**ISSN: 0067-2904**

# Enhancing Blockchain Security by Developing the SHA256 Algorithm

**Raghad K. Salih*[1,2], Ali H. Kashmar[3]**
[1]*Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq*
[2]*Department of Applied Sciences, University of Technology, Baghdad, Iraq.*
[3]*Al Farabi University College, Cyber Security Department*

Received: 14/6/2023 Accepted: 23/8/2023 Published: 30/10/2024

**Abstract**

Security plays a vital role in various domains, including blockchain technology. The Blockchain serves as a secure data structure for storing transactional records. Hash functions are employed in cryptography to ensure integrity and authentication within the blockchain. The widely used SHA256 algorithm has faced recent attacks, prompting the development of stronger hash functions. This paper presents a novel modification approach to enhance the performance of SHA256 by introducing an extended mechanism for generating a 288-bit message digest and reducing the number of rounds to 44 instead of 64 while preserving the diffusion of data through its complex iterative process, which involves multiple rounds of bitwise and logical operations. The change makes sure that even small changes to the input data cause noticeable variations in the output hash, thereby maintaining cryptographic properties. The suggested hash function SHA288 achieves improved security, collision resistance, and preimage resistance, while maintaining a faster execution time compared to SHA256. The tables and tests conducted on the suggested algorithm have revealed its remarkable safety and robustness in countering attacks as well as demonstrated outstanding performance in random tests, which further enhances its security measures.

**Keywords:** Hash function, SHA256, collision resistant, NIST tests of randomness and rounds.

# تعزيز أمان سلسلة الكتل من خلال تطوير خوارزمية SHA256

**رغد كاظم صالح*[1،2], علي حبيب كشمر[3]**
[1] قسم الرياضيات, كلية العلوم, جامعة بغداد, بغداد, العراق.
[2] قسم العلوم التطبيقية, الجامعة التكنولوجية, بغداد, العراق.
[3]كلية الفارابي الجامعة ، قسم الامن السبراني

**الخلاصة**

يلعب الأمن دورًا حيويًا في مختلف المجالات ، بما في ذلك تقنية سلسلة الكتل. تعمل سلسلة الكتل كهيكل بيانات آمن لتخزين سجلات المعاملات. يتم استخدام وظائف دالة التجزئة في التشفير لضمان النزاهة والمصادقة داخل سلسلة الكتل. واجهت خوارزمية SHA256 المستخدمة على نطاق واسع هجمات حديثة ،

*Email: Raghad.k.Salih@uotechnology.edu.iq

مما أدى إلى تطوير وظائف دالة تجزئة أقوى. يقدم هذا البحث طريقة تعديل جديدة لتحسين أداء SHA256 من خلال تقديم آلية موسعة لتوليد ملخص رسالة 288 بت وتقليل عدد الجولات إلى 44 بدلاً من 64 مع الحفاظ على انتشار البيانات من خلال عملية تكرارية معقدة تتضمن جولات متعددة من العمليات الأحادية والمنطقية. يضمن التغيير أنه حتى التغييرات الصغيرة في بيانات الإدخال تسبب اختلافات ملحوظة في تجزئة الإخراج, وبالتالي الحفاظ على خصائص التشفير. تحقق وظيفة التجزئة المقترحة SHA288 أمانًا محسنًا ومقاومة للتصادم ومقاومة ما قبل الصورة ، مع الحفاظ على وقت تنفيذ أسرع مقارنة بـ SHA256. كشفت الجداول والاختبارات التي أجريت على الخوارزمية المقترحة عن سلامتها وقوتها الملحوظين في التصدي للهجمات ، كما أظهرت أداءً متميزًا في الاختبارات العشوائية ، مما يعزز إجراءاتها الأمنية.

## 1. Introduction

A mathematical hash function is a cryptographic function, which accepts input messages of arbitrary size and outputs a fixed-size result known as the hash value or digest. The output is usually a combination of numbers and letters that act as a distinct digital "fingerprint" for the input information. Because of their strong anti-collision, pre-image and $2^{nd}$ pre-image resistance properties, hash functions are widely utilized in computer science and cryptography for a range of tasks such as verifying data integrity, storing passwords, blockchain transactions, and creating digital signatures. Second preimage resistance refers to the promise that it is computationally impossible to identify another input message that generates the same hash value as a given message. Collision resistance means that finding two different input messages that hash to the same output is computationally impossible. A perfect hash with an m-bit digest should take about $2^m$ operations to find a pre-image or a $2^{nd}$ pre-image, while finding a collision with a birthday attack requires about $2^{(m/2)}$ operations [1,2,3].

The output of the hash function must be deterministic, which means that for a given input, the output will always be the same. Secure Hash Algorithm 256-bit (SHA256) is one kind of hashing algorithms. The National Security Agency (NSA) of the United States created the SHA256 in 2001. It is frequently used in digital security applications to guarantee data integrity, authenticity, and confidentiality. It is a portion of the SHA-2 family. Because it is a one-way function, figuring out the input data from the hash value is computationally impossible. This characteristic makes SHA-256 practical for utilizes like password storage and digital signatures. SHA-256 is widely utilized in many sectors, including finance, healthcare, and government. It is regarded as being secure and reliable overall. In SHA-256, the message is split into blocks each one has 512-bit, which are then processed in 64 rounds utilizing a variety of bitwise operations and nonlinear functions, such as logical functions (AND, OR, XOR), arithmetic functions (ADD, SHIFT). During each round, the block is partitioned into sixteen words each one has 32-bit, and these words are used to update a set of eight 32-bit variables, known as the working variables [1,4,5]. This work aims to increase the security and effectiveness of the SHA256 algorithm, resulting in a more secure and reliable blockchain system.

This paper is designed as follows: Section 2 focuses on some previous work. Section3 describes the blockchain technology while Section 3 covers the details of SHA256 algorithm, the proposed method is presented in section 5. Section 6 discusses experimental results. section 7 presents the security analysis of the suggested algorithm. Finally, section 8 is devoted to the conclusions of the paper.

## 2. Related Work

In 2013, Algredo-Badillo et al. [6] proposed four novel hardware implementation schemes aimed at enhancing the performance of SHA256. The architectures focus on optimizing the inner loop computation by rearranging operations, precomputing certain values, balancing paths, and integrating additional registers to reduce path complexity. These modifications are achieved without increasing clock cycles.

In 2018, Rachmawati1 et al. [7] made a comparative study and found that the intricacy of SHA256 and MD5 algorithms was equal. However, MD5 demonstrated faster running times compared to SHA256.

In 2019, Xenya et al. [8] proposed the use of blockchain technology and cryptographic schemes to address fraudulent activities in accounting records. By employing the SHA256 algorithm for hashing records and enforcing a structure that prevents data manipulation, the system efficiently maintains the integrity of forensic accounting records, ensuring trust in digital evidence.

In 2021, Fotohi et al. [9] proposed a technique to enhance communication security among devices on a blockchain network through two procedures: node authentication using identity-based signatures and posting blocks utilizing device identities as public-keys for hashing. It improves scalability, throughput, average detection rate, and authentication time.

In 2022, Salagrama et al. [10] presents a blockchain-based method for data integrity assurance, where the message verification code is saved within blockchain blocks. The method involves encrypting the SHA256 value using public-key cryptosystem with the receiver's public-key, along with including a timestamp and nonce. Through comparisons with existing methods and penetration testing, the proposed approach showcases the superior strength and robustness of blockchain for ensuring data integrity. Al-Odat et al. [11] enhanced secure hash design by amalgamating SHA-1 and SHA-2, protecting against collision and length expansion attacks. It amalgamates the hash functions' round steps utilizing function manipulators. The design was verified by resistance collision and length extension attacks. Mohanty et al. [12] presented a hospital administration system that securely processes patient data for effective registration and diagnosis. Patient details are encrypted using a 256-bit SHA-256 hash value, ensuring secure data management. The system treats concerns about safety in long data encryption for insurance and medical data.

In 2023, Kumar et al. [13] proposes a secure and decentralized service recommendation framework utilizing SHA256 and blockchain. The system ensures privacy and offers personalized suggestions by combining blockchain technology and the SHA256 algorithm. Simulation trials validate its effectiveness in providing customized services while maintaining user privacy and security.

The objective of this work is to enhance the security and efficiency of blockchain technology through the enhancement of the SHA256 algorithm. The primary goals of this work are to fortify the algorithm's resistance against attacks, particularly collisions and brute force attacks. This will be achieved by increasing the output size to 288 bits, which will significantly bolster its ability to withstand potential threats. Simultaneously, we aim to optimize the algorithm's performance, aiming to reduce its overall running time while maintaining the randomness of the output digest and ensuring a secure cryptographic process.

## 3. Blockchain Technique

Blockchain is a transparent, decentralized digital ledger that records and Validates transactions made among several computers. It employs hash functions and cryptographic systems to construct a chain of blocks, each consisting of a timestamped and immutable record. It provides data storage that is secure, impenetrable, and auditable without relying on a centralized authority [4,5]. The hash functions are essential to preserving the integrity and security of the blockchain. In this work, the suggested hash algorithm SHA288 was utilized to boost the integrity and robust security of the blockchain. Every block in the chain of a blockchain contains a hash value that serves as a representation of the data it contains. The block's unique identifier is this hash value, which is produced by a hash algorithm. The interconnection of these hash values is what gives the blockchain its strength. Any alteration to the data within a block would result in a different hash value because the hash calculation takes the hash value of the preceding block into account. As any effort at tampering would be rapidly discovered by the network, this process ensures that the blockchain stays tamper-resistant and immutable, maintaining the security and integrity of the stored data as shown in Figure (1) [14,15].

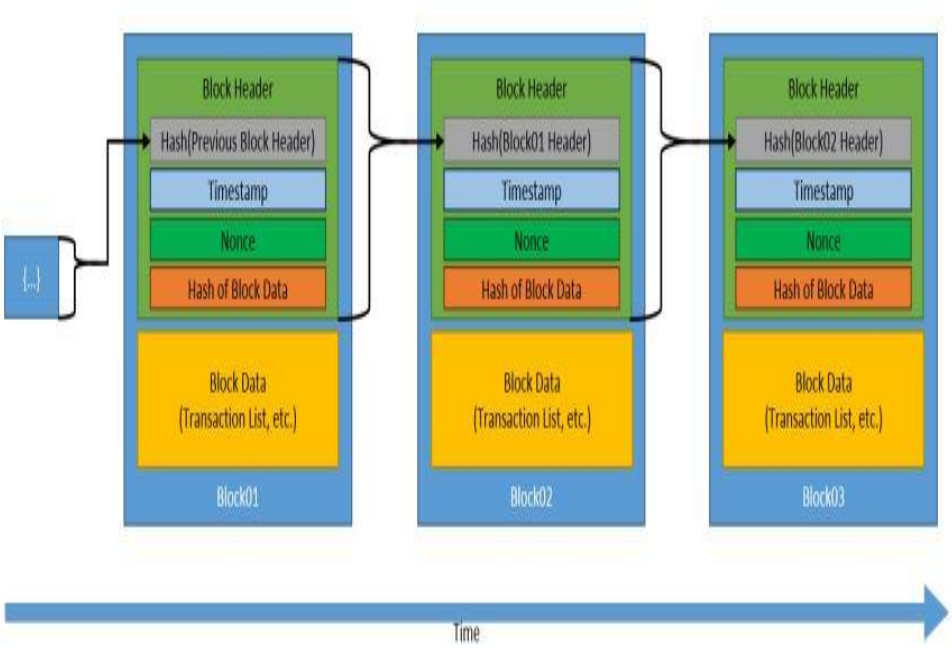


**Figure 1 :**Diagram showing how to link a blockchain.

## 4. SHA256 Function

SHA-256 belongs to the SHA-2 family. It is a robust cryptographic hash function and produces a message that has 256 bits called digest, which serves as a very secure manner of any given input message. SHA-256 is widely regarded for its ability to guarantee data safety and integrity across numerous applications. SHA-256 padding adds extra bits to the end of a message before hashing. It enforces that the message length is adjusted to be a precise multiple of the block size of 512 bits. SHA-256 creates a 256-bit hash by processing a padded message. Padding begins with a 1 bit, followed by zeros. A 64-bit integer representing the original message's length is appended. Since the hash function only works on 32-bit words and the starting hash values are eight 32-bit words in hexadecimal notation as explained in eq.(1), the modulo sum utilized in SHA-256 is $2^{32}$ [1,6].

$$\left.\begin{aligned} h_1^{(0)} &= A = 6a09e667 \\ h_2^{(0)} &= B = bb67ae85 \\ h_3^{(0)} &= C = 3c6ef372 \\ h_4^{(0)} &= D = a54ff53a \\ h_5^{(0)} &= E = 510e527f \\ h_6^{(0)} &= F = 9b05688c \\ h_7^{(0)} &= G = 1f83d9ab \\ h_8^{(0)} &= H = 5be0cd19 \end{aligned}\right\} \quad \ldots(1)$$

SHA-256 accepts a message of maximum length $2^{64} - 1$ bits, which is divided into n-blocks, $n \geq 1$, each one has length 512 bits such that ($m_1$, $m_2$, ... , $m_n$), then each $m_j$ block is split into 16 words each one has 32 bits as $m_j$= $w_1$, $w_2$…….$w_{16}$, $1 \leq j \leq n$. These 32 bit words are then extended into sixty- four words each word contains 32 bits as in eq.(1) [1].

$$W_i = \begin{cases} w_i & 1 \leq i \leq 16 \\ \sigma_1^{(256)}(w_{i-2}) + w_{i-7} + \sigma_0^{(256)}(w_{i-15}) + w_{i-16} & 17 \leq i \leq 64 \end{cases} \quad \ldots(2)$$

where

$$\sigma_0^{(256)}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \quad \ldots(3)$$

$$\sigma_1^{(256)}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \quad \ldots(4)$$

$ROTR^N(x)$ is the rotate right operation of $x$ by $N$ positions to the right and $SHR^N(x)$ is the right shift operation of $x$ ($SHR^N(x)=x>>N$).

After that, the blocks $m_1$, $m_2$,..., $m_n$ are processed one by one as follows:

For each j= 1,2,3,…,n construct the update eight working variable by doing 64 rounds consisting of:

$$\left.\begin{aligned} T_1 &= h + \sum_1^{(256)}(E) + Ch(E,F,G) + K_i^{(256)} + W_i \\ T_2 &= \sum_0^{(256)}(A) + Maj(A,B,C) \\ H &= G \\ G &= F \\ F &= E \\ E &= D + T_1 \\ D &= C \\ C &= B \\ B &= A \\ A &= T_1 + T_2 \end{aligned}\right\} \quad \ldots(5)$$

Where $W_i$ in eq.(2) , the SHA-256 keys $K_i$ , $i = 1, \ldots, 64$ are constant values,

$$\sum_0^{(256)}(A) = ROTR^2(A) \oplus ROTR^{13}(A) \oplus ROTR^{22}(A), \quad \ldots(6)$$

$$\sum_1^{(256)}(E) = ROTR^6(E) \oplus ROTR^{11}(E) \oplus ROTR^{25}(E), \quad \ldots(7)$$

$$Ch(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G), \quad \ldots(8)$$

$$Maj(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C) \quad \ldots(9)$$

and the update values can be obtained by using new values A,B,…,H as:

$$\left.\begin{aligned} h_1^{(j)} &= A + h_1^{(j-1)} \\ h_2^{(j)} &= B + h_2^{(j-1)} \\ h_3^{(j)} &= C + h_3^{(j-1)} \\ h_4^{(j)} &= D + h_4^{(j-1)} \\ h_5^{(j)} &= E + h_5^{(j-1)} \\ h_6^{(j)} &= F + h_6^{(j-1)} \\ h_7^{(j)} &= G + h_7^{(j-1)} \\ h_8^{(j)} &= H + h_8^{(j-1)} \end{aligned}\right\} \quad \ldots(10)$$

After the final block $m_n$ has been processed, the output hash $h_k^{(n)}, k = 1,2, \ldots 8$ is the string of the variables $SHA_{256} = h_1^{(n)} h_2^{(n)} h_3^{(n)} h_4^{(n)} h_5^{(n)} h_6^{(n)} h_7^{(n)} h_8^{(n)}$. Figure (2) describes how SHA256's algorithm works in two iterations [1,6,8].

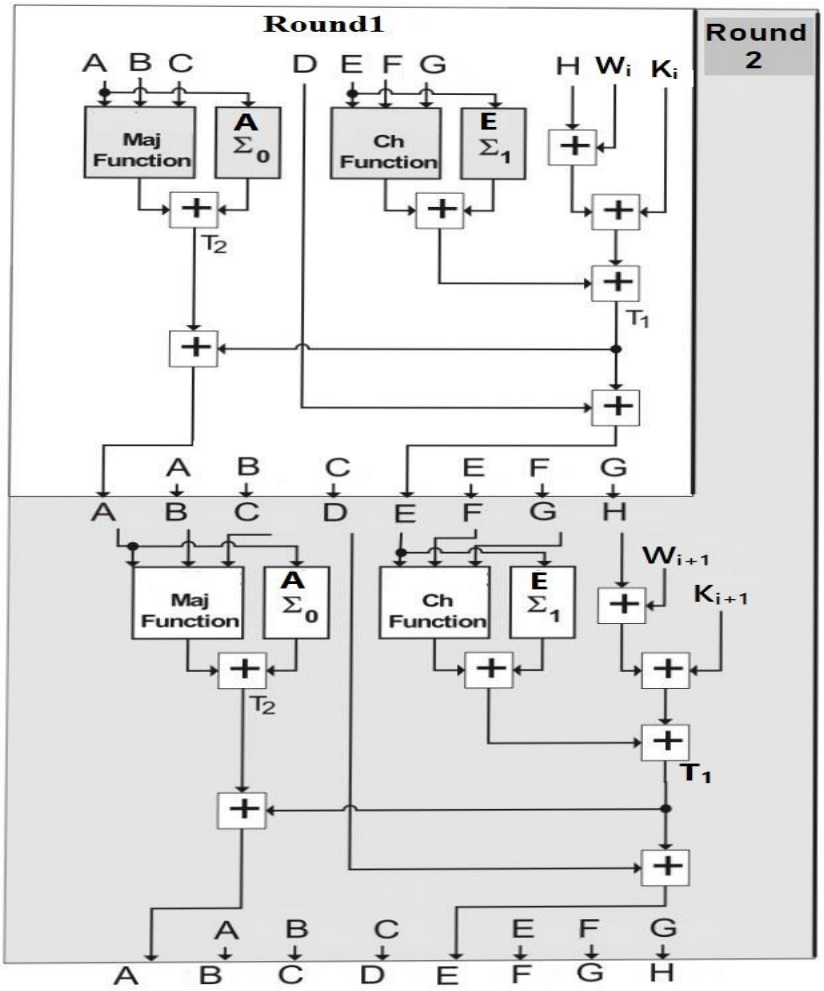


**Figure 2:** The Process of the SHA-256 algorithm involves two iterations.

## 5. The Proposed Method

The new suggested hash algorithm partakes a similar word size as SHA-256. and accepting a message of maximum length $2^{64} - 1$ bits which is divided into n-blocks ($m_1$, $m_2$, ... , $m_n$) each one has length 512 bits then each $m_j$ block is split into 16 words each one has 32

bits as $m_j = w_1, w_2, \ldots w_{16}$, $1 \leq j \leq n$. These 32-bit words are then extended into sixty- four words each word contains 32 bits, which are given as input to the round function. To bolster the algorithm's security, the number of intermediary state variables was augmented by one to be a total of nine. This enhancement donates a message digest length of 288 bits. The objective behind this modification is to fortify the algorithm's resilience against prospective attacks, thereby promoting its overall security by creating more bit variation in each working variable. A new modification approach is suggested to improve the functionality of SHA256 and address its time complexity. This approach reduces the number of rounds from 64 to 44, effectively streamlining the computation. Additionally, an extended mechanism is introduced to generate a larger 288 bit-length message digest, thereby strengthening the algorithm's security. Figure (3) shows the construction of the proposed hash function SHA288 while Figures (4) and (5) explained one iteration of it. In the proposed work, changes have been made to the working variables of the SHA-256 algorithm as well as the function mechanization and procedural aspects of the rounds keeping in mind maintaining the complexity and distribution of input data to augment security. These changes are outlined in the SHA288 algorithm steps.

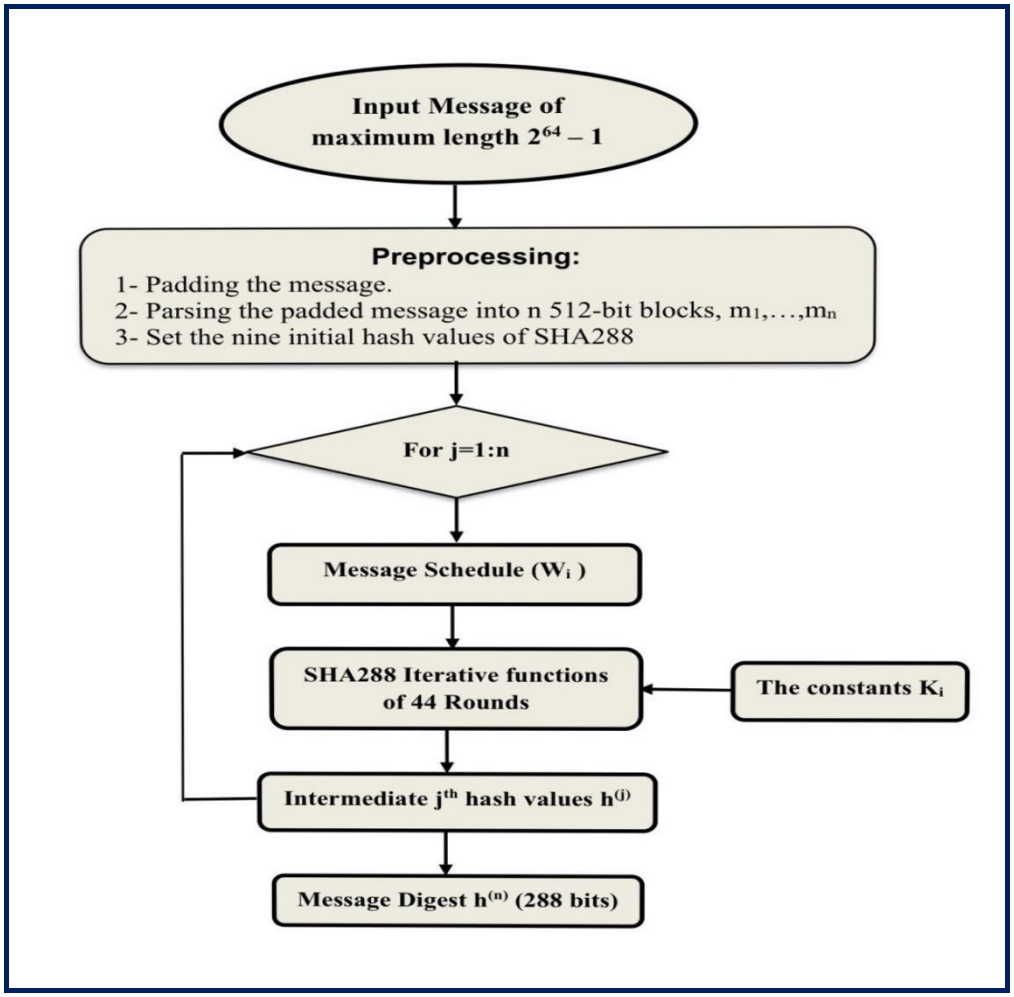


**Figure 3:** The proposed SHA288 construction.

***The SHA288 Algorithm***

**Input**

- The message M

**Output**

- The hash value in hexadecimal.

**1: 1:** affix the message with padded bits to make sure it is compatibility with 448 mod 512. A solo 1 bit is attached at the terminus and then more zeros are added to bring the overall length to a multiple of 512. considering the input data is 64 bits.

**2:** Partition the input by using (step1) into (n) blocks ($m_1$, $m_2$, ... , $m_n$) each one has 512 bits as described in section 4.

**3:** Initialize hash value by adding another initial value to eq.(1) to create a set of nine 32-bit words, in a hexadecimal pattern that is:

$$h_9^{(0)} = I = 6b6f58a9$$

**4:** Once the pre-processing is finished, each message block, starting from $m_1$ to $m_n$, is sequentially processed utilizing the below steps:

**i.** For j=1 to n

**ii.** Set up a message schedule, $W_i$ as:

$$W_i = \begin{cases} w_i & 1 \le i \le 16 \\ \sigma_1^{(256)}(w_{i-2}) + w_{i-7} + \sigma_0^{(256)}(w_{i-15}) + w_{i-16} & 17 \le i \le 40 \\ \sigma_1^{(256)}(w_{i-3}) + w_{i-7} + w_{i-5} + w_{i-17} & 41 \le i \le 64 \end{cases}$$

**iii.** Initialize the nine intermediary state variables A, B, C, D, E, F, G, H and I by using $(j-1)^{st}$ hash value in (step 3):

$$\left.\begin{aligned} A &= h_1^{(j-1)} \\ B &= h_2^{(j-1)} \\ C &= h_3^{(j-1)} \\ D &= h_4^{(j-1)} \\ E &= h_5^{(j-1)} \\ F &= h_6^{(j-1)} \\ G &= h_7^{(j-1)} \\ H &= h_8^{(j-1)} \\ I &= h_9^{(j-1)} \end{aligned}\right\}$$

**iv.** Compute the following:

For i= 1 to 64

If i = 2,4,6,…,40 then find:

$$T_1 = I + \sum_1^{(256)}(E) + Ch(E,F,G) + K_{i-1}^{(256)} + W_{i-1}$$

$$T_2 = \sum_0^{(256)}(A) + Maj(A,B,C) + K_i^{(256)} + W_i$$

$I = H + T_2$

$H = G$

G = F
F = E
E = D + $T_1$
D = C
C = B
B = A
A = $T_1$ + $T_2$
Elseif i = 41 to 64 then find

$$T_1 = \text{I} + \sum_{1}^{(256)} (E) + Ch(E,F,G) + K_i^{(256)} + W_i$$

$$\text{T}_2 = \sum_{0}^{(256)} (A) + Maj(A,B,C)$$

I = H
H = G
G = F
F = E
E = D + $T_1$
D = C
C = B
B = A
A = $T_1$ + $T_2$

where $W_i$ is explained in (step 4(ii)) and $K_i$ are constant values of SHA-256 keys. Figures (4) and (5) explained one iteration of SHA288.

**v.** Calculate the j[th] intermediate hash value as:

$$h_1^{(j)} = A + h_1^{(j-1)}$$
$$h_2^{(j)} = B + h_2^{(j-1)}$$
$$h_3^{(j)} = C + h_3^{(j-1)}$$
$$h_4^{(j)} = D + h_4^{(j-1)}$$
$$h_5^{(j)} = E + h_5^{(j-1)}$$
$$h_6^{(j)} = F + h_6^{(j-1)}$$
$$h_7^{(j)} = G + h_7^{(j-1)}$$
$$h_8^{(j)} = H + h_8^{(j-1)}$$
$$h_9^{(j)} = I + h_9^{(j-1)}$$

**vi.** Write the sha288 bit digest output of the message M after the final block $m_n$ has been processed: $SHA_{288} = h_1^{(n)}\ h_2^{(n)}\ h_3^{(n)}\ h_4^{(n)}\ h_5^{(n)}\ h_6^{(n)}\ h_7^{(n)}\ h_8^{(n)} h_9^{(n)}$

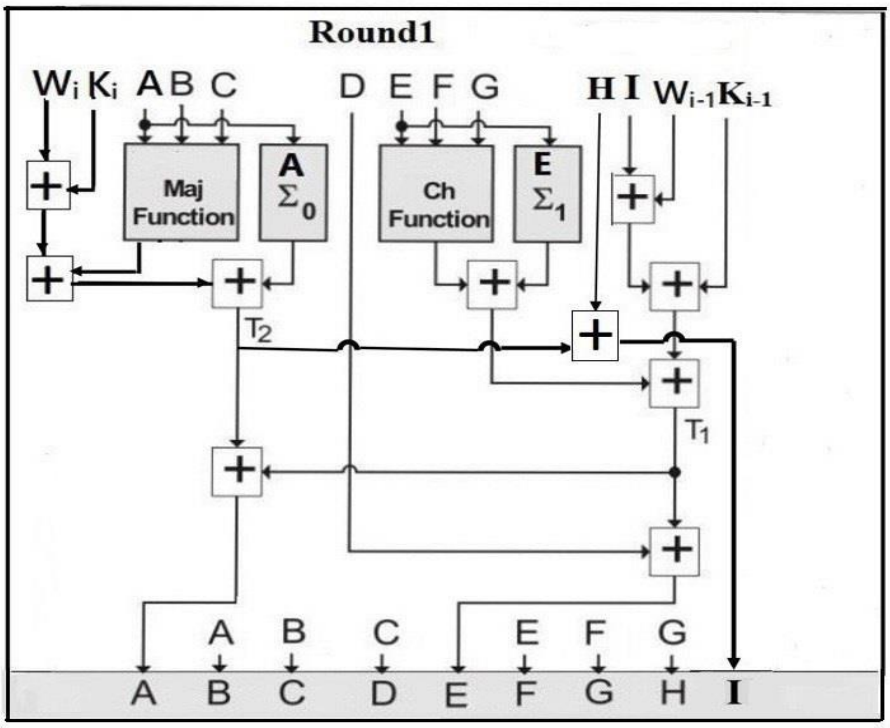


**Figure 4:** One iteration process of the SHA288 algorithm where i=2,4,…,40

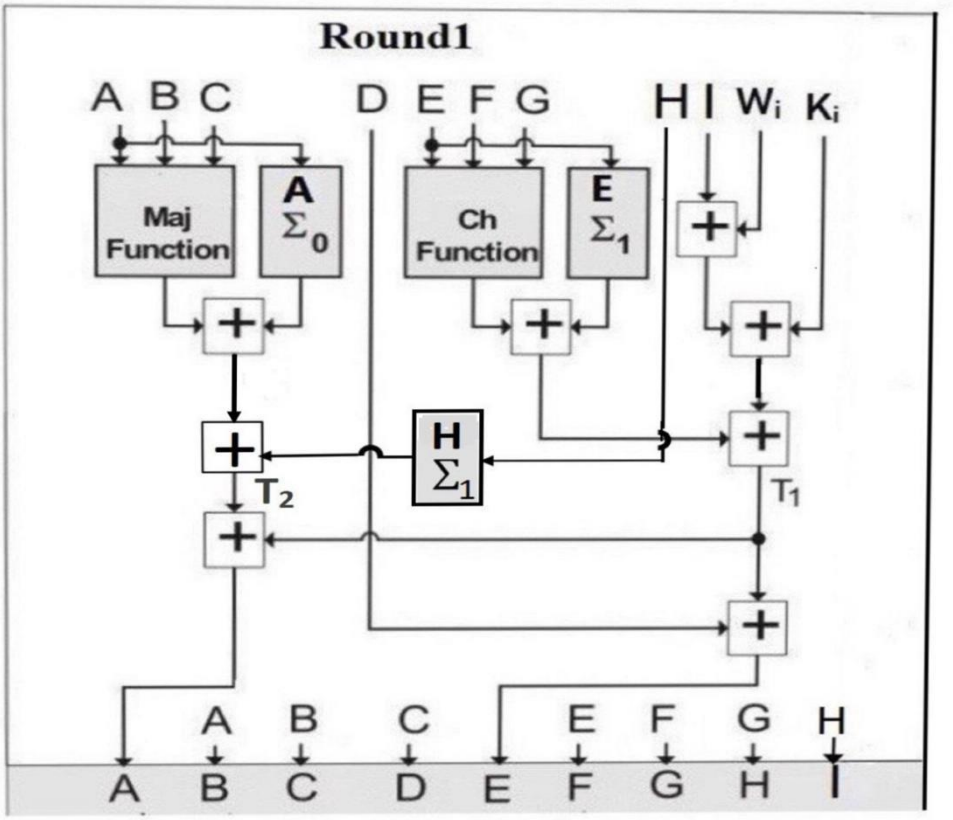


**Figure 5**: One iteration process of the SHA-288 algorithm where i=41,42,43,…,64

## 6. Experimental Results

The modified algorithm was subjected to rigorous testing using various messages of varying lengths. The results demonstrated notable improvements in algorithm efficiency, where the changeover mechanism ensures that even minor alterations in the input data lead to entirely distinct hash outputs, thereby preserving the robust and stringent properties of the cryptographic, with reduced execution times compared to the original SHA256 algorithm. Interestingly, despite generating longer message digests, the modified algorithm SHA288 offered superior performance. This observation is summarized in Table (1) below, showcasing the comparative results of the two algorithms.

**Table 1:** Message Digest and Running Time for Different Message Lengths

| No. | Message Type (M) | SHA256 | SHA288 | The elapsed time in seconds | |
|---|---|---|---|---|---|
| | | | | SHA256 | SHA288 |
| 1 | Abc | 06D90109C8CCE34EC0C776950465421E176F08B831A938B3C6E76CB7BEE8790B | 15A40AA2393D0FF4728227C259BE6BC290A12C0A7934E95BA1C38A0D3D7A080C05D26763 | 0.18858 | 0.162676 |
| 2 | abc | BA7816BF8F01CFEA414140DE5DAE2223B00361A396177A9CB410FF61F20015AD | F98AA371582AB12577AF1FC8D70B63E0AAD98FB33756C307B897D40CC3D8BA3E44EE02BE | 0.18896 | 0.163474 |
| 3 | 0000000000aaaaaa | 61321C480217CE2E2D278A563DB3C8696D92AE60C1578723E9DF38B73916EE43 | 5C941018B62697715D638704BFB0E56BAC3DA433CCA228BC802F9F925314EC14C35DC13E | 0.235469 | 0.182458 |
| 4 | 000000000aaaaaa | 7FE1C85C162FD58E04A25CC6564275877E33DEEB1A8D6FC6B978D18052873D58 | D2248506DABE6293228212CD6857CB5AF591B529EE6C2566C0E78D5F8C3C6FA57BF2B4B3 | 0.199469 | 0.179458 |
| 5 | 0000000000aaaaab | EC83DED5BEEE4858884851CCE32D25F1C445349A1829DE5EBA083268423DD22E | 68A44312F4455F327DC34BB4CC9A67F1AE331FA066CA28F283A171EB7E65401B73C56303 | 0.215598 | 0.180807 |
| 6 | 1000 bits | E133A99B761F90C353BD9BEED497146006AF756F5BCD838517072218E1B25749 | F3A0E02ACB6DE9217F0ADC54080878C2E25CAE6175EE5DA77F21B538A60A3CEA3284F43F | 2.591164 | 2.261733 |
| 7 | 10000 bits | E485FCD54BB04C357E7D37B69420C2FAE3A2E5AB37272A39877E15CE068ABCD7 | AC54076E80E67C081A8C8F2F6161393491610544B4ECBD571CFB650BFDC4305D5D6AF4DD | 24.06546 | 21.00639 |
| 8 | 30000 bits | 7E771686B586F0829EF513BA397C382584938AA59B081CC59AF03BC3460CF53D | 8AF716F7DF0A8CFCBE870A0FE314C5BC238F7F520EC4A7315FD39C1E440870F39B2A64B9 | 111.0545 | 101.3534 |
| 9 | 100000 bits | 8062238FBE34EF9356DEB57DDCC9D98B6B0A3D1E419FC9DBCAC71720E90D47FE | 59472AF20DA2EC587775AAE11BC7C645A758BC4D9B7337C9DD25FAFD28D8A099EDF4EF25 | 548.5551 | 512.1653 |

## 7. Security Analysis

In this section, we will subject the proposed algorithm to a series of tests to verify its security and efficiency, as described below.

### *7.1 Collision Attack*

The collision attack is an attack on a cryptographic hash function with the objective of finding two or more distinct input messages that produce the same hash output [16]. The proposed SHA288 with 288-bit hash output provides a larger output space compared to SHA256's 256-bit output, contributing to its heightened resilience against collision attacks. The increased bit length of SHA288 enhances its ability to offer a higher level of collision resistance. In contrast, the comparatively shorter hash length of SHA256 renders it more prone to potential collisions, making it less secure in the face of advanced cryptanalytic techniques. Table (2) contains the comparison among SHA1, SHA256 and SHA288 which share the same functional structure with variations in the internal operations, hash digest size, rounds and number of security bits.

**Table 2:** Secure Hash Algorithms

| **Algorithm** | **SHA1** | **SHA256** | **SHA288** |
|---|---|---|---|
| Message size | $<2^{64}$ | $<2^{64}$ | $<2^{64}$ |
| Block size | 512 | 512 | 512 |
| Hash digest size | 160 | 256 | 288 |
| Rounds | 80 | 64 | 44 |
| Preimage attack | $2^{160}$ | $2^{256}$ | $2^{288}$ |
| Second Preimage attack | $2^{160}$ | $2^{256}$ | $2^{288}$ |
| Collision complexity (Security) | $2^{80}$ | $2^{128}$ | $2^{144}$ |

### 7.2 *Brute Force Attack*

A brute force attack against a hash function involves systematically trying all possible inputs to discover the one that produces a specific hash value, with the intention of revealing the original input data [17]. Based on the analysis of simulation results presented in Table (1), it has been observed that SHA-288 exhibits superior cryptographic strength and resistance against brute force attacks compared to SHA-256. This can be attributed to SHA-288's larger output space of 288 bits, which allows for a wider range of possible hash values. Notably, even minor changes in the input of SHA-288 lead to significantly different outputs, indicating its heightened sensitivity to input variations. These findings emphasize the increased security provided by SHA-288, making it a more robust choice for data protection compared to SHA-256.

### *7.3 Running Time*

From the observations in Table (1), it is evident that the execution time of SHA288 is shorter than that of SHA256, even though the resulting message is longer. This improvement can be attributed to two factors: first, the reduction in the number of rounds from 64 to 44 in the SHA288 algorithm; second, the modifications made to the mathematical functions used in SHA288, which enhance its efficiency and security. These advancements in SHA288 highlight its superior performance in terms of efficiency and security when compared to SHA256. Figure (6) visually represents the difference in running times between two hash functions. It effectively displays how the execution times vary with an increase in the length of the message. This figure emphasizes the notable contrast in performance between the two algorithms, highlighting the impact of message length on their respective execution times.

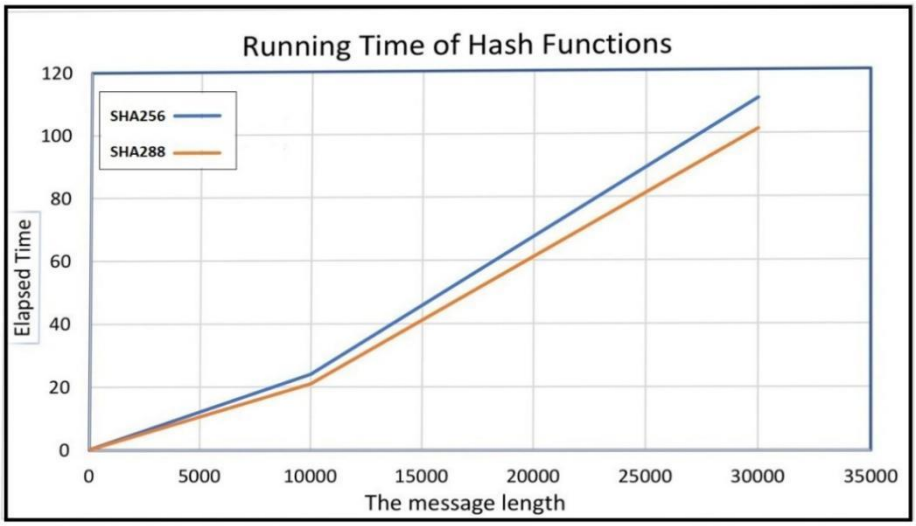


**Figure 6:** The running time of the hash functions SHA288 and SHA256

*7.4 NIST Statistical Suite Tests*

The NIST Test is a statistical program developed by the National Institute of Standards and Technology (NIST). It consists of 16 tests designed to assess the randomness and statistical properties [18] of a bit sequence generated by the SHA288 hash function. The test evaluates various aspects of randomness, ensuring that the SHA288 hash function meets the required standards. For the NIST Test, the input parameters used in the 16 tests are derived from multiple message files of different lengths. The goal is to evaluate the performance of the SHA288 hash function across various scenarios. The test results are presented in Tables (3), (4), and (5), where the percentage of passed sequences for each individual test is computed. These results are then compared with the appropriate P-value > 0.01 to determine the level of statistical significance.

**Table 3:** Presents NIST test results for hash functions SHA256 and SHA288 on message1 / Table 1.

| **The Message *abc*** | | | | |
|---|---|---|---|---|
| **Hash Function** | **SHA256** | | **SHA288** | |
| **Statistical Tests** | ***P*- Values** | **Result** | ***P*- Values** | **Result** |
| Frequency Monobit | 0.317 | Pass | 0.555 | Pass |
| Block Freq. | 0.535 | Pass | 0.731 | Pass |
| Runs | 0.068 | Pass | 0.983 | Pass |
| L. Run of Ones | 0.712 | Pass | 0.833 | Pass |
| Binary MatrixRank | -1 | fail | -1 | fail |
| DFT Spectral | 0.422 | Pass | 0.9138 | Pass |
| Non Overlapping Templates | 0.0001 | fail | 0.999 | Pass |
| Overlapping Templates | NaN | fail | NaN | fail |
| Maurer's Universal Statistical | -1 | fail | -1 | fail |
| Linear Complexity | 0.921 | Pass | 0.264 | Pass |
| Serial | 0.841 | Pass | 0.083 | Pass |
| Appro. Entropy | 1 | Pass | 1 | Pass |
| Cumulative sums Forward<br>Cumulative sums Reverse | 0.469<br>0.236 | Pass<br>Pass | 0.792<br>0.737 | Pass<br>Pass |
| Rand. Excursions | 0.604 | Pass | 0.350 | Pass |
| Rand. Excursions Variant | 0.288 | Pass | 0.789 | Pass |

**Table 4**: Presents NIST test results for hash functions SHA256 and SHA288 on message4 / Table 1.

| **The Message 000000000aaaaaa** | | | | |
|---|---|---|---|---|
| **Hash Function** | **SHA256** | | **SHA288** | |
| **Statistical Tests** | ***P*- Values** | **Result** | ***P*- Values** | **Result** |
| Frequency Monobit | 0.802 | Pass | 0.813 | Pass |
| Block Freq. | 0.731 | Pass | 0.314 | Pass |
| Runs | 0.534 | Pass | 0.076 | Pass |
| L. Run of Ones | 0.781 | Pass | 0.196 | Pass |
| Binary MatrixRank | -1 | fail | -1 | fail |
| DFT Spectral | 0.051 | Pass | 0.130 | Pass |
| Non Overlapping Templates | 0.999 | Pass | 0.999 | Pass |
| Overlapping Templates | NaN | fail | NaN | fail |
| Maurer's Universal Statistical | -1 | fail | -1 | fail |
| Linear Complexity | 0.759 | Pass | 0.499 | Pass |
| Serial | 0.841 | Pass | 0.498 | Pass |
| Appro. Entropy | 1 | Pass | 1 | Pass |
| Cumulative sums Forward<br>Cumulative sums Reverse | 0.687<br>0.906 | Pass<br>Pass | 0.524<br>0.350 | Pass<br>Pass |
| Rand. Excursions | 0.434 | Pass | 0.497 | Pass |
| Rand. Excursions Variant | 0.844 | Pass | 1 | Pass |

**Table 5:** Presents NIST test results for hash functions SHA256 and SHA288 on message 9 / Table 1.

| **The Message 100000 bits** | | | | |
|---|---|---|---|---|
| **Hash Function** | **SHA256** | | **SHA288** | |
| **Statistical Tests** | ***P*- Values** | **Result** | ***P*- Values** | **Result** |
| Frequency Monobit | 0.211 | Pass | 0.157 | Pass |
| Block Freq. | 0.404 | Pass | 0.560 | Pass |
| Runs | 0.511 | Pass | 0.191 | Pass |
| L. Run of Ones | 0.272 | Pass | 0.718 | Pass |
| Binary MatrixRank | -1 | fail | -1 | fail |
| DFT Spectral | 0.818 | Pass | 0.665 | Pass |
| Non Overlapping Templates | 0.053 | Pass | 0.999 | Pass |
| Overlapping Templates | NaN | fail | NaN | fail |
| Maurer's Universal Statistical | -1 | fail | -1 | fail |
| Linear Complexity | 0.239 | Pass | 0.498 | Pass |
| Serial | 0.498 | Pass | 0.498 | Pass |
| Appro. Entropy | 1 | Pass | 1 | Pass |
| Cumulative sums Forward<br>Cumulative sums Reverse | 0.378<br>0.091 | Pass<br>Pass | 0.251<br>0.174 | Pass<br>Pass |
| Rand. Excursions | 0.901 | Pass | 0.202 | Pass |
| Rand. Excursions Variant | 0.654 | Pass | 0.376 | Pass |

The comparative analysis between the SHA288 algorithm and SHA256 reveals improved performance in random tests, evident in Table (3). The SHA288 algorithm successfully passes 14 out of 16 tests, surpassing SHA256's achievement of 13 successful tests. Additionally, in other examples as in Tables (4) and (5), both algorithms achieve an equal number of successful tests.

Therefore, The SHA288 algorithm possesses several notable advantages compared to its predecessor, SHA256. It produces longer message digests, leading to enhanced security by expanding the output space and reducing the likelihood of collisions. Additionally, its faster execution time improves overall efficiency, making it more practical for real-world applications. Lastly, the algorithm's robust randomness properties provide resistance against various hacking techniques, thereby bolstering its overall protection capabilities.

## 8. Conclusions

The suggested algorithm SHA288 addresses the enhancement of security and efficiency in blockchain technology through improvements to the widely utilized SHA256 algorithm. The primary objectives encompass augmenting the security of SHA256's intermediary state variables by expanding the output size to 288 bits, and optimizing execution time by reducing the number of rounds to 44 iterations, while simultaneously enhancing the mechanization of working functions. The proposed modifications, meticulously detailed in tables 1 to 5 and figures 3 to 6, have undergone thorough testing to substantiate their robust security, exceptional anti-collision capabilities, resistance against preimage and second image attacks, as well as adherence to NIST tests of randomness.

## References

**[1]** Federal information processing standard (fips), " Sucre Hash Standard,"180-2. National Institute of Science and Technology, 2002.

**[2]** A. H. Jasim and A. H. Kashmar , "An Evaluation of RSA and a Modified SHA-3 for a New Design of Blockchain Technology," in *Artificial Intelligence for Smart Healthcare*, Cham: Springer International Publishing, 2023, pp. 477-489.

**[3]** H. Abdulsalam and A. A. Fahad, " Evaluation of Two Thresholds Two Divisor Chunking Algorithm Using Rabin Finger print, Adler, and SHA1 Hashing Algorithms," *Iraqi Journal of Science*, vol. 58, no.4c, pp.2438-2446, 2017.

**[4]** A.H. Kashmar, K. H. Ahmed and S. I. Eddie, "Hybrid chaotic keystream generation (HCKG) for symmetric image encryption," *Journal of theoretical and applied information technology*, vol.97, no.3, pp.984-993, 2019.

**[5]** H. k. Hussein, R. A. Muhajjar and B. S. Mahdi, "Using Visual Cryptography and hash function for Fragile Watermarking to Detect Electronic Document Forger." *Iraqi Journal of Science*, vol. 64, no.7, pp.4557-4569, 2023.

**[6]** I. Algredo-Badillo, C. Feregrino-Uribe, R. Cumplido and M. Morales-Sandoval, "FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256," *Microprocessors and Microsystems*, vol.37, no.6, pp.750-757, 2013.

**[7]** D. Rachmawati, J. T. Tarigan, and A. B. C. Ginting, " A comparative study of Message Digest 5 (MD5) and SHA256 algorithm," *Journal of Physics: Conference Series, IOP Publishing*, vol. 978, no.1, p. 012116, 2018.

**[8]** M.C. Xenya and K. Quist-Aphetsi., "A cryptographic technique for authentication and validation of forensic account audit using SHA256," in *International Conference on Cyber Security and Internet of Things (ICSIoT), IEEE*, pp. 11-14. 2019.

**[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," *Computer Networks*, vol.197, pp.108331, 2021.

**[10]** S. Salagrama, V. Bibhu and A. Rana, "Blockchain Based Data Integrity Security Management. Procedia Computer Science," vol.215, pp.331-339, 2022.
Available online at: www.sciencedirect.com .

**[11]** Z. A. Al-Odat, S. U. Khan and E. Al-Qtiemat, "A modified secure hash design to circumvent collision and length extension attacks," *Journal of Information Security and Applications*, vol.71, p.103376, 2022.

**[12]** M.D. Mohanty, A.Das, M.N. Mohanty, A. Altameem, S.R. Nayak, A.K. Saudagar and R.C. Poonia. "Design of Smart and Secured Healthcare Service Using Deep Learning with Modified SHA-256 Algorithm," In *Healthcare, MDPI*, vol. 10, no. 7, p. 1275, 2022.

**[13]** K.P Kumar, N. H. Varma, N. Devisree and M. S. Ali, "Implementation of Associative Service Recommendation Scheme Applying Sha256 Algorithm Through Blockchain," *Journal of Survey in Fisheries Sciences*, vol.10 no.2S, pp.2741-2747, 2023.

**[14]** A. A. S. Al-karkhi, N.F. Hassan and R. A. Azeez, "A Secure Private Key Recovery Based on DNA Bio-Cryptography for Blockchain," *Iraqi Journal of Science*, vol. 64, no.2, pp.958-972, 2023.

**[15]** R.F. Ghani, A.A.S. Al-Karkhi and S.M. Mahdi, "Proposed Framework for Official Document Sharing and Verification in E-government Environment Based on Blockchain Technology," *Baghdad Science Journal*, vol.19, no.6 (Suppl.), pp.1592-1602, 2022.

**[16]** R. Verma, N.Dhanda and V. Nagar, "Enhancing security with in-depth analysis of brute-force attack on secure hashing algorithms," in *Proceedings of Trends in Electronics and Health Informatics: TEHI 2021*, pp. 513-522. Singapore: Springer Nature Singapore, 2022.

**[17]** A.H. Alwan and A. H. Kashmar, "Block Ciphers Analysis Based on a Fully Connected Neural Network," *Ibn AL-Haitham Journal for Pure and Applied Sciences*, vol.36. no.1, pp.415-427, 2023.

**[18]** A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson et al., "A Statistical Test Suite for Random and Pseudo Random Number Generators for Cryptographic Applications," vol. 22. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2001.