

DOI: <http://doi.org/10.32792/utq.jceps.10.01.01>

Cryptographic Key Exchange Using Blockchain Technology

Fatima Muhsin Naeem¹

Kadhim H. Alibraheemi²

msc21co7@utq.edu.iq

kalibraheemi@utq.edu.iq

¹University of Thi-Qar, College of Education for Pure Science, Computer Science, Iraq

Received 25/11/2023

Accepted 21/1/ 2024, Published 1/3/2024



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Abstract:

Cryptographic key management systems (KMS) are an important component of secure communications systems and organisations must protect sensitive information by maintaining confidentiality, integrity, and reliability. In this paper, we create an efficient cryptographic key management application that uses blockchain technology (BC) and fingerprint biometrics to meet this demand. A private network of ten authorised users is created according to the Fingerprint Verification Competition 2000 (FVC2000) database. They create and exchange encrypted keys according to security standards that include confidentiality, authenticity, trust, and limited access to the system. The system requires each authorized user to log in using their email and password, in addition to their fingerprint.

The fingerprint is converted to binary if it is grayscale. If the fingerprint already exists in binary format, the system will extract the fingerprint features in matrix form by processing any missing or incorrect information with an emphasis on accurate representation. A 256-bit encrypted key is produced from this matrix after feature retrieval, by converting the feature matrix into a long string which is then fed through the secure hash algorithm 256 (SHA256) function. The app uses blockchain technology to encrypt the key, apply additional encryption, and associate it with a unique hash. The encrypted key is contained within a block. Users then securely exchange this encrypted block. The application strictly accepts authorised users and rejects unauthorised users. The encryption KMS application provides a high level of security and reliability for organizations looking to protect their encryption keys by combining the use of fingerprint biometrics and BC technology.

Keywords: Blockchain technology, SHA256, Fingerprint, Hybrid Shape and Orientation Descriptor, Key Management System

1-Introduction

Key management involves supervising cryptographic keys within a cryptosystem, including processes related to key generation, exchange, storage, use, and replacement at the user level[1]. A KMS includes key servers, user procedures, and protocols, including cryptographic protocol design. Effective key

management is pivotal in determining the security of a cryptosystem, ensuring that cryptographic keys are securely delivered to the appropriate keyholders and used by relevant protocols[2].

Cryptographic key management networks may include both long-lived and short-lived keys, depending on the systems for which they are created[3]. The complexity of cryptographic keys varies based on factors such as the length, source, or probability of the key being guessed, along with its resistance to unauthorized access[4]. The traditional methods by which the key is generated, such as those based on computers or natural phenomena, may be common and uncomplicated, but they are not robust enough to meet security challenges. Alternative methods have been used, including key generation through biometric authentication and hashing.[5]. In this approach, keys are generated based on biometrics and combined with the SHA256 hash function. This is because biometric authentication provides ease of use and good storage, and the hash function is distinguished by creating a unique, non-repeatable key. It is also impossible to obtain the raw data from which the key was generated during it.

Other challenges remain, including the speed of key creation and exchange, and whether the key will be distributed to all participants or only to the requested party. As a result of the protection, transparency and credibility that BC technology possesses when messaging, encrypted keys are exchanged between authorized users within the network, so that the encrypted key is included and sent explicitly to the party to be delivered. As for the rest of the users, they are notified of the messaging process by sending the key after encrypting it again, and this makes users verify that an actual messaging transaction has taken place without them knowing the key.

2-Related Works

Numerous studies have explored key management systems utilizing Blockchain technology, employing a variety of methods and techniques.

Ribeiro, Victor et al. In (2020) examined a KMS for long range wide area network environments. To improve security and availability in long range wide area network (LoRaWAN), a secure key management architecture built on smart contracts and permissioned BC was developed. A functional prototype was built using open-source software and affordable hardware to demonstrate the feasibility of the proposed LoRaWAN BC-based architecture. Performance investigation reveals that, especially for small and medium-sized LoRaWAN, the prototype exhibits execution time and reaction time values that are comparable to the conventional system[6].

Jia, Chenxi et al. In (2021) developed a dynamic key management based on BC technology using the built and modelled wireless sensor network framework of integrate building and energy management system (IBEMS). The test results showed that the proposed solution reduces the data storage time and space of each sensor and improves the control of IBEMS after thoroughly examining the security of BC technology. The results of the study serve as a guide for creating secure and trustworthy IBEMS based on spatial distribution and promote the use of BC technology in other internet of thing's situations[7].

Pal, Om Alam et al. In (2021) provided a review of the BC, a look at existing PKI for the BC, and key management for a BC wallet. The findings indicate that BC technology is very useful in terms of security and the distribution of authority because it removes the need for a third party and improves the security of the data in circulation, which leads to greater efficiency. Additionally, they suggested a group key

management method for collective secure communication to ensure the security of sensitive records through the BC network[8].

Fotohi, Reza et al. In (2021) have suggested a method to handle the enormous amount of data being transferred globally via the Internet of Things. There are two steps in the procedure. The first step in securing communication between devices on a BC platform is to authenticate each node using a file identity-based signature. The second stage involves sending blocks using hashing. Device identities are used as public keys for this reason. The suggested solution outperformed S-LoRaWAN and DLBA-IoT in terms of average detection rate, throughput, scalability, time required for block authentication, and energy consumption[9].

Liu, Qingyuan et al. In (2023) presented a BC-based architecture for certifying the identities of distributed Internet devices. This paradigm blends cloud computing with BC technology to suggest a network architecture comprising a hardware, middle, and cloud layer. This mesh architecture provides good interference resistance and high-security features. A key distribution model based on network security cryptography and an identity authentication model based on hash chains are proposed within the framework of this network architecture[10].

Mihaljevic, Miodrag Knezevic, et al. In (2023) presented a study on the problem of data access control when subscribers are IoT devices with a configuration that cannot be updated during the entire life cycle. A general framework and a specific example for conditional data access control within the Internet of Things are proposed. The general framework is based on the use of a custom secret key-based broadcast encryption scheme where the encrypted credentials are available for conditional access to data in the BC and the encrypted data subject to conditional access is available in an off-chain source of the data stream. Reduced key management overhead compared to direct delivery of decryption keys is demonstrated. An instance of the proposed framework built on the Ethereum BC platform is developed and experimentally evaluated[11].

3-Data set

In this system, the dataset evaluated in August 2000, FVC2000, was used for ten users, each of them has ten fingerprints, depending on the difference in the impression of each fingerprint. The dataset was carefully selected from the training set to pass validation testing despite challenges and modifications. These fingerprints have undergone a number of tests to evaluate their quality, including wet, dry, scratched, twisted and rounded fingerprints. The tests are not for identification but to verify fingerprints. Table 1 is a description of the dataset that was used in the application.

Table 1. Fingerprints dataset

No	User name	Test set fingerprint			Training set fingerprint								
1	A												
2	B												
3	C												
4	D												
5	E												
6	F												
7	G												
8	H												
9	I												
10	J												

4. Research Methodology

The proposed paper presents an effective application for managing encryption keys using BC technology to take advantage of its high security features. The application steps are divided into three main stages. The first is the login process by entering your email, password and fingerprint. When entry is authorized, the second stage comes, which is the process of generating encryption keys based on fingerprints after extracting features using the fine details extraction algorithm based on “Hybrid Shape and Orientation Descriptor” and feeding them to the SHA256 to generate an encrypted key with a length of 256 bits. The combination of fingerprint biometrics and SHA256 hash function provides strength and robustness to encryption keys, as it is not possible to re-decrypt and obtain the key generation source, in addition to the ease of key generation as it is based only on the user's fingerprint.

The third step is the process of exchanging the encrypted keys that were created between authorized users within the network, where the user who created the key based on his fingerprints sends the encrypted key to a party and notifies the rest of the users of the practical correspondence, relying on BC technology, which guarantees the key to be sent within. block and give it a previous unique hash and a current hash. The block is sent to all users for verification, of course with the original key kept encrypted for all users except the intended party. When the block is verified, the key is sent to the requested party in a secure and transparent manner. Fig.1 shows the application of a KMS.

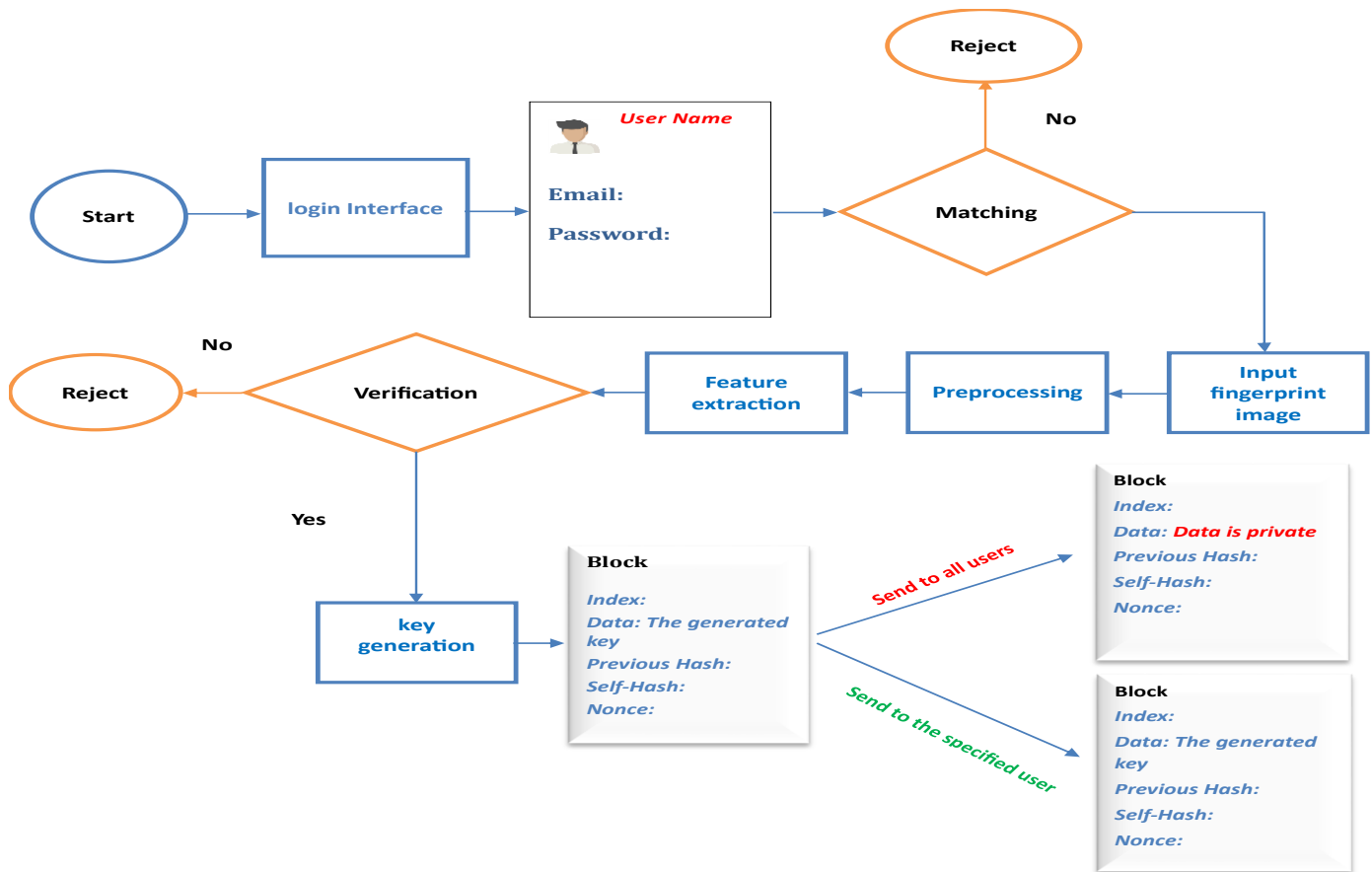


Figure 1. Key management system application

4.1. Application implementation stages

The proposed application consists of three parts:

- Registration and authentication part
- The encrypted key generation part
- The security part of the encrypted key

4.1.1. Registration and authentication part

Authentication takes place in two stages: Email and password authentication and fingerprint authentication.

4.1.2. Email and password authentication

To enhance system security and limit unauthorized access, authorized users are required to register their email and password. The verification process is done to authenticate their identities, thus preventing unauthorized individuals from accessing their login credentials.

4.1.3. Fingerprint authentication:

The FVC2000 database was employed, comprising 100 fingerprint images, with each individual having ten images of their fingerprint captured in various positions. Authentication tests were

conducted on these images using optical scanners to obtain ten different fingerprint scans. Subsequently, fingerprint matching using the mixed shape and orientation descriptor method was applied to extract intricate minutiae from the fingerprint images. The process involves multiple steps in discerning fingerprint patterns and confirming the identity of users.

4.1.4. Fingerprint Feature

The following parameters are contained in the feature vector produced for each fine minutia point using the fine minutia-based algorithm: Three coordinates are given: X, Y, and orientation. The extraction portion is summarized in the next section.

4.1.5. Minutiae Extraction

minutiae information is regarded as a highly significant feature of Automated Fingerprint Recognition Systems (AFRS). The two main methods of minutiae feature extraction either require the grey-scale image to be converted to a binary image or work directly on a raw or enhanced grey-scale image. In the binary image-based method, the binarization of the gray-scale image is the initial step. This requires each grey-scale pixel intensity value to be transformed to a binary intensity of black (0) or white (1) [12].

$$I(x, y) = \begin{cases} 1 & \text{if } (x, y) \geq t, \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Spurious minutiae can be introduced by dry skin, creases, feature detection algorithms, and other potential noise-causing agents. The general process of a fingerprint-matching algorithm is presented in

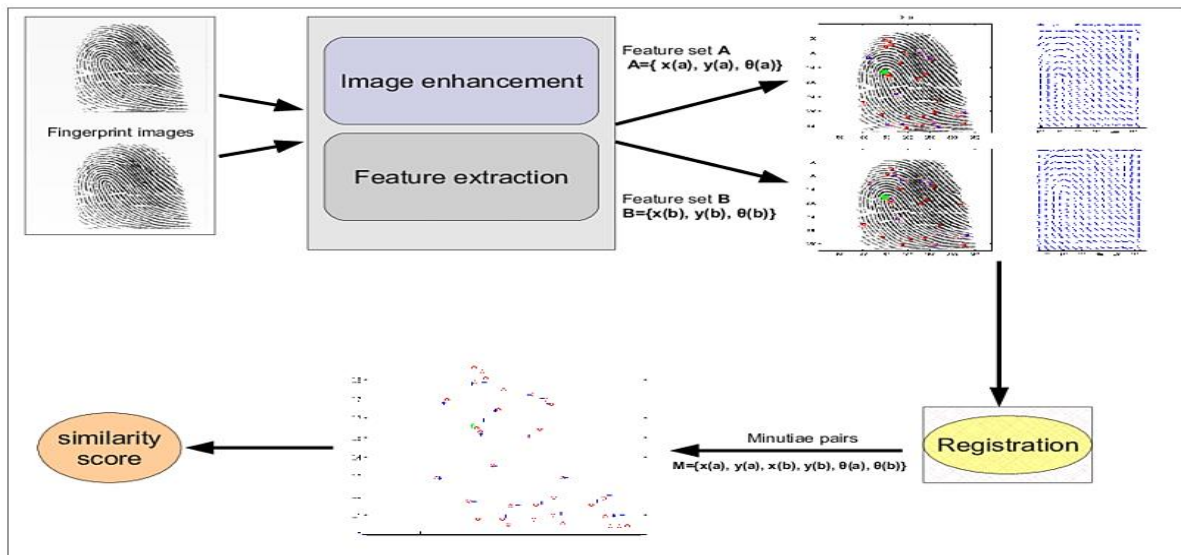


Figure 2. General processes for minutiae-based fingerprint matching[13].

In minutiae-based matching, minutiae are frequently represented in minutiae structures known as minutia triplets, where a minutia, m_i , is defined as $m_i = fx, y, qg$, where x, y represents the minutia's x-y position

and q the main ridge's angular orientation. One-to-one mapping or pairing of minutiae points from a test picture minutiae collection is the basic goal of minutiae-based matching.

$$A = \{m_{A1}, m_{A2} \dots \dots m_{Ap}\} \quad , \text{ where } m_{Ai} = \{x_{Ai}, y_{Ai}\} \text{ and } 1 \leq i \leq p \quad (2)$$

To a template image, a set of minutiae

$$B = \{m_{B1}, m_{B2} \dots m_{Bp}\} \text{ , where } m_{Bi} = \{x_{Bi}, y_{Bi}\} \text{ and } 1 \leq i \leq p \quad (3)$$

Forming the minutia pairs $(m_{AK}, m_{B\pi(k)})$ with $\pi(k)$ as the mapping permutation of pairs from set A to B.

Without some sort of pre-processing, we are unable to extract tiny pairs from triplets for the following reasons:

- The direction field in the triplet is worthless because fingerprint impressions might vary in their orientation.
- The x-y fields in the triplet are worthless due to the variability in offset between fingerprint impressions.
- When variable directional pressure is applied, skin elasticity causes non-linear distortion, or "warping," which results in triplet x-y variations. In general, the triplet structure cannot be used to speed up the search for minute pairs since it lacks invariant properties.

Global registration is necessary to address the problems with the triplet structure's lack of invariance. Global registration involves aligning and superimposing the template and test fingerprints so that corresponding regions have the smallest geometric separation from one another. Applying a heuristically directed affine transform to either the test or template fingerprint minutiae set can achieve registration geometrically, where the minutiae triplet field values are changed with

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{\Delta}) & -\sin(\theta_{\Delta}) \\ \sin(\theta_{\Delta}) & \cos(\theta_{\Delta}) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_{\Delta} \\ y_{\Delta} \end{bmatrix} \quad (4)$$

$$\theta_{new} = \theta - \theta_{\Delta} \quad (5)$$

Where θ_{Δ} is the orientation difference and (x_{Δ}, y_{Δ}) is the displacement difference in order to superimpose one fingerprint impression on top of the other with accurate overlap and uniform direction. Even with the introduction of substantial distortion, it is still predicted that, after alignment is attained, minutiae point within a fingerprint image would maintain their general global placement in reference to the majority of other minutiae points and other significant landmarks (such cores and deltas). To be more specific, even in distorted photos, the spatial distribution or geometric characteristics of nearby minutiae should alter only slightly. The global registration process significantly reduces the search space if we take into account that there are clear limitations in terms of minute landmarks relative to positioning variability (even with high distortion) and while taking into account that different fingerprint impressions have orientation and displacement differences. Since matching pairings are produced in smaller local neighborhoods (i.e., restrictions provided for minutiae mappings) once aligned, this reduces algorithm complexity for discovering minutiae pairs. This makes it possible to avoid using a crude brute force minutiae pairing technique. After the registration procedure, we can now generate geometric constraints, such as geometric distance, enabling the discovery of minute matching pairs.

$$dist_r(m_{Ai}, m_{Bi}) = \sqrt{(x_{Ai} - x_{Bi})^2 + (y_{Ai} - y_{Bi})^2} < r \delta \tag{6}$$

Alternatively, if we are comparing images obtained from different resolution scanners, we can adjust for scale differences.

$$dist_r(m_{Ai}, m_{Bi}) = \sqrt{(x_{Ai} - k_x x_{Bi})^2 + (y_{Ai} - k_y y_{Bi})^2} < r \delta \tag{7}$$

also, little angle differences.

$$dist_r(m_{Ai}, m_{Bi}) = \min(|\theta_{Ai} - \theta_{Bi}|, 360^\circ - |\theta_{Ai} - \theta_{Bi}|) < r_\theta . \tag{8}$$

The suggested technique in[14] finds a list of minutiae pairings before doing global registration, in contrast to most algorithms that perform local registration or minutiae pairing before global registration. Each minutia in the template and test fingerprints have an extended triplet structure defined as a 2-neighbourhood structure in the form of {x, y, θ, r1, θ1, r2, θ2}, where r1 and θ1 are the polar co-ordinates of the closest minutia, and likewise for the second closest minutia, r2 and q2. The list is then constructed by selecting couples from each alignment of a minute structure and assessing similarity. False pairs might be present in this initial collection of minutiae. The best registration is then determined using a least squares method using the biggest group of pairs that align using roughly the same transform parameters[15].

4.2. The encrypted key generation part

In this system, an encrypted key is generated based on the output of the fingerprint verification algorithm, which appears as a set of minutiae features. This array is converted into a string of numbers of a certain size and fed into the SHA256 function to give us a new string with a fixed length of 64 digests, equivalent to 256 bits in binary. This algorithm was used because it is one-way, meaning that the input text cannot be re-decrypted, and as a result, the original image of the data cannot be guessed, in addition to the advantage of data integrity. Where the output remains constant when the input remains unchanged. Any slight modification to the input values leads to a corresponding change in the output. This feature is useful for verifying the user's identity. It is necessary to realize that even a slight modification to the user's fingerprint will produce different outputs, which confirms the reliability of the system in verifying



identity. Fig. 3 explains the steps for generating encryption keys.

Figure 3. The cryptography key generation steps

SHA-256 is an iterated cryptographic hash function based on a compression function that updates the eight 32-bit state variables A, \dots, H according to the values of 16 32-bit words M_0, \dots, M_{15} of the message. The compression function consists of 64 identical steps as presented in Fig. 4. The step transformation employs the bitwise Boolean functions f_{MAJ} and f_{IF} , and two GF(2)-linear functions[16].

$$\Sigma_0(x) = ROTR2(x) \oplus ROTR13(x) \oplus ROTR22(x). \quad (9)$$

$$\Sigma_1(x) = ROTR6(x) \oplus ROTR11(x) \oplus ROTR25(x). \quad (10)$$

At each i -th step in the process, a fixed constant K_i and the i -th word W_i of the expanded message are utilised. where the input message undergoes padding and is then divided into 512-bit message blocks. Denoting the message expansion function as ME, it takes a vector M with 16 coordinates as input and produces a vector W with N coordinates as output. The coordinates W_i of the expanded vector is generated from the initial message M using the following formula:

$$W_i = \begin{cases} M_i & \text{for } 0 \leq i < 16 \\ \sigma_1(W_{i-2}) + W_{i+7} + \sigma_0(W_{i-15}) + W_{i+16} & \text{for } 16 \leq i < N \end{cases} \quad (11)$$

Taking a value for N different to 64 results in a step-reduced (or extended) variant of the hash function. The functions $\sigma_0(x)$ and $\sigma_1(x)$ are defined as follows:

$$\sigma_0(x) = ROTR7(x) \oplus ROTR18(x) \oplus SHR3(x) \quad (12)$$

$$\text{and } \sigma_1(x) = ROTR17(x) \oplus ROTR19(x) \oplus SHR10(x) \quad (13)$$

After several cycles of compression, the final hash is output, which is 256 bits. Fig. 4 shows one of the update cycles for the SHA256 state.

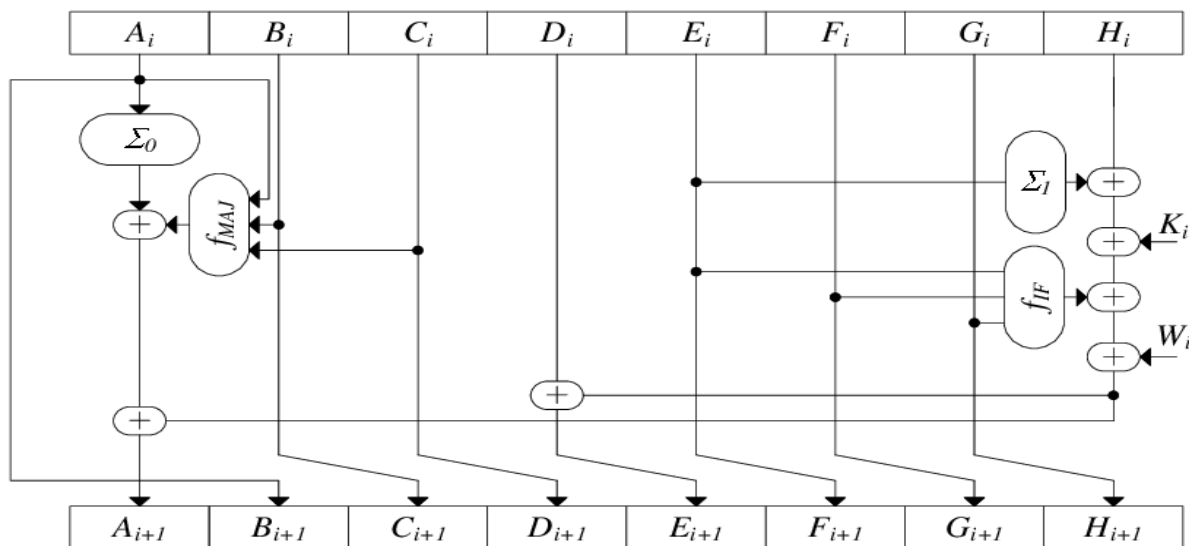


Figure 4. One step of the state update transformation of SHA-256

4.3. The security part of the encrypted key

The basic principle of the proposed system is the encrypted key protection part, where the encrypted key, once generated using fingerprint biometrics and hashing algorithm, is included in the first block and is represented as the transaction or data exchanged between users after it is encrypted again by the mining algorithm proof of work (POW) that it ensures that the block is validated and added in the appropriate sequence, as in the fig.5.

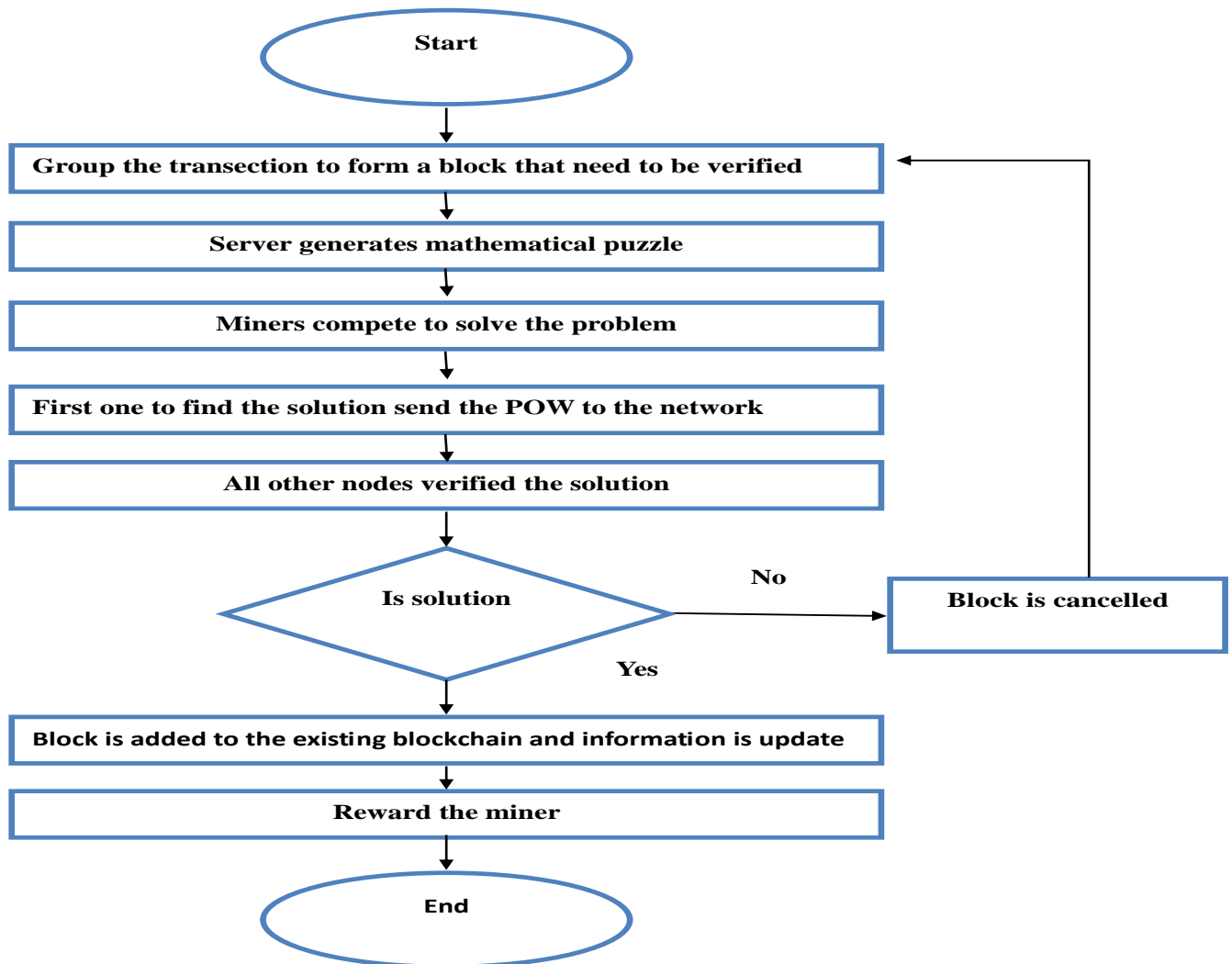


Figure 5. Flowchart of POW algorithm

4.3.1. Block component

In reality, a block is a data structure that may hold any kind of data. The data recorded in the majority of real-world BC applications is often transaction data. In MATLAB, a block can be represented by a class.

One block consists of several components:

- index
- The data that is entered
- Self-hash (The self-hash of the cryptographic key that is included in the transaction)
- Previous hash
- Nonce

The block class specification lists the following attributes: self hash (the block's own hash value), previous hash (the previous block's hash value), index (block number), data (transaction data), and random value (nonce). To implement the system correctly, at least two parameters must be available in the genesis block, which is (transaction data) and (index=0), and three parameters in subsequent blocks. Algorithm 1 explains how to create the block.

Algorithm (1): - Create block algorithm
Input: - Key, previous index, previous hash, and specified user.
Output: - Key, index, previous hash, self-hash and nonce.
Begin Step One: $index = previous\ index + 1;$ Step two: inter key as transaction data. Step three- Constructor ($obj = Block(index, key, previous\ Hash)$): Initialize the index, data, and optionally, the previous Hash of a new block. Automatically call the self-Mine method to mine the block when it is created. Step four: Mine Block ($mine\ Block(difficulty)$): Implement proof-of-work mining. Define a mining target based on the specified difficulty level (e.g., a string with leading zeros). Generate random nonce values and calculate the hash of the block repeatedly until the hash meets the mining target. Step five: Calculate Hash: Calculate the hash of the block based on its properties, including index, data, previous hash, and nonce Use a cryptographic hash function (SHA-256) to compute the hash. Step six: shear block and key with the specified user. Step seven: shear block with all users. End Algorithm.

4.3.2. Hash algorithm and mining

For ease of understanding, a hash algorithm can be thought of as a mapping method that converts a variable-length string of characters into another string of fixed length.

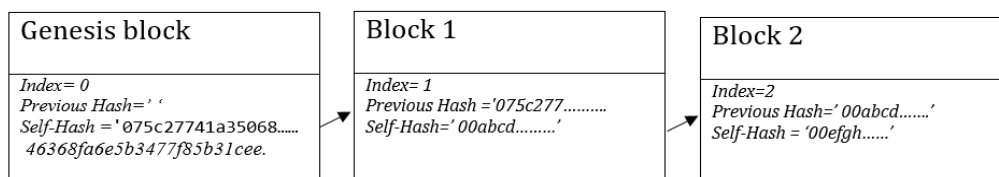
Function output=hash operation (input)

4.3.3. connecting block objects into chains

To establish a connection for newly created blocks initiated by the first user, it is necessary for the underlying building block to exist inherently. This initial block has a previous hash value of null and is set as a pointer with a value of 0. The current hash is the key generated using the fingerprint, which consists of 256 bits. As for the block created by the first user with an index of 1, it carries a previous hash equivalent to the hash value with an index of 0, the aforementioned 256-bit key. The current block value starts with "00" and includes a randomly generated crypto number. To be integrated into the blockchain, this block must successfully undergo a proof-of-work (POW) test, a mathematical puzzle that requires verification of the solution. This process is explained further in Algorithm 2.

Algorithm (2): - implementing a blockchain
Input: - the new block that was created.
Output: - Interconnected blockchain
<p>Begin</p> <p>Step 1: Create a Blockchain Class: Create a class called Blockchain that extends the handle class.</p> <p>Step 2: Define Properties: Define two properties within the Blockchain class: <i>total Count:</i> To keep track of the total number of blocks. <i>block Array:</i> To store the blocks in the blockchain.</p> <p>Step 3: Constructor: Create a constructor method for the Blockchain class. Initialize block Array with a single block called the "Genesis Block." Set the total count to 1. Calculate the Genesis block hash using SHA-256 encryption</p> <p>Step 4: Get the Latest Block: Create a method to get the latest block in the blockchain (get Latest).</p> <p>Step 5: Calculate Genesis Block Hash: Create a method to calculate the hash of the Genesis Block (calculate Genesis Block Hash) by using SHA-256. Combine the block's index and data as a string and hash it. Update the self-hash property of the Genesis Block with the calculated hash.</p> <p>Step 6: Add a New Block: Create a method to add a new block to the blockchain (add Block). Validate the new block using the validate new Block method. If the new block is valid, add it to the blockchain.</p> <p>Step 7: Validate a New Block: Create a method to validate a new block (validate new Block). Calculate the hash of the new block's combined data and nonce. Check if the first two characters of the calculated hash are "00" (for simplicity) and compare it with the block's stored hash. Return true if the block is valid; otherwise, return false.</p> <p>End Algorithm.</p>

The "chain" is the self-hash data of the preceding block, which is kept in the previous hash attribute of the following block. block 1 thus contains the information of block 0, and block 2 contains the information of block 1. information, so block 2 also contains Cluster information 0. Fig.6 shows how the blocks are



interconnected.

Figure 6. BC interconnected

5. Results

This section will provide an overview of all the experimental results in this paper, including testing of system access restriction, cryptographic results, and security evaluations of the system design compared to previous work.

5.1. Test the application's accessibility

Users' fingerprints underwent extensive testing, with each of the 10 users' fingerprints tested in 10 different positions. The results of the matching process were found to be 100% accurate, indicating that the system achieved an exceptionally high success rate in accurately matching fingerprints. see table 2.

$$FAR = FA / TA \quad (14)$$

$$FRR = FR / TA \quad (15)$$

Table 2. The results of FAR and FRR







Input image \ person sample	Number of images used for Recognition	Number of images recognized correctly	FRR to 10 Samples for the same Person	FAR to all different fingerprints in a database
A (1-10)	10	10	0	0
B (1-10)	10	10	0	0
C (1-10)	10	10	0	0
D (1-10)	10	10	0	0
E (1-10)	10	10	0	0
F (1-10)	10	10	0	0
G (1-10)	10	10	0	0
H (1-10)	10	10	0	0
I (1-10)	10	10	0	0
J (1-10)	10	10	0	0
Overall	100	100	0	0

The zero values in the false acceptance rate (FAR) and false rejection rate (FRR) fields signify that there were no instances of false acceptance or rejection. All users were successfully identified, and there were no recorded errors in permitting unauthorized users to access the application. Similarly, there were no instances of failing to recognize an authorized user.

5.2. Results of the proposed system's cryptography

This section examines the proposed system's encryption process, from the creation of keys based on biometrics to the block creation and mining phases. As shown in the table 3.

Table 3. Results of the proposed system's cryptography

ID	Users	Users' fingerprints	Results	
			Index	
1.1	A		Index	1
			Data	E4276DA40EBE7D2BD057A3E5F562B65BE6C89B8AD95A733DACBAD4B986025070
			Previous hash	075c27741a3506846368fa6e5b3477f85b31ceee71a5716e2f12b40fa21d23aa
			Self-hash	002baa43ef3c188490594b0e30f1adcb
			Nonce	645
2.1	B		Index	2
			Data	C8895DE22357F4E65186E431DEED5F41E078EDCA5CF7331FEF2E926E001E9744
			Previous hash	002baa43ef3c188490594b0e30f1adcb
			Self-hash	007d0bc057bc5524ff8aade14ccc7da1
			Nonce	476
3.1	C		Index	3
			Data	C143BD57A8EE37D8D631738E9B56EB0B87044E2D433E78B0CB2F7ED1B2A980AD
			Previous hash	007d0bc057bc5524ff8aade14ccc7da1
			Self-hash	007a5a17466ac6e97a4543f2e00dd540
			Nonce	455
4.1	D		Index	4
			Data	07CA9BEB6616924240AEF08F5021987658FA98219A74A479A942E0E20663B969
			Previous hash	007a5a17466ac6e97a4543f2e00dd540
			Self-hash	005ebda340d42868942245418699cec2
			Nonce	16
5.1	E		Index	5
			Data	5228299BC0809E7D9B1B01CD521836D1B60FFFA32DF2B0500F050978ECEA97D
			Previous hash	005ebda340d42868942245418699cec2
			Self-hash	005c9212e41ea715b1bec6dece8680f7
			Nonce	108
6.1	F		Index	6
			Data	50BBF4C4F3593FA6A575BC6B02B66E8060D49AE37E9A978C7C2D66CF8A8BA766
			Previous hash	00c7b013fe24705d46dbf55b8ad3c776
			Self-hash	00f6a5eca4c187aa465841e3d4a45cd5
			Nonce	382
7.1	G		Index	7
			Data	60421B8897D321BD795BC5A0E84C8A42C898FED38406DDD6D545CB1EF971CBCB
			Previous hash	00f6a5eca4c187aa465841e3d4a45cd5
			Self-hash	002f940f075ccce15935a6dc99ae2968
			Nonce	1234
8.1	H		Index	8
			Data	DEBF6BF8FF646211D23AF0A2E09E76CD774CC38FACB472010D6D7AEE0E7710AE
			Previous hash	002f940f075ccce15935a6dc99ae2968
			Self-hash	001b6f05523702558781a1fb1978ca77
			Nonce	639
9.1	I		Index	9
			Data	9908BBFD4C60C0CF3F3EE0D266C4B4D7C44DE3373791C9BEEFC076F1D0A7350E
			Previous hash	001b6f05523702558781a1fb1978ca77
			Self-hash	0021b045d8fe3c873738173a56e3ab47
			Nonce	155
10.1	J		Index	10
			Data	9CBFFCA22EADEDD54920C1F7B987509294B3C7C336530D19C3381795286FB426
			Previous hash	0021b045d8fe3c873738173a56e3ab47
			Self-hash	004f247e4f2a181d2356fed590c94ae0
			Nonce	67

In Table 2, the results of encrypting the keys generated through the fingerprint features and the sh256 function are considered the first block, which is the first block that the first user enters and which carries the first encrypted key, given that there is a virtual block preceding it, called the genesis block, which takes an index value = 0 and previous hash =null, and the self hash value is equal to the key that was generated by the user. Therefore, the previous hash of block 1 was considered the value of the 64-digest key that was generated from the SHA256 function. The current block is considered a unique hash of this key consisting of 32 bytes in addition to a random encryption number, and the encryption continues. This way for the rest of the blocks.

5.3. The experimental results with POW

This section presents the results obtained from applying the POW algorithm that was used in the block mining process. Different results were obtained in (Nonce), which represents the number of attempts to obtain the required hash in addition to the time seconds taken to add each block and the total execution time. The results can be summarized in the table 4.

Table 4. POW's, Time results

Block. No	Nonce	Time to add one block/s
1	645	0.1488
2	476	0.2612
3	455	0.4147
4	16	0.3997
5	108	0.4234
6	382	0.1852
7	1234	0.1982
8	639	0.2902
9	155	0.1995
10	67	0.1770

Fig.7 and fig.8 show a diagram of the attempts to obtain the correct hash and the time required to hash each block.

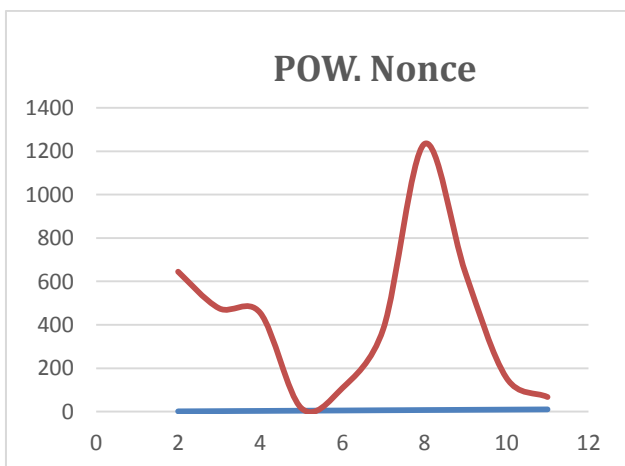


Figure 7. POW's Nonce

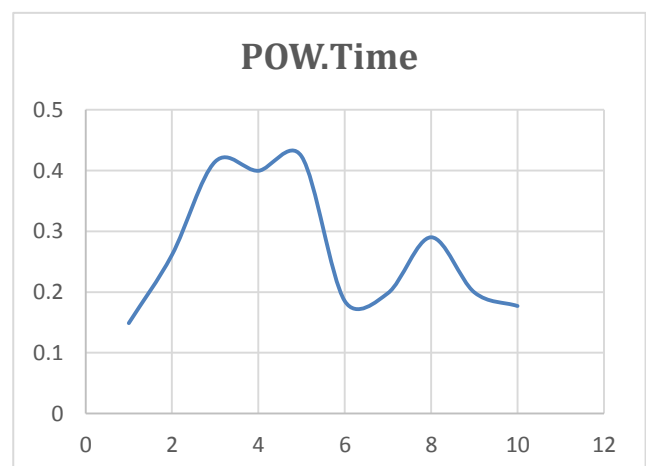


Figure 8. POW's Time

5.4. Comparing the proposed system with other studies

In this section, the proposed thesis is compared to other studies concerning the security requirements offered by BC technology, including various management techniques such as the handling of encrypted keys. Across all studies, there is a common utilization of the secure features inherent in BC technology to enhance protection and credibility, while also mitigating the risks associated with a centralized point of failure, which can expose user data.

In reference [19], BC technology is utilized to support identity-based systems through hash functions, resulting in a more expansive and powerful system. However, challenges related to privacy and large storage capacity persist. Reference [14], incorporates BC technology in conjunction with LORaWAN networks to connect and manage peripheral devices in the Internet of Things (IoT).

This application enhances network security and availability, eliminates central points of failure, and ensures secure storage. The potential drawback lies in the possible cost escalation due to the increasing number of peripheral devices and the demand for high storage capacity. In reference [18], BC technology is leveraged for key management, capitalizing on its attributes of openness, stability, traceability, and error tolerance. Keys are generated through hashing, ensuring a secure and stable KMS. Reference [84], focuses on using BC technology to support ad hoc vehicle networks. Here, a unique session key is provided for each vehicle, with real-time key generation and membership arrangement. Authentication relies on encryption without the need for a key certificate.

The proposed thesis excels in storage and authentication, employing fingerprints as the primary method and digital signature encryption as the secondary one. It ensures complete key privacy by generating keys biometrically and distributing them transparently. However, a limitation is noted in terms of availability, as the key is only accessible when both the sender and receiver are connected to the network. Table 5 shows the comparison of the proposed system with previous studies in terms of security requirements.

Table 5. Comparison of our system with other studies

Security requirements	References				
	[9]	[6]	[19]	[20]	Our system
Integrity	✓	✓	✓	X	✓
Scalability	✓	X	✓	✓	✓
Confidentiality	✓	✓	✓	✓	✓
Availability	X	✓	X	X	X
Verification	✓	✓	✓	✓	✓
Authentication	✓	✓	✓	✓	✓
Authorized	✓	✓	X	X	✓
User control	✓	✓	X	✓	✓
Non-repudiation	✓	✓	✓	✓	✓
Decentralized	✓	✓	✓	X	✓
Privacy	X	✓	✓	✓	✓
Reduce storage	x	X	x	X	✓

6. Conclusion

This paper presents a cryptographic KMS that integrates several advanced techniques to securely generate and distribute cryptographic keys. Access to the system is controlled through a combination of email authentication, password verification, and additional fingerprint authentication. Once the user's identity is successfully verified, he or she can log into the system. The system ensures a secure storage environment by creating uniquely encrypted keys for each user, rather than storing them. This method facilitates the direct generation of encrypted keys for users while enhancing the resistance of the database to unauthorized access or tampering. The system uses the SHA256 algorithm to generate 256-bit encrypted keys based on each user's unique fingerprint features. These fingerprint features are converted into encrypted keys by entering them into a SHA256 function. This approach not only maintains data integrity, but also allows users to regenerate the same key as long as they enter the same fingerprint. Furthermore, the system facilitates the distribution of encrypted keys among n users by leveraging BC technology. The encryption key is incorporated into the transaction, and then encrypted again. The block is validated by the mining algorithm, after which it is added to the BC. Security analysis of the proposed system reveals strong security performance and scalability, which significantly reduces the cost of onboarding new users to the system. The system's security is enhanced by its reliance on encryption rather than trust in a single entity, making it highly resilient to breaches. In addition, it has proven effective in generating and distributing encrypted keys compared to other studies in this field.

7. Reference

- [1] S. S. Chaeikar, M. Alizadeh, M. H. Tadayon, and A. Jolfaei, "An intelligent cryptographic key management model for secure communications in distributed industrial intelligent systems," *Int. J. Intell. Syst.*, vol. 37, no. 12, pp. 10158–10171, 2022.
- [2] T. W. van der Schaaf, "a Framework for Designing Near Miss Management Systems," *Near Miss Report. As a Saf. Tool*, no. April, pp. 27–34, 1991, doi: 10.1016/b978-0-7506-1178-7.50007-1.
- [3] M. A. Engelhardt, "Hitching healthcare to the chain: An introduction to blockchain technology in the healthcare sector," *Technol. Innov. Manag. Rev.*, vol. 7, no. 10, 2017.
- [4] C. M. Bruhner, O. Linnarsson, M. Nemeč, M. Arlitt, and N. Carlsson, "Changing of the guards: Certificate and public key management on the internet," in *International Conference on Passive and Active Network Measurement*, Springer, 2022, pp. 50–80.
- [5] S. Barman, S. Chattopadhyay, and D. Samanta, "An approach to cryptographic key distribution through fingerprint based key distribution center," *Proc. 2014 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2014*, no. November, pp. 1629–1635, 2014, doi: 10.1109/ICACCI.2014.6968299.
- [6] V. Ribeiro, R. Holanda, A. Ramos, and J. J. P. C. Rodrigues, "Enhancing key management in LoRaWAN with permissioned blockchain," *Sensors*, vol. 20, no. 11, p. 3068, 2020.

- [7] C. Jia, H. Ding, C. Zhang, and X. Zhang, "Design of a dynamic key management plan for intelligent building energy management system based on wireless sensor network and blockchain technology," *Alexandria Eng. J.*, vol. 60, no. 1, pp. 337–346, 2021, doi: 10.1016/j.aej.2020.08.019.
- [8] O. Pal, B. Alam, V. Thakur, and S. Singh, "Key management for blockchain technology," *ICT express*, vol. 7, no. 1, pp. 76–80, 2021.
- [9] R. Fotohi and F. S. Aliee, "Securing communication between things using blockchain technology based on authentication and SHA-256 to improving scalability in large-scale IoT," *Comput. Networks*, vol. 197, p. 108331, 2021.
- [10] Q. Liu, L. Luo, J. Wang, W. Li, R. Liu, and M. Yu, "Key management scheme of distributed IoT devices based on blockchains," *IET Commun.*, 2023.
- [11] M. J. Mihaljević, M. Knežević, D. Urošević, L. Wang, and S. Xu, "An Approach for Blockchain and Symmetric Keys Broadcast Encryption Based Access Control in IoT," *Symmetry (Basel)*, vol. 15, no. 2, p. 299, 2023.
- [12] Q. Xiao and H. Raafat, "Fingerprint image postprocessing: a combined statistical and structural approach," *Pattern Recognit.*, vol. 24, no. 10, pp. 985–992, 1991.
- [13] J. Abraham, P. Kwan, and J. Gao, "Fingerprint Matching using A Hybrid Shape and Orientation Descriptor," *State art Biometrics*, no. June 2014, 2011, doi: 10.5772/19105.
- [14] A. M. Bazen and S. H. Gerez, "Fingerprint matching by thin-plate spline modelling of elastic deformations," *Pattern Recognit.*, vol. 36, no. 8, pp. 1859–1867, 2003.
- [15] P. Kwan and J. Gao, "Fingerprint matching using a hybrid shape and orientation descriptor Abraham, Joshua," *State art Biometrics*, pp. 25–56, 2011.
- [16] J. Hilton, "The Hex Factor: The NIST Hash Function Competition".
- [17] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*, Springer, 1987, pp. 369–378.
- [18] A. Yadav, "Comprehensive Study on Incorporation of Blockchain Technology With IoT Enterprises," 2021, pp. 22–33. doi: 10.4018/978-1-7998-3295-9.ch002.
- [19] S. S. Panda, D. Jena, B. K. Mohanta, S. Ramasubbareddy, M. Daneshmand, and A. H. Gandomi, "Authentication and Key Management in Distributed IoT Using Blockchain Technology," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12947–12954, 2021, doi: 10.1109/JIOT.2021.3063806.
- [20] H. Tan and I. Chung, "Secure authentication and key management with blockchain in VANETs," *IEEE access*, vol. 8, pp. 2482–2498, 2019.