Enhancing Security in Medical Image Integrity: A 4D Hyperchaotic Keyed Hash Algorithm Approach

Ali hasan Alwan[1*], Ashwaq T. Hashim[2], Suhad A. Ali[3]

[1] Department of Computer Science, Faculty of Computer Science and Mathematics, Universityof Kufa, Kufa 54001, Iraq, Email: alih.alashour@student.uokufa.edu.iq
[2] Department of Control and Systems Engineering, Universityof Technology-Iraq, Baghdad 10011, Iraq.
[3] Department of Computer Science, Science College for Women, Universityof Babylon, Babylon 51001, Iraq
*Corresponding Author

Received: 12.07.2024 Revised: 15.08.2024 Accepted: 24.09.2024

ABSTRACT
In recent years, the use of chaotic iterative maps in cryptographic systems, particularly chaotic hash functions, has seen significant expansion. The inherent sensitivity to small changes in initial conditions and control parameters within chaotic systems makes chaotic hash functions particularly suitable. Preserving the integrity of digital medical images is of paramount importance due to the challenges posed by even slight modifications. Diagnosing a precise disease from a slightly modified digital medical image becomes an insuperable task. To address this, the paper introduces a secure keyed hash algorithm based on a 4D hyperchaotic system. A pre-processing step involves obtaining the Unicode of each pixel in the Region of Interest (ROI) in a medical image, generating four sequences from the Unicode as initial values for the 4D hyperchaotic system. The 256-bit user key (K) is extended to form four sequences, serving as parameters for the 4D hyperchaotic system. The generation process produces a hash value with adjustable length. the proposed method, illustrating results closely approximating the ideal values, with an average of 50.007 in a message length of 512 bits over 2048 iterations. It's essential to note that the ideal values typically exhibit a 50%. Furthermore, the algorithm's speed is compared with other proposed algorithms in the research, with a recorded speed of 0.037, outperforming alternative methods which range between 3.484 and 0.070 Ms. A thorough performance evaluation demonstrates the effectiveness and flexibility of the proposed hash algorithm. proposed algorithm's effectiveness and flexibility. Results are highly promising, closely approximating the ideals.

Keywords: Cryptographic systems, Chaotic iterative maps, Chaotic hash functions, Image integrity, 4D hyperchaotic system.

1. INTRODUCTION

Ahash function is a critical component in the field of computer science and information security, serving as a fundamental tool with different applications. Essentially, a hash function takes input data, often referred to as a message, of changeable length and transforms it into a fixed-size string (h = H(x)) as output, typically a hash code or digest. This transformation is deterministic, meaning the same input will always produce the same hash output. The resulting hash code is commonly of a fixed length, regardless of the input size [1]. Moreover, it fulfilled the following properties [2]:

- The input can be of variable length.
- The output maintains a constant length.
- Computing H(x) for any given x is relatively straightforward.
- H(x) functions unidirectionally.
- H(x) is resistant to collisions, ensuring unique hash values for distinct inputs.

Diverging from traditional cryptographic methods such as MD5 and SHA-1, New trends in encryption have been widely developed, such as chaos-based. Chaos refers to a mathematical concept describing deterministic systems with sensitive dependence on initial conditions, Chaos-based encryption has emerged as a key facet of contemporary cryptography, owing to its random behavior, responsiveness to initial conditions and parameter values, and operational efficiency, as well as its characteristics of confusion and diffusion. Chaos-based cryptography demonstrates substantial potential in safeguarding assets [3], chaotic systems have found extensive applications in various fields, including engineering, mathematics, physics, biology, chemistry, and cryptography. This is attributed to their unique properties,

particularly their sensitivity to small changes in initial conditions and control parameters[4]. Even a slight alteration in the initial states can result in entirely divergent outcomes, leading to either a completely different path or a shift to a continuous state. This phenomenon is commonly known as the butterfly effect[5]. Chaotic systems are categorized into continuous and discrete forms. Continuous chaotic systems are based on differential equations, while discrete systems are similarly based on difference equations. Utilizing numerical solutions for differences, as opposed to differential equations, offers a significant boost in encryption speed. Importantly, there is no inherent advantage in favor of continuous chaotic systems over simple discrete chaotic systems. Consequently, any continuous chaotic system can be substituted with a discrete counterpart without compromising security[6]. Digital chaotic systems can be classified into one-dimensional (1D) and multidimensional (M.D.) systems. The use of M.D. chaos maps in image security is growing due to their intricate structure and utilization of multiple parameters. Nevertheless, the heightened complexity poses challenges in both hardware and software implementation, leading to increased computational complexity. These characteristics of chaotic systems are Getting into the design of a hash function based on the hyperchaotic system.

The proposed method in this study innovatively transforms the input image into features utilized comprehensively in generating the hash value. This approach allows for the production of variable-length hash values, ranging from 128 to 1024, adapting seamlessly to diverse scenarios and specific requirements. The method demonstrates effectiveness against various attacks, attributing this robustness to the properties elaborated in subsequent chapters. Notably, the proposed technique exhibits a speed improvement of more than double compared to previous methods, as evidenced in the performance metrics outlined in the scale evaluation chapter. Furthermore, these results hold true across different environmental conditions where the methods were implemented.

The subsequent sections of the paper are structured as follows: section 2 introduce the related works section 3 provides a concise overview of the hyperchaotic system. Following that, section 4 details the presented hash algorithm. The experimental evaluation is expounded in section 5, and the paper concludes with section 6, where present the final remarks

2. Related Works

The hash function is a cryptographic method that maps messages of varying length into fixed-length hash values, called hash codes. It is used for authentication and error detection. Attacks on hash functions include cryptanalysis and brute-force attacks. Hash functions are often combined with salt to increase complexity and resist brute-force attacks. Common hash function algorithms include MD5 and SHA-1[6]. Message Digest 5 (MD5) is a message-digest algorithm developed by Ron Rivest in 1991. The MD5 algorithm takes input of any length and produces a 128-bit output digest. The input is processed in blocks of 512 bits, which are divided into 16 sub-blocks, each 32 bits in size. MD5 was designed to replace the MD4 algorithm. Initially, MD5 was considered secure, but weaknesses were discovered in 2004, specifically related to collisions, where two different inputs produce the same hash result. This weakness is due to a flawed design of the MD5 algorithm, allowing for quick discovery of collisions. As MD5 is used in various applications, its weaknesses could have significant impacts, such as unauthorized access to accounts and privileges through brute force attacks [7], [8]. The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a Federal Information Processing Standard (FIPS 180) in 1993. When weaknesses in the algorithm were found, various revisions were made to create a better algorithm. The revised standard was published as FIPS 180-1 in 1995, and it became a reference for the creation of the new algorithm called SHA-1. SHA-1 accepts input of up to less than 264 bits and produces a 160-bit long output. This input restriction is one of the strengths of the algorithm, as it reduces the potential for collisions. SHA-1 processes input in blocks of 512 bits, which are then divided into 16 sub-blocks, each measuring 32 bits. In comparing the two methods in terms of advantages and disadvantages, MD5 codes any stream of bytes into a 128-bit value, while SHA1 codes any stream of bytes into a 160-bit value. Therefore, SHA1 will provide more security compared to MD5. The MD5 algorithm is cheaper to compute; however, MD5 is found to be more vulnerable to collision attacks[9].

Hash algorithms based on chaotic maps have been proposed in several papers. El-Meligy et al. propose improving the security of SHA-512 by modifying the hash buffer values and additive constants using DNA sequences and chaotic maps the proposed algorithm limitation is use only 1D chaotic maps that led to limited or discontinuous chaotic range, non-uniform data distribution [5], Zellagui et al. present a new hash function structure inspired by the sponge construction, which uses chaotic maps such as Zaslavsky 2D maps, logistic maps, and Henon maps[10].

Zhou et al. propose a bit-level image encryption scheme using a combination chaotic system (NCCS) and SHA-512 hash function, demonstrating the effectiveness of NCCS in enhancing security While this

algorithm offers no flexibility by accommodating images in color or grayscale of any size, it comes with a significant computational cost. [11].
Liu et al. propose an image encryption algorithm based on a 4D chaotic map and steganography, where the hash value of the plaintext image is used to control the generation of chaotic sequences the proposed algorithm offer flexibility in handling images of various sizes in both color and grayscale, this algorithm is associated with a substantial computational cost [12].
Wu et al. propose an image encryption algorithm based on an improved Kent mapping, which increases sequence complexity and key space using a hash function, The proposed approach is limited to the encryption of grayscale photos exclusively [13].
B. Rahul et al proposes a method for audio encryption based on chaos It utilizes the Henon map, Lorenz system, and logistic map to generate chaotic sequences, this technique is suitable for audio signals [14].
Ashwaq Proposed system for secure and verifiable biometric template protection that Uses merging techniques of chaotic shift keying (CSK) and secret image sharing (SIS), contains Two phases: ID image protection and template protection, Retains template protection, and robustness to malicious attacks [15].
Amira et al suggested the Hash algorithm used to protect data in telemedicine applications, Slantlet Transformation (SLT), and Discrete Cosine Transform (DCT) employed for Region of Interest (ROI) localized from MRI images for hash code calculation, Hash code enciphered using secure Chaotic Shift Keying (CSK) the Proposed method useful for JPEG compression and resistant to image processing attacks, this article did not comprehensively address all the metrics associated with the hash function[16].
Qussay F et al Combines hash signature and watermarking based on frequency domains Extracts essential features from ROI using SLT and DCT transformations Produces a watermark combined with Electronic Patient Registration (EPR) Embeds signature in RONI using DWT Proposed algorithm improves average NC value larger than 0.90 under all attacks This method is time-consuming [17].
De Rosal Ignatius Moses Setiadi et al employ bit plane slicing, system chaos, and hash functions to enhance privacy. Bit plane slicing divides image bits for independent scrambling using the chaos method, while a hash function encrypts the user input. Dynamic parameters generated by a hash function enable the chaos method to confuse each bit plane. Bit-plane join and diffusion processes follow, with a logistic map using a second key. The method proves sensitive to modifications and exhibits satisfactory performance through various measurement tools. Decryption is verified by SSIM=1, PSNR=∞, and BER=0. The method is efficient, providing real-time encryption in less than one second, the proposed algorithm is limited to gray scale images [18].
To enhance biometric identification security due to its sensitivity to even minor alterations, a new 2D chaotic sine map is proposed by Rahman et al for generating encryption keys and enhancing security for biometric identification systems. Traditional encryption and compression methods are ineffective for encrypting biometric images because of their high execution time and algorithm complexity. The scheme uses various methods such as the Hénon-Sine Map, Secure Hash Algorithm, DNA computing, and Mersenne Twister for shuffling pixel values. Additionally, the proposed model has been tested on different RGB biometric images specifically the SOCOFing dataset and the COVID-19 chest X-ray dataset and evaluated using performance metrics such as entropy, key space, and robustness analysis[7].
The rapid advancement in technology and high computing power have increased security attacks, leading to a demand for innovative security algorithms. It is crucial to address concerns about security and storage volume in medical images. A Generative Adversarial Network (GAN) model was proposed to enhance the security of medical images using a 2D chaotic map, hash-table, and Deep Learning (DL) by Kumar et al. The encryption process involves confusion and diffusion using Mersenne Twister (MT) and the Hénon map, along with Differential Huffman Compression (DHC) for lossless compression. The proposed model has been tested on different medical images and evaluated using various performance metrics [19].

3. The Hyperchaotic System

The dynamic system under scrutiny in this article is derived from the oscillator previously introduced by Liu et al. [16]. As illustrated in Equation 1, this is achieved by substituting the cubic nonlinearity with hyperbolic sine nonlinearity, this modification was implemented to simplify the practical circuit design, considering that the hyperbolic sine nonlinearity is realized through two diodes connected antiparallelly. In contrast, cubic nonlinearity is achieved through analog multipliers.

$$\begin{Bmatrix} x_1 = -\beta_1(x_2 + 0.2(x_1 - \varepsilon \sinh(x_1))) \\ x_2 = \beta_2 * x1 - x_2 + x_3 + x_4 \\ x_3 = \beta_3 * x_2 - x_4 \\ x_4 = -\beta_4 * x_1 \end{Bmatrix} \quad (1)$$

where the state variables are $x1, x2, x3,$ and $x4$ and positive parameters are $\beta1, \beta2, \beta3,$ and $\beta4$. The singular state variable responsible for the complex behavior is $x1$, which is associated with the hyperbolic nonlinearity [20].

4. Proposed Hash Algorithm

An algorithm for constructing a one-way hash function based on hyperchaotic dynamics is presented. The distinctive features of the hyperchaotic system, coupled with a user key, are harnessed to generate a hash value. Chaotic systems are characterized by extreme sensitivity to initial conditions, and, in this context, the region of interest (ROI) pixels serve as the initial conditions for a hyperchaotic system. These pixels undergo pre-processing as float numbers, and a key is utilized as the initial set of parameters for the hyperchaotic system. these features are aptly utilized to produce a valid hash value that meets the necessary criteria. The inclusion of a key provides the recipient with robust evidence that the message was created by a known sender and has not been altered during transit. Initially, the ROI is identified through image processing steps [21]. Subsequently, a hash algorithm H (ROI, L, key) is generated using algorithm (1), employing a unique 256-bit key assigned to each ROI. **Error! Reference source not found.**illustrates the block diagram of the proposed system and Algorithm 1 is also define the steps of Generation of hash value, and the flowchart in Figure 2.

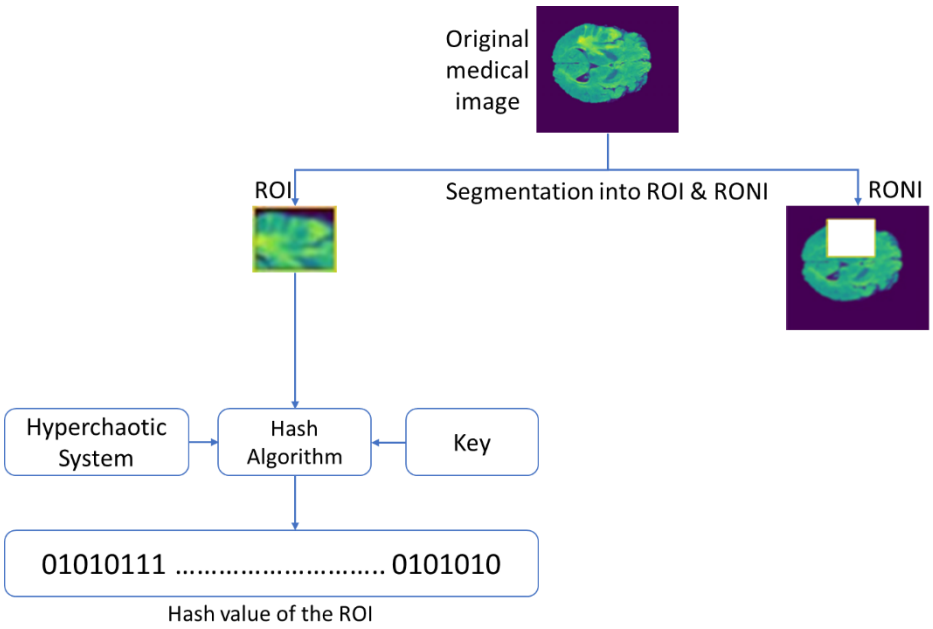


Fig 1. Block Diagram of the Proposed Hash Value Generating.

Algorithm 1: Generation of Hash Value
Input: ROI // Region of interest region Width, Height Key //256-bit key L // Hash value length
Output: H // hash value with L bit
Step 1: Convert the ROI sub-image into 1D as P with n=Width×Height pixels. **Step 2:** For each pixel $p(i) \in P, i = 1, 2, \ldots, n$, transform it into a real value using equation (3-9) to obtain m(i) which will serve as varying initial values $x1, x2, x3,$ and $x4$of equation (1). $m(i) = p(i) \times 10^{-4}$ 4-

1

Step 3: Divide m into four 1D arrays of$M1, M2, M3,$ and M4with the same length n/4such as follows:

$$\text{for } j = 1 \text{ to } n$$
$$M_1(j) = m(j),$$
$$M_2(j) = m(j+1),$$
$$M_3(j) = m(j+2),$$
$$M_4(j) = m(j+3),$$
$$j = j + 4$$
$$\text{Endfor}$$

Step 4: Set the parameter values for the hyperchaotic system in equation (1) by transforming a 256-bit key into its hexadecimal number K = (k(1), k(2), . . . , k(64)), and then generate four initial parameters $\beta_1, \beta_2, \beta_3,$ and β_4 using Eq. (3-10):

$$\begin{cases} \beta_1 = \sum_{i=1}^{32} k(i) \times 10^{-4}, \\ \beta_2 = \sum_{i=17}^{32} k(i) \times 10^{-4} \\ \beta_3 = \sum_{i=33}^{48} k(i) \times 10^{-4} \\ \beta_4 = \sum_{i=49}^{64} k(i) \times 10^{-4} \end{cases}$$

4-2

Step 5: Image concentration by repeating equation (1) sixteen rounds with initial values $x_1(1)$, $x_2(1)$, $x_3(1)$, and $x_4(1)$ such as following:

$x_1(1) = M_1(1),$
$x_2(1) = M_2(1),$
$x_3(1) = M_3(1),$
$x_4(1) = M_4(1)$

From the second round, set the initial values as follows:

$$\text{for } i = 1 \text{ to } n/4$$
$$\begin{cases} x'_1 = \beta_1(x_2 + 0.2\,(x_1 - \epsilon \sinh(x_1))) \\ x'_2 = \beta_2 x_1 - x_2 + x_3 + x_4 \\ x'_3 = -\beta_3 x_2 + x_4 \\ x'_4 = -\beta_4 x_1 \end{cases}$$
$$x'_1 = x'_1 + M_1(i+1),$$
$$x'_2 = x'_1 + M_2(i+1),$$
$$x'_3 = x'_1 + M_3(i+1),$$
$$x'_4 = x'_1 + M_4(i+1),$$
$$\text{Endfor}$$

Step 6: After repeating equation (1) 300 rounds to remove the transitory process, stay to iterate it L/16 times (i.e., L=512) using four initial values $x_1(n/4)$, $x_2(n/4)$, $x_3(n/4)$, and $x_4(n/4)$ within the turn to obtain four variable sequences $X_1(j)$, $X_2(j)$, $X_3(j)$, and $X_4(j)$, $j \in [1, L/16]$. Transform them into unsigned integers within the interval [0, 255], such as the following:

$$\text{for } j = 1 \text{ to } L/16$$
$$X(j) = \text{uint8}(\text{mod}(X_1(j), 256)),$$
$$Y(j) = \text{uint8}(\text{mod}(X_2(j), 256)),$$
$$Z(j) = \text{uint8}(\text{mod}(X_3(j), 256)),$$
$$W(j) = \text{uint8}(\text{mod}(X_4(j), 256)),$$
$$\text{Endfor}$$

Step 7: Extract the last h=16 iterations to get the values Width, Height, of X , Y , Z and W and generate 512-bit hash value H = H1|H2|H3|H4 in hexadecimal format by Eq. (3-11), finally, ‘as

$$\begin{cases} H_1 = \text{reshape}\left(\text{dec2hex}\left(\text{mod}(X(1{:}h) \times 10^{14}, 256)\right), 1, 2h\right) \\ H_2 = \text{reshape}\left(\text{dec2hex}\left(\text{mod}(Y(1{:}h) \times 10^{14}, 256)\right), 1, 2h\right) \\ H_3 = \text{reshape}\left(\text{dec2hex}\left(\text{mod}(Z(1{:}h) \times 10^{14}, 256)\right), 1, 2h\right) \\ H_4 = \text{reshape}\left(\text{dec2hex}\left(\text{mod}(W(1{:}h) \times 10^{14}, 256)\right), 1, 2h\right) \end{cases} \quad (3)$$

Step 8: Return H

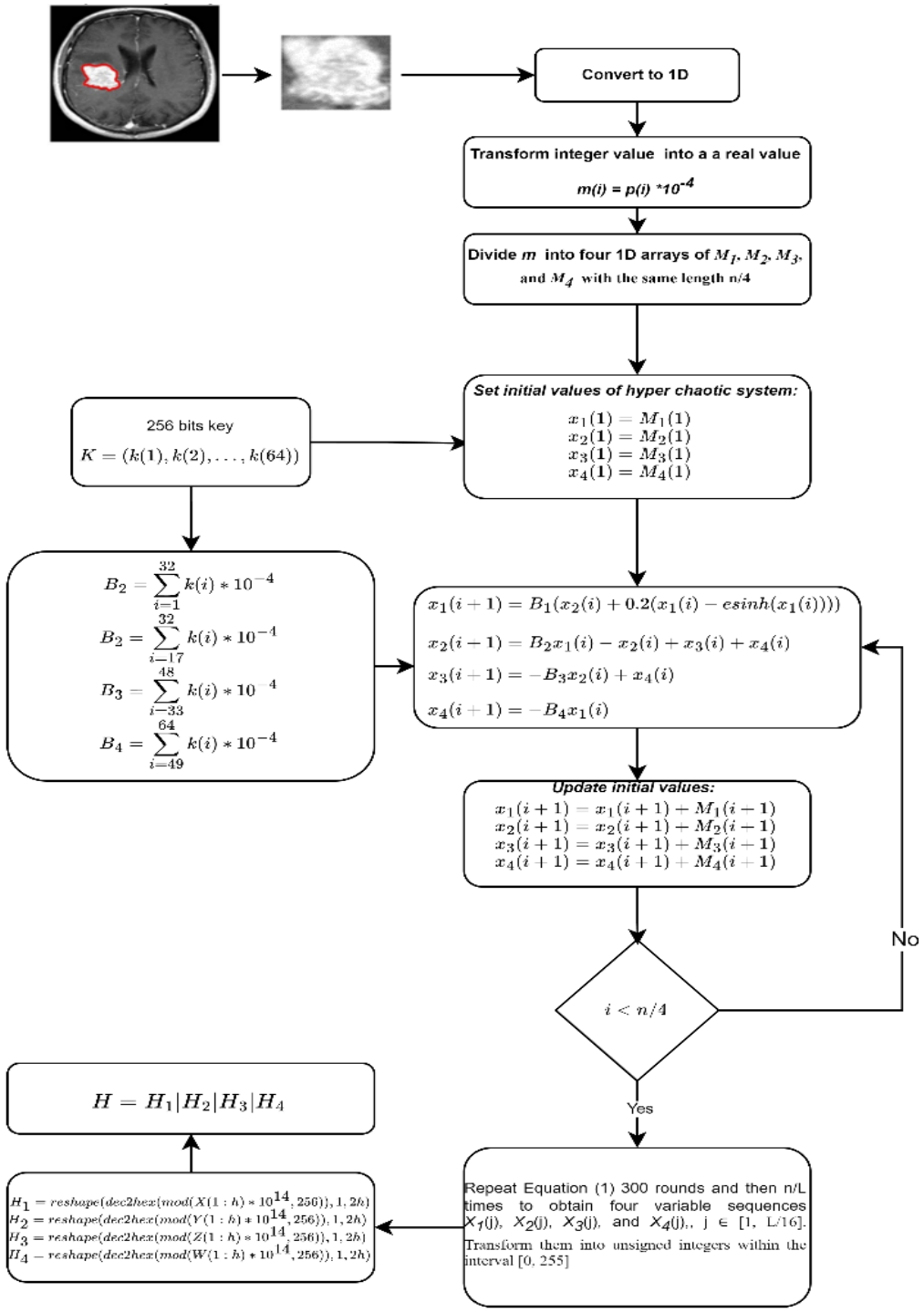


Fig 2. The flowchart of the keyed hash algorithm.

5. Experimental Results

The proposed algorithm was examined using an image from the Brats 2020 dataset after cropping a

portion of the tumour from the image. A single pixel value in the image was modified, and the results were evaluated after this alteration.

5.1 Distribution Statistics

The even distribution of hash values is intricately tied to the security of the hash function. In this context, employed 2D histograms to visually represent the distribution of the primary image, H 256, and H 512. In Fig (a), the Unicode values of the input image undergo translation. Meanwhile, in Figures 3(b) and 3(c), the hexadecimal hash values are uniformly dispersed.Uniformly dispersed values in hash functions are preferable to avoid collisions, achieve better load distribution, and ensure security. A uniform distribution contributes to more efficient searches, prevents stacking of values, balances load distribution, and enhances system security by reducing the likelihood of collisions.

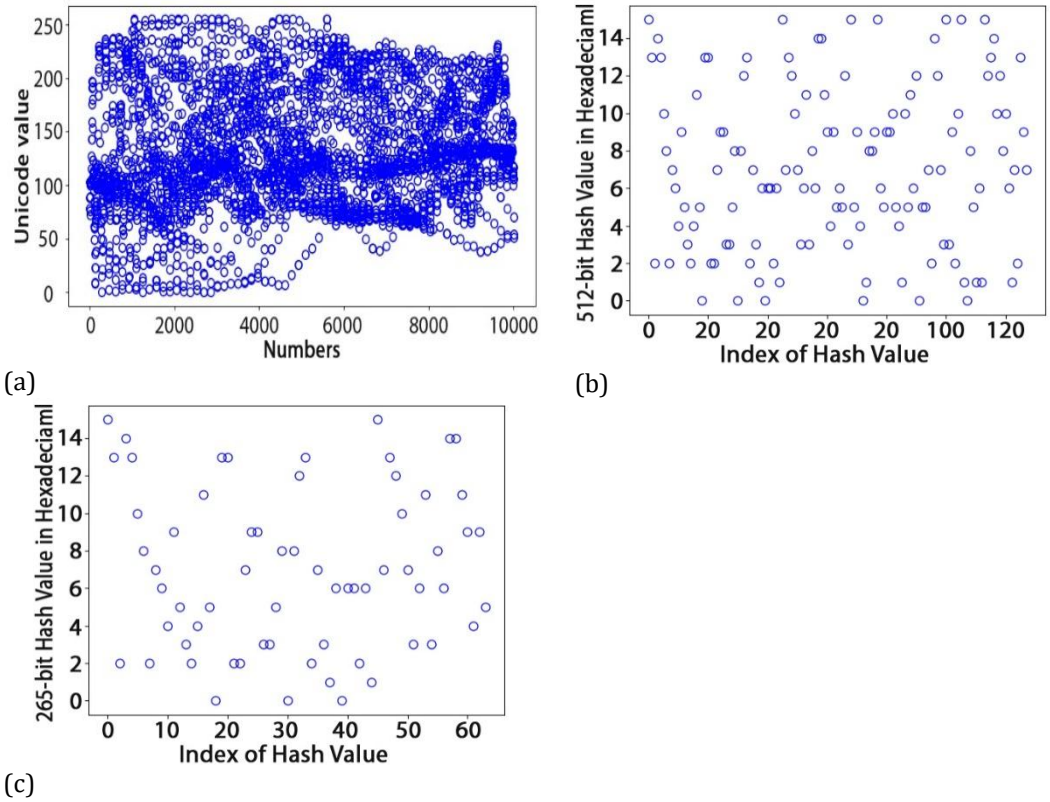


Fig 3**.** Distribution of message and 256, 512-bit hash values.

5.2 Key Space Analysis

The larger the key size, generally, the more secure the encryption, as it increases the difficulty for adversaries to perform brute-force attacks or other cryptographic attacks. The specific key size used depends on the encryption algorithm and the security requirements of the system .The proposed hash algorithm incorporates a one-time 256-bit external key, denoted as K, ensuring a key space $S = 2^{256}$ large enough to resist brute-force attacks [17]. The algorithm derives initial values, $\beta_1, \beta_2, \beta_3and\beta_4$, by summing non-symmetric values, as detailed in Algorithm (1). This approach ensures diverse and resilient values against attacks in the context of the discrete nonlinear differential hyperchaotic system. Notably, the precision of the state variables is impressive, reaching up to 10^{-15} [21],[22].A comparison was made between the proposed method and some previous methods, focusing on the key space. The results, as shown in
Table **1**below, highlight the superiority of the proposed technique in terms of key areas.

Table 1. Key space comparison

	Key Space size
[22]	2^{298}
[23]	2^{249}
[24]	2^{106}
[25]	2^{279}

Proposed	2^{256}

This integration emphasizes the importance of key size in encryption, highlighting the use of a 256-bit external key in the proposed hash algorithm. The mention of a key space of 2^{256} underscores the resistance against brute-force attacks, aligning with the general principle that larger key sizes enhance security. The precision of state variables and the comparative analysis further contribute to showcasing the robustness and superiority of the proposed technique, especially in terms of key areas

.

5.3 Hash Sensitivity

Based on the chaotic map, a robust hashing algorithm should be sensitive to small changes in the input message, conditions, and initial parameters. Here, conduct several experiments to verify this. In the following experiments, the proposed algorithm took an image, Alpha, extracted the return on investment (ROI), and converted it into a vector, denoted as $H_{Orignal}$, representing the original image. Randomly, altered one percentage of the image's numbers to obtain H_x. The corresponding hash values for them, with sizes of 256 bits and 512 bits, are shown below.

$H_{Orignal\ (256)}$ =fd2eda8276495324b50dd22799335808cd27316066261f7dca736b386eeb9495

$H_{1(256)}$ =a9d88fdf2387347310ba1d2499335808c925baae0fc71a6ebd60df3a5a00dcf1

$H_{2(256)}$ =fd2eda82907ef9aec60ed22799335808cd12e6fbc1e1147dca736b386eeb9495

$H_{3(256)}$ =fd2eda8276495324b50dd28488cfb27a80c28bef580d126a3e75b02b0b454f8a

$H_{4(256)}$ =fd1600b61dd36424b50dd2279d3366cc55ecae415a13937fca736b386eeb3b90

$H_{5(256)}$ =0e4c0e28773c378d524cbc9b9b337390271c46115b261f7dca736b386eeb9450

$H_{Orignal\ (512)}$ =
fd2eda8276495324b50dd22799335808cd27316066261f7dca736b386eeb949565c3f59401889f6599a541a5b6c05572ec73f392af1085161fcdeac8a6172d97

$H_{1(512)}$ =
ff2fda82ed2b9d7f6158cf2799335808cd27316066261f7dca736b386eeb949565c3f59401889f6599a541a5b6c05572ec73f392af1085161fcdeac8a6172d97

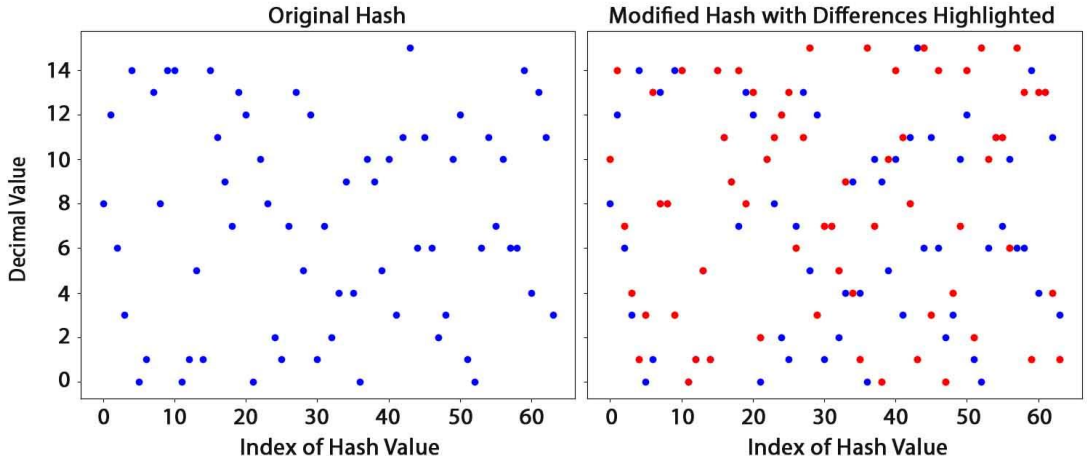
$H_{2(512)}$ =
b923d98276495380a6aa2ce28e335808cd27891d8d14c465a4d70a9229e0949565c31263363da89d8993b5a7b6c05572ec73f392f4ec7c0a0d41ecc8a6172d42

$H_{3(512)}$ =
b923d98276495324b50dd227541cf462881c890cff68cd5f3e756b386ee8a9ca0b4d079501889f6599a541a5b6c05572ec73f392af1085161f01826200d22297

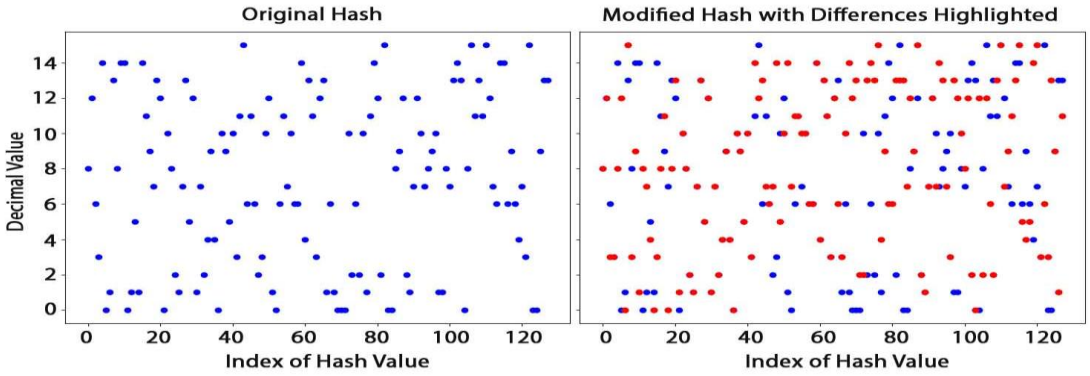
$H_{4(512)}$ =
e61c4e8476495324b50dd28488cfb2c3c227316066261f7dca736b386eeb949565c3f59401889f6599a5f248001c01bde873f392af1085161fcdeac8a6172d97

$H_{5(512)}$ =
fd2eda82ac6887ca3f1fd2279933580843ddcbbb221b1f7dca7312245ddfb5279b6a7fa601889f6599a541a571e68a197584f392af0cb09f028e0c289c04a199

It can also be made clearer by incorporating drawings and indicating the numbers that have changed between the original and modified hash values, as illustrated in the figure 4.



(a)



(b)

Fig 2. hexadecimal values of the original and modified pictures are highlighted to indicate the changed values, (a) for message 256-bit hash value (b) for message 512-bit hash value

5.4 Statistical Analysis of Diffusion and Confusion

The concepts of diffusion and confusion play pivotal roles in ensuring the robustness and security of algorithms. Diffusion involves the dispersal of input changes throughout the entire ciphertext, making it challenging for patterns to emerge. On the other hand, confusion aims to obscure the relationship between the plaintext and the ciphertext, adding a layer of complexity.

defining several metrics for quantifying the performance of diffusion and confusion in cryptographic algorithms. The summarize as the following:

$\boldsymbol{\beta_{min}}$

Definition: The minimum number of bits that change across all iterations.

Purpose: Indicates the least amount of alteration in the output over multiple iterations, providing insights into the stability of the algorithm.

$\boldsymbol{\beta_{max}}$

Definition: The maximum number of bits that change across all iterations.

Purpose: Reveals the highest degree of alteration in the output, highlighting the algorithm's capability to introduce variability.

$\check{\beta}$

Definition: The mean value of the number of changing bits.

Purpose: Represents the average level of alteration, offering a central tendency measure for the changing bits across iterations.

ρ

Definition: The mean probability value.

Purpose: Represents the average probability value associated with the cryptographic algorithm. This could relate to the probability of certain bits changing during the encryption process.

$\Delta\beta$

Definition: The standard deviation of the number of changing bits.

Purpose: Quantifies the degree of variability or dispersion in the changing bits, providing information on the algorithm's consistency or lack thereof.

$\Delta\rho$

Definition: The standard deviation of probability values.

Purpose: Measures the extent of variation in probability values, giving insights into the consistency of the algorithm's behavior.

These metrics collectively offer a comprehensive set of quantitative measures to assess the diffusion and confusion properties of cryptographic algorithms. Analyzing these metrics over multiple iterations provides a detailed understanding of how well the algorithm disperses changes and obscures relationships in the encrypted output.

It is crucial to establish a scale for measuring the distance between hash values. The distance metric used in the proposed scheme is the Bit Error Rate (BER) or the standard Hamming distance, defined as follows:

$$d_{xy} = \frac{1}{N}\sum_{i=0}^{N-1}|x_i - y_i| \quad (5)$$

when comparing two binary vectors of length N, denoted as x and y, the block hash distance is assessed against a predefined threshold. The two images (or blocks) are deemed similar if the distance is below

the threshold. Perfect similarity corresponds to a 0 BER value, while two entirely dissimilar images should yield a BER value close to 0.5, indicating a lack of similarity. Changing a single bit to the 'K' in a hash value, represented in hexadecimal form as detailed in Table 2, results in a Hamming distance that reveals even minor modifications to images or the key will result in a 50% difference in the hash value.

Table 2. Statistical results for 265-bit hash value.

N=	**256**	**512**	**1024**	**2048**	**4000**	**1000**
β_{min}	85	95	76	99	98	97
β_{max}	148	152	154	159	155	158
$\breve{\beta}$	127.98	128.625	128.135	127.50	127.84	127.92
ρ	50.03	50.24	50.05	49.99	49.94	50.03
Δβ	8.303	8.23	8.35	8.08	8.02	7.98
Δρ	3.16	3.21	3.26	3.15	3.13	3.11

Table 3. Statistical results for 512-bit hash value.

N=	**256**	**512**	**1024**	**2048**	**4000**	**1000**
$\boldsymbol{\beta_{min}}$	180	212	194	181	180	170
$\boldsymbol{\beta_{max}}$	294	286	304	299	291	299
$\breve{\boldsymbol{\beta}}$	256.3	255.5	255.6	256.0	255.9	256.05
ρ	50.06	49.91	49.93	50.00	49.99	50.01
Δβ	12.68	11.46	11.91	11.33	11.28	11.404
Δρ	2.47	2.23	2.32	2.21	2.20	2.22

Where β_{min} is the minimum number of changing bits in all iterations β_{max} the maximum number of changing bits $\breve{\beta}$ denoted the mean value. ρ mean probability value, ΔβStandard Deviation, Δρ Standard Deviation probability.

Table 4. Comparison with other algorithm for 256-bit hash value and N = 2048.

Hash Algorithm	$\boldsymbol{\beta_{min}}$	$\boldsymbol{\beta_{max}}$	$\breve{\boldsymbol{\beta}}$	**ρ**	**Δβ**	**Δρ**
SHA2-236	100	155	127.98	50.00	8.10	3.16
SHA3-236[26]	100	154	127.96	49.98	8.04	3.14
[27]	101	168	128.08	50.03	8.12	3.21
[28]	103	155	128.00	50.00	8.02	3.18
[29]	102	161	128.00	50.00	8.02	3.13
[30]	99	152	128.15	50.06	7.80	3.05
Proposed algorithm	99	159	127.50	49.99	8.08	3.15

Table 5. Comparison with other algorithm for 512-bit hash value and N = 2048

Hash Algorithm	$\boldsymbol{\beta_{min}}$	$\boldsymbol{\beta_{max}}$	$\breve{\boldsymbol{\beta}}$	**ρ**	**Δβ**	**Δρ**
SHA2-512	215	292	256.27	50.05	11.51	2.25
SHA3-512	216	287	254.75	49.76	11.32	2.21
[28]	219	296	255.89	49.98	11.27	2.20
[29]	223	296	256.10	50.02	11.142	2.17
[30]	220	294	255.69	49.94	11.12	2.24
Proposed algorithm	181	291	256.03	50.007	11.33	2.21

5.5 Speed Analysis and Comparison

As widely acknowledged, hash functions relying on chaos tend to exhibit lower performance due to the involvement of floating-point computations. In proposed algorithm employed a 4D hyper-chaotic system, leveraging fast operations to enhance the speed of the hashing process. The comparative performance of the proposed algorithm is detailed in **Error! Reference source not found.**, demonstrating its speed.

Table 6. Hashing speed comparison.

Algorithm	**Speed (Gbps)**	**Platform**
[31]	3.484	Intel core-i5, 8GB RAM
[32]	0.911	Intel core-i7, 16GB RAM

[23]	0.697	Intel core-i7, 16GB RAM
[33]	0.630	Intel core-i5, 16GB RAM
[34]	0.129	Intel core-i3, 6GB RAM
[29]	0.070	Intel Pentium IV (Dual core), 2GB RAM
Proposed algorithm	0.037	Basic Colab Subscription

the results revealed a notable performance improvement with the proposed algorithm, achieving a processing time of 0.037 seconds. In comparison, the previous method yielded the following results: 3.484, 0.911, 0.697, 0.630, 0.129, 0.070 seconds.
the significant reduction in processing time, as evidenced by the proposed algorithm's efficiency (0.037 seconds), underscores its noteworthy contribution to enhancing computational speed. This improvement is crucial in the context of image processing and security applications, as it can lead to more responsive and real-time processing, ultimately enhancing system performance and reliability.

6. CONCLUSIONS

In summary, a proposed keyed hash algorithm has been developed based on a hyperchaotic system, serving as a function to dissolve the Region of Interest (ROI) in a 4D hyperchaotic system. This algorithm exhibits flexibility by generating hash values of 256, 512, 1024 bits, or longer, controlled by parameters like L and h. The avalanche effect ensures that a single bit change causes a 50% probability of change in each output bit. Theoretical analysis and computer simulations confirm the algorithm's efficiency and flexibility, making it a suitable choice for data integrity and authentication.

REFERENCES

[1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of applied cryptography. CRC press, 2018.

[2] W. Stallings, “Cryptography And Networksecurity Principles Andpractice.” 2011.

[3] M. K. Khan, J. Zhang, and X. Wang, “Chaotic hash-based fingerprint biometric remote user authentication scheme on mobile devices,” Chaos Solitons Fractals, vol. 35, no. 3, pp. 519–524, Feb. 2008, doi: 10.1016/j.chaos.2006.05.061.

[4] S. H. Strogatz, Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering. CRC press, 2018.

[5] N. E. El-Meligy, T. O. Diab, A. S. Mohra, A. Y. Hassan, and W. I. El-Sobky, “A Novel Dynamic Mathematical Model Applied in Hash Function Based on DNA Algorithm and Chaotic Maps,” Mathematics, vol. 10, no. 8, p. 1333, Apr. 2022, doi: 10.3390/math10081333.

[6] A. A. Putri Ratna, P. DewiPurnamasari, A. Shaugi, and M. Salman, “Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system,” in 2013 International Conference on QiR, IEEE, Jun. 2013, pp. 99–104. doi: 10.1109/QiR.2013.6632545.

[7] M. Rahman, A. Murmu, P. Kumar, N. R. Moparthi, and S. Namasudra, “A novel compression-based 2D-chaotic sine map for enhancing privacy and security of biometric identification systems,” Journal of Information Security and Applications, vol. 80, p. 103677, Feb. 2024, doi: 10.1016/j.jisa.2023.103677.

[8] Taskeen Taj and Manash Sarkar, “A Survey on Embedding Iris Biometric Watermarking for User Authentication,” Cloud Computing and Data Science, pp. 203–211, Jul. 2023, doi: 10.37256/ccds.4220233051.

[9] S. Datta and S. Namasudra, “Blockchain-Based Smart Contract Model for Securing Healthcare Transactions by Using Consumer Electronics and Mobile-Edge Computing,” IEEE Transactions on Consumer Electronics, vol. 70, no. 1, pp. 4026–4036, Feb. 2024, doi: 10.1109/TCE.2024.3357115.

[10] A. Zellagui, N. Hadj-Said, and A. Ali-Pacha, “A new hash function inspired by sponge construction using chaotic maps,” Journal of Discrete Mathematical Sciences and Cryptography, pp. 1–31, Mar. 2022, doi: 10.1080/09720529.2021.1961900.

[11] W. Zhou, X. Wang, M. Wang, and D. Li, “A new combination chaotic system and its application in a new Bit-level image encryption scheme,” Opt Lasers Eng, vol. 149, p. 106782, Feb. 2022, doi: 10.1016/j.optlaseng.2021.106782.

[12] J. Liu, Y. Wang, Q. Han, and J. Gao, “A Sensitive Image Encryption Algorithm Based on a Higher-Dimensional Chaotic Map and Steganography,” International Journal of Bifurcation and Chaos, vol. 32, no. 01, Jan. 2022, doi: 10.1142/S0218127422500043.

[13] W. Wu, M. Zhang, X. Chen, and R. Jin, “Image encryption based on improved chaos and hash function,” in International Conference on Signal Image Processing and Communication (ICSIPC 2021), S. Chen and W. Qin, Eds., SPIE, Jun. 2021, p. 14. doi: 10.1117/12.2600146.

[14] B. Rahul, K. Kuppusamy, and A. Senthilrajan, "Chaos-based audio encryption algorithm using biometric image and SHA-256 hash algorithm," Multimed Tools Appl, vol. 82, no. 28, pp. 43729–43758, Nov. 2023, doi: 10.1007/s11042-023-15289-x.
[15] A. T. Hashim, "Secure Verifiable Scheme for Biometric System based on Secret Sharing and CSK," International Journal of Computing, pp. 78–84, Mar. 2021, doi: 10.47839/ijc.20.1.2095.
[16] A. Jabbar, A. Hashim, and Q. Al-Doori, "Secured Medical Image Hashing Based on Frequency Domain with Chaotic Map," Engineering and Technology Journal, vol. 39, no. 5A, pp. 711–722, May 2021, doi: 10.30684/etj.v39i5A.1786.
[17] A. K. Jabbar, A. T. Hashim, and Q. F. Hassan, "Medical Image Authentication by Combining Hash Signature and Watermarking Based on Frequency Domains," J Phys Conf Ser, vol. 1963, no. 1, p. 012039, Jul. 2021, doi: 10.1088/1742-6596/1963/1/012039.
[18] "Medical Image Encryption Using Bit Plane Slicing, Dynamic Chaos, and Hash Function," International Journal of Intelligent Engineering and Systems, vol. 15, no. 6, pp. 293–302, Dec. 2022, doi: 10.22266/ijies2022.1231.28.
[19] W. Ahmad, H. Ali, Z. Shah, and S. Azmat, "A new generative adversarial network for medical images super resolution," Sci Rep, vol. 12, no. 1, p. 9533, Jun. 2022, doi: 10.1038/s41598-022-13658-4.
[20] L. Liu and C. Liu, "Theoretical Analysis and Circuit Verification for Fractional-Order Chaotic Behavior in a New Hyperchaotic System," Math ProblEng, vol. 2014, pp. 1–14, 2014, doi: 10.1155/2014/682408.
[21] T. A. Dawood, A. T. Hashim, and A. R. Nasser, "Automatic Skull Stripping of MRI Head Images Based on Adaptive Gamma Transform," Mathematical Modelling of Engineering Problems, vol. 10, no. 1, pp. 304–310, Feb. 2023, doi: 10.18280/mmep.100136.
[22] R. I. Abdelfatah, E. A. Baka, and M. E. Nasr, "Keyed Parallel Hash Algorithm Based on Multiple Chaotic Maps (KPHA-MCM)," IEEE Access, vol. 9, pp. 130399–130409, 2021, doi: 10.1109/ACCESS.2021.3113855.
[23] 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, doi: 10.1109/ACCESS.2019.2896661.
[24] C. Li, G. Luo, K. Qin, and C. Li, "An image encryption scheme based on chaotic tent map," Nonlinear Dyn, vol. 87, no. 1, pp. 127–133, Jan. 2017, doi: 10.1007/s11071-016-3030-8.
[25] Y. Zhou, L. Bao, and C. L. P. Chen, "A new 1D chaotic system for image encryption," Signal Processing, vol. 97, pp. 172–182, Apr. 2014, doi: 10.1016/j.sigpro.2013.10.034.
[26] E. Wang, W. Xu, S. Sastry, S. Liu, and K. Zeng, "Hardware module-based message authentication in intra-vehicle networks," in Proceedings of the 8th International Conference on Cyber-Physical Systems, New York, NY, USA: ACM, Apr. 2017, pp. 207–216. doi: 10.1145/3055004.3055016.
[27] J. Guo, G. Liu, L. Song, and Y. Tu, "Exploring SAT for Cryptanalysis: (Quantum) Collision Attacks Against 6-Round SHA-3," 2022, pp. 645–674. doi: 10.1007/978-3-031-22969-5_22.
[28] A. Kanso, H. Yahyaoui, and M. Almulla, "Keyed hash function based on a chaotic map," Inf Sci (N Y), vol. 186, no. 1, pp. 249–264, Mar. 2012, doi: 10.1016/j.ins.2011.09.008.
[29] 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, doi: 10.1016/j.cnsns.2012.06.019.
[30] H. Liu, A. Kadir, C. Ma, and C. Xu, "Constructing Keyed Hash Algorithm Using Enhanced Chaotic Map with Varying Parameter," Math ProblEng, vol. 2020, pp. 1–10, Jul. 2020, doi: 10.1155/2020/4071721.
[31] J. Sen 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, doi: 10.1007/s10586-018-2870-z.
[32] M. Ahmad, S. Khurana, S. Singh, and H. D. AlSharari, "A Simple Secure Hash Function Scheme Using Multiple Chaotic Maps," 3D Research, vol. 8, no. 2, p. 13, Jun. 2017, doi: 10.1007/s13319-017-0123-1.
[33] M. AsgariChenaghlu, S. Jamali, and N. NikzadKhasmakhi, "A novel keyed parallel hashing scheme based on a new chaotic system," Chaos Solitons Fractals, vol. 87, pp. 216–225, Jun. 2016, doi: 10.1016/j.chaos.2016.04.007.
[34] 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, doi: 10.1007/s11071-016-2652-1.