Ibn Al-Haitham Journal for Pure and Applied Sciences
Journal homepage: jih.uobaghdad.edu.iq
PISSN: 1609-4042, EISSN: 2521-3407
IHJPAS. 2025, 38(4)

Development of the Hash Function Using Modified Skew Tent Map to Improve Blockchain Technology

Raghad K. Salih[1], **Ali A.Aubad**[2*], **Mohammed Yasin**[3] and **Hadi Hamad**[4]

[1,2] Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq.
[1] College of Applied Sciences, University of Technology, Baghdad, Iraq.
[3,4]Department of Mathematics, An-Najah National University, Nablus P400, Palestine
* Corresponding Author.

Received: 23 April 2025 Accepted: 24 July 2023 Published: 20 October 2025
doi.org/10.30526/38.4.4153

Abstract

Using SHA256 in the Blockchain system for security purposes, as it is important in linking blocks and preventing tampering efficiently and securely. In order to further confirm the security of SHA256 and protect it and increase its susceptibility to resist threats that it is exposed to in one way or another, its algorithm was developed by utilizing the modified skew tent map (MSTM). The developed SHA256 algorithm (D-SHA256) is distinguished by two essential features: less time and more enhanced security than its predecessor SHA256. This distinction arises from the strongly chaotic behavior and the highly randomness properties of the MSTM. Moreover, the proposed D-SHA256 algorithm consist of 32 rounds while preserving the randomness properties of the compression function by combining 48 hash constants and 48 words with the MSTM to obtain high randomness with less rounds. D-SHA256 guarantees that in the event of small changes that may occur in the input message leading to large changes in the output hash digest, while confirming the preservation of the properties of the cryptographic hash, containing collision resistance and ideal confusion and diffusion. The proposed algorithm was compared with SHA256 and other current hash algorithms, the results showed that D-SHA256 has increased collision resistance, higher output randomness, better cryptographic hashing properties, and lower execution time.

Keywords:SHA256, Blockchain, Skew tent chaotic map, NIST randomness tests, and Compression function.

1. Introduction

Blockchain is a decentralized and distributed ledger technology that securely records transactions across multiple computers in a way that prevents any changes or tampering (1, 2). It contains of:

- Data: Transaction information.
- Hash: A unique digital representation of the block.
- Previous Block Hash: Refers to the hash of the block that precedes it in the chain.

Blocks are linked using hashes, a numerical value extracted from the contents of the block, making it impossible to modify any block without changing all the blocks that follow it (2, 3). Blockchain systems are useful in many fields, such as healthcare, banking transactions, and many other applications that require a high level of security (2-6). **Figure 1** illustrated how to link a blockchain. SHA256 is the most widely used hash function in blockchains due to its efficient properties. Therefore, many researchers have improved the security of this function (7-9) and some of them have developed hash functions (10-12) and encryption methods (13, 14) by using chaotic maps (15, 16), because chaotic systems are highly sensitive to parameters and initial conditions and are characterized by random paths (17) with strong confusion and diffusion properties that satisfy Shannon's principles (18). However, many existing hash functions neglect running time computation, which is critical for efficient and reliable performance. Our work introduces a strong hashing algorithm to enhance the security of SHA256 and achieve a secure blockchain by preventing tampering of the join blocks. The developed SHA256 algorithm (D-SHA256) using modified skew tent map (MSTM) ensures that any modification to the data of any block changes its hash. Hence maintains the integrity of the information within each block and prevents any changes. D-SHA256 provides strong resistance to collisions and has efficient confusion and diffusion. D-SHA256 has 32 rounds, and by using MSTM in the compression function a balance between security and efficiency was achieve. Reducing D-SHA256 execution time improves data integrity verification. The programs for evaluating the performance of developed hash algorithm were implemented in MATLAB R2023b. This paper covers the following: Section 2 offers the developed hash algorithm. Section 3 offers the results, and Section 4 discusses the results of the proposed hash algorithm. Finally, section 5 concludes the work.

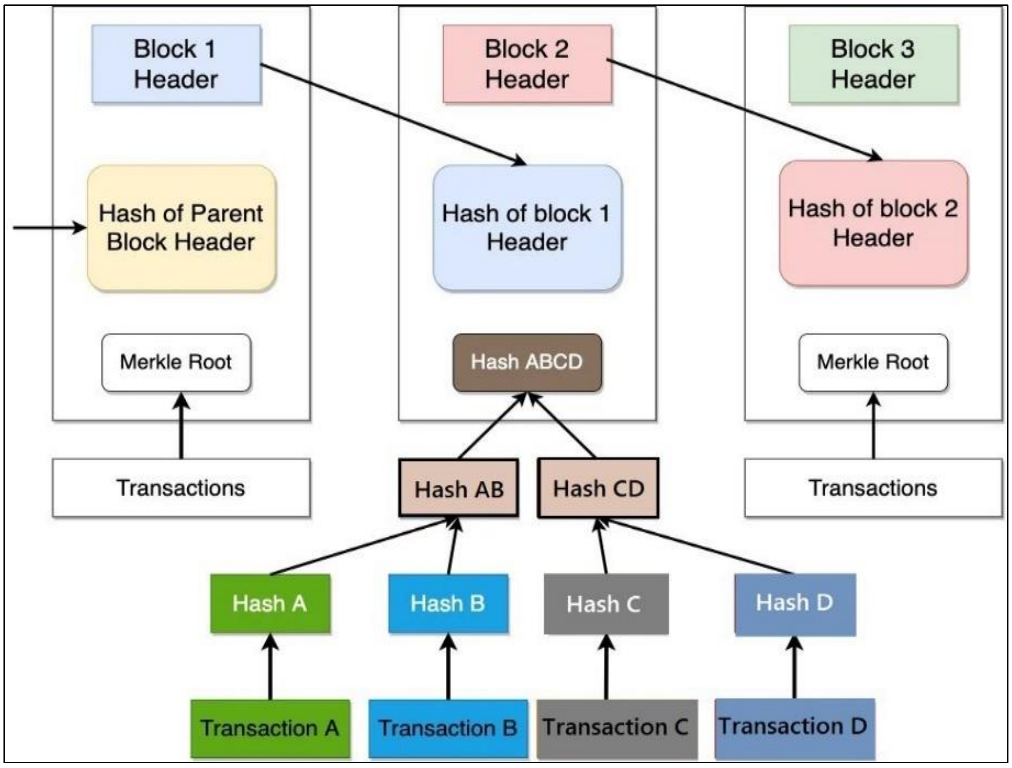


Figure 1. Diagram showing how to link a blockchain.

2. Materials and Methods

2.1. Hash Function

The Secure Hash Algorithm 256 bit (SHA256) belongs of the SHA-2 family. It converts the inputs data into a fixed value of 256 bits (32 bytes). This function works in a one-way, meaning that the original data cannot be obtained from the hashed value. In SHA256, the data is divided into blocks of 512 bits, and each block is split into 16 words of 32 bits. After that, padding is added to ensure that the length of the data is divisible by 512 (2,19,20). The initial values of SHA256 are introduced in **Equation 1** (19). Any ideal hash function should have the properties below (2, 20).

i. Collision resistance, which is defined as the impossibility of finding two different inputs that generate the same resulting hash value.
ii. Preimage resistance, which means that for a given hash, finding the original input is almost impossible.
iii. Second preimage resistance ensures that for a given hash value, it is very hard to find another input message that generates the same hash.

The hashing algorithm with output values of length m needs 2^m operations to find a preimage or second preimage, while finding a collision by a birthday attack needs $2^{(m/2)}$ operations (2,21,22).

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

The maximum length inputs of SHA256 it can handle is $2^{64} - 1$ bit. These inputs are partitioned to $\left(m^{(1)}, m^{(2)}, \ldots, m^{(n)}\right),\ n \geq 1$ blocks, each one is partitioned into 16 words of 32 bits as $m^{(j)} = w_1, w_2, \ldots, w_{16}\ ,\ 1 \leq j \leq n$. Then the words extended into (64) words of 32 bits as shown in **Equation 2** (19).

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

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

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

$ROTR^N(x)$ is the rotate right operation of x by N positions to the right and $x>>N = SHR^N(x)$ is the right shift operation of x. Thereafter, the blocks $m^{(1)}, m^{(2)}, \ldots, m^{(n)}$ are treated one after another by constructing the updated 8 state variables out of 64 rounds, using the constants $K_i\ , i = 1, \ldots, 64$ which represent the fixed SHA-256 key values (16). The update values can be obtained by using new values $A, B, \ldots, H$. After the block $m^{(n)}$ has been treated, the output hash $H_k^{(n)}\ , k = 1,2, \ldots 8$ is: $SHA_{256} = H_1^{(n)} || H_2^{(n)} || H_3^{(n)} || H_4^{(n)} || H_5^{(n)} || H_6^{(n)} || H_7^{(n)} || H_8^{(n)}$, where `||` represents the operation of connection hash values in computation (11,19). **Figure 2** described how SHA256's algorithm works (11).

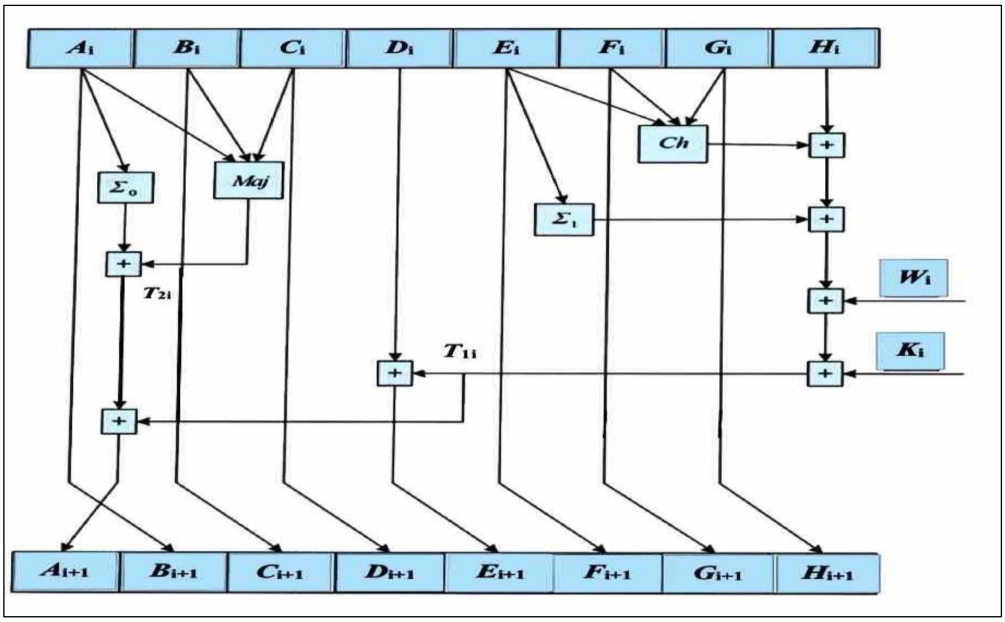


Figure 2. Illustrates a single round of the SHA256 algorithm process.

2.2.Skew Tent Map.

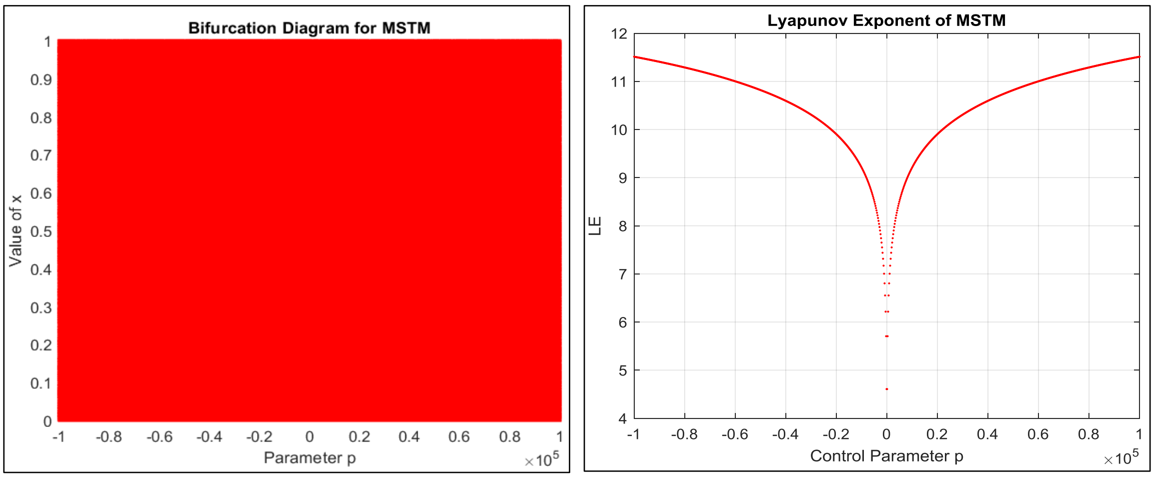
The one-dimensional skew tent map (STM) is defined in **Equation 5** (23-25).

$$x_{n+1} = \begin{cases} \frac{x_n}{p} & if \;\; 0 \le x_n \le p \\ \frac{1-x_n}{1-p} & if \;\; p < x_n \le 1 \end{cases} \tag{5}$$

Where $p \in (0,1)$ and $n = 0,1,2,\ldots$. The STM has some drawbacks. If the control parameters or initial conditions are not chosen carefully or go beyond certain limits, the chaotic behavior may weaken or disappear, leading to the cancellation of chaos (26). A modification of STM (MSTM) was shown in **Equation 6**.

$$x_{n+1} = \begin{cases} mod\left(\frac{1}{p}\left(x_n + p^2\left(x_n - 1\right)\right), 1\right) & if \;\; 0 \le x_n \le p \\ mod\left(p\left(x_n + \frac{1-x_n}{(1-p)}\right), 1\right) & if \;\; p < x_n \le 1 \end{cases} \tag{6}$$

MSTM exhibits a more evenly distributed chaotic sequence and operates over a broad control parameter range, $p \in (-\infty, \infty) - \{0, 0.5, 1\}$, effectively avoiding blank areas. In **Equation 6**, x_0 represents the initial value within n=0,1,2,…. **Figure 3** illustrates the chaotic behavior of MSTM.



(a) (b)

Figure 3. Chaotic behavior of MSTM (a) Bifurcation diagram and (b) Lyapunov Exponent (LE).

2.2.1. NIST Statistical Suite Tests

The NIST suite (27) comprises 15 statistical tests for evaluating the randomness of binary sequences. In this analysis, 100 binary sequences, each containing10^6 bits, are derived from 100 distinct MSTM sequences binary sequences of **Equation 6**. We apply the threshold function T(x) as defined in **Equation 7** to find binary sequences. The results are assessed by comparing the P-values where P > 0.01, as summarized in **Table 1.**

$$T(x) = \begin{cases} 0 & if \;\; x < 0.5 \\ 1 & otherwise \end{cases} \qquad (7)$$

Table 1. NIST test of 10^6-bit binary sequence of MSTM.

No.	MSTM Statistical Tests	x_0=0.1 and *r*=999 *P*- Values	 Result
1	Frequency Monobit	0.79177	✓
2	Block Freq.	0.76693	✓
3	Runs	0.019287	✓
4	Long Run of Ones	0.06776	✓
5	Binary Matrix Rank	0.93068	✓
6	DFT (Spectral)	0.31276	✓
7	Non-Overlapping Templates	0.28969	✓
8	Overlapping Templates	0.37208	✓
9	Maurer's Universal Statistical	0.15794	✓
10	Linear Complexity	0.61058	✓
11	Serial test	0.44381	✓
12	Approximate Entropy	0.748247	✓
13	Cumulative sums test	0.56396	✓
14	Rand. Excursions	0.34042	✓
15	Rand. Excursions Variant	0.17102	✓
	Pass rate	15/15	

2.3. A Novel Development of SHA256

The suggested algorithm D-SHA256 has the same output size as SHA256 and treats inputs of up to $2^{64} – 1$ bit. The message is split into *n*-blocks, each one is split into 16 words similar to the structure used in SHA256, as described in Section 2. To enhance the speed, security and randomness of SHA256, D-SHA256 integrates the compression function of SHA256 with MSTM with working an additional modification to increase the randomness of the state variables and ensure strong resistance to current attacks. Additionally, the proposed algorithm employes 32 rounds with 8 working variable and 48 hash words and constants (*K*). D-SHA256 algorithm offers efficient calculations, reduced running time, strong security, and a uniform distribution of output data.

2.3.1. The D-SHA256 Algorithm

Input: The message (M)
Output: The D-SHA256 value in hexadecimal.

1: Start
2: Use ASCII stream to convert the M into its binary form.
3: Append a '1' bit to the message and then append '0' bits until the total length is 448 mod 512.
4: Split the padded message in Step 3 into 512-bit blocks $\left(m^{(1)}, m^{(2)}, \dots, m^{(n)}\right), n \geq 1$.
5: Ensure the last 64 bits of the final block in Step 4 store the original message (M) length in bits.

6: Use **Equation 6** with **Equation 7** to compute the binary sequence of MSTM by taking x_0=0.1, p=999 and the number of iterations is 1536.

7: Split the sequence in step 6 into $(S_0, S_1, \ldots, S_{48})$, keeping in mind that each part of S_i , $i = 1,2, \ldots 48$ contains 32 bits.

8: $\forall\ m^{(1)}\ to\ m^{(n)}$ block apply the steps below:

i. For j=1 to n

ii. Define the message schedule W_i as:

$$W_i = \begin{cases} m_i^{(j)} & 1 \le i \le 16 \\ \sigma_1^{(256)}(w_{i-2}) + w_{i-7} + \sigma_0^{(256)}(w_{i-15}) + w_{i-16} + S_i & 17 \le i \le 48 \end{cases}$$

iii. Set up the intermediate state variables A, B, …, H by utilizing the initial value in **Equation 1** as:

$$\mathrm{A} = \mathrm{H}_1^{(\mathrm{j}-1)}$$
$$\mathrm{B} = \mathrm{H}_2^{(\mathrm{j}-1)}$$
$$\mathrm{C} = \mathrm{H}_3^{(\mathrm{j}-1)}$$
$$\mathrm{D} = \mathrm{H}_4^{(\mathrm{j}-1)}$$
$$\mathrm{E} = \mathrm{H}_5^{(\mathrm{j}-1)}$$
$$\mathrm{F} = \mathrm{H}_6^{(\mathrm{j}-1)}$$
$$\mathrm{G} = \mathrm{H}_7^{(\mathrm{j}-1)}$$
$$\mathrm{H} = \mathrm{H}_8^{(\mathrm{j}-1)}$$

i. For i= 2:2:32

{

$$T_1 = h + \textstyle\sum_1^{(256)}(E) + Ch(E,F,G) + K_{i-1}^{(256)} + W_{i-1} + S_{i-1}$$
$$T_2 = \textstyle\sum_0^{(256)}(A) + Maj(A,B,C)$$
$$T_3 = \textstyle\sum_0^{(256)}(E) + K_i^{(256)} + W_i + S_i$$

H = G, G = F, F = E+ T_2 *, E = D +* T_1 *, D = C, C = B, B = A and A =* $T_1 + T_3$

}

For i = 33:48

{

$$T_1 = \mathrm{h} + \textstyle\sum_1^{(256)}(E) + Ch(E,F,G) + K_i^{(256)} + W_i + S_i$$
$$T_2 = \textstyle\sum_0^{(256)}(A) + Maj(A,B,C)$$

H = G, G = F, F = E, E = D + T_1 *, D = C, C = B, B = A and A =* $T_1 + T_2$

$$\textstyle\sum_0^{(256)}(A) = ROTR^2(A) \oplus ROTR^{13}(A) \oplus ROTR^{22}(A),$$
$$\textstyle\sum_1^{(256)}(E) = ROTR^6(E) \oplus ROTR^{11}(E) \oplus ROTR^{25}(E),$$
$$Maj(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C),$$
$$Ch(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$

}

ii. Compute j[th] intermediate hash working value as:

$$\left.\begin{array}{l} H_1^{(j)} = A + H_1^{(j-1)} \\ H_2^{(j)} = B + H_2^{(j-1)} \\ H_3^{(j)} = C + H_3^{(j-1)} \\ H_4^{(j)} = D + H_4^{(j-1)} \\ H_5^{(j)} = E + H_5^{(j-1)} \\ H_6^{(j)} = F + H_6^{(j-1)} \\ H_7^{(j)} = G + H_7^{(j-1)} \\ H_8^{(j)} = H + H_8^{(j-1)} \end{array}\right\}$$

End For j

iii. Represent the D-SHA256 value of the M, after processing the final block $m^{(n)}$, as:

$$D - SHA_{256} = H_1^{(n)} || H_2^{(n)} || H_3^{(n)} || H_4^{(n)} || H_5^{(n)} || H_6^{(n)} || H_7^{(n)} || H_8^{(n)}$$

9: End

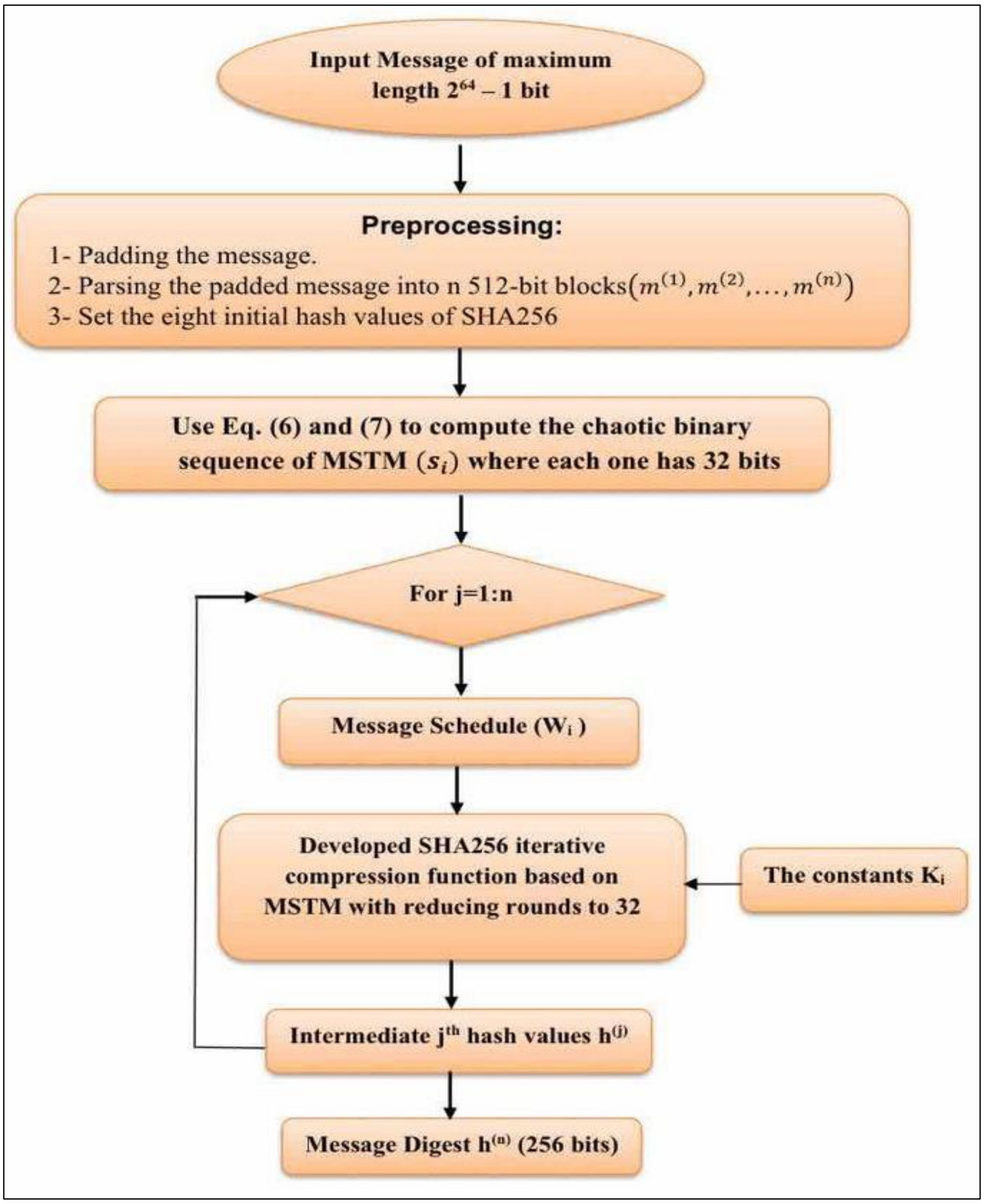


Figure 4. Construction of the D-SHA256 structure.

3. Results

3.1. Hash Value Distribution

To evaluate D-SHA256 algorithm security, tests were conducted using both random and extreme plaintext messages. For instance, **Figure 5a** illustrates the use of random characters with ASCII values primarily in the range 50 to 95 as input. **Figure 5b** demonstrates the use of identical characters with unchanged ASCII values. As shown in **Figures 5c** and **d**, the plaintext messages are limited to a finite range, while their corresponding hash values are distributed randomly and evenly. The results offers that a compression function based on MSTM in D-SHA256 algorithm effectively conceals the statistical properties of the plaintext, making it impossible to infer the plaintext from the hash value.

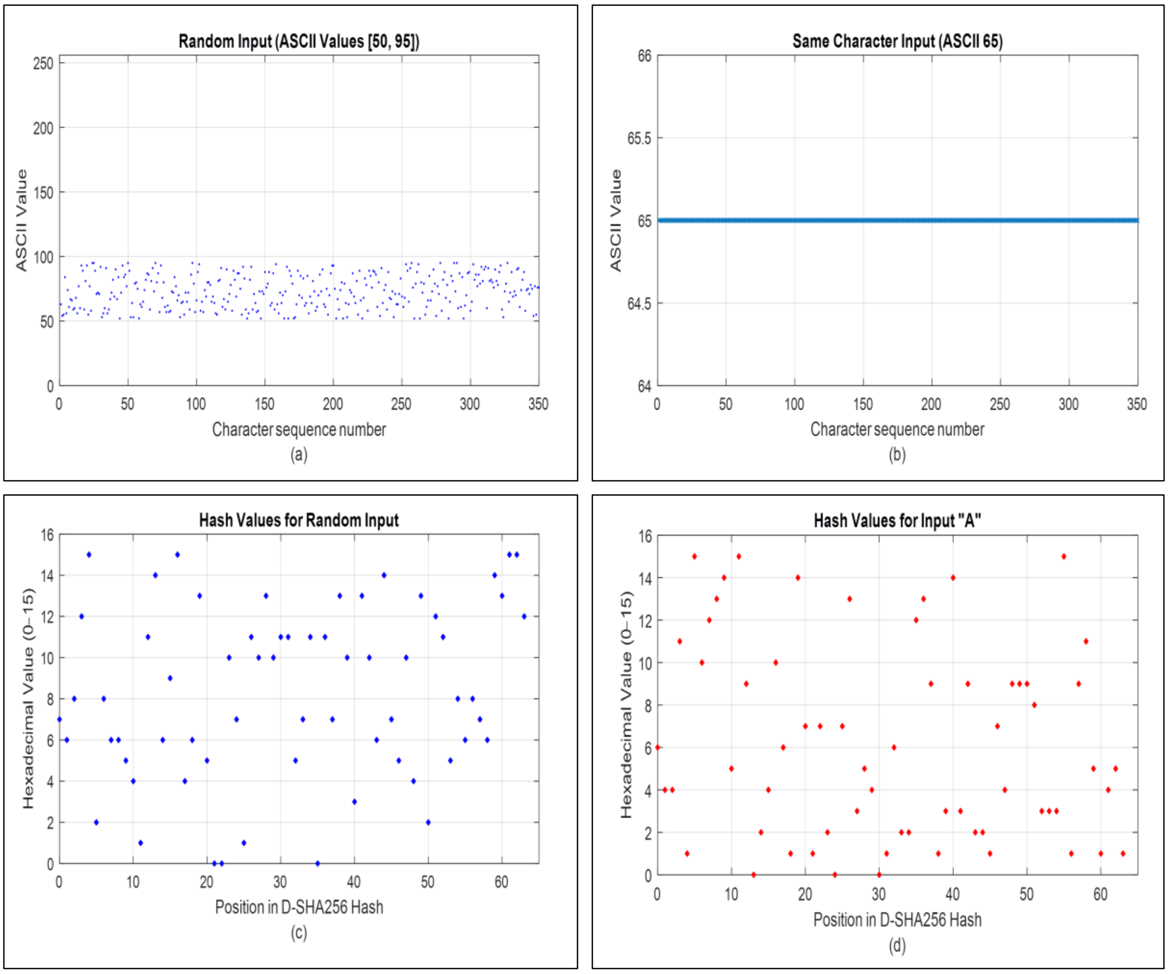


Figure 5. D-SHA256 value distribution.

3.2. D-SHA256 Sensitivity

To illustrate the sensitivity of the D-SHA256 algorithm, six trials were done, each including small changes. C1 is the input message, while C2–C6 are its altered of C1.

a) C1: The original message: “The Department of Mathematics and Computer Science”
b) C2: C1 with delete space between "of Mathematics"'.
c) C3: Replace the first letter 'T' in "The" with 't'.
d) C4: Remove the letter 'r' from " Computer ".
e) C5: Add a dot (.) at the end.
f) C6: Replace 'a' in "and" with 'A'.

Table 2 presents the hash digest values in hexadecimal format and the number of altered bits for the D-SHA256 algorithm. The results, as illustrated in **Figure 6**, highlight the suggested hash algorithm's high sensitivity to minor changes in the input message bits.

Table 2. The output of the D-SHA256 algorithm and the number of bits altered relative to the C1.

C	D-SHA256	Change bits
C1	8CBC119118029E394A2E3EF484CA2A2AB3504E4CC80AAAE7BD48559D0D64832A	-
C2	228B227878E11070816852D313562E2E1C8E2E5B6F7964CCD2844A519D334C16	133
C3	B3EF7E0405A55D8F89622C0B46171D4B67F6B69AD683ADFFCEB44834900D9B93	137
C4	7D6906EA2524A57D86935F56AE2E34DD82DB864203B64C89078A649CC99F60A3	132
C5	77C54EBFF292F0CF18607DB158DC58174B7FDA707233BE82D756CF7B723DEBF1	139
C6	8A368E8DEEF802C875E73D8D5486B086440D18F29A01829F37A88FF9645C0F9C	129

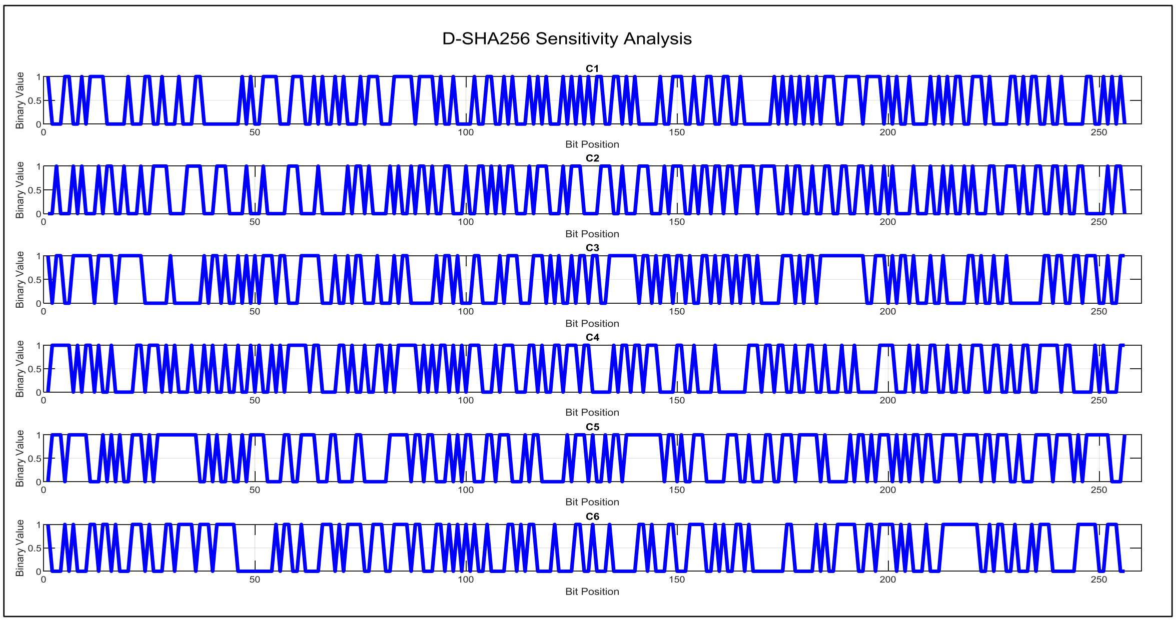


Figure 6. The representation of output of D-SHA256 for C1-C6.

3.3. Confusion and Diffusion of D-SHA256

Claude Shannon's principles (18) of confusion and diffusion, avalanche effect, are necessary for secure hash design. Confusion obscures the link between input and hash, while diffusion guarantees that small input changes affect the entire hash value. Secure hash algorithms must produce uniformly distributed values, with an ideal 50%-bit change probability for binary hashes, making them resistant to collision attacks and computational weaknesses. The goal of the confusion and diffusion test is to statistically analyses how hash values change when small perturbations are made to input messages. This done by the steps below:

1. Use D-SHA256 algorithm to compute the hash digest value of an original message.
2. Swap one bit of the original message and recalculate the output hash value.
3. Make a comparison between the original and altered hash values.
4. Iterate N times the process with different input messages.

Here take N = 256, 512, 1024, 2048 and 10000 to find the statistical behavior of D-SHA256 under minor input changes. Results in **Table 3** demonstrate the algorithm's effectiveness in achieving confusion and diffusion, ensuring maintaining strong security properties and **Table 4** offers the comparison when N=10^4. We use the formulas below for these statistics (18).

- $B_i, i = 1, \dots, N$ represents the total number of bits that changed in the ith test of the hash after modification.
- Minimal number of $B_i : B_{min} = \min(B_i)_{i=1}^{N}$ (8)
- Maximal number of $B_i : B_{max} = \max(B_i)_{i=1}^{N}$ (9)
- Average of $B_i : \bar{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$ (10)

- Average changed probability: $P = \left(\frac{\bar{B}}{n}\right) \times 100\%$ (11)
- Standard deviation of B_i: $\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \bar{B})^2}$ (12)
- Standard deviation of P: $\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{B_i}{n} - P\right)^2}$ (13)

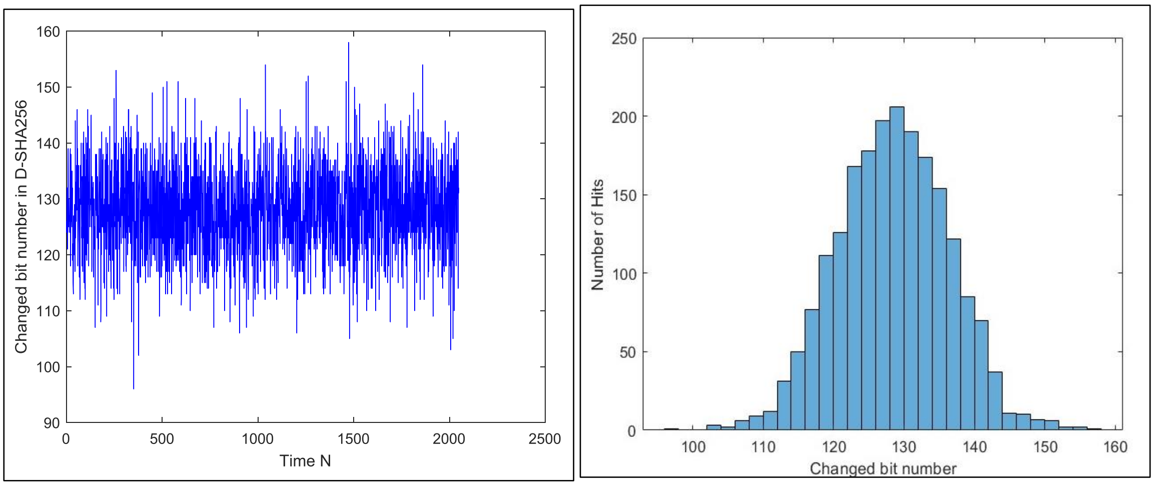
where N is the number of times the test is done, and n is defined as the resulting hash size. **Figure 7** shows the histogram and the statistical analysis of B_i for N=2048 tests.

Table 3. Confusion and diffusion analysis of D-SHA256 for different lengths.

N	D-SHA256					
	Bmin	**Bmax**	**Mean ($\bar{B}$)**	**P%**	**ΔB**	**Δp**
256	107	151	128.20	50.01	7.97	0.0306
512	107	152	128.47	50.04	7.90	0.030
1024	101	153	128.23	50	8.01	0.03
2048	107	157	128.01	50.01	8.22	0.032
100000	100	157	128.10	50.00	7.921	0.0302

Table 4. The confusion and diffusion comparison of different hash algorithms where N=104.

Algorithm	**Bmin**	**Bmax**	**Mean ($\bar{B}$)**	**P%**	**ΔB**	**Δp**
Ref. (9)	99	155	128.08	50.03	8.09	0.0316
Ref.(10) (Str-1)	97	159	128	50.01	7.921	0.0309
Ref.(10) (Str-2)	100	161	128.1	50.04	8.016	0.03131
Ref.(10) (Str-3)	100	161	128	50.00	7.911	0.03091
Ref.(10) (Str-4)	95	153	128	50.02	8.131	0.03176
SHA3-256 (31,10)	101	153	128.1	50.02	8.01	0.0313
SHA256 (19,10)	104	154	128.00	50.00	7.940	0.0310
D-SHA256	100	157	128.10	50.00	7.921	0.0302



(a) (b)

Figure 7. Statistical analysis of D-SHA256: (a) Distribution of B_i and (b) B_i histogram

3.4. NIST Statistical Suite Tests

The National Institute of Standards and Technology (NIST) suite of tests are statistical tests for verifying the randomness of binary bit sequences (27). It comprises 15 tests designed to assess the performance of the D-SHA256 algorithm. **Tables 6** and **7** show the performance of the NIST test for several input messages of different sizes mentioned in **Table 5** along with their execution time. These tables show the success rate which must be with a P-value > 0.01 for each test. The three NIST tests were excluded: binary matrix rank, overlapping templates and Maurer's universal statistical, because they require input sequences longer than 256 bits to be accurate, while the hash output is limited to this length.

Table 5. Output hash values and execution Time for different input lengths

No.	Different Message Length (M)	SHA256	D-SHA256	The running time in seconds	
				SHA256	D-SHA256
1	abc	BA7816BF8F01CFEA4 14140DE5DAE2223B00 361A396177A9CB410F F61F20015AD	42F45A9A77053D4E8EA 9E821BDBD7E430D2681 79B38BA6160212CC7CD 75D05B6	0.125977	0.070893
2	0000000	20FDF64DA3CD2C78E C3C033D2AC628BACF 701711FA99435EE37B EF0304800DC5	5A13C1D2C9D52432E93 EF2FB8E86B5C1ADBB0 22C21700D22CEEB3FB5 3AD5C23A	0.117267	0.068670
3	abcooooooooooooooooo ooouuuuuuuuuuuuuuuu uuuuuuuuuxxxxxxxxxx xxxxxxxxxxxx	7FBBDF20A0C98C42B D20482279FF86825464 F3FF3D4069B1CD14C BAD6BD7B9D6	5BEF030A360537D3D8F 9F278A1D849CB40CFC AA2E04C9172617B7E1A 515028E0	0.232181	0.081639
4	abcooooooooooooooooo ooossssssssssssssssssss ssssxxxxxxxxxxxxxxxxx xxxxxx	53BCE3B7773F4B72D7 C099713B9F251F7E41 B7EB7BE287787559D7 EAEB72161E	FD6A75B10B15C0FF959 9DEC21812606622406A1 CF2C99BFCFF5AA8F960 20C7AD	0.242044	0.088812
5	10000 bits	1D9CE838421F7D8EAF 552161F70F9B3339AD 999F5EF2E61951E11B6 976F9298B	ECA20DA52D7D89B328 8A39FD2978EC491E20B 3E3E6FA54DD10E8660F 0FA58384	13.67671	7.965537
6	100000 bits	D5A0A8E4300F485BB A28174BCA0B5172313 9B5C50029880227CCB A76326D211F	08B1324F93429A3CAAB F152861EB1187DF6A8F5 1014CCB0DC1549D4F61 E96439	352.33897	177.19982

Table 6. NIST randomness test of SHA256 and D-SHA256 for message 2 / Table 5

The Message 0000000				
Hash Function	**SHA256**		**D-SHA256**	
Statistical Tests	***P*- Values**	**Result**	***P*- Values**	**Result**
Frequency Monobit	0.8025	✓	1.000	✓
Block Freq.	0.9394	✓	0.9692	✓
Runs	0.0807	✓	0.3815	✓
L. Run of Ones	0.3091	✓	0.3262	✓
Binary MatrixRank	-1	×	-1	×
DFT Spectral	0.4220	✓	0.8185	✓
Non Overlapping Templates	0.0002	×	0.0537	✓
Overlapping Templates	NaN	×	NaN	×
Maurer's Universal Statistical	-1	×	-1	×
Linear Complexity	0.4985	✓	0.9218	✓
Serial	0. 4989	✓	0.8413	✓
Appro. Entropy	1	✓	1	✓
Cumulative sums test	0.7458	✓	0.8579	✓
Rand. Excursions	0.4265	✓	0.5625	✓
Rand. Excursions Variant	0.3763	✓	0.6510	✓
Pass rate	**11/15**		**12/15**	

Table 7. Number of successful NIST tests of hash algorithms

Input Message (M) in Table 5	Number of successful NIST tests of SHA256	Number of successful NIST tests of D-SHA256
abc	11/15	12/15
0000000	11/15	12/15
abcoooooooooooooooooooouuuuuuuuuuuuuuuuuuuuuuuuuxxxxxx xxxxxxxxxxxxxxxx	10/15	12/15
abcoooooooooooooooooooosssssssssssssssssssssssssxxxxxxxxxxx xxxxxxxxxxx	9/15	12/15
10000 bits	12/15	12/15
100000 bits	12/15	12/15

The comparison between the D-SHA256 and SHA256 algorithms highlights improved performance in randomness tests. As shown in **Table 7**, the D-SHA256 algorithm passes 12 out of 15 tests, outperforming SHA256, which achieves 11, 10, and 9 successful tests, respectively. D-SHA256 offers distinct advantages over SHA256, including faster execution times and improved efficiency, making it more suitable for practical applications.

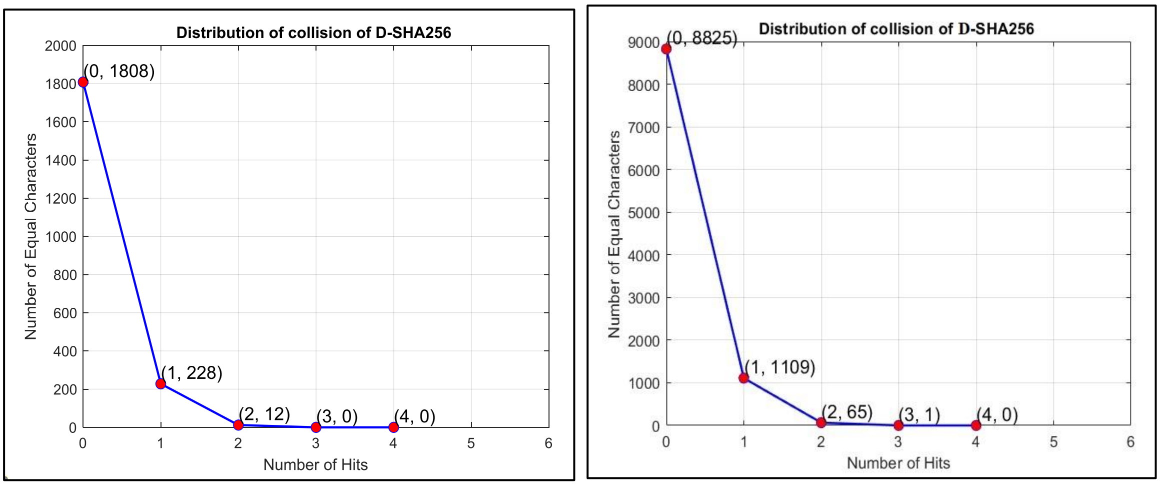
4. Discussion

4.1. Collision Resistance Analysis of D-SHA256

In this study, collision resistance is evaluated using the method described in (28-30). A random input message is selected, its hash is resulted, and it is saved in ASCII format. Then, a single bit in the input message is randomly modified to produce a new hash value, which is also saved in ASCII format. By comparing the ASCII characters at corresponding positions in both hash values, the highest count of identical characters determines the collision degree. A lower value indicates weaker collision resistance. Any single matching character is considered a collision, and the total number of such instances is recorded. The number of hits in the N^{th} test is provided in (30), where the theoretical number of expected collisions is computed using the following formula:

$$W_N(w) = N \times prob(w) = N \times \frac{s!}{w!(s-w)!}\left(\frac{1}{2^8}\right)^w \left(1-\frac{1}{2^8}\right)^{s-w} \quad (14)$$

Here, the number of hits (w) when comparing two hash values refers to the number of ASCII characters that are identical and occur at the same position in both hash values. N represents the number of tests, 8 refers to the number of bits in an ASCII character, and $s = \left(\frac{length\ of\ hash}{8}\right) = \frac{256}{8} = 32$. **Figure 8** presents the distribution of how often hash values contain identical characters at the same position for the D-SHA256 algorithm after running N=2048 and N=10^4 tests. Moreover, **Table 8** shows the number of hits comparison for hash algorithms in 2048 and 10000 random tests. It also shows that the values of the proposed D-SHA256 algorithm are closer to the theoretical values compared to the other hash functions.



(a) (b)

Figure 8. The distribution of the number of hits for D-SHA256 when (a) N=2048 and (b) $N=10^4$

Table 8. The number of hits comparison of hash algorithms.

	Hits numbers (*w*) N=2048					
	0	**1**	**2**	**3**		**32**
Theoretical value	**1806.91**	**226.75**	**13.78**	**0.54**		$\mathbf{1.75 \times 10^{-74}}$
Ref.(10) (Str-1)	1819	219	10	0		0
Ref.(10) (Str-2)	1806	222	20	0		0
Ref.(10) (Str-3)	1800	240	8	0		0
Ref.(10) (Str-4)	1799	235	14	0		0
Ref.(12) (Str-1)	1803	232	13	0		0
Ref.(12) (Str-2, r=8)	1817	215	16	0		0
Ref.(12) (Str-2, r=24)	1815	226	7	0		0
Ref.(10,19) (Str-3)	1824	213	11	0		0
Ref.(10,31) (Str-1)	1931	114	3	0		0
Ref.(10,31) (Str-2)	1929	114	5	0		0
Ref.(10,31) (Str-3)	1942	106	0	0		0
Ref.(11)	1823	215	10	0		0
Ref.(12,19) SHA256	1817	220	11	0		0
D-SHA256	1808	228	12	0		0
	Hits numbers (*w*) N=10000					
	0	1	2	3	4	32
Theoretical value	8822.81	1107.18	67.30	2.64	0.075	8.64×10−74
D-SHA256	8825	1109	65	1	0	0

4.2. Running Time

The modified algorithm underwent extensive testing with messages of varying lengths, demonstrating notable gains in efficiency and running time. **Table 5** and **Figure 9** offer that D-SHA256 has a shorter execution time than SHA256. The D-SHA256. Integrate MSTM into the modified D-SHA256 enhances performance, while reducing the number of rounds and optimizing the compression function minimizes execution time. The average execution time of SHA256 and D-SHA256 is 5.768 seconds and 4.4674 seconds respectively. As a result, the D-SHA256 algorithm achieves robust security, strong randomness, and improved efficiency.

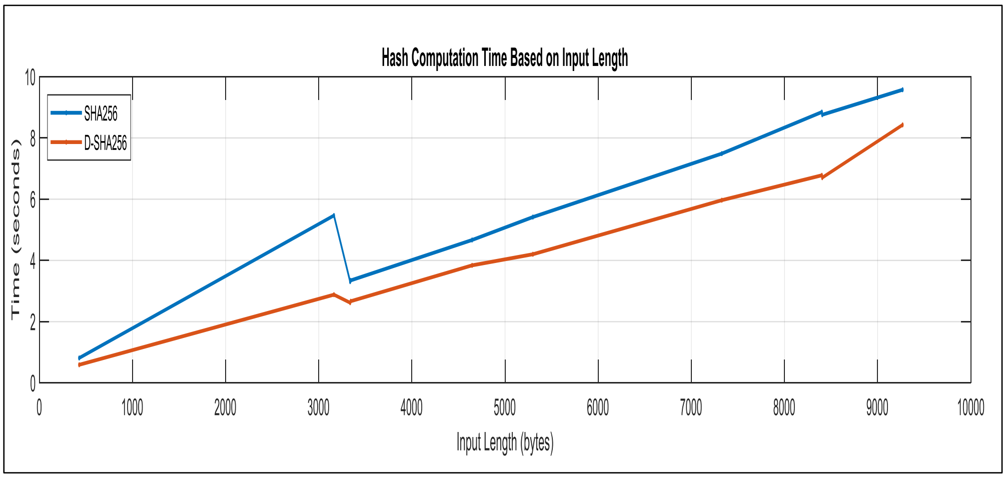


Figure 9. The running time of the hash algorithms D-SHA256 and SHA256

5. Conclusion

The objective of this work is how to boost the level of security of the blockchain system based on increasing the security of hashing function. The proposed D-SHA256 algorithm enhances the intermediate state of the compression function by using the modified skew tent map (MSTM) for strength blockchain security. The number of rounds were reduced to 32 for faster execution while maintaining the robust security. The proposed algorithm employs 48 hash constants and 48 words. It has strong collision resistance, and protection against preimage attacks, satisfies the NIST randomness tests, and reduces execution time. This approach achieves a balance between security and efficiency, providing a reliable framework for secure hash implementation and offering a novel direction for hash algorithm development, particularly in compression function design. Its enhanced sensitivity makes it resilient to all known attacks.

Acknowledgment

Our thanks and appreciation to the reviewers and publishers of Ibn Al-Haitham Journal for Pure& Applied Sciences.

Conflict of Interest

The authors declare that they have no conflicts of interest.

Funding

This work is not supported by any the Foundation.

References

1. Salagrama S, Bibhu V, Rana A. Blockchain based data integrity security management. Procedia Comput Sci. 2022;215:331–339. https://doi.org/10.1016/j.procs.2022.12.035
2. Salih RK, Kashmar AH. Enhancing blockchain security by developing the SHA256 algorithm. Iraqi J Sci. 2024;65(10):5678–5693. https://doi.org/10.24996/ijs.2024.65.10.30
3. Jasim AH, Kashmar AH. An evaluation of RSA and a modified SHA-3 for a new design of blockchain technology. In: Artificial Intelligence for Smart Healthcare. Cham: Springer; 2023;477–489. https://doi.org/10.1007/978-3-031-23602-0_28
4. Kumar KP, Varma NH, Devisree N, Ali MS. Implementation of associative service recommendation scheme applying SHA256 algorithm through blockchain. J Surv Fish Sci. 2023;10(2S):2741–2747. https://sifisheriessciences.com/journal/index.php/journal/article/view/1325

5. Fotohi R, Aliee FS. Securing communication between things using blockchain technology based on authentication and SHA-256 to improving scalability in large-scale IoT. Comput Netw. 2021;197:108331. https://doi.org/10.1016/j.comnet.2021.108331
6. Abid Ali AAM, Hazar MJ, Mabrouk M, Zrigui M. Proposal of a modified hash algorithm to increase blockchain security. Procedia Compute Sci. 2023;225:3265–3275. https://doi.org/10.1016/j.procs.2023.10.320
7. Tutueva AV, Karimov AI, Moysis L, Volos C, Butusov DN. Construction of one-way hash functions with increased key space using adaptive chaotic maps. Chaos Solitons Fractals. 2020;141:110344. https://doi.org/10.1016/j.chaos.2020.110344
8. Alawida M, Samsudin A, Alajarmeh N, Teh JS, Ahmad M, Alshoura WH. A novel hash function based on a chaotic sponge and DNA sequence. IEEE Access. 2021;9:17882–17897. https://doi.org/10.1109/ACCESS.2021.3049881
9. Kanso A, Yahyaoui H, Almulla M. Keyed hash function based on a chaotic map. Inf Sci. 2012;186(1):249–264. https://doi.org/10.1016/j.ins.2011.09.008
10. Serag Eldin SM, Abd El-Latif AA, Chelloug SA, Ahmad M, Eldeeb AH, Diab TO. Design and analysis of new version of cryptographic hash function based on improved chaotic maps with induced DNA sequences. IEEE Access. 2023;11:101694–101709. https://doi.org/10.1109/ACCESS.2023.3298545
11. Wang J, Liu G, Chen Y, Wang S. Construction and analysis of SHA-256 compression function based on chaos S-box. IEEE Access. 2021;9:61768–61777. https://doi.org/10.1109/ACCESS.2021.3071501
12. Abdoun N, El Assad S, Deforges O, Assaf R, Khalil M. Design and security analysis of two robust keyed hash functions based on chaotic neural networks. J Ambient Intell Humaniz Comput. 2020;11(5):2137–2161. https://doi.org/10.1007/s12652-019-01244-y
13. Saleh FF, Ali NHM. Generating streams of random key based on image chaos and genetic algorithm. Iraqi Journal of Science. 2022;63(8):3652–3661. https://doi.org/10.24996/ijs.2022.63.8.39
14. Taqi IA, Hameed SM. A new beta chaotic map with DNA encoding for color image encryption. Iraqi J Sci. 2020;61(9):2371–2384. https://doi.org/10.24996/ijs.2020.61.9.24
15. Abdulmunem IA., Harba ES., Harba HS. Advanced Intelligent Data Hiding Using Video Stego and Convolutional Neural Networks, Baghdad Science Journal. 2021; 18(4):1317-1327. https://doi.org/10.21123/bsj.2021.18.4.1317
16. Abbaas HD, AbdulSalam AA. Hybrid efficient stream cipher key generator based on LFSR's and chaotic map. Ibn AL-Haitham Journal for Pure and Applied Sciences. 2024;37(1):464–476. https://doi.org/10.30526/37.1.3321
17. Hua Z, Zhou Y. One-dimensional nonlinear model for producing chaos. IEEE Trans Circuits Syst I Regul Pap. 2017;65(1):235–246. https://doi.org/10.1109/TCSI.2017.2717943
18. Shannon CE., Weaver W. The mathematical theory of communication. Champaign, IL: University of Illinois Press. 1949.
19. Federal Information Processing Standard (FIPS). Secure hash standard. FIPS PUB 180-2. National Institute of Standards and Technology; 2002.
20. Algredo-Badillo I, Feregrino-Uribe C, Cumplido R, Morales-Sandoval M. FPGA-based implementation alternatives for the inner loop of the Secure Hash Algorithm SHA-256. Microprocess Microsyst. 2013;37(6):750–757. https://doi.org/10.1016/j.micpro.2012.06.007
21. Kundu R, Dutta A. Cryptographic hash functions and attacks: A detailed study. Int J Adv Res Comput Sci. 2020;11(2). https://doi.org/10.26483/ijarcs.v11i2.6508
22. Verma R, Dhanda N, Nagar V. Enhancing security with in-depth analysis of brute-force attack on secure hashing algorithms. In: Trends in Electronics and Health Informatics: TEHI 2021; 2022:513–522. https://doi.org/10.1007/978-981-16-8826-3_44

23. Hasler M, Maistrenko YL. An introduction to the synchronization of chaotic systems: Coupled skew tent maps. IEEE Trans Circuits Syst I Fundam Theory Appl. 1997;44(10):856–866. https://doi.org/10.1109/81.633874
24. Lawnik M, Berezowski M. New chaotic system: M-map and its application in chaos-based cryptography. Symmetry. 2022;14(5):895. https://doi.org/10.3390/sym14050895
25. Elmanfaloty RA, Abou-Bakr E. Random property enhancement of a 1D chaotic PRNG with finite precision implementation. Chaos Solitons Fractals. 2019;118:134–144. https://doi.org/10.1016/j.chaos.2018.11.019
26. Umar T, Nadeem M, Anwer F. A new modified skew tent map and its application in pseudo-random number generator. Comput Stand Interfaces. 2024;89:103826. https://doi.org/10.1016/j.csi.2023.103826
27. Bassham LE, Rukhin AL, Soto J, Nechvatal JR, Miles E, Stefan D, Levenson M, Vangel M, Banks DA. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22. 2010. https://doi.org/10.6028/NIST.SP.800-22r1a
28. Kwok-Wo W. A combined chaotic cryptographic and hashing scheme. Phys Lett A. 2003;307(5–6):292–298. https://doi.org/10.1016/S0375-9601(02)01770-X.
29. Di X, Xiaofeng L, Shaojiang D. One-way hash function construction based on the chaotic map with changeable-parameter. Chaos Solitons Fractals. 2005;24(1):65–71. https://doi.org/10.1016/j.chaos.2004.07.003
30. Zhang J, Wang X, Zhang W. Chaotic keyed hash function based on feedforward–feedback nonlinear digital filter. Phys Lett A. 2007;362(5–6):439–448. https://doi.org/10.1016/j.physleta.2006.10.052
31. Dworkin MJ. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Gaithersburg, MD, USA: Information Technology Laboratory National Institute of Standards and Technology. 2015. https://doi.org/10.6028/NIST.FIPS.202.