times T \quad (11)$$

where T is the clock cycle time, P is the number of instructions in the program, and CPP is the average number of cycles per instruction. The analysis of the execution time is shown in Fig. 9.

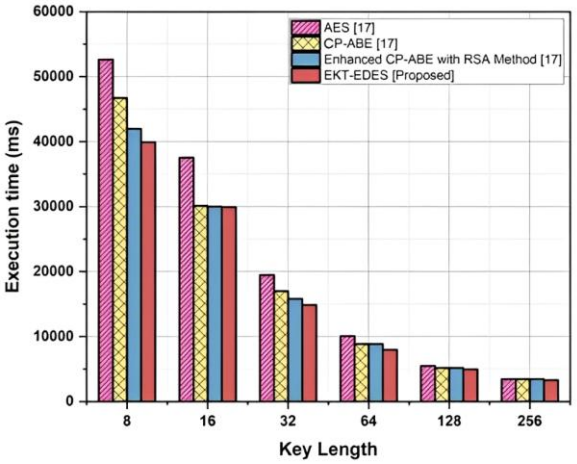

Fig. 9. Analysis of execution time.

Table II compares the execution times (measured in milliseconds) of various encryption algorithms, including AES, CP-ABE, Enhanced CP-ABE with RSA, and EKT-EDES, with key lengths ranging from 8 to 256 bits.

TABLE II. COMPARISON OF EXECUTION TIME WITH AN EXISTING ALGORITHM

Key Length	Execution time (ms)			
	AES [17]	CP-ABE [17]	Enhanced CP-ABE with RSA Method [17]	EKT-EDES [Proposed]
8	52,617	46,717	41,977	39,895
16	37,509	30,112	29,990	29,910
32	19,468	16,993	15,822	14,860
64	10,061	8859	8856	7952
128	5494	5196	5198	4955
256	3449	3459	3437	3295

The execution timings of the suggested method (EKT-EDES 3295 ms) are significantly shorter than those of the current process [17] (AES 3449 ms, CP-ABE 3459 ms, and CP-ABE with RSA 3437 ms). Due to its improved encryption approach, EKT-EDES regularly performs better than other methods.

This increased effectiveness shows that the proposed approach improves the security of sensitive medical record cloud storage and retrieval.

The term "encryption time" describes the time required to encrypt data before cloud storage and decrypt it upon retrieval in the context of access control policies for storing and retrieving sensitive healthcare records. Ensuring the security and confidentiality of medical records requires this procedure. The Eq. (12) of the encryption time is as follows:

$$\textit{Encryption time} = \textit{Time taken to encrypt data} + \textit{Time taken to decrypt data} \quad (12)$$

The encryption time analysis is shown in Fig. 10.

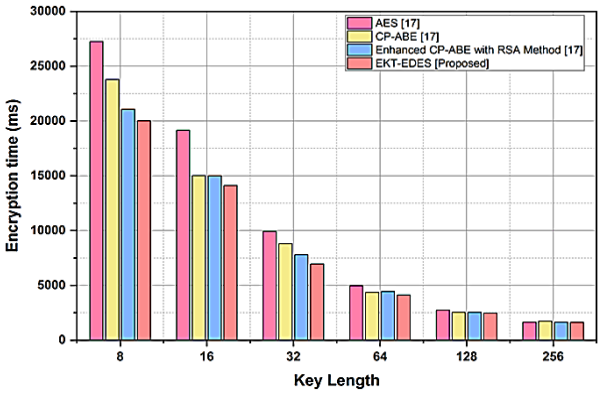

Fig. 10. Analysis of encryption time.

For patient healthcare records, Table III shows the encryption time (in milliseconds) for several encryption algorithms with different key lengths (8, 16, 32, 64, 128, 256 bits). Comparisons are made between AES, CP-ABE, enhanced CP-ABE with RSA [17], and the proposed EKT-EDES technique. In general, encryption time gets faster as the length of the key increases. The suggested EKT-EDES (EKT-EDES 1628ms) technique works better than existing ones (AES 1648ms, CP-ABE 1721 ms, CP-ABE with RSA 1638ms) because it provides effective encryption for protecting patient health records, maintaining data privacy, and reducing processing overhead, a crucial component of medical data management.

TABLE III. COMPARISON OF ENCRYPTION TIME WITH AN EXISTING ALGORITHM

Key Length	Encryption time (ms)			
	AES [17]	CP-ABE [17]	Enhanced CP-ABE with RSA Method [17]	EKT-EDES [Proposed]
8	27,241	23,783	21,070	20,010
16	19,158	15,010	14,994	14,120
32	9920	8794	7795	6950
64	4972	4365	4436	4115
128	2742	2542	2544	2480
256	1648	1721	1638	1628

The time required to decrypt encrypted records and make them available to authorized users is called the "decryption time" in the context of security for storing and retrieving sensitive healthcare records. It is a crucial factor in healthcare information safety to guarantee that private patient data is safe from unwanted access while readily available to medical experts when needed.

Time needed to decrypt data can be calculated using the following formula:

$$\textit{Decryption time} = D/P \quad (13)$$

where D is the encrypted data's size or complexity, and P is the system's processing capacity or decryption efficiency.

The decryption time is shown in Fig. 11.

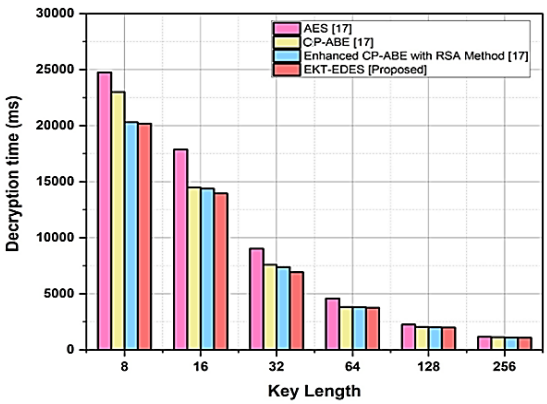

Fig. 11. Outcome of decryption time.

To retrieve private healthcare information, Table IV compares the key lengths of various encryption techniques and their related decryption durations, with special emphasis on security and privacy issues. With increasing decryption time as key lengths grow, the commonly used encryption technology AES ensures strong security but may slow access speed. CP-ABE, in contrast, provides quicker decryption times, making it appropriate for quick data retrieval while protecting privacy by enabling access based on characteristics rather than direct keys. Performance and security are balanced by enhanced CP-ABE with RSA integration. While the remarkable decryption speed of the recommended EKT-EDES technique makes it ideal for speedy access to medical information, its security implications should be carefully evaluated. Finding a balance between these factors is necessary to protect confidential healthcare data while guaranteeing convenient access. The suggested method (EKT-EDES1087 ms) is beneficial for protecting patient data since it improves the security of storing and retrieving sensitive healthcare records and has a faster decryption time than the current way (AES1175 ms, CP-ABE 1134 ms, CP-ABE with RSA1102 ms).

TABLE IV. COMPARISON OF DECRYPTION TIME WITH AN EXISTING ALGORITHM

Key Length	Decryption time (ms)			
	AES [17]	CP-ABE [17]	Enhanced CP-ABE with RSA Method [17]	EKT-EDES [Proposed]
8	24,747	23,005	20,299	20,190
16	17,873	14,512	14,382	13,958
32	9042	7592	7394	6942
64	4593	3836	3813	3765
128	2265	2053	2019	1995
256	1175	1134	1102	1087

The rate at which information can be analysed or transferred is referred to as throughput. It could be used to describe the speed at which records can be accessed or obtained from a database or storage system in the context of medical record storage and retrieval.

In the healthcare industry, high throughput is frequently necessary to guarantee prompt access to patient data. Fig. 12 represents the outcome of throughput.

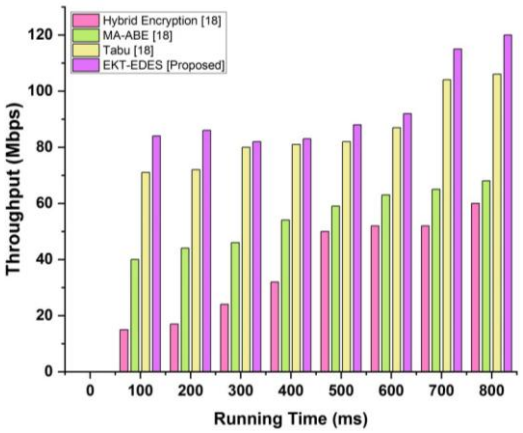

Fig. 12. Outcome of throughput.

A useful comparison of several techniques for collecting private and sensitive medical information using their throughput (in Ms) and privacy and security consequences is presented in Table V.

TABLE V. COMPARISON OF THROUGHPUT WITH AN EXISTING ALGORITHM

Running Time (ms)	Through put (Mbps)			
	Hybrid Encryption [18]	MA-ABE [18]	Tabu [18]	EKT-EDES [Proposed]
0	0	0	0	0
100	15	40	71	84
200	17	44	72	86
300	24	46	80	82
400	32	54	81	83
500	50	59	82	88
600	52	63	87	92
700	52	65	104	115
800	60	68	106	120

The suggested EKT-EDES approach outperforms other encryption algorithms in terms of efficiency when it comes to the storage and retrieval of private medical information. EKT-EDES outperforms current methods [18], including Hybrid Encryption, MA-ABE, and Tabu, with running durations between 71 and 120 ms and throughput between 82 and 92 Mbps, guaranteeing strong security measures for protecting medical data. The result of latency is shown below, in Fig. 13.

Data loss in security refers to the unintentional or unauthorized removal, fraud, or lack of crucial patient data, harming its integrity, privacy, or accessibility, putting the privacy of patients and healthcare operations in danger. With an emphasis on security and privacy, the Fig. 14 shows a comparative examination of various analysis of data loss used for obtaining sensitive medical information. Epochs between 0.4 and 1.5 reflect the computational cost (in an undefined unit) related to each approach at various security levels. With minimal data loss rates across different intensity coefficients, the proposed EKT-EDES system shows promising performance in evaluating the security elements involved in storing and accessing confidential healthcare records. EKT-EDES continuously maintains reduced data loss rates, ranging from 0.05% to 0.5%, in comparison to IRC4, FVSCA, and FPGA.

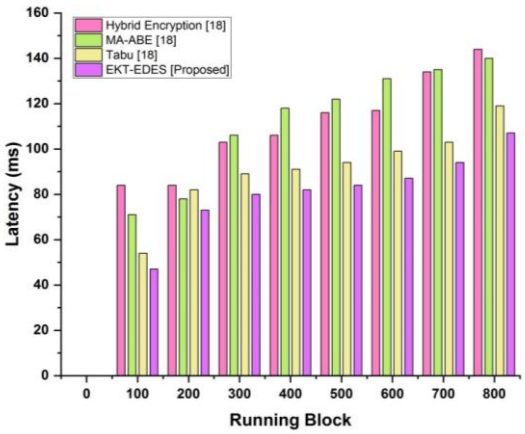

Fig. 13. Result of latency.

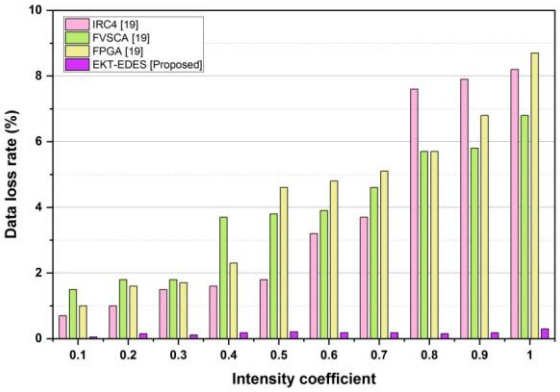

Fig. 14. Analysis of data loss.

V. DISCUSSIONS AND LIMITATIONS

The proposed work presents several advantages and challenges for securing healthcare records. One of its primary benefits is robust security, providing strong protection for sensitive patient information against unauthorized access and cyber threats. The use of smaller key sizes, such as 256-bit, not only enhances security but also reduces storage and transmission requirements, which is particularly advantageous in managing large volumes of Electronic Health Records (EHRs). Additionally, ECC operations demand less computational power, making them suitable for mobile health applications and devices with limited resources, resulting in faster encryption and decryption processes that improve the efficiency of data sharing and transactions. Furthermore, ECC's scalability allows it to accommodate the growing amounts of EHRs without significant increases in resource demands, helping

healthcare organizations comply with strict security regulations like HIPAA. However, ECC also comes with challenges. Its mathematical complexities can complicate the implementation, requiring specialized knowledge to ensure secure deployment. Additionally, as a relatively newer technology, ECC has not been extensively tested in real-world healthcare applications, raising potential concerns about undiscovered vulnerabilities. Integration with existing systems may pose compatibility issues, particularly with legacy technologies that do not support ECC. Moreover, poor implementation choices can introduce security risks, emphasizing the need for careful parameter selection and coding practices. While many ECC patents have expired, some proprietary implementations might still face licensing issues. Using 3DES with only two unique keys (2-key 3DES) instead of three (3-key 3DES) significantly impacts the security of the proposed healthcare record system. In 2-key 3DES, encryption is still applied three times (encrypt-decrypt-encrypt) but with only two distinct keys, reducing the effective key length from 168 bits to about 112 bits due to key reuse. While 112-bit encryption can still provide moderate security, it is weaker than the 168-bit strength of 3-key 3DES and is more vulnerable to brute-force attacks and certain cryptanalytic methods like meet-in-the-middle attacks.

For sensitive applications like healthcare records, where data privacy and integrity are critical, 3-key 3DES or stronger algorithms like AES are preferred to ensure robust security. AES, for instance, offers better performance and security, with support for 128, 192, and 256-bit key lengths, making it a more future-proof choice.

Lastly, ECC, like all public key cryptography, is vulnerable to future quantum computing threats, prompting the exploration of quantum-resistant alternatives. In general, ECC offers significant benefits for securing healthcare records, but careful consideration of its complexities and integration challenges is essential to effectively protect sensitive patient data.

A hybrid encryption technique such as tri-iterative DES (EKT-EDES) combined with elliptic curve cryptography (ECC) provides a robust level of security, which is critical for safeguarding sensitive healthcare data. Although computational intensity may pose challenges for devices with limited processing power, this trade-off ensures that data confidentiality, integrity, and authenticity are maintained to the highest standards. In addition, the adoption of these advanced encryption techniques encourages healthcare companies to upgrade their infrastructure, fostering long-term technological growth and resilience. For mobile devices or low-powered systems, optimization techniques such as pre-computed keys, selective encryption, or task offloading to cloud servers can be implemented to mitigate delays. However, tri-iterative DES (EKT-EDES) and Elliptic Curve Cryptography (ECC) hybrid encryption techniques, should be built on the provided context:

1. Future-proof security: ECC is recognized for its efficiency in providing strong security with smaller key sizes, making it more resistant to brute-force attacks. This enables the system to remain secure even as computational power increases in the future.

2. Compliance with regulations: strong encryption mechanisms help healthcare organizations meet strict regulatory requirements, such as HIPAA, GDPR, or similar standards, ensuring legal compliance for data security.

3. Interoperability: A hybrid encryption system can support secure data sharing across different platforms and devices, improving collaboration and interoperability in healthcare ecosystems.

4. Protection against emerging threats: combining multiple encryption techniques, like EKT-EDES and ECC, reduces vulnerabilities by leveraging the strengths of each method, providing defense-in-depth against increasingly sophisticated cyber threats.

5. High confidentiality for sensitive data: healthcare records often include highly sensitive information, such as patient diagnoses and treatment plans. This system ensures that only authorized personnel can access such data, maintaining confidentiality.

6. Enhanced trust: patients and stakeholders are more likely to trust a healthcare provider that prioritizes data security, enhancing the organization's reputation and reliability.

7. Support for distributed systems: in environments like cloud computing, where healthcare data might be stored or processed across multiple locations, the hybrid technique provides robust end-to-end encryption, safeguarding data at every point.

8. Reduced risk of single point of failure: the use of two encryption methods ensures redundancy; even if one method is compromised, the other continues to provide protection.

9. Adaptability for critical applications: the system can be adapted to secure high-priority, mission-critical data without compromising other operations, ensuring healthcare services continue efficiently.

To support a hybrid encryption program using tri-iterative DES (EKT-EDES) and Elliptic Curve Cryptography (ECC), the system must meet certain hardware and software requirements. Minimum hardware includes a multi-core processor (2.5 GHz or higher) with cryptographic acceleration, 8 GB RAM (16 GB preferred), a 256 GB SSD (512 GB recommended), a high-speed network interface (1 Gbps), and optional GPUs for enhanced performance. For software, the system should run on a secure OS like Windows 10/11, Ubuntu 20.04 and utilize cryptographic libraries such as OpenSSL or Libgcrypt, with programming support in Python, C++, or Java. A secure database (e.g. PostgreSQL, MySQL with encryption) is essential, along with Transport Layer Security (TLS) 1.2/1.3 for data transmission and monitoring tools like Splunk or Elasticsearch, Logstash, and Kibana (ELK) for performance and security. Cloud integration with AWS, Azure, or Google Cloud, featuring Hardware Security Modules (HSM), can further enhance operations. These specifications ensure secure and efficient handling of sensitive healthcare data.

VI. Conclusions

We have developed a framework to address the safe sharing and storage of existing EMRs in the cloud using blockchain and innovative contract technologies. The technology allows doctors to submit the ciphertext to IPFS after first encrypting the EMR with the proper access rules. By combining IPFS with blockchain, doctors can handle large volumes of electronic health information in the cloud using IPFS, avoiding uploading data directly to the chain and saving network capacity on the blockchain. To ensure the confidentiality of electronic medical records in the cloud, physicians' and patients' information is encrypted. A keyword index is also created in the cloud to facilitate the search of encrypted medical data.

The blockchain stores the encrypted keyword index data, which is used by the smart contract in the cloud to conduct keyword searches inside the distributed system. We use EKT-EDES for role-based access control for data requesters in the cloud. Data security, privacy protection, and safe tracking are shown through security analysis. Furthermore, we evaluated the overall cost of SC in the cloud. We carry out a thorough performance analysis, examining variables such as key generation time (3), encryption time (1628 ms), decryption time (1087 ms), execution time (3295 ms), latency (107 ms), throughput (120 ms), and data loss. In this work, smart contracts are designed to give patients complete control over their health records, allowing them to decide with whom to share their data. All transactions are securely logged onto the blockchain. Various test cases are presented to demonstrate the functionality and cost efficiency of the proposed algorithm.

Our results show that the proposed decentralized system effectively secures and shares EHRs with authorized users without exposing private keys. This approach addresses key challenges such as data integrity, security, scalability, and immutability.

Future work will focus on enhancing response times, reducing latency, and minimizing costs, making the system suitable for devices with limited resource capacity.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, S.A. and N.P.; methodology, S.A., N.P., L.G. and R.-V.R..; software, S.A., R.-V.R. and L.G.; validation, S.A., N.P.; formal analysis, S.A., N.P., R.-V.R. and L.G.; investigation, S.A., N.P., R.-V.R. and L.G.; resources, S.A., N.P., R.-V.R. and L.G.; data curation, S.A., N.P., R.-V.R. and L.G.; writing—original draft preparation, S.A., N.P.; writing—review and editing, R.-V.R. and L.G.; visualization, S.A., N.P., R.-V.R. and L.G.; supervision, S.A. and N.P.; project administration, N.P.; funding acquisition, L.G. and R.-V.R.; all authors had approved the final version.

Acknowledgment

The last two authors wish to thank Babes-Bolyai University, Cluj-Napoca, Romania, for financial support.

References

[1] F. A. Reegu, H. Abas, Y. Gulzar, Q. Xin, A. A. Alwan, A. Jabbari, R. G. Sonkamble, and R. A. Dziyauddin, "Blockchain-based framework for interoperable electronic health records for an improved healthcare system," *Sustainability*, vol. 15, no. 8, 6337, 2023. https://www.mdpi.com/2071-1050/15/8/6337

[2] Q. Zhang and Z. Zhao, "Distributed storage scheme for encryption speech data based on blockchain and IPFS," *The Journal of Supercomputing*, vol. 79, no. 1, pp. 897–923, 2023. https://doi.org/10.1007/s11227-022-04702-1

[3] C. Thapa and S. Camtepe, "Precision health data: Requirements, challenges and existing techniques for data security and privacy," *Computers in Biology and Medicine*, vol. 129, 104130, 2021. https://doi.org/10.1016/j.compbiomed.2020.104130

[4] K. Keshta and A. Odeh, "Security and privacy of electronic health records: Concerns and challenges," *Egyptian Informatics Journal*, vol. 22, no. 2, pp. 177–183, 2021. doi: 10.1016/j.eij.2020.07.003

[5] S. Mohapatra, B. Sainath *et al.*, "Application of blockchain technology in the agri-food system: A systematic bibliometric visualization analysis and policy imperatives," *Journal of Agribusiness in Developing and Emerging Economies*, 2023. https://doi.org/10.1108/JADEE-10-2022-0237

[6] A. H. Eyeleko and T. A. Feng, "Critical overview of industrial internet of things security and privacy issues using a layer-based hacking scenario," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21917–21941, 2023. doi: 10.1109/JIOT.2023.3308195

[7] H. Xu, Q. He, X. Li, B. Jiang, and K. Qin, "BDSS-FA: A blockchain-based data security sharing platform with fine-grained access control," *IEEE Access*, vol. 8, pp. 87552–87561, 2020. doi: 10.1109/ACCESS.2020.2992649

[8] M. T. D. Oliveira *et al.*, "Towards a blockchain-based secure electronic medical record for healthcare applications," in *Proc. 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761307

[9] G. Xu, S. Xu, Y. Cao, F. Yun, Y. Cui, Y. Yu, and K. Xiao, "PPSEB: A post-quantum public-key searchable encryption scheme on blockchain for E-healthcare scenarios," *Security and Communication Networks*, 2022. doi: 10.1155/2022/3368819

[10] Y. Chen, S. Ding, Z. Xu, H. Zheng, and S. Yang, "Blockchain-based medical records secure storage and medical service framework," *Journal of Medical Systems*, vol. 43, pp. 1–9, 2019. doi: 10.1007/s10916-019-1460-5

[11] A. Amanat, M. Rizwan, C. Maple, Y. B. Zikria, A. S. Almadhor, and S. W. Kim, "Blockchain and cloud computing-based secure electronic healthcare records storage and sharing," *Frontiers in Public Health*, vol. 10, 938707, 2022. doi: 10.3389/fpubh.2022.938707

[12] B. Arunkumar and G. Kousalya, "Blockchain-based decentralized and secure lightweight e-health system for electronic health records," in *Proc. Fifth International Conference on Intelligent Systems, Technologies, and Applications*, 2019, pp. 273–289. doi: 10.1007/978-981-15-3914-5_21

[13] L. Chen, W. K. Lee, C. C. Chang, K. K. R. Choo, and N. Zhang, "Blockchain-based searchable encryption for electronic health record sharing," *Future Generation Computer Systems*, vol. 95, pp. 420–429, 2019. https://doi.org/10.1016/j.future.2019.01.018

[14] J. Gong and L. Zhao, "Blockchain application in healthcare service mode based on health data bank," *Frontiers of Engineering Management*, vol. 7, no. 4, pp. 605–614, 2020. doi: 10.1007/s42524-020-0138-9

[15] A. Ali, M. F. Pasha, J. Ali, O. H. Fang, M. Masud, A. D. Jurcut, and M. A. Alzain, "Deep learning based homomorphic secure search-able encryption for keyword search in blockchain healthcare system: A novel approach to cryptography," *Sensors*, vol. 22, 528, 2022. doi: 10.3390/s22020528

[16] B. Chandrasekaran, Y. Nogami, and R. Balakrishnan, "An efficient hierarchical multi-authority attribute-based encryption scheme for profile matching using a fast ate pairing in cloud environment," *Journal of Communications Software and Systems*, vol. 14, no. 2, pp. 151–156, 2018. doi: 10.24138/jcomss.v14i2.461

[17] M. N. Ramachandra, M. S. Rao, W. C. Lai, B. D. Parameshachari, J. A. Babu, and K. L. Hemalatha, "An efficient and secure big data storage in cloud environment by using triple data encryption standard," *Big Data and Cogn. Comput*., vol. 6, no, 4, 101, 2022. https://doi.org/10.3390/bdcc6040101

[18] T. Nhan, K. Upadhyay, and K. Poudel, "Towards patient-centric healthcare: Leveraging blockchain for electronic health records," in *Proc. the 2024 Computers and People Research Conference*, 2024, vol. 13, pp. 1–8. https://doi.org/10.1145/3632634.3655883

[19] T. Górski, "Smart contract design pattern for processing logically coherent transaction types," *Appl. Sci.*, vol. 14, 2224, 2024. https://doi.org/10.3390/app14062224