Received March 19, 2020, accepted April 8, 2020, date of publication April 23, 2020, date of current version May 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989820

HMNT: Hash Function Based on New Mersenne Number Transform

ALI MAETOUQ AND SALWANI MOHD DAUD, (Member, IEEE)
Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur 54100, Malaysia

Corresponding author: Salwani Mohd Daud (salwani.kl@utm.my)

This work was supported in part by the Ministry of Education, Malaysia, and Universiti Teknologi Malaysia, under Grant Q.K130000.3556.05G71.

ABSTRACT In the field of information security, hash functions are considered important as they are used to ensure message integrity and authentication. Despite various available methods to design hash functions, the methods have been proven to time inefficient and have security flaws (such as a lack of collision resistance or susceptibility to birthday attacks). In the current study, we propose a novel hash function scheme based on a new Mersenne number transform. The suggested hash function called Hash Mersenne Number Transform (HMNT) takes an arbitrary length as input to generate a hash value with variable lengths (128, 256 and 512-bits or longer). The proposed scheme is evaluated in terms of the sensitivity of the hash value to the message, secret key and image, distribution of hashes, confusion and diffusion, robustness against collision and birthday attacks, alongside flexibility. Based on the simulation outcomes, the suggested scheme possess high sensitivity to the original message, the secret key and images, along with strong collision resistance. In conclusion, the proposed hash scheme is simple and efficient compared with the existing hash functions, making it viable for practical implementation.

INDEX TERMS Hash function, Mersenne number transform, message digest authentication.

I. INTRODUCTION

Cryptographic hash functions, one of the most important cryptographic primitives can be used to ensure the security of many cryptographic applications and protocols, including message authentication code, integrity, digital signature and random number generation [1], [2]. A hash function is able to take a message of arbitrary length to produce a fixed-length code (or hash value) [3].

To ensure efficiency, a hash function must satisfy three security properties, namely: (i) collision resistance (i.e. it is computationally infeasible to find any two different input messages m and m' with the same output hash value, $h(m) = h(m')$); (ii) preimage resistance (i.e. it is computationally infeasible to find any input message which is hashed to the given output hash value); and (iii) second preimage (i.e. it is computationally infeasible to find any second input that has the same output as any specified input [4], [5].

Among the many algorithms designed for the implementation of the hash function, the Race Integrity Primitives Evaluation Message Digest (RIPMD), Message Digest 5 (MD5), Secure Hash Function 1 (SHA-1) and Secure Hash Function 2 (SHA-2) are the most preferred. The strength of these algorithms is based on the use of block ciphers, logical operations and the number of rounds [6], [7]. However, the abovementioned algorithms have been found to be vulnerable to collision and other types of attacks. For example, Wang *et al*. in 2004 discovered that MD4, MD5, RIPEEMD, and HAVAL-128 were weak against collision attacks [8]. In addition, successful collision attacks against SHA-1 were also reported [9], where large companies such as Google and Microsoft announced plans to abandon SHA-1 as a result of the collision attacks [10]. Neither MD5 nor SHA-1 is robust against collision attacks, therefore, National Institute of Standards and Technology (NIST) announced gradual elimination of SHA-1 [10] and its replacement with the SHA-2 family, a collection of several different hash functions (i.e. SHA-224, SHA-256, SHA-348 and SHA-512) [4].

Contrary to the presumption, the occurrences of collision and some partial attacks were identified in SHA-2 [11], [12]. Hence, these algorithms were not preferred to ensure integrity since they are not as time-efficient as SHA-1 [13]. Following several demonstrations of successful collision attacks, NIST reported a standard secure hash function called Secure

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

Hash Function 3 (SHA-3) in 2015. This function is based on the KECCAK algorithm, which was selected as the winner of the SHA-3 cryptographic hash algorithm competition by NIST [14]. Unfortunately, the first potential collision attack on SHA-3 has been recently presented [15] and the time required for hashing was greater than that of SHA-2 [3].

Since the traditional hash functions are no longer safe, the need to design a novel hash function has become a priority in the information security research. Although many hash function algorithms were proposed by researchers, some are based on chaotic maps where most of these algorithms use floating point representation for their digital chaotic maps. These algorithms also have high computational complexity, as they have also been detected to suffer from interoperability problems [16]. Unfortunately, despite the effort in designing an efficient hash function, some of the current chaos-based hashing scheme were proven to be insecure and less efficient compared to the classical hash function [17], [18].

Therefore, this study aims to design a hash function scheme based on a new Mersenne number transform (NMNT) that is both statistically secure and reasonably efficient.

This paper is organized as follows: Section II provides a brief introduction to the NMNT. Section III explains the transform parameters. Next, section IV describes the steps of implementing the proposed hash function scheme. While section V presents the security and performance evaluation of the proposed hash function. Then section VI compares the proposed scheme with other hash functions. Finally, section VII presents our conclusion.

II. THE NEW MERSENNE NUMBER TRANSFORM

The NMNT is part of the number-theoretic transform (NTT) family of algorithms and was introduced by Boussakta and Holt in 1995 [19]. It is defined as the modulo of the Mersenne numbers, where arithmetic operations are simple equivalents to ones' complement. NMNT also possesses the property of cyclic convolution used for fast calculation of error-free convolutions [20]. Furthermore, it has a long transform length with a power of two facilitating fast algorithms. On the other hand, NMNT can be used in one or several dimensions. Moreover, NMNT has several inherent advantages, such as its sensitivity to slight input variation, the long transform length and a variable block size [21]. These properties can be exploited to design a hash function that is more secure and efficient.

The forward one-dimension NMNT, $X(k)$ of an integer sequence $x(l)$, with a length equal to L is defined as in (1):

$$X(k) = \langle \sum_{l=0}^{L-1} x(l)\beta(lk) \rangle_{Mp} \quad k = 0, 1, 2, \ldots, L-1 \quad (1)$$

where $\langle\ \rangle_{Mp}$ is the Mersenne prime in the form of;
$Mp = 2^p - 1$ (for p = 2, 3, 5, 7, 13, 17, 19, 31, ...), and;

$$\beta(lk) = \beta_1(lk) + \beta_2(lk) \quad (2)$$

$$\beta_1(lk) = \langle \mathrm{Re}(\alpha_1 + \alpha_2)^{lk} \rangle_{Mp} \quad (3)$$

$$\beta_2(lk) = \langle \mathrm{Im}(\alpha_1 + j\alpha_2)^{lk} \rangle_{Mp} \quad (4)$$

$$\alpha_1 = \pm\langle 2^q \rangle_{Mp} \quad \text{and } \alpha_2 = \pm\langle -3^q \rangle_{MP} \quad (5)$$

$$\mathrm{q} = 2^{\mathrm{p}-2} \quad (6)$$

The values of $\beta_1(lk)$ and $\beta_2(lk)$ in (3) and (4) are calculated for a maximum transform length $L = 2^{p+1}$. For transform length less than that, their values can be obtained using the following (7) and (8):

$$\beta_1(lk) = \langle \mathrm{Re}((\alpha_1 + \mathrm{j}\alpha_2)^d)^{lk} \rangle_{Mp} \quad (7)$$

$$\beta_2(lk) = \langle \mathrm{Im}((\alpha_1 + \mathrm{j}\alpha_2)^d)^{lk} \rangle_{Mp} \quad (8)$$

where Re() and Im() are the real and imaginary parts of the enclosed term, respectively, d is an integer with a power of two.

III. CALCULATION OF THE TRANSFORM PARAMETERS

The calculation of the transform parameters starts by choosing a prime number (p). The value of the prime number depends on the desired transform length and dynamic range. For example, let us choose for simplicity a prime number $\mathrm{p} = 5$. The modulus for the chosen prime is $M_p = 2^5 - 1 = 31$ and the maximum transform length, $L_{max} = 32, q = 2^{p-2} = 2^3 = 8$. Using (5) to calculate α_1 and α_2 as in (9) and (10) respectively:

$$\alpha_1 = \pm\langle 2^8 \rangle_{31} = \pm 8 \quad (9)$$

$$\alpha_2 = \pm\langle 3^8 \rangle_{31} = \pm 20 \quad (10)$$

The initial values of α_1 and α_2 can be (8, 20), (−8, 20), (−8, −20), (8, −20). These values are for the transform length, $L = 2^{p+1} = 64$, and they vary according to the transform length. Selecting pair (8, 20), the corresponding α_1 and α_2 for transform length $2^p = 32$, can be calculated as in (11) and (12):

$$\alpha_1 \langle \mathrm{Re}(8 + j20)^2 \rangle_{31} = 5 \quad (11)$$

$$\alpha_2 \langle \mathrm{Im}(8 + j20)^2 \rangle_{31} = 10 \quad (12)$$

Hence, $\beta_1(lk)$ and $\beta_2(lk)$ can be computed by (13) and (14) as follows:

$$\beta_1(lk) = \langle \mathrm{Re}(5 + j10)^{lk} \rangle_{31} \quad (13)$$

$$\beta_2(lk) = \langle \mathrm{Im}(5 + j10)^{lk} \rangle_{31} \quad (14)$$

The same procedure can be replicated to estimate other transform for different transform size and moduli.

This numerical example illustrates all the required calculations to transform a string of numbers from one form to another. Let's assume an input array X containing four elements, each element is presented in decimal, X = [72 105 32 32]. This array can then be transformed to another form using the NMNT.

The first task is to choose the modulus which should be a Mersenne number in the form of $Mp = 2^p - 1$. The modulus should definitely be higher than any elements in the input array X. Thus, the minimum Mersenne prime number that can be selected in this example is $p = 7$ which makes the modulus $Mp = 127$. The input array is 4 and hence, the transform

length is chosen to be $L = 4$, then, $\alpha_1 = 0, \alpha_2 = 1$. The next step is to compute $\beta(l)$ using (3) and (4).

$$\beta(0) = 1,\ \beta(1) = 1,\ \beta(2) = -1,\ \beta(3) = -1.$$

Then, applying (1) to the input ($X = [72\ 105\ 32\ 32]$), the array elements will be transformed in the following way: The first element is:

$$\begin{aligned} &X(0) \\ &= x(0) \times \beta(0 \times 0) + x(1) \times \beta(1 \times 0) + x(2) \\ &\quad \times \beta(2 \times 0) + x(3) \times \beta(3 \times 0) = (72 \times 1 + 105 \times 1 + 32 \\ &\quad \times 1 + 32 \times 1) \bmod 127 = 241 \bmod 127 = 114 \end{aligned}$$

The second element is:

$$\begin{aligned} &X(1) \\ &= x(0) \times \beta(0 \times 1) + x(1) \times \beta(1 \times 1) + x(2) \\ &\quad \times \beta(2 \times 1) + x(3) \times \beta(3 \times 1) = (72 \times 1 + 105 \times 1 + 32 \\ &\quad \times -1 + 32 \times -1) \bmod 127 = 113 \bmod 127 = 113 \end{aligned}$$

The third element is:

$$\begin{aligned} &X(2) \\ &= x(0) \times \beta(0 \times 2) + x(1) \times \beta(1 \times 2) + x(2) \\ &\quad \times \beta(2 \times 2) + x(3) \times \beta(3 \times 2) = (72 \times 1 + 105 \times -1 \\ &\quad + 32 \times 1 + 32 \times -1) \bmod 127 = -33 \bmod 127 = 94 \end{aligned}$$

The fourth element is:

$$\begin{aligned} &X(3) \\ &= x(0) \times \beta(0 \times 3) + x(1) \times \beta(1 \times 3) + x(2) \\ &\quad \times \beta(2 \times 3) + x(3) \times \beta(3 \times 3) = (72 \times 1 + 105 \times -1 \\ &\quad + 32 \times -1 + 32 \times 1) \bmod 127 = -33 \bmod 127 = 94 \end{aligned}$$

Finally, the transform output of the input array is given by:

$$[X(0)\ \ X(1)\ X(2)\ X(3)] = [114\ 113\ 94\ 94].$$

(Note that $\beta(2 \times 2) = \beta(\langle 4 \rangle_L) = \beta(0)$, $\beta(2 \times 3) = \beta(\langle 6 \rangle_L) = \beta(2)$ and $\beta(3 \times 3) = \beta(\langle 9 \rangle_L) = \beta(1)$

IV. DESCRIPTION OF THE PROPOSED HASH FUNCTION SCHEME

In this section, the proposed hash function scheme is described in detail. The proposed hash function in this study is called HMNT. It takes an input message M of arbitrary length to generate a variable hash value H. Usually, HMNT supports three lengths of hash values, i.e. H = 128, 256 and 512 bits or longer. The HMNT process consists of the following steps:

Step 1: Convert the input message M into corresponding ASCII code value.

Step 2: The original message M is partitioned into number of blocks (m): $M = \{M_0, M_1, \ldots, M_{m-1}\}$. The length of each block is denoted as n, where n is the length of the hash value. The shortage in the last block is padded with the equivalent number of space characters in the ASCII code, which is 32.

Step 3: The secret key K is a series of characters, that modify the input message M. These characters also convert into corresponding ASCII code values. If the character length is less than the length of the hash value (n), the block is padded with the equivalent number of space characters in the ASCII code. Then, elements are added one by one to each block of the input message M.

Step 4: Upon modifying the input message using the secret key K, NMNT (a mathematical formula that performs mathematical operations to transfer each block of the message to the transform domain) is applied to each block in the input message as explained in Section III.

Step 5: The final hash value H of message M is obtained by the summation (element-by-element addition) of transform output NMNT to each block. The structure of the proposed HMNT hash function is illustrated in Fig.1.

V. SECURITY AND PERFORMANCE EVALUATION

In this section, the proposed hash function HMNT is evaluated in terms of the sensitivity of the hash value to the message, secret key and image, confusion and diffusion, the distribution of hash value, its collision resistance, its resistance to birthday, exhaustive key search and meet-in-the-middle attacks along with its flexibility. The results obtained from the simulation are then compared with some of the existing hash functions.

In order to evaluate the performance of our proposed scheme, we adopted methods that were used in previous literature [4], [6], [22]–[24]. The randomly used input message M and the secret key K in this evaluation are as follows:

M : "*Cryptographic hash functions is one of the most useful primitives for data security, which offers message authentication, data integrity, and digital signature.*"

K : "*abcdefghigkl12345*".

A. SENSITIVITY OF HASH VALUE TO THE MESSAGE

Theoretically, a good hash function must be sensitive to slight changes in the input message. In particular, any small change in the input message should lead to a 50% difference in the hash value, i.e. a Hamming distance of approximately $n/2$ (where n is the length of the hash value) between the two hash values. Therefore, to demonstrate the sensitivity of the proposed hash function scheme, we constructed several different messages by modifying the input message M given in Section V. We calculated the hash values of all resulting messages and compared under the following five conditions.

C1: The original message M is the same as that given in Section V;

C2: The first character "C" in the original message is changed to "c";

C3: The full stop "." at the end of the original message is changed to a comma ",";

C4: The word "hash" in the original message is changed to "Hash";

C5: Space is added to the beginning of the original message. The secret key, K given in Section V is fixed for all conditions.

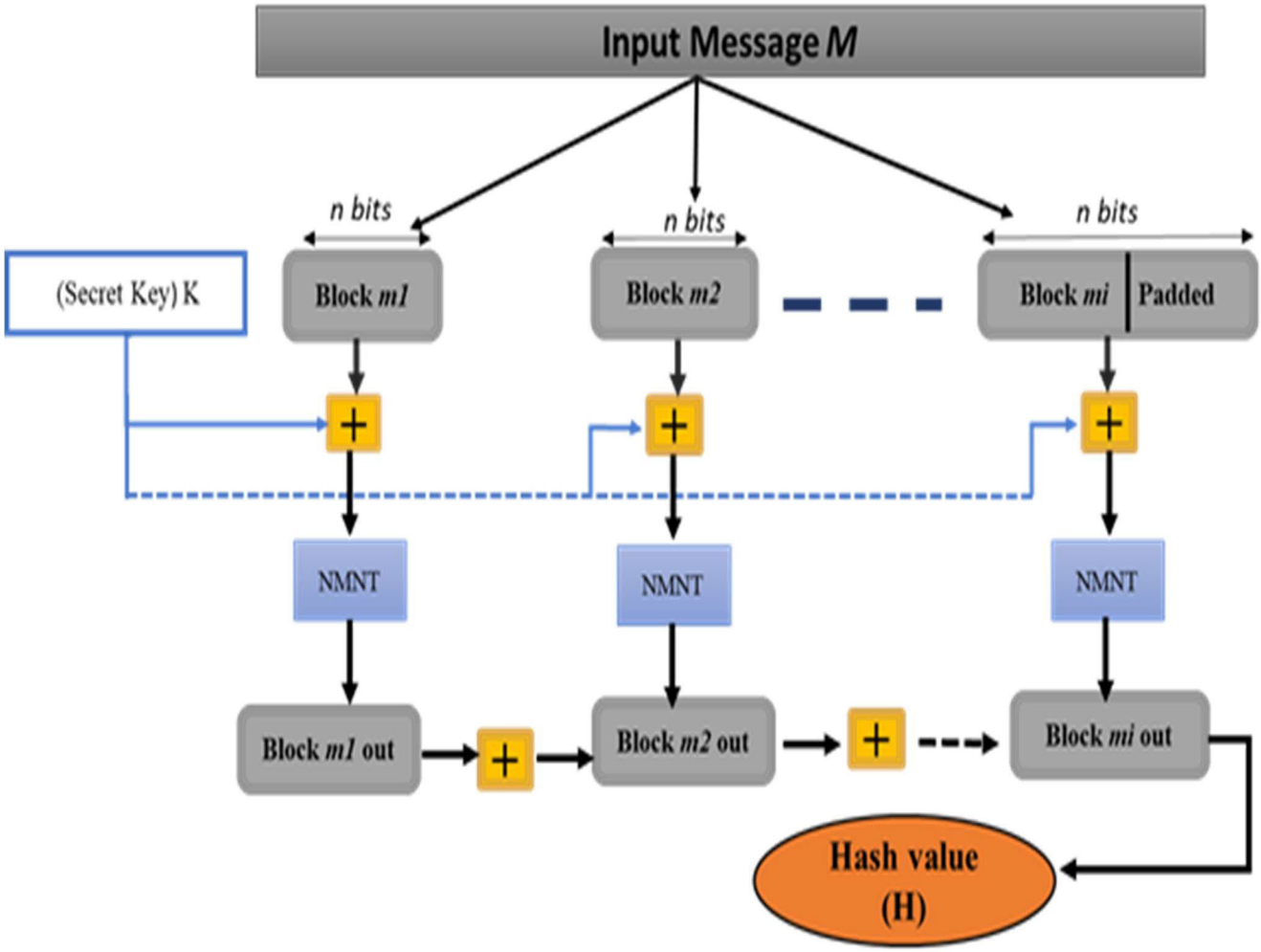


FIGURE 1. The structure of the proposed hash function HMNT.

Table 1, Table 2 and Table 3 summarize the results obtained for the corresponding lengths of 128, 256, and 512-bits hash values, respectively, in hexadecimal format for each condition. The resulting values of different bits are compared to the hash value obtained for C1, as well as the percentage of number of bits changed.

Based on the hexadecimal representation of the obtained hash values, a slight difference in the original message M causes a huge change in the hash value by about 50%. Thus, proving that the sensitivity of this proposed scheme is very high.

B. SENSITIVITY OF HASH VALUE TO THE SECRET KEY

Next, in order to evaluate the sensitivity of the hash value to the secret key, simulation experiments are carried out under five different conditions, as follows:

C1: The original secret key K as given in Section V;

C2: The first character "a" in the original secret key is changed to "A";

C3: The last character "d" in the original secret key is changed to "b";

C4: A "1" is added to the beginning of the original secret key;

C5: A "0" is added to the end of the original secret key;

The original input message M given in Section V is fixed for all conditions.

Table 4, Table 5 and Table 6 present the results obtained for the corresponding lengths of 128, 256, and 512-bits hash values, respectively, in hexadecimal format for each condition.

The tables also included the different bits of the hash value compared with the hash value obtained for C1 and the percentage of number of bits changed.

The results summarized in Table 4, Table 5 and Tables 6, clearly demonstrate that a small change in the secret key causes a significant change in the hash value. Hence, indicating that the proposed hash function is very sensitive to the secret key.

C. SENSITIVITY OF HASH VALUE TO THE IMAGES

In order to evaluate the sensitivity of the hash value to the image, a gray-scale Lena image with 256 × 256 image size (Fig. 2) is applied under three different conditions as follows:

TABLE 1. Corresponding hash values of length 128 bits in hexadecimal format associated with C1-C5, and number of different bits compared with the hash value of C1 for the message, *M*.

Condition	128 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	71e63a3d1c3b35e3c653317a38e238ce	0	0
C2	aaba5b4f550953d755075b875694ad70	68	53.12
C3	551856ae51345f32afa45a4faee1559a	67	52.34
C4	1ef43f1634751dc61db43547337d3a05	58	45.31
C5	3c2019c037c71ddb38ec317036641d6e	60	46.87

TABLE 2. Corresponding hash values of length 256 bits in hexadecimal format associated with C1-C5, and number of different bits compared with the hash value of C1 for the message, *M*.

Conditions	25 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	5dab5eac54b3ad33b5895b7e5847afb2aebdafe4ac84a94bae595ddb5b715e07	0	0
C2	ab6751e1d562d63cd59f50916bd952c555b9a8ded5995cbdaeae588c5b7255cc	131	51.17
C3	6bae5fe758ddd5675e55acb5af7ea9355b8e55116b7c58875aa4acfe d423 6a7c	143	55.85
C4	b5a9a9bdaf105d565e6bad245fa1544aac6e5ee1ab3451b8d7285ec8d5455b95	129	50.39
C5	6bb9ad41539ba8f8581ba8e6d6445eccac6aad5b5ebfad6b51c755505557a89e	126	49.21

TABLE 3. Corresponding hash values of length 512 bits in hexadecimal format associated with C1- C5, and number of different bits compared with the hash value of C1 for the message, *M*.

Conditions	512 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	6a156bbcdb4b6a8cdf6ed47fd9fadd95dfe66c0fdb8ed7f8d6bbd737db45 db16 b6f56f3fdb4bd77cde5ad445d3e96aafdfbfd2856a43dd6cb6eb6cb468d5 d2d8	0	0
C2	af31aff7b8585d00d7f5b881d6fdb4365c395c065fc2b168ad65b245bc70 b087 5c00b3f95d71b3fbb65cbd82588c5977ac77bbf0b286bec35cf9b350 5d7faceb	257	50.19
C3	b5f2bdff5fa0ebe7bd51b602b0eb59f3be1bac69bf25ba42b9fbb52bb919ba8e b13bd626b48faf91b308afa4b3685dc9bae7ae0bb64db58bb177eb50b613b625	260	50.78
C4	5a3cebaa5920d751eb7bf580b65cac9fb87cb101b4e15a4ab0fcbba9b3deb493 bea0d71ed60bb78bb7bb5a53aca05a615e7ebbf3ba31aff0b613bea3b238ba28	248	48.43
C5	b5f2f5c95f84f58ab2bdb6e158caadac5a2aba9bbf695e08ae135860b216bbe5b 13bad34bc62d60db3325c8deb90bbd2b41059c2bbb0b276b8b2b627b427aee9	262	51.17

C1: The original gray-scale Lena image with 256×256 image size as shown in Fig. 2.

C2: Add 1 to the gray value of the pixel located at the upper left corner;

C3: Subtract 1 to the gray value of the pixel located at the upper right corner.

The secret key, K given in Section V is fixed for all conditions.

The corresponding hash values of length 128, 256, and 512 bits in hexadecimal format for each condition; the different bits of the hash value compared with the hash value obtained for C1, and the percentage of the number of bits changed are listed in Table 7, Table 8 and Table 9.

Fig. 3 illustrates the wave graphic of 128 bits hash values for C1, C2 and C3.

Based on the hexadecimal and binary representations of the obtained hash values, any modification in the original image leads to significant differences in the hash value. This observation yet again proves the sensitivity of our proposed hash function, HMNT.

D. DISTRIBUTION OF HASH VALUE

Uniform distribution of hash value which is regarded as the most important properties of a hash function is directly related to the security of the hash function. In this section, we used two-dimensional graphs to present the distribution

TABLE 4. Corresponding hash values of length 128 bits in hexadecimal format associated with C1-C5, and the number of different bits compared with the hash value of C1 for the secret key, *K*.

Conditions	128 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	71e63a3d1c3b35e3c653317a38e238ce	0	0
C2	8dda8c6d3e2e3b061ab88dc919a93cfe	60	46.87
C3	3a9237ac3eb030343fa48dbb1f7a1e27	64	50.00
C4	312234c8c66d8e5d71cec69a3b338c45	69	53.90
C5	1ef43f1634751dc61db43547337d3a05	58	45.31

TABLE 5. Corresponding hash values of length 256 bits in hexadecimal format associated with C1-C5, and the number of different bits compared with the hash value of C1 for the secret key, *K*.

Conditions	256 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	5dab5eac54b3ad33b5895b7e5847afb2aebdafe4ac84a94bae595ddb5b71 5e07	0	0
C2	95da97b0c9a0918fcd29cf17cfba9e42940f9033978a9d59ced3904f 96d7cb45	124	48.43
C3	cab4c8676638caa092ec9e44cd9a94f919b092c63217c8829318cca8 959ece7f	135	52.73
C4	90229c1f93c290029569ca02929a335d98ce33b896529524cf5a6757 cf68c9a0	130	50.78
C5	66f191ab6711958091fbca95991dcefa65e5935ac833931a93ca9ea9 cd8c918d	131	51.17

TABLE 6. Corresponding hash values of length 512 bits in hexadecimal format associated with C1-C5, and the number of different bits compared with the hash value of C1 for the secret key, *K*.

Conditions	512 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	6a156bbcdb4b6a8cdf6ed47fd9fadd95dfe66c0fdb8ed7f8d6bbd737db45db16b6f56f3f db4bd77cde5ad445d3e96aafdfbfd2856a43dd6cb6eb6cb468d5d2d8	0	0
C2	af145918b196bb15b9d6afb75cabbdf0ba4cbd06aeb8b4b8b58d5844b859bf5ab5a6b0ce bedeae1ad79db23cb72bb9555ee9ad4db836b8dcb6825e2ebe44b6bb	243	47.46
C3	ac8eeb015adbb596b709bdbcad2bb3c1d618f5caad5c5e885abfb6d7d6f45f88ae615cb9 5e31ae25ebd25c19be205a70582ab4a8b62f b0195d58b3ffbbebbe15	253	49.41
C4	cbea9fcb9a9dce066599cb6e94b298abcafa928491586478661db201ceab9e81ccfe9347 c9c79a21c92dcf309016ca899e60cabe91fc9a469cedca259dd4d99b	260	50.78
C5	bfa7b55ceb65d680be40eb97be673d775c11aed25816b227b580b2beb65cad6dbe5a5cd 8 baecb6a7bc1bd6abbb8cb112ad9c5c51b348b122ba485d61b8d7bcff	259	50.58

of the original message *M* and its hash values (128, 256 and 512-bits).

First, the original message *M* is plotted as in Fig. 4(a). Referring to this plot, it is observed that the decimal ASCII code values of the original message *M* are distributed in a small range of [32.124]. Whereas Fig. 4(b), (c) and (d) which demonstrated the distribution of the corresponding hash values in hexadecimal format are uniform.

The extreme case of an ''all zero'' message with the same length is selected for a comparison in this study.

The distributions of the message are observed in Fig. 5(a) and the distributions of the corresponding hash values in Fig. 5(b), 5(c) and 5(d) are also uniform.

In short, based on the simulation results, no information about the message remained following the confusion and diffusion.

TABLE 7. Corresponding hash values of length 128 bits in hexadecimal format associated with C1- C3, and number of different bits compared with the hash value of C1 for gray-scale lena image.

Condition	128 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	d999d88d1a8edc0edb316afb46b96c9a	0	0
C2	981d4f183c5d27c83a0f 994536829cde	67	52.34
C3	781dfbd03d7fa1e3cd2f0c47b801ef4	66	51.56

TABLE 8. Corresponding hash values of length 256 bits in hexadecimal format associated with C1- C3, and number of different bits compared with the hash value of C1 for gray-scale lena image.

Conditions	256 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	ddc5bc2aed1bb886bf05ecdfdbc2b2c0bd9feeeedbbbde75d98c7639df89 bb2d	0	0
C2	714777b45dcd71ce78aa5c067e882f607013b8e97d4c9726714b70b85c5a76d0	128	50.00
C3	11478bd2885763d4c5201c0ac794c4371013164a898611068c61158e8cfd1bae	136	53.12

TABLE 9. Corresponding hash values of length 512 bits in hexadecimal format associated with C1- C3, and number of different bits compared with the hash value of C1 for gray-scale lena image.

Conditions	512 bits hash value in hexadecimal format	Number of different bits	Percentage of number of bits changed, %
C1	35bbcdef381d9d3a36579f2334013ab29fe29e9c377e3d7673b2cfcf34a6cde89c6c e76b9aaa9c483c2c37799e75e6113d519d833629cea1cd2adb93cd3466b2	0	0
C2	fe2ffb86ffa1f5a77fa5fb04ff85fff5f608f9e4f559f35af914ffb7feb5ff1df10ef8e4ff0e f5d2f39cff6bfd06f9d7f81cfed9f01df9d9ff55f436fcf8fc4b	259	50.58
C3	267e89dd97724fb999a325664f5727a24bab9c1925e892894c669e859fcc24c29859 13319aba4cc54fbb999c9db6978d9ca598869155494e4dd8120490d1949b	251	49.02

FIGURE 2. The standard gray-scale Lena image with 256 × 256 image size.

E. STATISTIC ANALYSIS OF CONFUSION AND DIFFUSION

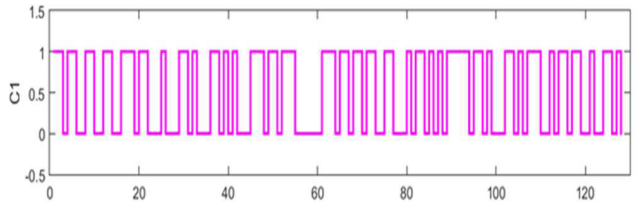
In cryptography, confusion and diffusion carry their own distinct definitions. Confusion defines the relationship between a message, whereby its corresponding hash code is complex and difficult to predict. Diffusion, on the other hand, explains that the hash value is extremely dependent on the message [23]. Therefore, for good diffusion, a one bit modification on an original input message will lead to a 50% change in the probability of each output bit. In order to analyze the confusion and diffusion capabilities of the proposed hash function, we have performed the following experiment. First, a random message M was selected and its hash value was calculated. Secondly, a single bit in the message M was randomly chosen and toggled to compute a new hash value. Finally, the two hash values were compared bit by bit, where the count number of the changed bits were marked as B_i. This experiment was repeated N times with a different length of hash values (128, 256 and 512) respectively. The evaluation of diffusion and confusion capabilities usually require six statistics that are defined as follows:

Minimum number of bits changed:

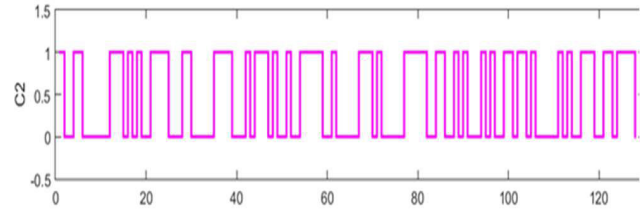
$$B_{min} = min\{B_1, B_2, \ldots, B_N\} \tag{15}$$

Maximum number of bits changed:

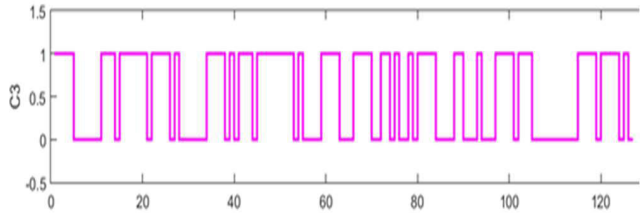
$$B_{max} = max\{B_1, B_2, \ldots, B_N\} \tag{16}$$



(a) The representation of the 128-bit hash value for C1



(b) The representation of the 128 bits hash value for C2.



(c) The representation of the 128 bits hash value for C3.

FIGURE 3. Corresponding binary sequence of 128 bits hash value for C1, C2 and C3. C1 is the original gray-scale Lena image.

Mean number of bits changed:

$$\bar{B} = \frac{1}{N}\sum_{i=1}^{N} Bi \tag{17}$$

Mean changed probability:

$$P = \left(\frac{\bar{B}}{h}\right) x100\% \tag{18}$$

Standard variation of the changed bit number:

$$\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \bar{B})^2} \tag{19}$$

Standard variation of the changed probability:

$$\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{B_i}{h} - P\right)^2} \times 100\% \tag{20}$$

where B_i denotes the changed bit number in the i^{th} test, N indicates the total number of the experiment and h represents the length of hash value.

Table 10, Table 11 and Table 12 were obtained by changing the single bit in original message M (given in Section V) and by executing the proposed hash function N times (for N = 256, 512, 1024 and 2048) in order to generate hash values with different size (128, 256, 512-bits). The number of changed bits between the original hash value and a new hash value is computed every time.

TABLE 10. Statistical results for 128 bits hash value.

Parameters	N=256	N=512	N=1024	N=2048	Mean
B_{min}	47	42	39	35	40.75
B_{max}	72	86	83	88	82.25
$\bar{B}$	73.58	68.69	70.32	71.18	70.94
$P(\%)$	57.49	53.66	54.93	55.61	55.42
ΔB	5.29	5.45	5.50	5.56	5.45
ΔP (%)	4.64	4.74	4.79	4.80	4.74

TABLE 11. Statistical results for 256 bits hash value.

Parameters	N=256	N=512	N=1024	N=2048	Mean
B_{min}	94	91	74	76	83.75
B_{max}	163	164	158	141	156.5
$\bar{B}$	133.83	133.38	142.66	135.58	136.36
$P(\%)$	52.28	52.10	55.73	52.96	53.26
ΔB	7.43	7.71	6.58	6.66	7.09
ΔP (%)	2.19	2.12	2.24	2.23	2.20

TABLE 12. Statistical results for 512 bits hash value.

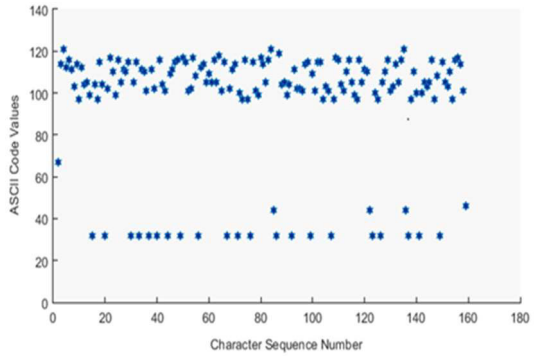
Parameters	N=256	N=512	N=1024	N=2048	Mean
B_{min}	165	184	171	161	170.25
B_{max}	308	313	318	323	315.5
$\bar{B}$	274.91	272.06	274.97	256.75	269.67
$P(\%)$	53.69	53.14	53.71	50.15	52.67
ΔB	5.38	5.32	5.37	5.02	5.27
ΔP (%)	2.78	3.16	3.38	3.27	3.14

According to the data presented in the Table 10, Table 11 and Table 12, we can conclude that the proposed hash function generated the mean number of bits changed $\bar{B}$ and the mean changed probability P values are very close to the ideal values (i.e. 64, 128, 256 bits, half of the length of hash value) and 50%. Furthermore, the standard variation of ΔB and ΔP are very small indicating storing capabilities for confusion and diffusion of the proposed hash function.

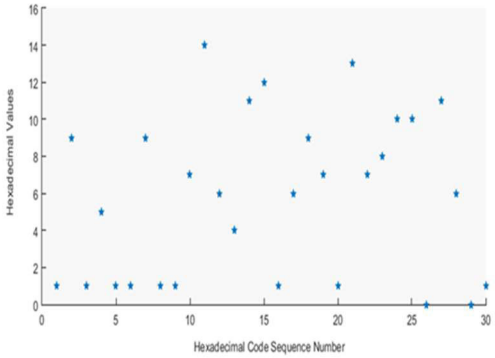
F. RESISTANCE TOWARDS COLLISION ATTACK

Two different input messages producing the same hash value is called a collision. In general, a property for a good hash function must possess is collision resistance. Hence, to analyze the collision resistance of the proposed scheme, the following experiment was carried out. First, a random message was generated and its hash value was calculated. Followed by a random modification of one bit of the original message to estimate the new hash value of the modified message. Then, both the hash values were transformed into hexadecimal format. Both hash values were compared and the number of equal two-hexadecimal characters at the same location was estimated. This count is referred to as the number of hits, N_h.

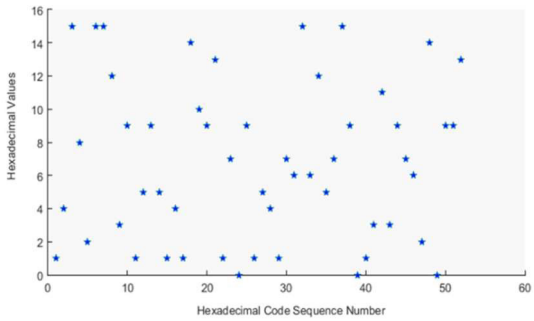
For example, the hexadecimal hash value of "*abcdefghijk*" is "19 0f 17 0b 14 62 18 40 05 4e 07 d4 14 5a 00 1d". If 'a' (01100001) is changed to 'A' (01000001) the hexadecimal hash value of the new message "*Abcdefghijk*"



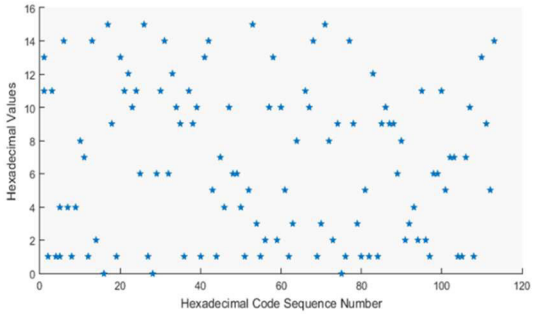
(a) Distribution of the original message *M* in ASCII code.



(b) Distribution of the length 128 bits hash value in hexadecimal format.



(c) Distribution of the length 256 bits hash value in hexadecimal format.



(d) Distribution of the length 512 bits hash value in hexadecimal format.

FIGURE 4. **Distribution of the original message *M* and its hash values of 128, 256 and 512-bits.**

becomes "0e c4 1b ac 00 90 01 62 19 9e 1a 75 19 e8 06 ed". Since there are no equal two-hexadecimal characters at the same location, $N_h = 0$.

This test was repeated $N = 2048$ times for hash values lengths of 128, 256 and 512-bits. Table 13 tabulates the numbers of hits, N_h where the two-hexadecimal characters are equal at the same location.

TABLE 13. **Number of hits in collision test for 128, 256 and 512-bits with $N = 2048$.**

PROPOSED HASH FUNCTION	Number of hits (N_h)				
	0	1	2	3	4
HMNT-128 BITS	1926	119	3	0	0
HMNT-256 BITS	1914	129	5	0	0
HMNT-512 BITS	1898	135	9	6	0

The maximum number of equal two-hexadecimal character in proposed hash function at 128 and 256- bits is only two, while at 512-bit it is three. These numbers indicate a very low collision probability. In addition, the absolute difference, *d,* between the 128, 256 and 512-bits original and modified hash values were also calculated using the following (21):

$$d = \sum_{i=1}^{N} (|t(a_i) - t(b_i)|) \tag{21}$$

where a_i and b_i are the ASCII characters of the original and the new hash values at position i respectively. The function t converts the ASCII to the decimal equivalent. Table 14 summarizes the corresponding results of the maximum, minimum and the mean values of d upon completing the collision test $N = 2048$.

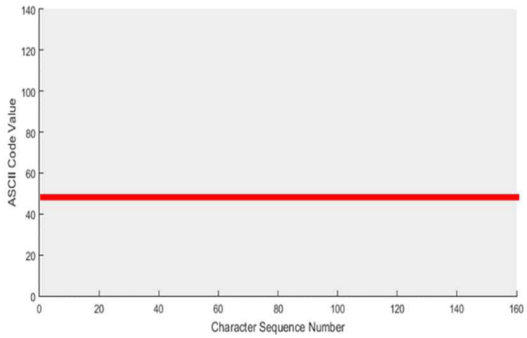
TABLE 14. **Absolute difference for 128, 256 and 512-bits hash values with $N = 2048$.**

Proposed hash function	Max	Min	Mean
HMNT-128 BITS	2.432	537	1372.7
HMNT-256 BITS	4235	1651	2740.3
HMNT-512 BITS	7234	3980	5348.6

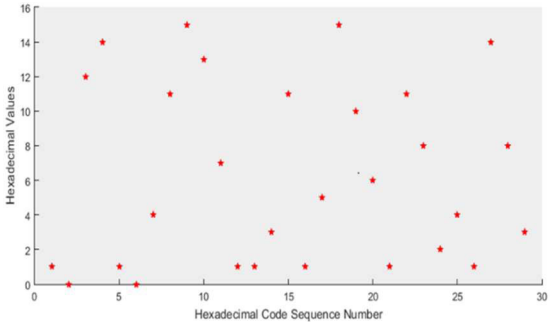
Meanwhile, the mean values of d between the original and modified hash values at 128, 256 and 512-bits are 1372.7, 2740.3 and 5348.6 as shown in Table 14 respectively. The mean values of d are very close to the theoretical values [25], suggesting strong collision resistance capability of the proposed hash function, HMNT.

G. RESISTANCE TOWARDS BIRTHDAY ATTACK

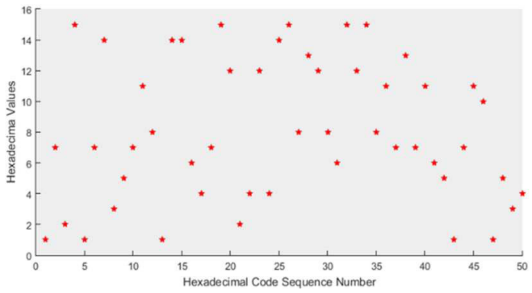
A birthday attack is a type of attack that is independent of the construction and can be applied on any hash function algorithm [26]. The attacker in this attack seeks to find two distinct messages (M, M') that have the same hash values h within fewer than $2^{n/2}$ trials (where n is the length of hash value) [4]. Thus, for $n = 128$, 256 and 512-bits, the proposed



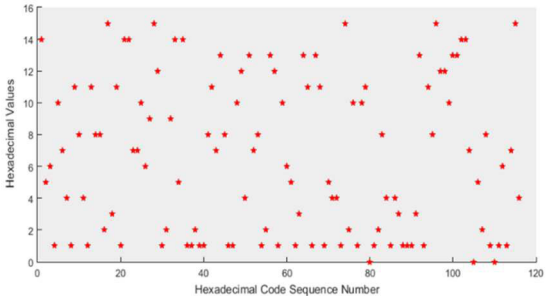
(a) Distribution of the "zero message" in ASCII code.



(b) Distribution of the length 128 bits hash value in hexadecimal format.



(c) Distribution of the length 256 bits hash value in hexadecimal format.



(d) Distribution of the length 512 bits hash value in hexadecimal format.

FIGURE 5. **Distribution of the "zero message "and its hash values of 128, 256 and 512-bits.**

hash function is identified sufficient to resist this type of attacks.

H. RESISTANCE TOWARDS MEET-IN-THE-MIDDLE ATTACK

This attack aims to find a collision in the intermediate hash chaining values rather than the final hash values. A collision can be found if there is a match between two intermediate hash chaining values [7]. However, this attack revealed that the proposed hash function is ineffective because we used a secret key as the initial value. The secret key made the inverse computation extremely difficult even with a small modification to the initial values. For instance, the original message was presented as $M = (M_1, M_2, M_3, \ldots, M_n)$, and expected contradicted message was represented as $M' = (M_1, M_2, M_3, \ldots, M'_n)$. The attack only replaced the last block in the message M_n: "*digital signature*", with a randomly selected message block M'_n: "*abcdefghigklmn*". The corresponding 128 bits hash value of the original message M_n and the replaced message M'_n in the hex-formats are:

M_n: "17e9 1831 170a 0657 1801 1a87 1138 09ee"

M'_n: "0997 1b1a 0250 186d 112b 01a2 03ed 1344"

Based on the output, M_n is clearly different from M'_n, Hence, the proposed hash function is identified to be resistant to the meet-in-the-middle (MITM) attack.

I. RESISTANCE TOWARDS EXHAUSTIVE KEY SEARCH ATTACKS

An exhaustive key search attack can be applied to any hash function that employs a secret key as an input. In a keyed hash function, if the attacker has access to a message/hash value pair, then the key can be found through exhaustive searching. So, on average, the attacker needs 2^{k-1} tries, where k is the size of the key. The proposed scheme is flexible, allowing the size of the secret key to be tuned. If the size of the key is set to 512 bits, the difficulty of the attack is 2^{512}. Since k = 512 bits, the proposed scheme is immune against this kind of attack.

J. FLEXIBILITY

On the other hand, the proposed scheme is also designed to manage problems including the length of the hash and resistance against common attacks like a collision. Since the proposed scheme is highly flexible, it can be used to produce hashes with the length of 128, 256 and 512-bits or longer, unlike the traditional fixed-length hash functions such as MD5 and SHA-1.

VI. COMPARISON WITH OTHER HASH FUNCTIONS

This section discusses the assessed comparison between the proposed hash function with existing and standard hash functions such as SHA-2 and SHA-3, that is based on statistical performance, collision resistance and speed.

A. COMPARISON OF STATISTICAL PERFORMANCE

Table 15, Table 16 and Table 17 present the comparison of statistical performance between the suggested hash function and recent hash functions. The results outlined in Table 15 are based on $N = 2048$ random test and 128 bits hash value, while Table 16 for $N = 2048$ random test and 256 bits hash value, and Table 17 focuses on $N = 2048$ random test and

TABLE 15. Comparison on statistical performance for 128 bits hash value with $N = 2048$ random tests.

Hash functions	Statistical performance of the hash functions					
	B_{min}	B_{max}	$\bar{B}$	$P(\%)$	ΔB	ΔP (%)
Ahmad *et al.* [4]	45	82	63.873	49.901	5.581	4.360
Li and Li [6]	47	81	64.1700	50.13	5.7488	4.49
Sen The, Tan and Alawida [16]	48	83	64.00	50.00	5.44	4.25
Li 's [23]	45	83	63.9438	49.9561	5.607089	4.3805
Todorova's [27]	45	89	63.99	49.99	5.51	4.33
Our proposed HMNT-128bit	35	88	71.18	55.56	5.56	4.80

TABLE 16. Comparison on statistical performance for 256 bits hash value with $N = 2048$ random tests.

Hash functions	Statistical performance of the hash functions					
	B_{min}	B_{max}	$\bar{B}$	$P(\%)$	ΔB	ΔP (%)
Liu, Kadir and Liu [24]	100	155	128.13	50.05	7.94	3.10
Todorova *et al.* [27]	102	161	127.94	49.97	7.94	3.1
Yang *et al.* [28]	N/A	N/A	131.6021	49.8493	8,1163	3.0744
SHA2-256	100	155	127.98	50.00	8.10	3.16
SHA3-256	100	155	128.01	50.00	8.07	3.13
Our proposed HMNT-256 bit	76	141	135.58	52.96	6.66	2.23

TABLE 17. Comparison on statistical performance for 512 bits hash value with $N = 2048$ random tests.

Hash functions	Statistical performance of the hash functions					
	B_{min}	B_{max}	$\bar{B}$	$P(\%)$	ΔB	ΔP (%)
Liu, Kadir and Liu [24]	222	289	265.19	50.04	11.20	2.19
Todorova *et al.* [27]	214	293	255.74	49.95	11.44	2.23
Kanso and Ghebleh [26]	224	292	256.24	50.04	11.02	2.15
SHA-512	N/A	N/A	256.084	50.016	11.232	2.194
Our proposed HMNT-512 bit	161	323	256.75	50.25	5.02	3.27

512 bits hash value. The analyses revealed that the statistical performance of the proposed hash function is very close to that of an ideal hash function algorithm. Furthermore, in comparison with the existing hash function, the proposed scheme performs statistically better than most of the hash algorithms presented in all the tables discussed in this section.

B. COMPARISION OF SPEED ANALYSIS

In order to evaluate the speed of the proposed hash function with varying hash lengths (128, 256 and 512-bits) for different sizes of input message, we implemented the proposed hash function in C# programming language on a device with an Intel Core i5-3110M CPU, 2.4 GHz 4 GB RAM and Windows 7 OS to calculate the hashing time (HT) in millisecond and the hashing-throughput (HTH)(Mb/s). Besides that, the number of cycles to hash one BYTE *NCpB* (Cycles/Byte) is estimated using the formula in [29] as follows

$$HTH(MBytes/s) = \frac{Message\ size(MBytes)}{Average\ hashingtime(s)} \quad (22)$$

$$NCpB(cycles/Byte) = \frac{CPU(Hz)}{HTH(Byte/s)} \quad (23)$$

The outcomes obtained are summarized in Table 18. Furthermore, the speed of the performance is compared between the proposed hash function with some of the recent hash functions, in terms of the *NCpB* along with their specified platforms as depicted in Table 19. Based on the results, the *NCpB* of the proposed HNMT is much faster than that of the *NCpB* obtained by Abdoun's [29] and the standard hash function SHA-2.

C. COMPARISION OF COLLISION RESISTANCE

Table 20 and Table 21 represent the comparison between the proposed hash function with existing hash functions in terms of the total number of position where the equal characters are identical in the 128 and 256-bits hash values when $N = 2048$.

As described in these tables, the values yielded by the proposed hash function are in agreement with some of those presented by existing hash schemes. Hence, the proposed hash function has very low collision.

Next, Table 22 summarizes the comparison between the proposed hash function with some of the recent hash functions available in the literature. The comparison is made in terms of the mean value of d of the two hash values for

TABLE 18. Hashing time hashing throughput and the number of cycles per byte for 128, 256 and 512-bits hash values.

Message of different sizes	HMNT-128			HMNT-256			HMNT-512		
	HT	*HTH*	*NCpB*	*HT*	*HTH*	*NCpB*	*HT*	*HTH*	*NCpB*
1MB	556.2	14.38	159.16	613.1	13.04	175.52	730.7	10.94	209.2
10MB	678.2	117.95	19.40	711.7	112.40	20.36	798.1	100.23	22.8
20MB	738.4	432.19	5.29	813.5	196.68	2.81	856.8	267.51	3.82

TABLE 19. Comparison of NCpB of the proposed hash function with other hash function for 100 mb message.

Hash functions	NCpB	Platform
Abdoun and Assad [29]	15.36	Intel Core-i5, 4 GB RAM
SHA-256 [29]	11.87	N/A
Our proposed HMNT-128	5.29	Intel Core i5, 4 GB RAM
Our proposed HMNT-256	2.81	Intel Core i5, 4 GB RAM
Our proposed HMNT-512	3.82	Intel Core i5, 4 GB RAM

TABLE 20. Comparison of collision resistance for 128 hash value with $N = 2048$.

Hash functions	Length of hash values in bits	Number of equal characters (N_h)			
		0	**1**	**2**	**3**
Ahmad *et al.* [4]	128	1923	121	4	0
Li, Ge and Xia [7]	128	1928	116	4	0
Li [18]	128	1919	126	3	0
Li [22]	128	1957	82	9	0
Our proposed HMNT-128	128	1926	119	4	0

TABLE 21. Comparison of collision resistance for 256 hash value with $N = 2048$.

Hash functions	Length of hash values in bits	Number of equal characters (N_h)			
		0	1	2	3
Abdoun and Assad [29]	256	1825	207	15	1
SHA-2	256	1817	215	16	0
Our proposed HMNT-256	256	1914	129	5	0

128 bits where $N = 2048$. The finding indicated that the mean of absolute difference for the proposed hash function is very close to the ideal value given in [25] than other hash functions. Hence, the proposed hash function yields a stronger collision resistance than most of the schemes used in this comparison.

In addition, Table 23 presents a comparison between the proposed hash function with the standard hash functions

TABLE 22. Comparison of absolute differences for 128 bits hash value with $N = 2048$.

Hash Algorithm	Maximum	Minimum	Mean
Li and Li [6]	1972	685	1253
Li [18]	2139	645	1388.6
Li [22]	2096	646	1425
Abdoun and Assad [29]	2213	730	1426.23
Yang *et al.*[30]	2554.3	861.7	1707.9
Proposed HMNT-128	2432	537	1372.7

TABLE 23. Comparison of the properties of the proposed scheme with standard hash functions.

Hash functions	Hash size (in bits)	Block size (in bits)	Max message size	Collision Found
SHA-2	256/512	512/1024	$2^{64-1}/2^{128-1}$	No
SHA-3	256/512	1088/576	∞	No
Our proposed HMNT	128/256/512	128/256/512	∞	No

namely SHA-2 and SHA-3, in terms of hash size, block size and collision occurrence.

Based on the values recorded in Table 23, the properties of the proposed hash function are comparable with those of other modern hash functions namely SHA-2 and SHA-3. Hence, the proposed HMNT is a viable candidate for a new hash function.

VII. CONCLUSION

This paper proposed a novel hash function scheme based on NMNT, called HMNT. It took an arbitrary input message to generate hash values of 128, 256 or 512-bits. The proposed hash function was evaluated in terms of the sensitivity of the hash value to the message, image and secret key, the distribution of hash values, statistical performance, the resistance of the scheme to birthday attacks and collision, along with the comparison with available hash functions. The results indicated that the suggested hash function scheme has a higher sensitivity to the original message, image and the secret key, and strong collision resistance. Moreover, the results also demonstrated that the proposed HMNT is flexible and efficient, hence, can be applied for authentication to ensure data integrity.

ACKNOWLEDGMENT

The authors would like to express their appreciation to Advanced Informatics Department, Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia for realizing and supporting this research work.

REFERENCES

[1] D. Wang, Y. Jiang, H. Song, F. He, M. Gu, and J. Sun, "Verification of implementations of cryptographic hash functions," *IEEE Access*, vol. 5, pp. 7816–7825, 2017.
[2] H. Tiwari and K. Asawa, "A secure and efficient cryptographic hash function based on NewFORK-256," *Egyptian Informat. J.*, vol. 13, no. 3, pp. 199–208, Nov. 2012.
[3] N. Abdoun, S. El Assad, M. A. Taha, R. Assaf, O. Deforges, and M. Khalil, "Secure hash algorithm based on efficient chaotic neural network," in *Proc. IEEE Int. Conf. Commun. (COMM)*, Bucharest, Romania, Jun. 2016, pp. 405–410.
[4] M. Ahmad, S. Khurana, S. Singh, and H. D. AlSharari, "A simple secure hash function scheme using multiple chaotic maps," *3D Res.*, vol. 8, no. 2, pp. 1–15, Jun. 2017.
[5] M. Turcanik, "Hash function generation based on neural networks and chaotic maps," in *Proc. Commun. Inf. Technol. (KIT)*, Vysoke Tatry, Sovakia, Oct. 2017, pp. 1–5.
[6] Y. Li and X. Li, "Chaotic hash function based on circular shifts with variable parameters," *Chaos, Solitons Fractals*, vol. 91, pp. 639–648, Oct. 2016.
[7] Y. Li, G. Ge, and D. Xia, "Chaotic hash function based on the dynamic S-Box with variable parameters," *Nonlinear Dyn.*, vol. 84, no. 4, pp. 2387–2402, Jun. 2016.
[8] X. Wang, X. Lai, and H. Yu, "Collisions for hash functions MD4 , MD5 , HAVAL-128 and RIPEMD," *IACR Cryptol. ePrint*, vol. 2004, pp. 4–8, Aug. 2004.
[9] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Proc. Annu. Int. Cryptol. Conf*, 2005, pp. 17–36.
[10] K. Wu, Y. Li, L. Chen, and Z. Wang, "Research of integrity and authentication in OPC UA communication using whirlpool hash function," *Appl. Sci.*, vol. 5, no. 3, pp. 446–458, 2015.
[11] P. Zhang, X. Zhang, and J. Yu, "A parallel hash function with variable initial values," *Wireless Pers. Commun.*, vol. 96, no. 2, pp. 2289–2303, Sep. 2017.
[12] S. Sanadhya and P. Sarkar, "New collision attacks against up to 24- step SHA-2," in *Proc. Int. Conf. Cryptol*, vol. 2, 2008, pp. 91–103.
[13] S. Verma and G. S. Prajapati, "Robustness and security enhancement of SHA with modified message digest and larger bit difference," in *Proc. Symp. Colossal Data Anal. Netw. (CDAN)*, Mar. 2016, pp. 0–4.
[14] M. J. Dworkin, *Sha-3 Standard?: Permutation-Based Hash and Extendable-Output Function*, Standard NIST FIPS-202, 2015.
[15] D. I. , O. Dunkelman, and A. Shamir, "Collision attacks on up to 5 rounds of sha-3 using generalized internal differentials," in *Proc. Int. Workshop. Fast. Soft. Encrypt*, Singapore, Mar. 2013, pp. 219–240.
[16] J. S. Teh, K. Tan, and M. Alawida, "A chaos-based keyed hash function based on fixed point representation," *Cluster Comput.*, vol. 22, no. 2, pp. 649–660, Jun. 2019.
[17] A. Maetouq, S. Mohd, N. Azurati, N. Maarop, N. Nur, and H. Abas, "Comparison of hash function algorithms against attacks: A review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 98–103, Sep. 2018.
[18] Y. Li, "Collision analysis and improvement of a hash function based on chaotic tent map," *Optik*, vol. 127, no. 10, pp. 4484–4489, May 2016.
[19] S. Boussakta, "New transform using the mersenne numbers," *IEE Proc.-Vis., Image, Signal Process.*, vol. 142, no. 6, pp. 381–388, 1995.
[20] M. T. Hamood and S. Boussakta, "Efficient algorithms for computing the new Mersenne number transform," *Digit. Signal Process.*, vol. 25, pp. 280–288, Feb. 2014.
[21] M. F. Al-Gailani and S. Boussakta, "Evaluation of one-dimensional NMNT for security applications," in *Proc. 7th Int. Symp. Commun. Syst., Netw. Digit. Signal Process. (CSNDSP)*, Newcastle upon Tyne, U.K., Jul. 2010, pp. 715–720.
[22] Y. Li, X. Li, and X. Liu, "A fast and efficient hash function based on generalized chaotic mapping with variable parameters," *Neural Comput. Appl.*, vol. 28, no. 6, pp. 1405–1415, Jun. 2017.
[23] Y. Li and G. Ge, "Cryptographic and parallel hash function based on cross coupled map lattices suitable for multimedia communication security," *Multimedia Tools Appl.*, vol. 78, no. 13, pp. 17973–17994, Jul. 2019.
[24] H. Liu, A. Kadir, and J. Liu, "Keyed hash function using hyper chaotic system with time-varying parameters perturbation," *IEEE Access*, vol. 7, pp. 37211–37219, 2019.
[25] A. Akhavan, A. Samsudin, and A. Akhshani, "A novel parallel hash function based on 3D chaotic map," *EURASIP J. Adv. Signal Process.*, vol. 2013, no. 1, pp. 1–12, Dec. 2013.
[26] A. Kanso and M. Ghebleh, "A fast and efficient chaos-based keyed hash function," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 1, pp. 109–123, Jan. 2013.
[27] M. Todorova, B. Stoyanov, K. Szczypiorski, and K. Kordov, "SHAH?: Hash function based on irregularly," *INTL J. Electron. Telecommun.*, vol. 64, no. 4, pp. 457–465, Oct. 2018.
[28] Y.-G. Yang, J.-L. Bi, D. Li, Y.-H. Zhou, and W.-M. Shi, "Hash function based on quantum walks," *Int. J. Theor. Phys.*, vol. 58, no. 6, pp. 1861–1873, Jun. 2019.
[29] N. Abdoun, S. El Assad, O. Deforges, R. Assaf, and M. Khalil, "Design and security analysis of two robust keyed hash functions based on chaotic neural networks," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 5, pp. 2137–2161, May 2020.
[30] Y. Yang, F. Chen, X. Zhang, J. Yu, and P. Zhang, "Research on the hash function structures and its application," *Wireless Pers. Commun.*, vol. 94, no. 4, pp. 2969–2985, Jun. 2017.

ALI MAETOUQ received the B.S. degree in computer engineering from the Faculty of Electronic Technology, Bani Waleed, Libya, in 2003, and the M.S. degree in computer engineering from The Libyan Academy, Tripoli, Libya, in 2010. He is currently pursuing the Ph.D. degree in computer engineering with the Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia. His current research interests include cryptographic hash functions, data security, and blockchain technology.

SALWANI MOHD DAUD (Member, IEEE) received the B.Eng. degree (Hons.) in electronics engineering from the University of Liverpool, in 1984, and the M.Eng. and Ph.D. degrees in electrical engineering from Universiti Teknologi Malaysia (UTM), in 1989 and 2006, respectively. She is currently a Professor with the Advanced Informatics Department, Razak Faculty of Technology and Informatics, UTM. She has been with UTM for more than 30 years, and has vast experience in teaching and research. She is also teaching Machine Learning and System Design for Security for postgraduate program. She is also leading few research grants in the related topics and had secured more than RM2 millions of Research and Development funds. She is also heading the Cyber Physical Systems Research Group. She also has published more than 100 academic articles in journal, proceedings, and books. Her research area is focusing on artificial intelligence, blockchain, and the IoT. She is a member of registered Professional Technologist from Malaysia Board of Technologists (MBOT) and registered Graduate Engineer with the Board of Engineers Malaysia (BEM).