Al-Khwarizmi Engineering Journal
ISSN (printed): 1818 – 1171, ISSN (online): 2312 – 0789
Vol. 21, No. 4, December, (2025), pp. 45- 64

Rabbit-256 Optimisation for Secure Blockchain Hashing in IoT-Healthcare Data

Khalid Jamal Jadaa[1*], Aymen Mudheher Badr[2], Waleed Noori Hussein[3], and Latifah Munirah Kamarudin[4]

[1] *Department of Computer Engineering, College of Engineering, University of Diyala, Diyala, Iraq*
[2] *College of Medicine, University of Diyala, Diyala, Iraq*
[3] *Al-Zahraa College of Medicine, University of Basrah, Basrah, Iraq*
[4] *Centre of Excellence for Advanced Sensor Technology (CEASTech), Perlis, Malaysia*
[4] *Department of Computer and Communication Engineering, University Malaysia Perlis, Perlis, Malaysia*
*Corresponding Author's E-mail: kalid.jamal.jadaa@gmail.com

(Received 8 July 2025; Revised 5 October 2025; Accepted 27 October 2025; Published 1 December 2025)
https://doi.org/10.22153/kej.2025.10.001

Abstract

The recent trend towards the use of a blockchain as a means to guarantee the security of health data has raised concerns with regard to its applicability in Internet of Things (IoT) scenarios due to computationally heavy primitives (e.g. hashing functions) and lack of scalability. As a solution to this problem, this article introduces Rabbit-256: an addition–rotation–XOR-based sponge construction derived from the Rabbit stream cipher that is twisted and adapted to a lightweight hash function, suitably adapted for distributed solutions in healthcare systems with a blockchain nature. Rabbit-256 is a lightweight encryption cipher that wears the mask of a hash function but with better diffusion and avalanche through an official buildup in Merkle trees. The presented system is evaluated using common cryptographic measures against SHA-256, i.e. grid operators of 100, 500, and 1000 inputs for the avalanche effect, Hamming distance, and mean standard deviation. We observe that Rabbit-256 exhibits a higher security margin and lower computational overhead, and thus, it is an optimal alternative to resource-constrained IoT systems given its resistance against attacks. Although the current work is developed in simulation, Rabbit-256 can be utilised for actual deployment to ensure the privacy of e-health records and medical sensor data in IoT and clinical services over a blockchain. In the future, we will focus on hardware design, energy efficiency, and integration (i.e. to be compliant with the Health Insurance Portability and Accountability Act in the U.S. and the General Data Protection Regulation in Europe).

Keywords: *healthcare; hashing; medical internet of things; security; cloud computing; blockchain*

1. Introduction

The dispersion of processing power worldwide has enabled the development of various technologies, including blockchain technology [1]. Examples include coins, such as bitcoins, which are recorded publicly and temporarily. The most interesting aspect of this research is the public element; that is, anyone in the world can download the code and either 'mine' for bitcoins or participate in new network concepts created on the Ethereum platform [2]. As blockchain technology rapidly develops, it is currently being implemented in various medical data scenarios [3], personal data protection, and data allocation systems. It is even used to stimulate the inclusion of renewable energy sources in power grids, which is beyond its original application in the scope of cryptocurrency. Thus, reducing emissions in the global shipping industry enables banks to process transfers faster at lower costs [4].

The cryptographic algorithm (Rabbit 128/128 bit) is the foundation of the blockchain hash algorithm proposed in the current work. This highly efficient algorithm is used to model the blockchain hash algorithm design, providing resilient security and integrity of data. Furthermore, high speed can be achieved for the purpose of data transfer. This scheme can truly improve the security and efficiency of data processing in a blockchain.

Through the use of the Internet of things (IoT) technology in health systems, patients can be remotely diagnosed. Their health can be analysed online and even remotely observed in real time. However, major hurdles exist for the entire digital transformation, including privacy management. As a novel technology, blockchain provides a possible solution to the aforementioned problems by authenticating and securely storing data. However, the mainstream use of blockchain necessitates costly commands, such as hashing functions (specifically SHA-256) and cryptographic primitives, which require heavy computations. This situation is the reason why we will likely not see the use of blockchain technology in low-power and similar devices within the IoT paradigm.

The current article primarily discusses the aforementioned limitations and attempts to address them. A lightweight hash function based on the Rabbit stream cipher is included. In this article, we demonstrate a hardware-efficient (and easily scalable) Rabbit-256 function. The following research questions are posed:

1. Could Rabbit-256 be used as an encryption that is sufficiently strong, and perhaps, less computationally expensive, and with a lower power requirement than SHA-256 for technologies such as IoT?
2. How does Rabbit-256 function when subjected to key cryptographic properties, such as the avalanche effect and Hamming distance (HD)?
3. What are the implications of blockchain-based healthcare systems on scalability and regulations?

To resolve these problems, the research presented here makes the following contributions:

1. A new adaptation of the Rabbit stream cipher for encryption in blockchain security.
2. Experimental evaluation with different sizes of datasets
3. Promotion of the light encryption method for a blockchain-based healthcare system with a holistic solution

The current work contributes a step towards paving the way for the implementation of improved device applications in the future, bridging the gap between the limitations of IoT devices and blockchain security in healthcare.

1.1. Immutability and compatibility concepts

Although several features are associated with blockchain security, the two most critical ones are consensus and immutability [5]. In a distributed blockchain network, the most critical feature is the capability of nodes to agree on the authentic state of a network and the validity of transactions. Consensus algorithms are basically used to reach consensus. [6].

For a clear explanation, consensus is the agreement of nodes on the actual state of a network and the legitimacy of transactions; meanwhile, immutability is the capability of blockchains to block changes made to valid transactions. Although cryptocurrencies are the most widely used technology at present, they can also be utilised for other digital data that have no connection to financial transactions. In blockchain networks, immutability and consensus work together to produce a data protection system. One of the key tasks of a consensus algorithm is to ensure that all involved parties agree on the current state of a network and the rules of a system are strictly followed. After simultaneously verifying the validity of each new dataset, the role of immutability ensures the integrity of data and transaction records [7].

1.2. Use of cryptography to secure blockchains

One of the key factors for ensuring the security of blockchain networks is the adoption of cryptographic hashing operations. Hashing is the process of generating output values with a fixed length and not considering the size of the input. The result of hash changes in accordance with changes that occur in the input data, and the data remain the same if the input data do not change [8].

In a blockchain, data blocks are unambiguously identified as unique identifiers based on their hash, which is generated for each block built on the hash of the previous block. This structure is called a blockchain. A blockchain confirms transactions to the rest of the network. If block data are modified, then the hash value of this block should also be changed to preserve the immutability principle (integrity) of a blockchain [9].

Consensus algorithms, such as proof of work in bitcoins, use hashing to check the validity of transactions and for mining. The SHA-256 function (a hash of 256 bits) is commonly employed [10]. In addition, cryptography is used in the preservation

of transaction information and crypto wallets with asymmetric encryption. Users can receive and transmit payments by using public key/private key pairs [11]. The ownership of transferred coins is verified using digital signatures that are created with private keys. Wallet balances are protected through authentication by using an asymmetrically encrypted private key.

The remainder of this paper is organised as follows. Section 2 provides a quick overview of related background and literature on blockchain security and healthcare data management. The proposed Rabbit-256-based optimisation algorithm for blockchain hashing is described in Section 3, including its detailed design and implementation procedure. The description of the modified Rabbit-256 algorithm is provided in Section 4, whilst Sections 5 and 6 respectively present the experiment setup and discuss the results obtained by the performance evaluation and security analysis of the proposed algorithm. Finally, Section 7 concludes the article with a synthesis of the key findings, contributions and future research directions.

2. Related Work

2.1. Blockchain technique based on IoT

Blockchain technology is considered a proposed solution to the security and privacy issues in IoT networks. Blocks can be distributed amongst devices, providing provable techniques for the future. The influences of blockchain technology and cryptocurrency on the development of IT in society are the subject of prospective research. Such research is crucial because several stakeholders, including the United Nations [12], should start dealing with these technologies to understand how they work and learn from them. Block-based IoT schemes proposed by academicians exhibit the potential for effectively incorporating resource-constrained IoT devices into a blockchain.

In [13], a decentralised access control design for IoT based on a blockchain was introduced with the capability to accommodate numerous devices. Involving IoT devices in a blockchain network is challenging due to constrained resources. Instead, a management hub ‘talks’ to a blockchain network for these devices. In [14], the authors provided a collaborative mining network to deal with the restricted communication and computational requirements of mobile IoT devices. They used resources that remained free beyond mining devices and cloud–edge to perform some tasks related to exploitation in mobile blockchains. In [15], a decentralised capability-based access control architecture, called IoT-consortium capability-based access control, for IoT consortium networks was presented. This structure adopted a blockchain database for high throughput performance, overcoming conflicts from data leakage and failure of centralised processing systems.

In [16], a collaborative computing architecture was introduced to satisfy the quantum computing requirements of a blockchain-enabled IoT. It consists of computer servers that are virtualised in numerous data access points to form a resource pool with elasticity. Data are collected based on block size, and a correct nonce is created using blockchain calculation. Security is further achieved through a cloud cache-based storage of the block, whilst adding its hash value to the blockchain.

2.2. Hash function with a blockchain

The significance and roles of hashes in the blockchain architecture have been recognised. Here, we present studies related to this subject. In [17], the authors presented a robust but straightforward hashing mechanism that could be used with a blockchain to safeguard the confidential data of healthcare Internet of medical things systems. This overall process reduces energy utilisation and computational requirements, thus; it is useful for medical devices with restricted resources in contrast with the traditional hash reporter. The test results indicated good avalanche effects, unpredictability and anti-attack performance. The proposed mechanism is efficient and reliable for applications in healthcare IoT.

Accordingly, we introduce a novel hash function that uses a genetic algorithm to improve data integrity for blockchain-based healthcare systems. The researchers developed the genetic algorithm-based hashing technique (GAHBT) for health data categorisation and preservation. This technique provides more robustness against data collision and higher randomness than the conventional method. Studies have confirmed that the GAHBT scheme provides progressive data security and is less affected by common cryptographic attacks. The paper Clarion recognises this idea as a successful solution to protecting patient data in blockchain healthcare systems and discusses how it can be employed in the scalable management of healthcare data. [18]. To guarantee transparency and data accuracy whilst eliminating intermediaries, a document’s hash information and transactions are registered in a blockchain.

In [19], the Edwards-curve digital signature algorithm (EdDSA) was used in a secure and strong pseudorandom number generator by comparing it with the elliptic curve digital signature algorithm. Accordingly, researchers have proposed adopting EdDSA to generate hash functions in transactions.

In [20], the researchers suggested using Chameleon hash functions to modify one block in a blockchain without affecting others. They employed multiparty computing to recover a shared trapdoor key for block debugging. Revision occurs when lead auditors digitally sign proposed modifications, eliminating the need for a trusted party. Another trapdoor switch has been proposed to prevent block revision without the creator's permission. In [21], the researchers introduced policy-based Chameleon segmentation with black box accountability (PCHBA), which enables the attribute authority to link updated transactions with accountable transaction rates by using black box accountability. Public users can use the access device/black box to identify these rates. This previous study provides a foundation for PCHBA.

Recent research has significantly enhanced the body of literature on the use of lightweight cryptography (LWC) in blockchain systems for the healthcare sector. The researchers conducted experiments on lightweight algorithms with miniature-sized microcontrollers, verifying the implications of security over efficiency trade-offs on IoT devices [22]. Meanwhile, the current research performed a more precise analysis of lightweight cryptographic programmes and identified metrics for validation, such as HD, the avalanche effect and bit independence [23]. This line of work gave rise to a recent hash function based on the SPECK cipher and demonstrated that cipher reuse could provide us with secure yet efficient hashing, albeit its relation with a blockchain was not trivial [24]. Although the aforementioned researchers examined blockchain in healthcare applications, they pointed out that regulatory barriers and scale and interoperability challenges were two issues points [25] [26]. In contrast, tailored lightweight hash also functions for IoT, but in the sense of a trade-off between energy saving and resistance against possible system access [27]. The aforementioned studies suggest the critical requirement for the development of hashing schemes similar to Rabbit-256 that maintain a potential for lightweight efficiency and simultaneously support wide cryptographic strength in addition to blockchain applications in healthcare.

3. Proposed System

3.1. LWC algorithms

The LWC project, launched in April 2018 by the U.S. National Institute of Standards and Technology (NIST), aims to design cryptographic algorithms for resource-constrained devices. The current work was inspired by the growing IoT, in which effective and secure communication between devices must be assured for new emerging applications, such as autonomous cars and smart grid operations [28].

Lightweight encryption, also known as LWC, is a form of encryption designed for devices with limited resources. To offer secure solutions for network-constrained resources, lightweight encryption technology employs less memory, fewer computational resources and lower electricity consumption.

AES and SHA are unsuitable for resource-constrained IoT environments because they require excessive computational resources [29]. To address this issue, low-power cryptographic devices have been developed for IoT/radio-frequency identification devices. International and NIST groups have defined techniques for LWC that are specifically designed for low-resource systems. Figure 1 illustrates a classification of simple cryptography algorithms.

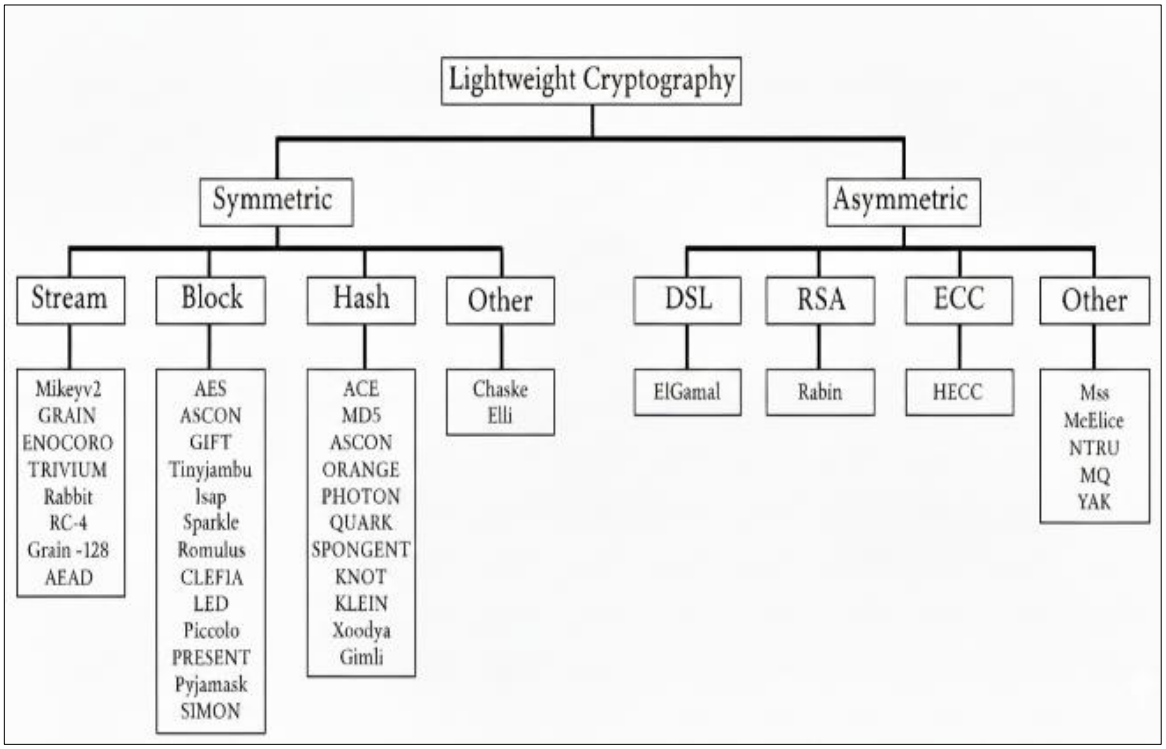

Fig. 2. LWC Algorithms

3.2. Rabbit algorithm

The Rabbit algorithm, which enables a powerful nonlinear mixing of the inner state between two repetitions, is particularly compact for encrypting and decrypting sensitive messages. The Rabbit algorithm was introduced in 2003. One of the earliest attempts at LWC was this algorithm. It utilises 0.18 nm complementary metal–oxide–semiconductor technology and 3800 GE [30]. It employs the original messages shown in Figure 2, combined with a secret key of 128 bits for encryption. The files, which are exchanged amongst authorised users, are encrypted and decrypted using the keys.

Each cycle uses 128 randomly chosen internal state bits to create the output block by converting plaintext into ciphertext, and vice versa, using the XOR technique during encryption and decryption. Eight counters with 32 bits each, one state variable with 32 bits and one carry counter bit make up the 513 bits that comprise internal state size. State variables are updated using the paired nonlinear octet function. Situational variables are the minimum duration of the time promised by the counters.

The basic objective of this technique is to encode 128 bits of data every iteration and generate the cipher as a big stream. The strength of encryption depends on the robust mixing of internal states over two successive repetitions. The mixing function uses the g-function related to arithmetical squaring, XOR, a bitwise rotation and modulo 2 additions.

The Rabbit-256 algorithm has been preferred over other lightweight algorithms for blockchain applications, especially in healthcare, because it offers strong security, superior avalanche properties and lightweight performance, making it ideal for the current research. Table 1 provides a comparison between Rabbit-256 and other lightweight algorithms (SHA-256, AES, SPECK, SIMON, PRESENT) in terms of computational efficiency, memory efficiency, power usage, avalanche effect, bit independence, HD, suitability for IoT and blockchain integration [31]-[34].

Table 1,
Comparison of Rabbit-256 with other LWC algorithms

Feature	**Rabbit-256 (our work)**	**SHA-256 (our work)**	**AES (128 bits) [Daemen & Rijmen 2013]**	**Other lightweight ciphers (SPECK, SIMON, PRESENT) [Beaulieu et al. 2015; Bogdanov et al. 2007; El-Hajj et al. 2023]**
Computational Efficiency	High	Moderate	Moderate	Variable (optimised for IoT)
Memory and Power Usage	Low	High	Moderate	Low
Avalanche Effect	≈58% (strong)	≈50%	Not considered for hashing	Moderate
Bit Independence	Strong	Strong but slower	N/A	Varies; some ciphers are weak
HD	Higher than SHA-256	Lower than Rabbit-256	N/A	Lower
Suitability for IoT	Excellent	Poor	Limited	Good
Blockchain Integration	Merkle tree + blockchain	Heavy for IoT	Non-straight applicable	Not widely tried in blockchain

3.3. Key Management System (KMS)

In a cryptosystem, managing cryptographic keys is referred to as key management, which encompasses handling key creation, transfer, storage, usage, crypto-shredding (destruction) and replacement [35]. Figure 2 illustrates the basic function of the Rabbit algorithm.

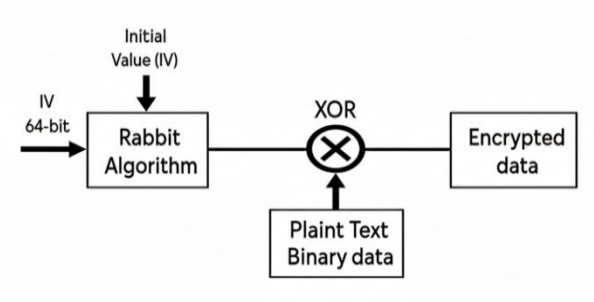

Fig. 2. Basic Function of the Rabbit Algorithm

The exchange of keys is less problematic at present because of the development of public key cryptography in the 1970s. The danger of key leakage during distribution has significantly decreased since the Diffie–Hellman key exchange protocol was developed in 1975. This protocol made exchanging a key across an unsecured communication channel feasible. Key indications can be linked to an encrypted communication as clear text by using a method that is analogous to a book code. This form of encryption was utilised by Richard Sorge's code clerk; it was a code that referred to a page in a statistics handbook [36]. The symmetric encryption key used by the German army during World War 2 was a mixed type; it was composed of a privately disseminated key schedule component and a user-selected session key component for each transmission.

KMS is a systematic approach for generating, distributing and preserving cryptographic keys for hardware and software. It consists of client-side capacities for storing and managing keys, and back end functionality for the generation and distribution of keys [37]. Key management is fundamental to the security of cryptosystems, and it has social engineering components, such as system policy, user education and organisation coordination [38].

3.4. Merkle trees

A Merkle tree is a perfect binary tree with an associated hash function and an assignment function [39]. Merkle trees (hash trees, authentication trees) are data structures used in cryptography and computer science for the purpose of efficiently checking large volumes of data for changes or consistency within a distributed system [40]. The nodes of Merkle trees are essentially their leaves and inner nodes. Individual data points are represented as leaf nodes, and each leaf node is associated with its unique hash value. Meanwhile, non-leaf nodes connected by the hash value of their child nodes are known as internal nodes. A tree's root node is a single hash value, which is the result of combining the hash values of its child nodes through recursive hashing [41]. If $\Phi(n)$ is the hash function used to compute the hash of a given input n and '||' denotes concatenation, then the equation for generating a Merkle root hash can be expressed as

$$\Phi(n_{parent}) = hash(\Phi(n_{left})||\Phi(n_{right})). \quad \ldots(1)$$

The structure of a Merkle tree comprises a full binary tree that is accompanied by a hash function and an assignment function, denoted by ϕ, which maps the nodes in the tree to κ length strings in the set $n \rightarrow \Phi(\mathrm{n}) \in \{0,1\}^k$. In particular, for any interior node (n_{parent}) in the tree, its two child nodes (n_{left}) and (n_{right}) must satisfy the condition that the assignment function Φ maps n parent to the concatenation of the values of Φ (n_{parent}) and Φ (n_{right}). Figure 3 illustrates the process of generating a Merkle tree in the proposed system after data are hashed and how to obtain the root hash.

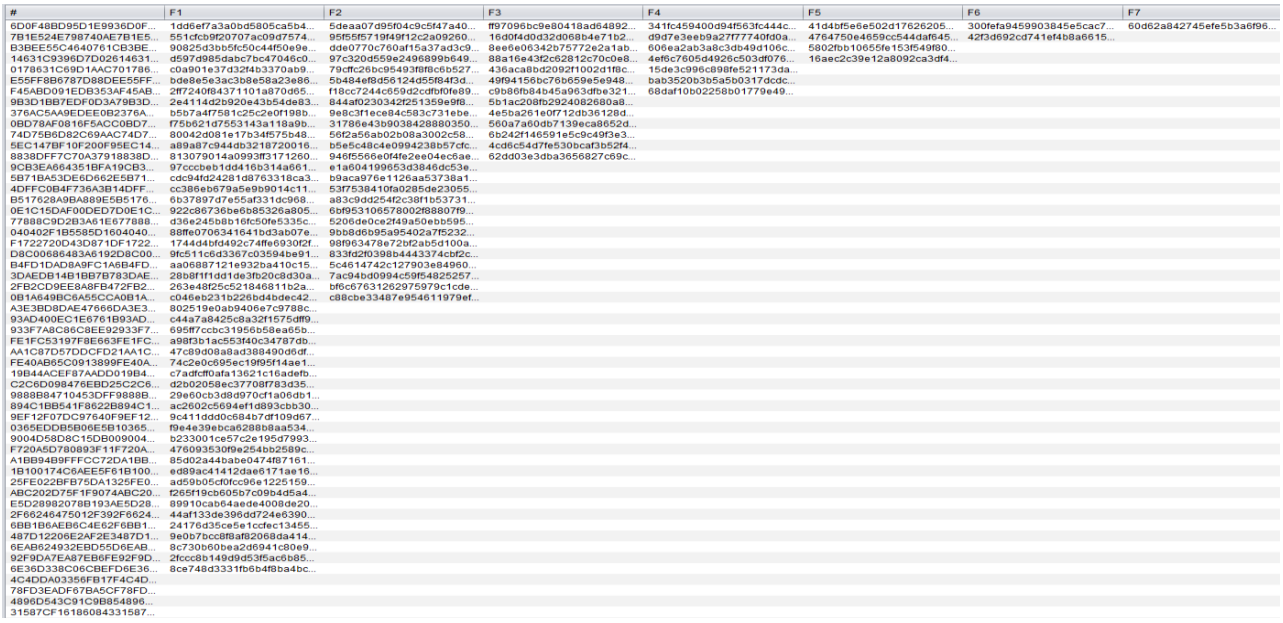

Fig. 3. Merkle Tree for Generating Top Hash

4. Modified Rabbit Algorithm to Work as Hash in a Blockchain (MRHB)

Cryptography is extremely important with regard to securing private information, such as medical records. With the advent of blockchain technology, a decentralisation strategy that secures data integrity and improves security is established.

The current work extends the Rabbit algorithm, which is an efficient and compact segmentation approach used in blockchain systems. A multilayered system is shown in Figure 4, in which medical data are produced by health professionals during patient examination and diagnostics on different devices.

In the second layer, the Rabbit-256 algorithm is used to transform data into session unique hashes. It scrambles data by nonlinearly operating on them with random keys, increasing privacy and cryptographic resistance. The Rabbit algorithm is a symmetric key stream cipher that produces a key stream by combining input data, a secret key and a nonce. The key stream is XORed with the input data, and the process is repeated multiple times for security.

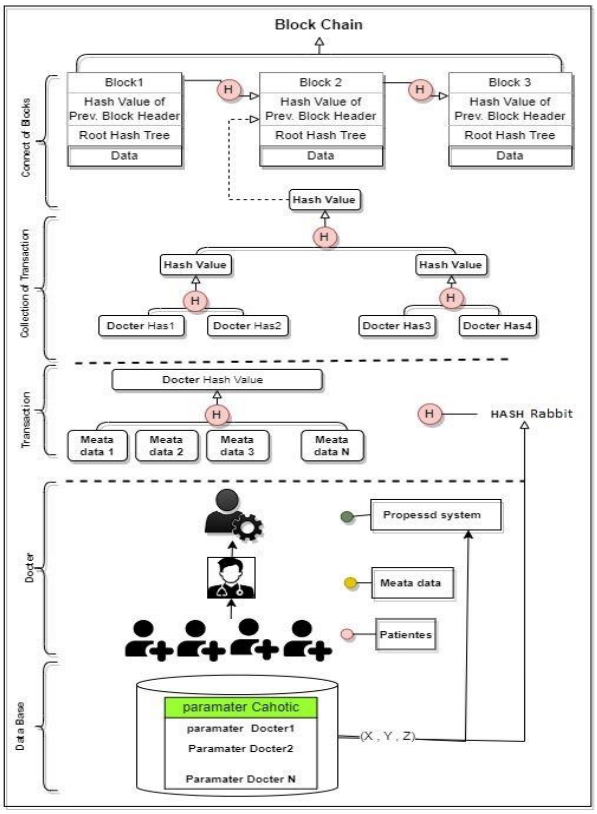

Fig. 4. Structure of MRHB

The basic Rabbit is a stream cipher, generating a keystream for encryption, and MRHB tweaks this structure such that it functions as a hash. In MRHB, metadata (imprint time, nonce and patient ID) are embedded into message blocks and then manipulated through state updates, S-box/chaotic substitutions and block mixing. The iterative aggregation processed with final compression works as a 256-bit optimised hash value for secure and lightweight healthcare blockchain applications. Figure 5 illustrates a comparative view of the original Rabbit cipher and the candidate MRHB hash function.

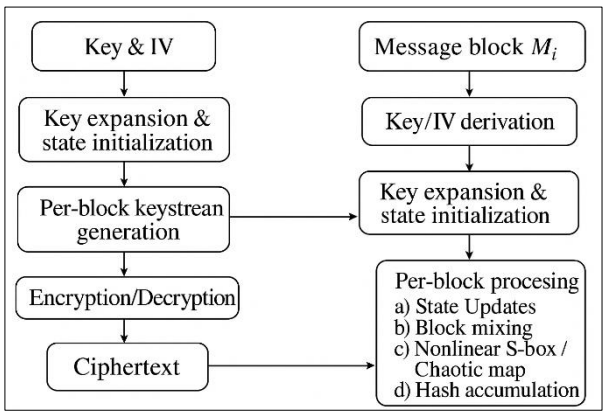

Fig. 5. Comparative Architecture of the Original Rabbit Cipher and the Proposed MRHB Hash Function

To clarify the transformation process from a stream cipher into a hash function, the researchers detail the steps of transforming the internal state of the Rabbit stream cipher into a secure, lightweight hash primitive (i.e. MRHB). The process of transforming Rabbit from a stream cipher into a hash primitive (MRHB) works by using Rabbit's fast state update and nonlinear counter functions, adding deterministic seeding, block-based mixing and additional nonlinear tweaks. The objective is to create a lightweight, unkeyed 256 bits hash output that works to verify public blockchains in IoT healthcare settings.

Design parameters:

- Internal state: 513 bits (Rabbit core state)
- Message block size: 256 bits (32 bytes)
- Internal word size: 32 bits
- Output length: 256 bits (fixed)
- Rounds per block: 4 Rabbit state updates (configurable for security/performance trade-off)

Transformation steps:

1- Initialisation

- Rabbit's secret key/IV is replaced with fixed public constants.
- A 256 bits initial chaining value (H_0) is derived from predefined constants.
- This step ensures that identical input messages always produce identical output (public verifiability).

2- Message preprocessing

- Input message (metadata ‖ payload) is serialised and padded using Merkle–Damgård style: append 0×80, followed by 0 bytes, and finally, a 64 bits big-endian message length (bits).
- The padded message is divided into 256 bits blocks: M[1..N].

3- Per-block processing

-To put through a piece block M_i (where i ranges from 1 To N):

a. Seed creation: A deterministic seed is created from the current chaining value M_i and the block counter combined.

b. Rabbit keystream generation: This seed is used as input for Rabbit (mapping it onto an internal state) and then a 256 bits keystream K_i is calculated.

c. Block transformation: The XOR function is used on M_i with K_i and T_i is obtained.

d. Nonlinear substitution: Byte-per-byte is substituted with a substitution box $U_i = SBox(T_i)$, which can be an AES S-box or a simple chaotic map as an option.

e. Hash accumulation: The chaining value is changed nonlinearly:

- H_i is equal to Rabbit round (H_{i-1} XOR U_i), where Rabbit round refers to a single update of the Rabbit state that uses U_i as the input. This method prevents simple XOR-based accumulation and strengthens defence against collision-style attacks.

4- Finalisation

- The length of the encoded message is added to the state once all the blocks are processed.
- Four extra Rabbit updates are performed using a zero block to ensure that all leftover structures are mixed.
- The final 256 bits chaining value is taken and used as the resulting hash.

5- Output

- A 256 bits MRHB digest is generated as the final hash.
- The fixed and accessible digest can be employed as a blockchain system's block header hash or a Merkle leaf.

The resulting encrypted data are the hash, merged with randomly generated keys. Figure 6 shows the encrypted data for private healthcare.

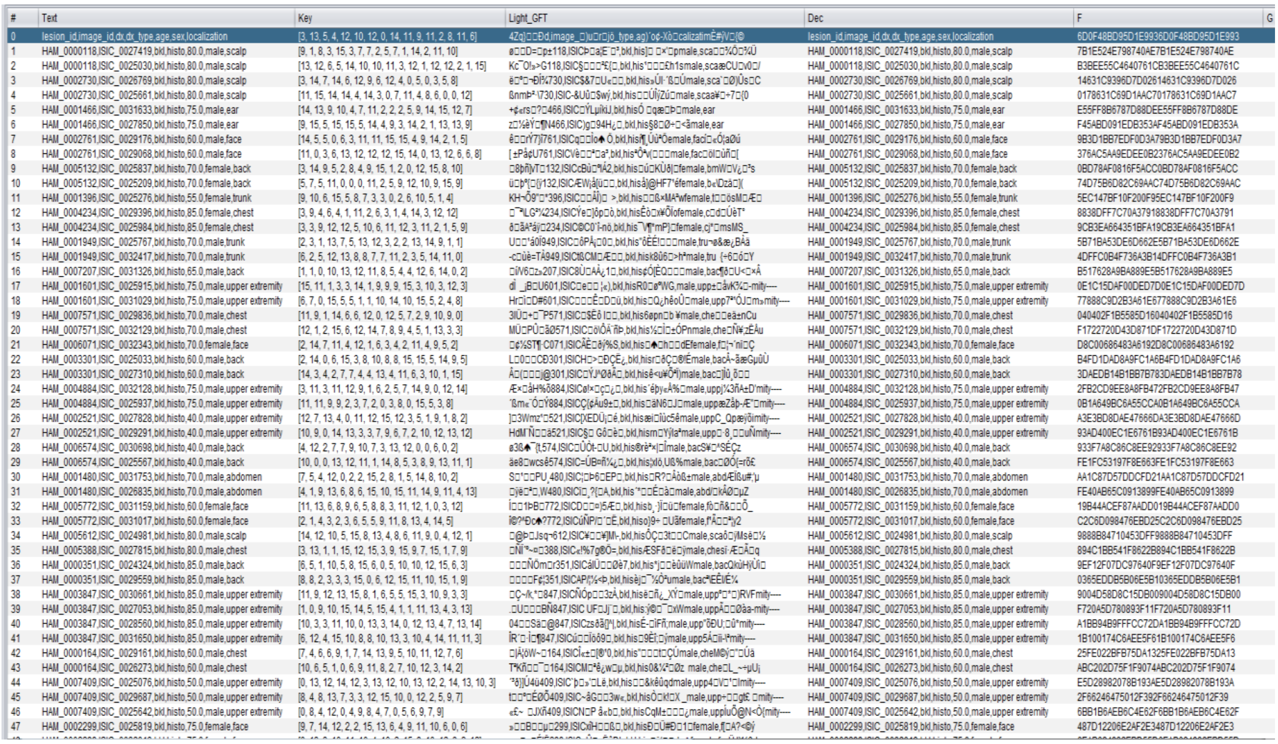

Fig. 6. Private Healthcare Data After Encryption

In the third layer, a Merkle tree is constructed based on hashes from several sessions. This tree guarantees security in transmitting data and avoiding spurious file transmission. Every leaf node is a hash of a session, and its parent nodes are the hashes of pairs of child nodes. File integrity verification is achieved by network users by comparing hashes with the root hash of the Merkle tree. Any interference will cause a mismatch between the root hash that is stored and the calculated one, indicating tampering.

To ensure data integrity, the root hash of the Merkle tree is stored in a blockchain and in a new block. The use of a distributed ledger is beneficial because it can store multiple network nodes. One of the important features of adopting blockchain technology is that the blocks are essentially immutable and cannot be tampered with after they are added, and thus, they are considered highly secure. By relying on data replication technology in the network, adopting this technology helps reduce individual failures and mitigate the severity of attacks, which, in turn, leads to an increase in reliability. Figure 7 and Algorithm 1 illustrate the flowchart and pseudocode that present the methodology of the proposed Rabbit-256-based blockchain hashing for healthcare data security, respectively.

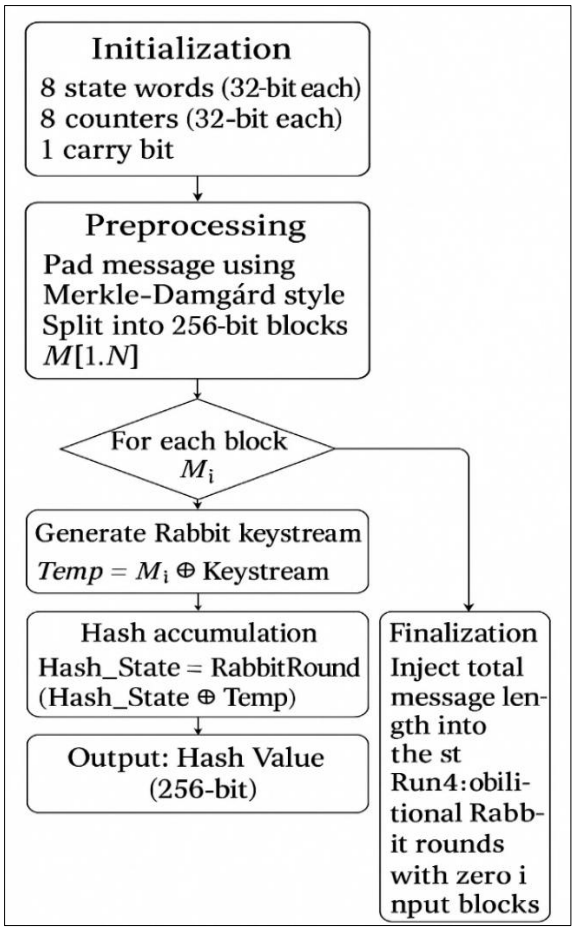

Fig. 7. Flowchart of the Research Methodology

Algorithm MRHB_Hash(Input: Message)
Output: Hash_Value (256-bit)

1- Initialization:
- Define internal state as Rabbit's original structure:
- **8 state words (32-bit each)**
- **8 counters (32-bit each)**
- **1 carry bit**

Total = 513 bits.
- Initialize all state variables with fixed public constants.

2- Preprocessing:
- Pad Message using Merkle–Damgård style:
Append 0x80, followed by zeros, then append 64-bit message length (in bits).
- Split padded message into 256-bit blocks M[1..N].

3- For each block M_i:
a) Generate Rabbit keystream (4 rounds of state update using g-function.(
b) Mix block with keystream:
Temp = $M_i \oplus$ Keystream
c) Nonlinear layer:
Temp = SBox(Temp) (AES S-box or chaotic map applied bytewise)
d) Hash accumulation:
Hash_State = RabbitRound(Hash_State $\oplus$ Temp)

4- Finalization:
- Inject total message length into the state.
- Run 4 additional Rabbit rounds with zero input blocks.
- Extract 256 bits of the final state (concatenate 4 × 64-bit words) as Hash_Value.

5- Return Hash_Value.

Algorithm 1. Pseudocode Block of the Modified Rabbit-256 Hashing Process

5. Simulation Environment

To ensure reproducibility, all experiments in the current research were performed in a managed software–hardware setup. The specifications are summarised below:
• Hardware platform:
- Processor: Intel Core i7-11800H @ 2.30 GHz (8 cores, 16 threads)
- Memory: 16 GB DDR4 RAM
- Storage: 512 GB SSD
- Operating system: Windows 11 Pro, 64 bit
• Software environment:
- Programming language: Python 3.11
- Libraries: NumPy (v1.26), SciPy (v1.12), Matplotlib (v3.8) and custom cryptographic routines for Rabbit and SHA-256
- Simulation IDE: Jupyter Notebook/PyCharm Community Edition
- Randomness source: Python's built-in secrets library for nonce/IV generation
• Dataset and experimental setup:
- Input sizes: 100, 500 and 1000 random test messages (each 128-bit block is padded)
- Message metadata: Time stamp, patient ID and nonce values were included to simulate healthcare transactions.
- Evaluation metrics: the avalanche effect, HD and mean standard deviation (MSD).

Experiments were repeated on the same dataset with different seeds to validate that they were consistent. Resource usage monitoring indicated that typical memory utilisation did not exceed 350 MB. It also provided evidence of suitability to resource-constrained IoT-like environments with a CPU load of less than 40%.

6. Results and Discussion

Hashing in the system is also examined, tested and proven effective in the following subsections. For thoroughness and to provide a full account of the studies, multiple evaluation criteria were adopted as follows: bit independence, avalanche metrics computation, HD, mean changed and MSD. The framework was tested with three datasets in the current study: 100, 500 and 1000 hashes.

The hash function was compared with the SHA-256 algorithm, which is generally employed in this area. In addition, the dimension of each hash value in the dataset is 256 bits. The hash function apparently exhibits the following property of 'independent bit': Informally, for a hash function, proving that any subset selection of bits in the output can be generated independently from any other (which is essential for the Markle–Damgård structure) should be possible. Through the observation of output bits, such a property can be tested, whether it is uniform or not.

Balancing the ARAB-256 hashing scheme based on different data sizes (100, 500 and 1000) is employed in the current paper. A balanced assessment can be obtained by focusing on the number 100 as a baseline, increasing it to 500 for medium-range testing, and then up to 1000 for large-scale evaluation. This procedure gives rise to computational simulations but retains statistical 'robustness'. These use cases are helpful, and they genuinely represent the healthcare situation with IoT.

Table 2, A
Use of 100 hash input to determine the average of the metrics

Metric	Rabbit-256	SHA-256
BIC	0.0004534	0.0004580
Avalanche	128.64	80.05
HD	36.58	58.44
NMCB	50.41796	50.01953
MSD	73.54271	73.887815

Table 2, B
Use of 500 hash input to determine the average of the metrics

Metric	Rabbit-256	SHA-256
BIC	0.0004541	0.0004574
Avalanche	128.172	80.214
HD	59.79	37.334
NMCB	49.815	50.025
MSD	73.73788	73.65656

Table 2, C
Use of 1000 hash input to determine the average of the metrics

Metric	Rabbit-256	SHA-256
BIC	0.0004588	0.0004563
Avalanche	127.872	80.39
HD	60.038	37.57
NMCB	49.948882	49.969921
MSD	73.73166	73.72074

The performance and security of hash functions are evaluated based on two essential factors: confusion and diffusion. To clarify these concepts, confusion complicates the relationship between input and output, whilst diffusion distributes the effect of input on output. Statistical tests measure the bits that change in the output when any modification occurs in the input, as approved by NIST. The obtained results are used to demonstrate the efficiency and effectiveness of the hash functions, enabling various comparisons to be made.

From Tables 2A–2C, the avalanche effect and HD distribution of Rabbit-256 are always better than those of SHA-256 under any size (100, 500, 1000) of message digest. In crypto terms, a high avalanche effect indicates that a small number of input bits are spread over the output space, such that a single bit change in input causes approximately half of the bits to change output.

Whilst the above experiments put Rabbit-256 against SHA-256, researchers have considered other lightweight hash functions, as indicated in Table 2.

Table 3,
Comparative features of Rabbit-256 (this work) and lightweight hash functions from the literature

Metric (average)	Rabbit-256 (our work)	BLAKE 2s [Aumasson et al., 2015]	SPONGENT-128 [Bogdanov et al., 2012]
Avalanche Effect (%)	~58%	~54%–56%	~48%–50%
HD (bits)	58–60 average	~56 average	~32–34 average
Bit Independence	Strong	Strong	Weak–Moderate
Memory/Power Needs	Low	Moderate	Very Low
IoT Suitability	Excellent	Good	Excellent
Blockchain Integration	Direct Merkle tree)	Direct	Rarely explored

To create a reliable benchmarking comparison, the researchers do not adopt the exact values. The original values differ in the test dataset [42] [43], leading to the adoption of the average ranges that have been reported in the literature. Table 3 provides a wider view by comparing Rabbit-256, BLAKE2s and SPONGENT.

Rabbit-256 exhibits a stronger avalanche effect (about 58%) than SPONGENT (about 48%–50%) and comes close to BLAKE2s (about 54%–56%). It also reaches a higher HD (59–60 bits) than SPONGENT. BLAKE2s spreads data well but requires more memory. SPONGENT uses less resources but does not separate data as well as BLAKE2s. These results indicate that Rabbit-256 achieves good balance and is fit for blockchain-based healthcare IoT, where security and speed are important.

Moreover, well-balanced HD values imply that our proposed MRHB scheme is slightly independent, eliminating the possibility of an adversary guessing the relationships between input and output.

In the health domain, such GCs directly translate into enhanced security for electronic health records and IoT sensor data. For example, if one information changes slightly in a patient's vital sign record, then this slight change will lead to a completely different hash, and the blockchain transaction will immediately go awry. Determining if data were altered, which is an exceedingly important detail for tracking compliance with the rules imposed by the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR), will be difficult. In addition, no Rabbit-256 slowdown is observed on larger sets (500–1000 blocks), enabling Rabbit-256 to handle the high volume of data traffic from a busy hospital and real-time IoT data flows.

That is, Rabbit-256 has more benefits to offer than the SHA-2 family, and the difference frequently matters in the real world. Rabbit-256 provides a lighter but more secure method of hashing data; therefore, it is applicable to securing blockchain systems in healthcare.

The next subsection explains the key metrics employed for the evaluation, as follows.

6.1. Bit Independence criterion (BIC) analysis

In the hash function, we can evaluate the independence of output bits with BIC. The more robust the BIC, the more challenging controlling and forecasting its product will be. Attacker attempts to forge and manipulate hash values become harder.

Figures 8A–8C compare the proposed hashing method with SHA-256 in terms of BIC values. The graph shows the BIC values of the proposed method and SHA-256 when hashing 100/500/1000 length strings. Comparing the BIC results of Rabbit-256 and SHA-256, as presented in Tables 2A–2C and Figures 8A–8C, both algorithms generally have high BIC values, indicating that their output bits are independent and unpredictable. However, some results indicate that the BIC values for SHA-256 are slightly higher in certain cases compared with the Rabbit-256 results. This discrepancy suggests that Rabbit-256 and SHA-256 may be more difficult for attacks that use correlations between output bits.

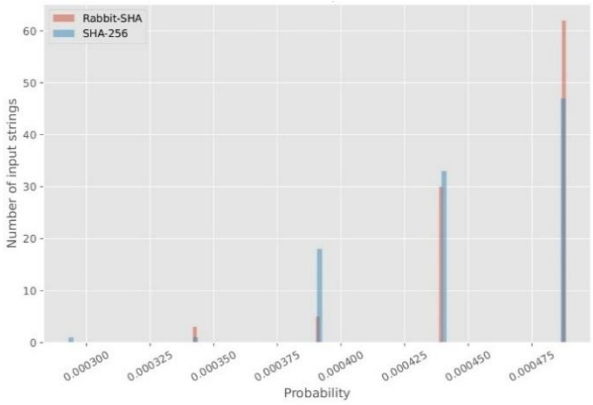

Fig. 8. A. BIC Values for the [Rabbit-256, SHA-256] Case Study [100]

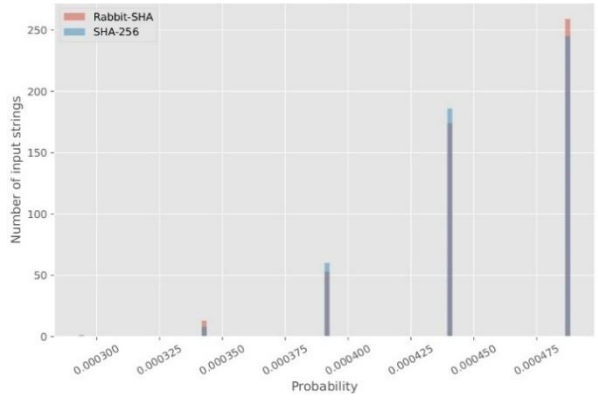

Fig. 8. B. BIC Values for the [Rabbit-256, SHA-256] Case Study [500]

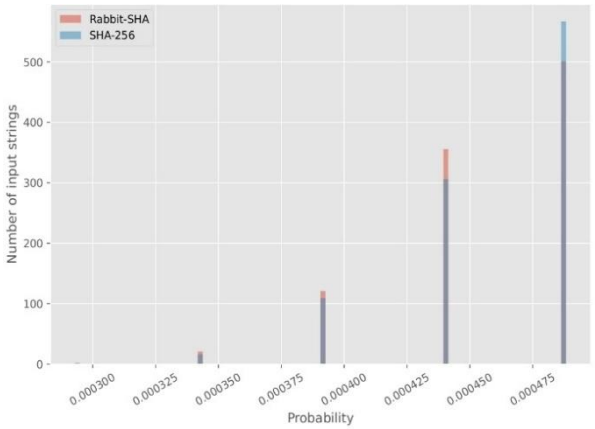

Fig. 8. C. BIC Values for the [Rabbit-256, SHA-256] Case Study [1000]

6.2. Avalanche metrics analysis

The 'avalanche effect' in hash functions is a well-known phenomenon in cryptography. It can be summarised as follows: when a single bit of the input is flipped, an extremely large change occurs in the output hash value. This phenomenon can be quantified by calculating the percentage of bits changed in the output of two input that differ by 1 bit. In hash functions, when the value of the avalanche effect is close to 50%, which is the highest value, the case is considered optimal. The following well-known equation, called the standard percentage calculation, is utilised to find the measure:

$$D = \frac{X}{Y} * 100\%, \qquad \ldots(2)$$

where D represents the avalanche effect, X represents the number of modified bits in the resulting hash value, and Y represents the total number of bits in the hash value. As mentioned previously, the avalanche effect becomes more important for a stronger hash function, and vice versa. This condition makes constructing two messages that hash to the same value virtually impossible for an attacker.

When comparing the avalanche effect measures of the modified Rabbit-256 and SHA-256, we can see that for a bit change probability, the modified Rabbit-256 offers a 58% average bit change rate compared with up to 50% average bit change rate for SHA-256. This result implies that the modified Rabbit-256 algorithm exhibits better avalanche effect, i.e. any slight modification on the input will cause a major change in the output, as indicated in Tables 2A–2C and Figures 9A–9C.

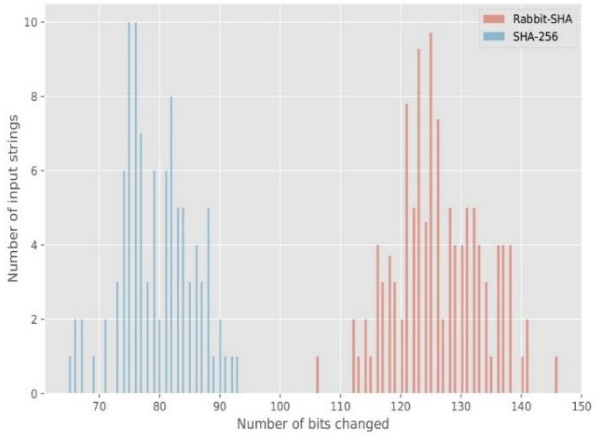

Fig. 9. A. Calculation Avalanche Metrics for the [Rabbit-256, SHA-2056] Case Study [100]

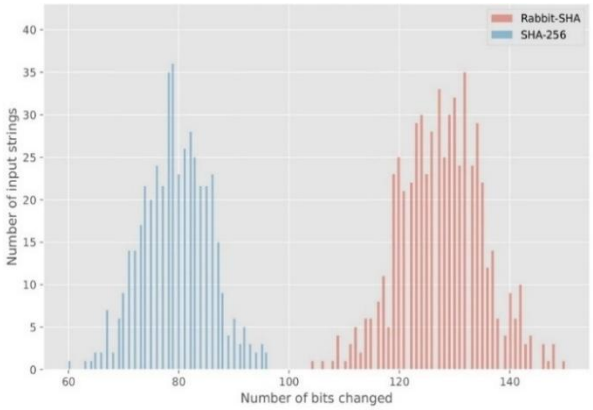

Fig. 9. B. Calculation Avalanche Metrics for the [Rabbit-256, SHA-2056] Case Study [500]

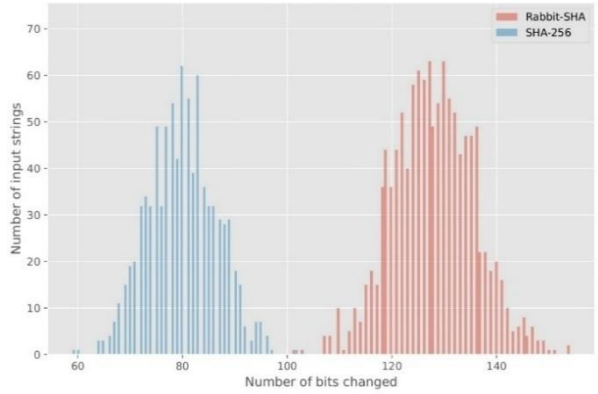

Fig. 9. C. Calculation Avalanche Metrics for the [Rabbit-256, SHA-2056] Case Study [1000]

6.3. HD Metrics analysis

HD indicates the similarity between two hash values by computing the number of positions at which their respective symbols are different. It determines the similarity between the two input by comparing their hashes. A distance with a low Hamming value indicates that the two items are similar, whereas a large distance indicates high dissimilarity. This metric is useful in applications such as near-duplicate detection, fuzzy matching or similarity analysis amongst large-scale datasets.

HD between two strings of equal length can be calculated using the following equation:

$$d_H(A,B) = \sum_{i=1}^{n} (a_i \neq b_i), \quad \ldots(3)$$

where $d_H(A,B)$ indicates HD between strings A and B, $\sum_{i=1}^{n}$. indicates the sum of the differences between corresponding symbols in the two strings, from $i = 1$ to $i = n$, where n is the length of the strings; a_i and b_i are the symbols at position i in strings A and B, respectively; and $\neq$ indicates inequality, such that the sum is only incremented when $a_i$and b_i are different.

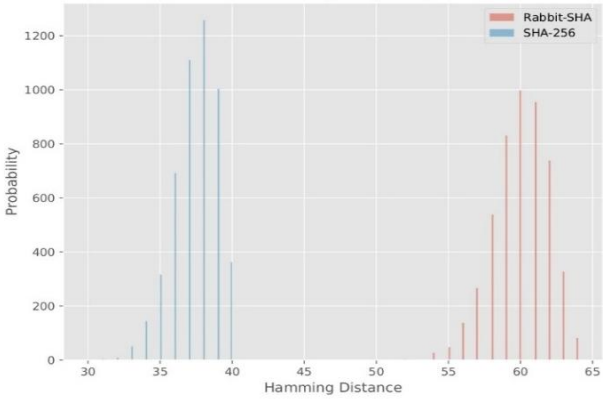

Fig. 10. A. Calculation of HD in the [Rabbit-256, SHA-256] Case Study [100, 4950]

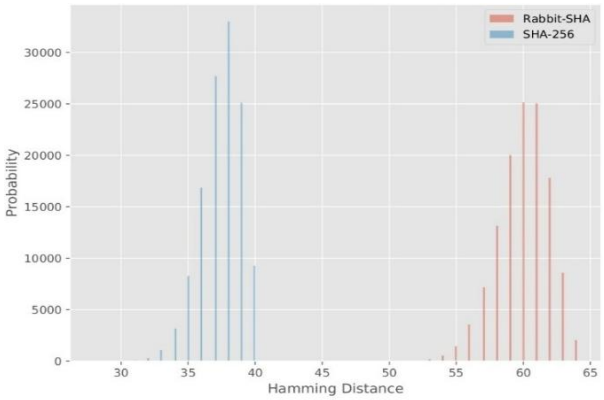

Fig. 10. B. Calculation of HD in the [Rabbit-256, SHA-256] Case Study [500, 124750]

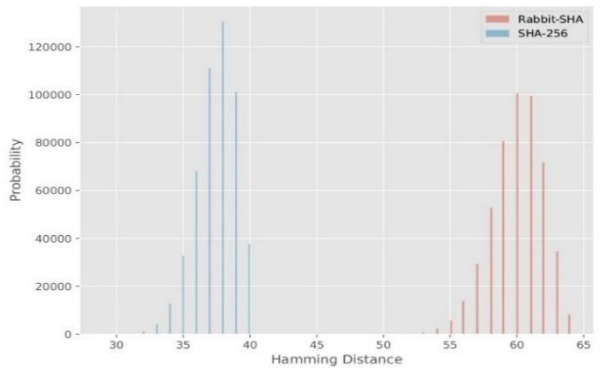

Fig. 10. C. Calculation of HD in the [Rabbit-256, SHA-256] Case Study [1000, 499500]

Comparing the HD results of the modified Rabbit-256 and SHA-256, we can see that the modified Rabbit-256 output has a higher average HD than the SHA-256 output. This result indicates

that the modified Rabbit-256 algorithm produces output that is more different from each other than the output of SHA-256, as presented in Tables 2A–2C and Figures 10A–10C.

6.4. Analysis of the number of mean changed bits (NMCB)

NMCB analysis evaluates cryptographic hash functions by measuring the average number of bits that change the hash value when a single bit is altered in the input. It assesses the avalanche effect, where a slight change in input results in a significant change in output.

$$NMCB = \frac{1}{N} * \sum_{i=1}^{n}(|x_i - y_i|), \quad \ldots(4)$$

where N is the total number of bits in the two binary sequences being compared, $x_i$and y_i are the corresponding bits in the two binary sequences and $\sum_{i=1}^{n}$. The sum of the absolute differences between each corresponding bit in the two binary sequences is denoted.

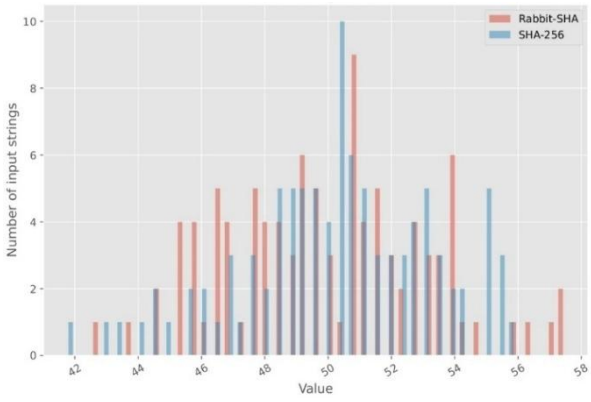

Fig. 11. A. Percentage of Mean Changed Bits in the [Rabbit-256, SHA-256] Case Study [100]

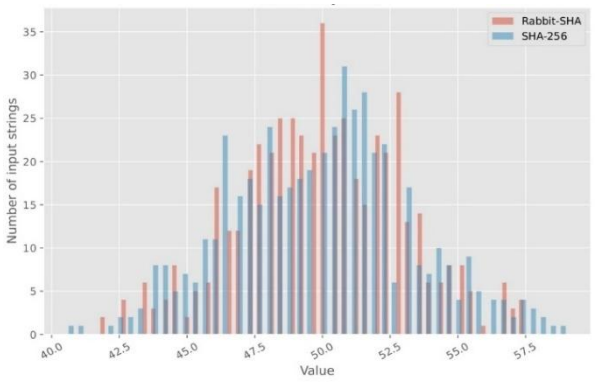

Fig. 11. B. Percentage of Mean Changed Bits in the [Rabbit-256, SHA-256] Case Study [500]

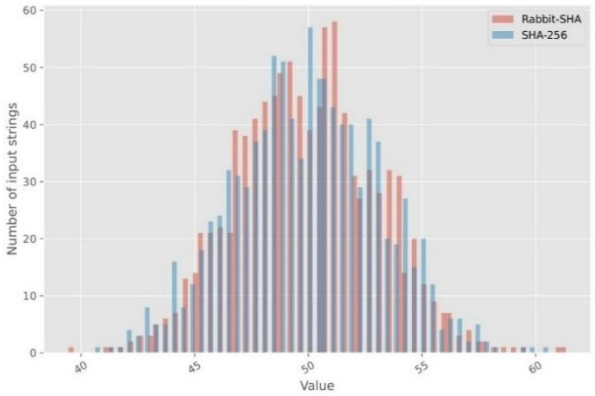

Fig. 11. C. Percentage of Mean Changed Bits in the [Rabbit-256, SHA-256] Case Study [1000]

By comparing the obtained results, we determine that the modified Rabbit-256 algorithm is superior because it provides a result of 50.366% in terms of NMCB, whilst the result of SHA-256 does not exceed 49.853%. The significance of this result is twofold: a minute modification to the input will exert several effects on the output hash. This result is a clear sign that the behaviour of the modified Rabbit-256 algorithm depends more strongly on its input than that of SHA-256. The two algorithms can be compared in terms of strength and security by using the HD shown in Figures 11A–AC. Tables 2A–2C present the comparison results between the modified Rabbit-256 and SHA-256.

6.5. MSD

The quality of cryptographic hash functions is assessed by adopting a statistical technique that measures the difference between the average HD of all possible pairs of hash values and the expected value of HD under a uniform distribution.

The mean μ and standard deviation σ of a dataset is calculated by utilising the following equation:

$$\mu = \frac{1}{2} * \sum_{i=1}^{n}(xi), \quad \ldots(5)$$

where n is the entire number of data points, and xi denotes individual data points.

$$\sigma = \sqrt{(\left(\frac{1}{n}\right) * \sum_{i=1}^{n}(xi - \mu)^2)}, \quad \ldots(6)$$

where (sqrt) represents the square root function, and $(xi - \mu)^2$ represents the squared deviation of each data point from the mean.

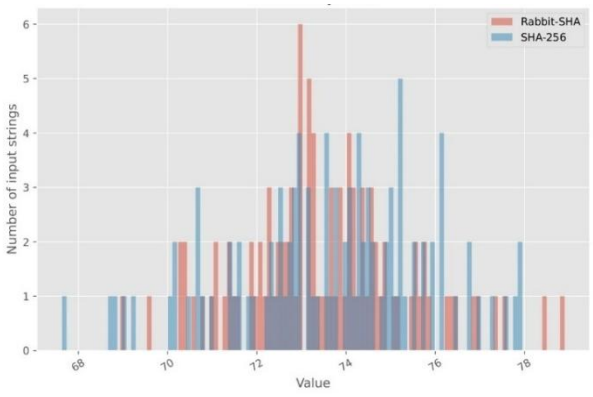

Fig. 12. A. Standard Deviation of the Changed Bit Number in the [Rabbit-256, SHA-256] Case Study [100]

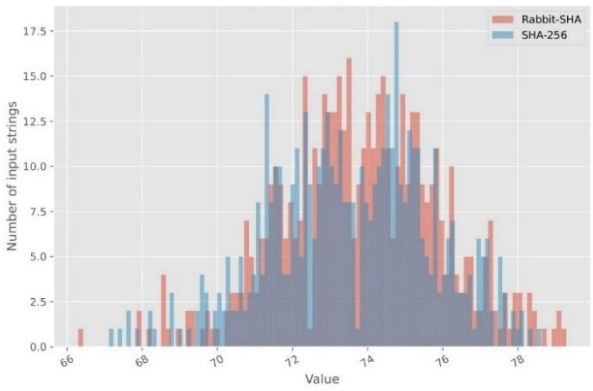

Fig. 12. B. Standard Deviation of the Changed Bit Number in the [Rabbit-256, SHA-256] Case Study [500]

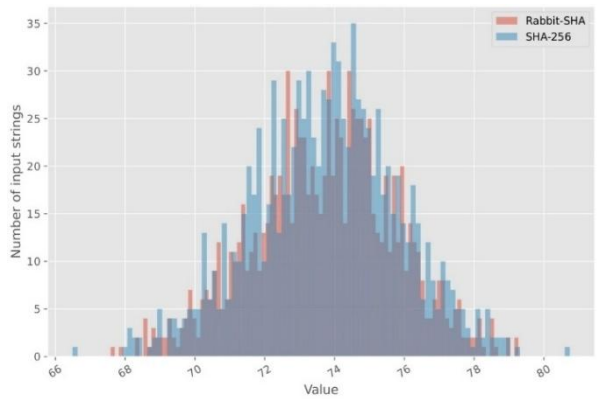

Fig. 12. C. Standard Deviation of the Changed Bit Number in the [Rabbit-256, SHA-256] Case Study [1000]

For the modified Rabbit-256, MSD = 1.635, and the values of the hash output are considered relatively stable if most of their differences from the mean are less than 1.635. The MSD for SHA-256 is 2.309, over the value for SHA-1. That is, most hash output values are close to 2.309 of the mean, as shown in Figures 12A–12C. When MSD is smaller, the dataset is more uniform and predictable. Meanwhile, a higher MSD indicates more variability and less predictability.

Expanding to what these outcomes mean in practice for healthcare IoT (latency, data transfer, battery lifetime), the enhancements observed in the avalanche effect, HD and MSD exert a direct effect on IoT-based healthcare systems. A stronger avalanche property indicates less likelihood of differential attacks, such that even slight changes in the output values from a medical sensor (e.g. heart rate or glucose level) generate unpredictable hash values. This condition increases data integrity when transferring and archiving. Trending system-wise, Rabbit-256 outperforms SHA-256 in terms of computational overhead, leading to reduced delay when providing security for fast IoT streams, e.g. continuous patient surveillance. The minimisation of delay is expected to lessen delays for queueing construction and verification, making real-time clinical decision-making more realistic. Moreover, lower computation per hash leads to less power consumption for on-device battery life against more computationally expensive approaches. This condition is particularly important for wearable or implantable sensors that need to continue to work well over a long period without frequent charging/replacement. An efficient hashing technique minimises intermediate exchanges within a blockchain, and thus, data transfer speed and the maximum scalability of healthcare data networks become faster/larger.

Introducing MRHB into the system of IoT-based healthcare will be a huge attempt to focus on efficiency and resource-saving. However, following regulations remains important. MRHB must be compliant with regulations, such as those of HIPAA and GDPR. These policies are in place to safeguard the security of patients' confidential information (confidentiality, integrity and availability). These rules are enforced by MRHB with reliance on an inspection programme that verifies integrity. It employs top avalanche and diffusion mechanisms for realising attempts to modify medical information. It also makes Rabbit-256 a frugal solution, consistent with the policy protected by GDPR of 'data protection by design and by default'. Such construction can achieve secure hashing for low-powered or slow IoT-based medical devices, but with reduced energy consumption and delay. In an HIPAA-compliant system, MRHB works with blockchain-based electronic health records to help generate tamper-proof patient records and audit logs. In addition to these standards, MRHB goes well beyond mere good cryptography, indicating that it can preserve

the privacy of patients and be trusted in real-life healthcare operations.

7. Conclusions

In accordance with the result analysis, the Rabbit-256-based optimisation of blockchain hashing for healthcare data security is a promising method. To our knowledge, the proposed algorithm is one of the first to combine the Rabbit-256 stream cipher and the SHA-256 hash function. The achieved entropy and randomness provide resistance against attacks and exhibit better security margin than competing ciphers.

The cited results indicate that the proposed Rabbit-256-based algorithm has sufficient strength and computational efficiency compared with its counterparts, including SHA-256. In addition, the probability of collision is small, which is a necessary condition for secure data storage and access in healthcare systems.

However, this work is only a simulated test and currently does not support real-world IoT deployments and hardware acceleration. The work will focus on potential directions in the future. We plan to validate Rabbit-256 in real clinical IoT scenarios, further optimising it to run on embedded hardware, and including integration with standards organisations, such as HIPAA and GDPR. Furthermore, the combination with machine learning-based anomaly detection will fortify its applicability.

Overall, Rabbit-256 offers a potential path towards realising secure, lightweight and efficient blockchain infrastructure that is specifically designed for future IoT-enabled multilevel healthcare systems.

References

[1] M. K. Thukral, "Emergence of blockchain-technology application in peer-to-peer electrical-energy trading: A review," Clean Energy, vol. 5, no. 1, pp. 104–123, 2021, doi: 10.1093/ce/zkaa033.

[2] S.-Y. Lin et al., "A survey of application research based on blockchain smart contract," Wireless Networks, vol. 28, no. 2, pp. 635–690, 2022, doi: 10.1007/s11276-021-02874-x.

[3] A. Adiyanto and R. Febrianto, "Authentication of transaction process in e-marketplace based on blockchain technology," Aptisi Transactions on Technopreneurship (ATT), vol. 2, no. 1, pp. 68–74, 2020, doi: 10.34306/att.v2i1.71.

[4] B. Düdder et al., "Event-based supply chain network modeling: Blockchain for good coffee," Frontiers in Blockchain, 2022, doi: 10.3389/fbloc.2022.846783.

[5] W. Viriyasitavat and D. Hoonsopon, "Blockchain characteristics and consensus in modern business processes," J. Ind. Inf. Integr., vol. 13, pp. 32–39, 2019, doi: 10.1016/j.jii.2018.07.004

[6] V. Gramoli, "From blockchain consensus back to Byzantine consensus," Future Generation Computer Systems, vol. 107, pp. 760–769, 2020, doi: 10.1016/j.future.2017.09.023.

[7] A. Kumar, R. Liu, and Z. Shan, "Is blockchain a silver bullet for supply chain management? Technical challenges and research opportunities," Decision Sciences, vol. 51, no. 1, pp. 8–37, 2020, doi: 10.1111/deci.12396.

[8] C. C. Agbo and Q. H. Mahmoud, "Comparison of blockchain frameworks for healthcare applications," Internet Technol. Lett., vol. 2, no. 5, p. e122, 2019, doi: 10.1002/itl2.122.

[9] S. Chen et al., "Study and implementation on the application of blockchain in electronic evidence generation," Forensic Sci. Int.: Digit. Invest., vol. 35, p. 301001, 2020, doi: 10.1016/j.fsidi.2020.301001.

[10] Z. E. Rasjid et al., "Implementation of Rail Fence Cipher and Myszkowski Algorithms and Secure Hash Algorithm (SHA-256) for Security and Detecting Digital Image Originality," in Proc. 2022 Int. Conf. Informatics, Multimedia, Cyber and Information Syst. (ICIMCIS), pp. 207–212, 2022, doi: 10.1109/ICIMCIS56303.2022.10017975.

[11] Y. Liu et al., "Optical image encryption algorithm based on hyper-chaos and public-key cryptography," Optics & Laser Technology, vol. 127, p. 106171, 2020, doi: 10.1016/j.optlastec.2020.106171.

[12] C. Stoll, U. Gallersdörfer, and L. Klaaßen, "Climate impacts of the metaverse," Joule, vol. 6, no. 12, pp. 2668–2673, 2022, doi: 10.1016/j.joule.2022.10.013.

[13] NOVO, Oscar. Blockchain meets IoT: An architecture for scalable access management in IoT. IEEE internet of things journal, 2018, 5.2:1184-1195.
DOI: 10.1109/JIOT.2018.2812239

[14] S. Guo et al., "Blockchain meets edge computing: Stackelberg game and double auction based task offloading for mobile

blockchain," IEEE Trans. Veh. Technol., vol. 69, no. 5, pp. 5549–5561, 2020, doi: 10.1109/TVT.2020.2982000.

[15] M. A. Bouras et al., "IoT-CCAC: a blockchain-based consortium capability access control approach for IoT," PeerJ Comput. Sci., vol. 7, p. e455, 2021, doi: 10.7717/peerj-cs.455.

[16] S. Fu et al., "Cooperative computing in integrated blockchain-based internet of things," IEEE Internet Things J., vol. 7, no. 3, pp. 1603–1612, 2019, doi: 10.1109/JIOT.2019.2948144.

[17] B. W. Aboshosha, M. M. Zayed, H. S. khalifa, and R. A. Ramadan, "Enhancing Internet of Things security in healthcare using a blockchain-driven lightweight hashing system," Beni-Suef University Journal of Basic and Applied Sciences, vol. 14, no. 1, May 2025, doi: https://doi.org/10.1186/s43088-025-00644-8.

[18] F. Hanif, U. Waheed, R. Shams, and A. Shareef, "GAHBT: Genetic Based Hashing Algorithm for Managing and Validating Health Data Integrity in Blockchain Technology," Blockchain in Healthcare Today, vol. 6, no. 2, Feb. 2023, doi: https://doi.org/10.30953/bhty.v6.244.

[19] S. J. Basha et al., "Security enhancement of digital signatures for blockchain using EdDSA algorithm," in Proc. 2021 3rd Int. Conf. Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 274–278, 2021, doi: 10.1109/ICICV50876.2021.9388411.

[20] K. Ashritha, M. Sindhu, and K. V. Lakshmy, "Redactable blockchain using enhanced chameleon hash function," in Proc. 2019 5th Int. Conf. Advanced Computing & Communication Systems (ICACCS), pp. 323–328, 2019, doi: 10.1109/ICACCS.2019.8728524.

[21] Y. Tian et al., "Policy-based chameleon hash for blockchain rewriting with black-box accountability," in Proc. Annu. Comput. Security Applications Conf., pp. 813–828, 2020, doi: 10.1145/3427228.3427247.

[22] Jesús Soto-Cruz, E. Ruiz-Ibarra, J. Vázquez-Castillo, A. Espinoza-Ruiz, A. Castillo-Atoche, and J. Mass-Sanchez, "A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers," Technologies, vol. 13, no. 1, pp. 3–3, Dec. 2024, doi: https://doi.org/10.3390/technologies13010003.

[23] P. S. Suryateja and K. Venkata Rao, "A Survey on Lightweight Cryptographic Algorithms in IoT," Cybernetics and Information Technologies, vol. 24, no. 1, pp. 21–34, Mar. 2024, doi: https://doi.org/10.2478/cait-2024-0002.

[24] A. Sevin and Ü. Çavuşoğlu, "Design and Performance Analysis of a SPECK-Based Lightweight Hash Function," Electronics, vol. 13, no. 23, p. 4767, Dec. 2024, doi: https://doi.org/10.3390/electronics13234767.

[25] A. L. A. Fonsêca et al., "Blockchain in Health Information Systems: A Systematic Review," International Journal of Environmental Research and Public Health, vol. 21, no. 11, p. 1512, Nov. 2024, doi: https://doi.org/10.3390/ijerph21111512.

[26] A. Arif, M. Hussain, and C. P. Subbe, "Blockchain: What is the use case for physicians in 2024? A rapid review of the literature," Future Healthcare Journal, vol. 11, no. 1, p. 100005, Sep. 2024, doi: https://doi.org/10.1016/j.fhj.2024.100005.

[27] N. F. Mufidah and Hilal Hudan Nuha, "Performance and Security Analysis of Lightweight Hash Functions in IoT," Jurnal Informatika Jurnal Pengembangan IT, vol. 9, no. 3, pp. 264–270, Dec. 2024, doi: https://doi.org/10.30591/jpit.v9i3.7633.

[28] B. B. Gupta and M. Quamara, "An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols," Concurrency Computat.: Pract. Exper., vol. 32, no. 21, p. e4946, 2020, doi: 10.1002/cpe.4946.

[29] P. Kietzmann et al., "A performance study of crypto-hardware in the low-end IoT," Cryptology ePrint Archive, 2021. [Online]. Available: https://ia.cr/2021/058.

[30] R. B. Gandara and M. Alaydrus, "Analysis of the IEEE 802.15.4 Protocol with Rabbit Encryption Algorithm for Industrial Applications in Oil and Gas Sector," in Proc. 2019 16th Int. Conf. Quality in Research (QIR), pp. 1–5, 2019, doi: 10.1109/QIR.2019.8898287.

[31] J. Daemen and V. Rijmen, The Design of Rijndael: AES — The Advanced Encryption Standard. New York, NY, USA: Springer, 2013. doi: 10.1007/978-3-662-04722-4.

[32] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proc. 52nd Annual Design Automation Conf. (DAC)*, San Francisco, CA, USA, Jun. 2015, pp. 1–6. doi: 10.1145/2744769.2747946.

[33] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An

ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems – CHES 2007* (Lecture Notes in Computer Science, vol. 4727), P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer, 2007, pp. 450–466. doi: 10.1007/978-3-540-74735-2_31

[34] M. El-Hajj, H. Mousawi, and A. Fadlallah, "Analysis of lightweight cryptographic algorithms on IoT hardware platform," *Future Internet*, vol. 15, no. 2, p. 54, Feb. 2023. doi: 10.3390/fi15020054.

[35] S. Ahmad, S. Mehfuz, and J. Beg, "Hybrid cryptographic approach to enhance the mode of key management system in cloud environment," J. Supercomput., 2022, pp. 1–37, doi: 10.1007/s11227-022-04964-9.

[36] V. I. Korzhik et al., "Information theoretically secure key sharing protocol executing with constant noiseless public channels," Math. Vopr. Kibernet., vol. 12, no. 3, pp. 125–141, 2021, doi: 10.4213/mvk378.

[37] P. S. Nakhate and R. T. Pansare, "CS 237 Project Paper – PII Data Security in Software Systems," Univ. California, Irvine, 2022. [Online]. Available: https://ics.uci.edu/cs237/projects2022/5_report.pdf.

[38] O. A. Khashan, R. Ahmad, and N. M. Khafajah, "An automated lightweight encryption scheme for secure and energy-efficient communication in wireless sensor networks," Ad Hoc Netw., vol. 115, p. 102448, 2021, doi: 10.1016/j.adhoc.2021.102448.

[39] K. S. Garewal, "Merkle trees," in Practical Blockchains and Cryptocurrencies, Apress, 2020, pp. 137–148, doi: 10.1007/978-1-4842-5893-4.

[40] U. Chelladurai and S. Pandian, "Hare: A new hash-based authenticated reliable and efficient modified Merkle tree data structure to ensure integrity of data in the healthcare systems," J. Ambient Intell. Humaniz. Comput., 2021, doi: 10.1007/s12652-021-03085-0.

[41] Y. Yang et al., "Fast wireless sensor for anomaly detection based on the data stream in an edge-computing-enabled smart greenhouse," Digit. Commun. Netw., vol. 8, no. 4, pp. 498–507, 2022, doi: 10.1016/j.dcan.2021.11.004.

[42] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, *The BLAKE2 cryptographic hash and message authentication code (MAC)*, RFC 7693, Aug. 2015. doi: 10.17487/RFC7693.

[43] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "SPONGENT: the design space of lightweight cryptographic hashing," IEEE Transactions on Computers, vol. 62, no. 10, pp. 2041–2053, Aug. 2012, doi: 10.1109/tc.2012.196.

تحسين Rabbit-256 لتجزئة البلوكتشين الأمنة في بيانات الرعاية الصحية لأنترنت الاشياء

خالد جمال جداع[1]*، ايمن مظهر بدر [2]، وليد نوري حسين [3]، لطيفة منيرة قمرالدين[4]

[1] قسم هندسة الحاسوب، كلية الهندسة، جامعة ديالى، ديالى، العراق

[2] كلية الطب، جامعة ديالى، ديالى، العراق

[3] كلية طب الزهراء، جامعة البصرة، البصرة، العراق

[4] قسم هندسة الحاسوب و الاتصالات، جامعة برليس الماليزية، برليس، ماليزيا

[4] مركز التميز لتكنولوجيا الاستشعار المتقدمة (CEASTech)، برليس، ماليزيا

*البريد الالكتروني : khalid.jamal.jadaa@gmailcom

المستخلص

تقيات البلوكتشين اصبحت اداة مطروحة لتأمين سجلات الرعاية الصحية، لكن بسبب التكلفة الحسابية العالية لدوال التجزئة ادت الى محدودية استخدامها في اغلب سناريوهات انترنت الاشياء. و لحل هذه المشكلة تقترح هذه الورقة البحثية دالة (Rabbit-256) و هي عبارة عن دالة محدثة و مشتقة من تشفير تدفق (Rabbit) للتحول الى دالة تجزئة خفيفة الوزن لتكون متناسبة و مخصصة الى انظمة الراعاية الصحية القائمة على تقنية البلوكتشين. وهي عبارة عن شفرة تشفير خفيفة الوزن مُقنّعة بشكل رقيق، حُوّلت إلى دالة تجزئة ذات خصائص انتشار قوية، وسلوك انهيار مُحسّن، وقدرة على العمل في (Merkle Tree). تم عمل مقارنة هذه الدراسة المُقترحة بخوارزمية SHA-256 باستخدام مقاييس تشفير قياسية، ومشغلات شبكة تتكون من ١٠٠ و٥٠٠ و١٠٠٠ مُدخل من حيث تأثير الانهيار، ومسافة هامينغ، ومتوسط الانحراف المعياري. تُظهر النتائج أن (Rabbit-256) يوفر هامش أمان أقوى وتعقيدًا حسابيًا أقل، مما يُظهر أنه حل فعال لأجهزة إنترنت الأشياء المحدودة مع الحفاظ على متانة التشفير. وبينما يُجرى العمل الحالي في المحاكاة، يُمهد Rabbit-256 الطريق لتحقيق أمن سجلات الصحة الإلكترونية وبيانات أجهزة الاستشعار الطبية في إنترنت الأشياء والخدمات السريرية على تقنية البلوك تشين. وفي المستقبل، سنعمل على تطوير تطبيقات الأجهزة، وتحسين استهلاك الطاقة، والتكامل بما يتوافق مع لوائح الرعاية الصحية مثل قانون التأمين الصحي والمساءلة (HIPAA) واللائحة العامة لحماية البيانات (GDPR).