# Comparative Assessment of Hash Functions in Securing Encrypted Images

**Ghayth Al-Asad**
Ministry of Local Administration, Amman, Jordan
ghayth.thelover@gmail.com

**Mohammed Al-Husainy**
Irbid National University, Irbid, Jordan
drAlHusainy@gmail.com (corresponding author)

**Mohammad Bani-Hani**
Jadara Research Center, Jadara University, Irbid, Jordan
mbbanihani@gmail.com

**Ala'eddin Al-Zu'bi**
Ministry of Education, Exams Department, Irbid, Jordan
alaaalzoubi1985@yahoo.com

**Sara Albatienh**
Irbid National University, Irbid, Jordan
batainehsara96@yahoo.com

**Hazem Abuoliem**
Irbid National University, Irbid, Jordan
hazemabuoliem@gmail.com

*Received: 9 September 2024 | Revised: 3 October 2024 and 13 October 2024 | Accepted: 16 October 2024*

*Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.8961*

## ABSTRACT

**Different encryption methods have been developed to securely transmit confidential images over the Internet and combat the increasing cybercrime. Many of these methods use hash functions to enhance encryption strength. Due to the lack of a comprehensive evaluation of how different hash functions affect image encryption, this study presents a comparative analysis of the performance of various hash functions as encryption keys and analyzes their security, speed, and efficiency. The source image is first processed as a series of bytes. The bytes are divided into byte vectors, each with a length that matches the length of the hash value of a specified hash function. An XOR operation is performed between the hash value bytes and the associated byte vector. The bytes are reordered in each vector according to the ascending order of the associated hash value. Several metrics, such as Normalized Mean Absolute Error (NMAE), Peak Signal to Noise Ratio (PSNR), entropy, key size, and hash time, were used to evaluate the performance of different hash functions in image encryption. The results showed a clear variation in using various hash functions in terms of security, speed, and efficiency. With NMAE>72%, PSNR<6.62 dB, and Entropy>7.999 bpp, the use of the SHA family and MD5 is recommended in applications that need to achieve a high level of distortion in encrypted images. To resist brute-force attacks on the key, Blake2b, SHA512, and Whirlpool are the best choices with a key size of 512 bits. The Tiger is the fastest hash function, requiring the least average time of 0.372 seconds to complete the encryption process, making it the best choice for real-time applications. These findings help to choose the appropriate hash function in developing cryptographic techniques for a particular area.**

***Keywords-cryptography; secure image; hash functions; XOR operation; cybersecurity***

## I. INTRODUCTION

People are increasingly using image sharing on the Internet to express their emotions, and many of them contain confidential and private information. As these images may be exposed to significant risks during transmission, encryption is used to protect the image content from unauthorized users [1]. The technique of converting a plain image into a coded form that can only be decoded by the intended recipient is known as image encryption [2]. Image encryption is a crucial strategy to safeguard people's privacy and information security [3]. Researchers are paying close attention to image encryption as a result of the rapid advancement of the Internet and multimedia technologies, which have led to an increase in the production and transmission of digital images containing a wide range of information over networks. As a result, the security of image information has become a critical concern [4].

Hash functions are one of many encryption techniques in which each secret image is converted to a completely noisy image using a secure hash function [5]. Mathematical methods known as hash functions are one-way functions that convert an arbitrary-sized input message into a fixed-size hash or message digest. A hash function is called a one-way function that is simple to calculate for each input, but determining its inverse value is an extremely challenging task. Furthermore, it is difficult to find two different data with the same hash value. Common hash algorithms include Secure Hash Algorithm 1 (SHA-1), SHA-2, Message Digest 5 (MD5) [5], etc. Each of them produces a specific hash value of a specific length of bits. The resulting hash value and its length depend primarily on the operations used in the hash function algorithm [6, 7]. Due to the lack of sufficient analytical information on the features of different hash functions and their performance when used in image encryption, this study presents a comprehensive performance assessment that investigates the effectiveness of using different popular hash functions in image encryption through deep analysis based on standard security metrics. To encrypt an image, this study starts by choosing a particular hash function and dividing the original image into vectors. The length of the vector is equal to the number of bytes in the hash value. Then, using an initial key, the necessary hash values are generated for all image vectors. To create the encrypted image, an XOR operation is performed between the bytes in each image vector and the associated bytes in the hash value. The bytes in each image vector are then reordered in ascending order according to the ascending order of the bytes in the relevant hash value.

In [8], a novel approach was proposed for high-security and high-sensitivity satellite image encryption using a hash key-based symmetric cryptography algorithm that combines generated keys with chaotic mapping parameters. This method utilized the SHA-1 to SAH-512 hash function to create an encryption key, determining the chaos mapping parameters using algebraic functions connected to the primary key, and employing encryption blocks to retrieve the encrypted image. This method used hash functions and chaos functions for encryption. Chaos-based algorithms focus on dispersion and confusion for data protection. The results of the proposed method were assessed using histogram analysis, entropy information, and key sensitivity analysis, demonstrating benefits over alternative techniques for encrypting satellite images.

In [9], an image encryption technique was proposed, built on the Galois field and SHA512. Through generating a random key matrix for encryption using a SHA512-built keystream generator, this technique concealed pixel information by multiplying the plain image and key matrix by transforming the data into the Galois Field GF($8^2$). Then the encrypted image's pixel locations were disordered using scrambling matrices. This method achieved exceptional security and sensitivity in protecting digital images. In [10], a dynamic DNA color image encryption strategy was described, which used two rounds of permutation-diffusion employing chaotic systems, dynamic DNA coding, DNA sequencing operations, and conditional shifting. This strategy was based on the SHA-512 encryption standard. The SHA-512 method and natural DNA sequences produce initial values for chaotic systems and intermittent parameters, ensuring sensitivity to variations in input images. The three steps of the encryption process were DNA encryption, diffusion, and decoding. The calculation of Hamming distances for chaotic systems and DNA matrices is necessary to update initial values and complete diffusion procedures. The security analysis and stimulation findings showed that this technique was resilient to plaintext attacks and could withstand a variety of attacks.

In [11], a hybrid paradigm of DNA computing, chaotic systems, and hash functions was proposed to offer great efficiency and security. DNA-level permutation and diffusion were used in the encryption process, along with logistic maps and pseudo-random vectors for encryption and decoding. The experimental results showed that this technique had good encryption effects, practical speed, and robustness to known attacks. To increase the sensitivity and security of the encryption technique, the algorithm generated starting values for chaotic systems and control parameters using a mix of SHA-256 and MD5 hash functions. In [12], a hybrid cryptosystem was proposed that combined symmetric and asymmetric encryption techniques to protect and authenticate numerical images. The SHA-256 algorithm created the first secret key based on the original image and the owner's signature. Safe key exchange and authentication were made possible by RSA asymmetric encryption, which encrypted both the original key and the owner's signature. The symmetric approach relied on the Lorenz chaotic system to produce high-quality keys and leveraged Pseudo-Random Number Generator- Counter mode (PRNG-CTR)-based XOR confusion for full-image encryption. The evaluation results showed that this technique offered minimal computational complexity, improved security, and great performance. This architecture is appropriate for real-time applications since it is adaptable, allows for key extension, and is simple to parallelize. A comparative analysis showed that this approach was very successful for image authentication and encryption in many fields, outperforming recent works.

In [13], an image encryption system was proposed that combined chaotic sequence sorting with a 3-D map to increase security. Along with dynamic DNA encryption and decoding rules, it presented a 3-D DNA level permutation and substitution strategy to shuffle and modify elements of the plain image matrix. This approach generated initial values using a hyperchaotic system and hashes using SHA-256. Experiments showed that a variety of images with high-security ratios could be successfully encrypted and decrypted, proving the efficacy of the encryption method.

One notable aspect of this study is the thorough comparison of a wide variety of hash functions, including both more recent functions like Blake2b and more well-known ones like SHA-512, as prior research focused on a smaller range of hash functions or performance elements. Additionally, this study emphasizes the use of these hash functions in a unique way to ensure image integrity, particularly during transmission, storage, and retrieval. Furthermore, the analysis includes a wide range of measures, including resistance to different cryptographic attacks, key size, and key space. This provides a more comprehensive performance rating compared to other works. The most important differences between the comparative study in this work and the related works are:

- Most previous works used a specific hash function based on the length of its hash value or the common use of a hash function without considering security, time, and efficiency. This study takes into account all these factors to make an accurate evaluation of several hash functions.
- Most related works used a hash function by combining it with other algorithms, making it difficult to evaluate the performance of the hash function alone. This study uses each hash function independently to encrypt images and presents a performance evaluation of each.
- There is no comprehensive evaluation of the performance of different hash functions in image encryption to facilitate the selection of the appropriate one for specific applications. This study helps researchers choose the proper hash function based on an evaluation scale of a particular factor of the hash function.

## II. MATERIALS AND METHODS

This study used several types of hash functions with different lengths of hash values to encrypt images. The source image to be encrypted is treated as a sequence of bytes. First, the source image is divided into a set of byte vectors. The length of each vector is equal to the number of bytes of the hash value of the hash function used. Second, the desired hash function is used to generate the necessary hash values for each vector of bytes of the source image. Table I shows the hash function types and lengths examined used in this study.

An XOR operation was performed between the bytes in the generated hash value and the corresponding vector. This process helps achieve the desired confusion effect in the bytes of each vector. Furthermore, to produce the desired diffusion effect in the bytes of each vector, the bytes within each vector are rearranged in ascending order based on the ascending order of bytes in the corresponding hash value. The main objectives for selecting the types of operations to encrypt images and the methods for implementing them are focused on:

- Achieving simplicity in implementation
- Achieving fairness in implementing the various hash functions
- Minimizing the hash speed (encryption time)
- Causing the necessary confusion and diffusion effects in the encrypted images
- Ensuring the effect of three factors of the hash value of different hash functions on the quality of the resulting encrypted image. These factors are:
    - The number of bytes in the hash value
    - The bytes values in the hash value
    - The order of bytes in the hash value.

TABLE I. HASH FUNCTIONS AND THEIR LENGTH

| Hash function | Length of hash value (bits) |
|---|---|
| Blake2b | 512 |
| Blake2s | 256 |
| MD5 | 128 |
| RipeMD160 | 160 |
| RipeMD320 | 320 |
| SHA1 | 160 |
| SHA256 | 256 |
| SHA384 | 384 |
| SHA512 | 512 |
| Tiger | 192 |
| Whirlpool | 512 |

The algorithm below illustrates the proposed scheme.

```
Algorithm 1: The proposed scheme
Input:
  Source Image (SImage): An image that
  will be encrypted using different hash
  functions
  Initial key (IKey): The initial set of
  bytes used in the first call of the hash
  function to create the keys required for
  SImage encryption.
Output:
  Encrypted Image (EImage): The resulting
  image after encrypting SImage using the
  hash function
Encryption Stage:
1: Represent SImage as a sequence of
   bytes. For example:
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *SImage* | 13 | F3 | 92 | 0A | 78 | 22 | 00 | CA | 5E | 65 |

```
2: Split SImage into vectors of size N,
   where N is the length of the hash value
   (in bytes). This ensures that when the
   hash function is changed, the size of
   the data block will be changed. For
   example: if N=5:
```

```
              0  1  2  3  4      0  1  2  3  4
     SImage |13|F3|92|0A|78|   |22|00|CA|5E|65|
                   V1                V2
3: Call the chosen hash function and use
   IKey as input data for the function to
   produce a new hash value H. Using a
   different IKey for each image will
   produce a completely different series
   of hash values used to encrypt the data
   blocks. For example:
                 0    1    2    3    4
             H | 13 | F3 | 92 | 0A | 78 |
4: For each vector of SImage
   a: Call the hash function and use the
      last hash value produced as input
      data for the function to produce a
      new hash value H.
                  0  1  2  3  4
                H|E7|69|A4|9B|14|
   b: XOR every byte in the new hash
      value with the corresponding bytes
      in the current block of SImage. The
      XOR operation is used here to create
      the desired confusion effect in the
      encrypted image. For example:
                  0  1  2  3  4
              H |E7|69|A4|9B|14|
             XOR
                  0  1  2  3  4
             V1 |13|F3|92|0A|78|
              ↓
             ----------------------
                  0  1  2  3  4
             V1 |F4|9A|36|91|6C|

   c: Arrange the bytes in H in ascending
      order and arrange the bytes in the
      corresponding vector in SImage in
      the same order as the bytes in H.
      The rearrangement process is used
      here to create the desired diffusion
      effect in the encrypted image.
        0    1    2    3    4          0    1    2    3    4
    H | E7 | 69 | A4 | 9B | 14 | →  H | 14 | 69 | 9B | A4 | E7 |

        0    1    2    3    4          0    1    2    3    4
   V1 | F4 | 9A | 36 | 91 | 6C | → V1 | 6C | 9A | 91 | 36 | F4 |

5: Construct the encrypted image EImage
   from the produced encrypted vectors of
   Simage.
```

## III. RESULTS AND DISCUSSION

This study compares the performance of different hash functions for image encryption based on several metrics: Normalized Mean Absolute Error (NMAE), Peak Signal-to-Noise Ratio (PSNR), entropy, key size and key space, and hash speed (encryption time). The metrics provided useful information on the stability, accuracy, and efficacy of the hash functions in Table I, helping to assess their suitability for various image processing and analysis applications.

Various standard color and grayscale images of different sizes were obtained from an image database [14] and used in the experiments. These images are commonly used in most image-processing applications in general and in cryptography applications in particular. Examples of these images include Baboon (512×512), Lake (512×512), Peppers (256×256), Lena (256×256), Petra (300×400), and Minaret (450×570).

### *A. Normalized Mean Absolute Error (NMAE)*

The NMAE metric is designed to enable the comparison of datasets with different scales of mean absolute error. An efficient encryption method achieves the highest possible NMAE value. NMAE is calculated using [15, 16]:

$$NMAE = \frac{\sum_{k=1}^{N} |S_k - E_k|}{\sum_{k=1}^{N} S_k} \times 100 \quad (1)$$

where $S$ and $E$ represent the source and encrypted data, respectively. Table II lists the NMAE values for different hash functions applied to the images, showing the degree of distortion produced in the encrypted images using each hash function, with larger average NMAE values indicating greater distortion. SHA512 had an average NMAE of 72.723%, making it the best performer among all the hash functions provided. This result is considered the best because it has the highest NMAE and generates the most significant image distortion compared to other hash functions.

TABLE II. NMAE VALUES FOR HASH FUNCTIONS.

| Hash function | NMAE (%) | | | | | | Avg |
|---|---|---|---|---|---|---|---|
| | Baboon | Lake | Peppers | Lena | Petra | Minaret | |
| Blake2b | 58.218 | 74.24 | 75.617 | 48.739 | 56.591 | 59.894 | 62.217 |
| Blake2s | 59.847 | 74.61 | 76.759 | 50.587 | 57.901 | 61.364 | 63.511 |
| MD5 | 68.603 | 80.803 | 83.331 | 62.861 | 68.009 | 72.439 | 72.674 |
| RipeMD160 | 60.956 | 75.272 | 77.691 | 50.567 | 59.297 | 62.799 | 64.430 |
| RipeMD320 | 59.188 | 75.501 | 76.103 | 50.462 | 57.277 | 60.755 | 63.214 |
| SHA1 | 68.739 | 80.887 | 83.276 | 62.817 | 68.141 | 72.354 | 72.702 |
| SHA256 | 68.783 | 80.82 | 83.129 | 62.934 | 68.079 | 72.422 | 72.695 |
| SHA384 | 68.707 | 80.882 | 83.321 | 62.836 | 68.165 | 72.412 | 72.721 |
| SHA512 | 68.763 | 80.796 | 83.218 | 62.97 | 68.264 | 72.327 | 72.723 |
| Tiger | 60.376 | 75.054 | 77.121 | 51.313 | 58.674 | 62.091 | 64.105 |
| Whirlpool | 58.497 | 75.104 | 76.094 | 49.85 | 56.547 | 65.643 | 63.623 |

### *B. Peak Signal-To-Noise Ratio (PSNR)*

PSNR is a metric used to evaluate the quality of the signal or data, expressed in dB. In encrypted circumstances, PSNR is frequently employed to assess the degree of distortion that arises during the encrypted process, as given by [17]:

$$PSNR = 10.\log_{10}\left(\frac{Max_S^2}{NMAE}\right) \quad (2)$$

where $S$ represents the source data and $Max_S$ is the maximum possible byte value of $S$. Recording a low PSNR value means that the encrypted method achieves high distortion in the encoded data. Table III shows the recorded PSNR during the experiments using the proposed method.

TABLE III. PSNR VALUES FOR HASH FUNCTIONS.

| Hash function | PSNR (dB) | | | | | | Avg |
|---|---|---|---|---|---|---|---|
| | Baboon | Lake | Peppers | Lena | Petra | Minaret | |
| Blake2b | 8.684 | 6.369 | 7.519 | 8.878 | 9.274 | 7.132 | 7.976 |
| Blake2s | 8.456 | 6.266 | 7.394 | 8.571 | 9.072 | 6.911 | 7.778 |
| MD5 | 7.32 | 5.526 | 6.795 | 6.634 | 7.825 | 5.573 | 6.612 |
| RipeMD160 | 8.276 | 6.119 | 7.321 | 8.376 | 8.878 | 6.7 | 7.612 |
| RipeMD320 | 8.557 | 6.315 | 7.477 | 8.587 | 9.167 | 7 | 7.851 |
| SHA1 | 7.305 | 5.525 | 6.8 | 6.644 | 7.816 | 5.58 | 6.612 |
| SHA256 | 7.301 | 5.528 | 6.799 | 6.632 | 7.83 | 5.571 | 6.610 |
| SHA384 | 7.304 | 5.528 | 6.799 | 6.641 | 7.818 | 5.572 | 6.610 |
| SHA512 | 7.303 | 5.534 | 6.806 | 6.627 | 7.811 | 5.583 | 6.611 |
| Tiger | 8.376 | 6.183 | 7.377 | 8.352 | 8.965 | 6.8 | 7.676 |
| Whirlpool | 8.659 | 6.309 | 7.488 | 8.777 | 9.276 | 6.322 | 7.805 |

The lowest average PSNR value for SHA256 and SHA384 means that a high distortion was achieved in the generated encrypted images, while Blake2b achieved a lower distortion.

## *C. Entropy*

Entropy is a measure of unpredictability or randomness in a data collection. Equation (3) yields an entropy number that quantifies the amount of information included in a message or dataset. The higher the entropy, the information is more uncertain or unpredictable [17, 18].

$$Entropy = \sum_{i=1}^{n} P_i . log_2(P_i) \qquad (3)$$

where $n$ is the number of different data values and $P_i$ is the occurrence probability of the data value. Table IV shows the level of randomness or unpredictability added by each hash function to the encrypted images. Higher entropy indicates greater unpredictability and randomness, which can affect the encrypted images' security and capacity to withstand decoding. As a result, comparing the entropy levels of various hash functions sheds light on how well they preserve the security and integrity of encrypted images. SHA256 and SHA384 provide the best average entropy result of 7.99949.

TABLE IV. ENTROPY VALUES FOR HASH FUNCTIONS.

| Hash function | Entropy | | | | | | Avg |
|---|---|---|---|---|---|---|---|
| | Baboon | Lake | Peppers | Lena | Petra | Minaret | |
| Blake2b | 7.28809 | 7.61571 | 7.75430 | 7.56580 | 7.79362 | 7.74919 | 7.62779 |
| Blake2s | 7.43590 | 7.71233 | 7.80908 | 7.65379 | 7.83838 | 7.80612 | 7.70927 |
| MD5 | 7.99976 | 7.99976 | 7.99901 | 7.99885 | 7.99950 | 7.99975 | 7.99944 |
| RipeMD160 | 7.58309 | 7.80270 | 7.85961 | 7.73594 | 7.88163 | 7.85967 | 7.78711 |
| RipeMD320 | 7.37989 | 7.67612 | 7.78778 | 7.61850 | 7.82034 | 7.78405 | 7.67778 |
| SHA1 | 7.99980 | 7.99975 | 7.99899 | 7.99913 | 7.99947 | 7.99977 | 7.99948 |
| SHA256 | 7.99978 | 7.99977 | 7.99906 | 7.99912 | 7.99944 | 7.99977 | 7.99949 |
| SHA384 | 7.99978 | 7.99974 | 7.99906 | 7.99902 | 7.99961 | 7.99975 | 7.99949 |
| SHA512 | 7.99974 | 7.99976 | 7.99906 | 7.99899 | 7.99948 | 7.99977 | 7.99947 |
| Tiger | 7.52483 | 7.76613 | 7.83903 | 7.70067 | 7.86394 | 7.83730 | 7.75532 |
| Whirlpool | 7.28741 | 7.61619 | 7.75384 | 7.56771 | 7.79459 | 7.93070 | 7.65841 |

## *D. Key Size and Key Space*

It is well known that the choice of key significantly affects the strength of cryptography techniques. A large key indicates a strong defense against attacks, especially brute force attacks [15, 16]. Table V provides a comparison of key sizes and key spaces for different hash functions, with Blake2b, SHA512, and Whirlpool, with key sizes of 512 bits having the largest key spaces of $2^{512}$ and indicating high security. MD5, with a key size of 128 bits and a key space of $2^{128}$, is considered less secure. Hash functions including SHA256, SHA384, Blake2s, Tiger, and RipeMD320 provide a medium level of security with key sizes ranging from 192 to 384 bits.

TABLE V. KEY SIZE AND SPACE FOR HASH FUNCTIONS

| Hash function | Key size (bits) | Key space |
|---|---|---|
| Blake2b | 512 | $2^{512}$ |
| Blake2s | 256 | $2^{256}$ |
| MD5 | 128 | $2^{128}$ |
| RipeMD160 | 160 | $2^{160}$ |
| RipeMD320 | 320 | $2^{320}$ |
| SHA1 | 160 | $2^{160}$ |
| SHA256 | 256 | $2^{256}$ |
| SHA384 | 384 | $2^{384}$ |
| SHA512 | 512 | $2^{512}$ |
| Tiger | 192 | $2^{192}$ |
| Whirlpool | 512 | $2^{512}$ |

## *E. Hash Speed (Encryption Time)*

Encryption time or hash speed refers to the amount of time it takes for an encryption or hashing technique to process a specific volume of data. This measure is crucial to assess the efficiency of cryptographic algorithms, especially in scenarios where speed is critical, such as real-time communications, high-frequency trading, or systems with limited processing capacity [19, 20]. Table VI shows the speed of different hash functions applied to certain images. Comparing hash speeds yields performance insights. Tiger had the lowest average hash speed, due to its effectiveness in processing images quickly, and MD5 needed more time to complete the hashing process.

TABLE VI. HASH SPEED OF HASH FUNCTIONS

| Hash function | Hash speed (s) | | | | | | Avg |
|---|---|---|---|---|---|---|---|
| | Baboon | Lake | Peppers | Lena | Petra | Minaret | |
| Blake2b | 0.97 | 0.961 | 0.228 | 0.163 | 0.285 | 0.654 | 0.544 |
| Blake2s | 0.929 | 0.827 | 0.145 | 0.142 | 0.268 | 0.558 | 0.478 |
| MD5 | 1.341 | 1.315 | 0.218 | 0.221 | 0.397 | 0.843 | 0.723 |
| RipeMD160 | 0.531 | 1.51 | 0.132 | 0.135 | 0.234 | 0.516 | 0.510 |
| RipeMD320 | 0.666 | 0.668 | 0.14 | 0.141 | 0.258 | 0.555 | 0.405 |
| SHA1 | 0.796 | 0.917 | 0.21 | 0.216 | 0.367 | 0.772 | 0.546 |
| SHA256 | 0.63 | 0.751 | 0.167 | 0.166 | 0.273 | 0.585 | 0.429 |
| SHA384 | 0.872 | 0.897 | 0.196 | 0.19 | 0.346 | 0.706 | 0.535 |
| SHA512 | 0.794 | 0.893 | 0.195 | 0.202 | 0.336 | 0.718 | 0.523 |
| Tiger | 0.564 | 0.656 | 0.13 | 0.127 | 0.244 | 0.513 | 0.372 |
| Whirlpool | 0.79 | 0.948 | 0.187 | 0.193 | 0.339 | 0.729 | 0.531 |

## IV. CONCLUSIONS

Due to the lack of comparative studies that evaluate the performance differences between different hash functions in image encryption, it is necessary to conduct such studies to make the right decision in choosing the appropriate hash function for each field. This work investigated how different hash functions perform in the context of image encryption, providing valuable insight into how well they work to protect image confidentiality. This study provides a comprehensive understanding of the performance of different hash functions in encryption using several standard evaluation metrics, such as NMAE, PSNR, entropy, key size, key space, and hash speed. The novelty of this study and its results contribute to enriching researchers' knowledge of different hash functions and their effectiveness in image encryption. Future directions involve extending this study considering the use of other hash functions, the use of more evaluation metrics, and the encryption of other types of data.

## ACKNOWLEDGMENT

The authors would like to thank Irbid National University, Irbid, Jordan.

## REFERENCES

[1] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, "Symmetric Encryption Algorithms: Review and Evaluation study," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 12, no. 2, 2020.

[2] U. Diaa, "A Deep Learning Model to Inspect Image Forgery on SURF Keypoints of SLIC Segmented Regions," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 12549–12555, Feb. 2024, https://doi.org/10.48084/etasr.6622.

[3] H. Gao and T. Gao, "Double verifiable image encryption based on chaos and reversible watermarking algorithm," *Multimedia Tools and Applications*, vol. 78, no. 6, pp. 7267–7288, Mar. 2019, https://doi.org/10.1007/s11042-018-6461-z.

[4] H. M. Ghadirli, A. Nodehi, and R. Enayatifar, "An overview of encryption algorithms in color images," *Signal Processing*, vol. 164, pp. 163–185, Nov. 2019, https://doi.org/10.1016/j.sigpro.2019.06.010.

[5] A. K. Chattopadhyay, S. Saha, A. Nag, and J. P. Singh, "A verifiable multi-secret image sharing scheme based on DNA encryption," *Multimedia Tools and Applications*, Apr. 2024, https://doi.org/10.1007/s11042-024-19033-x.

[6] I. Bashir, F. Ahmed, J. Ahmad, W. Boulila, and N. Alharbi, "A Secure and Robust Image Hashing Scheme Using Gaussian Pyramids," *Entropy*, vol. 21, no. 11, Nov. 2019, Art. no. 1132, https://doi.org/10.3390/e21111132.

[7] S. Dhall and K. Yadav, "Cryptanalysis of substitution-permutation network based image encryption schemes: a systematic review," *Nonlinear Dynamics*, vol. 112, no. 17, pp. 14719–14744, Sep. 2024, https://doi.org/10.1007/s11071-024-09816-0.

[8] M. Sedighi, S. K. Mahmoudi, and A. S. Amini, "Proposing a new method for encrypting satellite images based ON hash function and chaos parameters," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-4-W18, pp. 949–953, Oct. 2019, https://doi.org/10.5194/isprs-archives-XLII-4-W18-949-2019.

[9] H. Lan and R. Ye, "A Novel Image Encryption Algorithm Based on Secure Hash Function and Galois Field," in *2019 2nd International Conference on Safety Produce Informatization (IICSPI)*, Chongqing, China, Nov. 2019, pp. 98–102, https://doi.org/10.1109/IICSPI48186.2019.9095933.

[10] S. Zhou, P. He, and N. Kasabov, "A Dynamic DNA Color Image Encryption Method Based on SHA-512," *Entropy*, vol. 22, no. 10, Oct. 2020, Art. no. 1091, https://doi.org/10.3390/e22101091.

[11] E. Z. Zefreh, "An image encryption scheme based on a hybrid model of DNA computing, chaotic systems and hash functions," *Multimedia Tools and Applications*, vol. 79, no. 33, pp. 24993–25022, Sep. 2020, https://doi.org/10.1007/s11042-020-09111-1.

[12] M. Gafsi, M. A. Hajjaji, J. Malek, and A. Mtibaa, "Efficient Encryption System for Numerical Image Safe Transmission," *Journal of Electrical and Computer Engineering*, vol. 2020, no. 1, 2020, Art. no. 8937676, https://doi.org/10.1155/2020/8937676.

[13] C. Zhu, Z. Gan, Y. Lu, and X. Chai, "An image encryption algorithm based on 3-D DNA level permutation and substitution scheme," *Multimedia Tools and Applications*, vol. 79, no. 11, pp. 7227–7258, Mar. 2020, https://doi.org/10.1007/s11042-019-08226-4.

[14] "Support, Image Databases - ImageProcessingPlace.Com." [Online]. Available: https://www.imageprocessingplace.com/downloads_V3/root_downloads/image_databases/standard_test_images.zip.

[15] A. Saini and R. Sehrawat, "Enhancing Data Security through Machine Learning-based Key Generation and Encryption," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14148–14154, Jun. 2024, https://doi.org/10.48084/etasr.7181.

[16] M. Abbas Fadhil Al-Husainy, H. A. A. Al-Sewadi, and B. Al-Shargabi, "Image Encryption using a Binary Search Tree Structure-Based Key," *International Journal of Computing and Digital Systems*, vol. 12, no. 1, pp. 823–836, Sep. 2022, https://doi.org/10.12785/ijcds/120168.

[17] E. Aruna and A. Sahayadhas, "Blockchain-Inspired Lightweight Dynamic Encryption Schemes for a Secure Health Care Information Exchange System," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15050–15055, Aug. 2024, https://doi.org/10.48084/etasr.7390.

[18] M. A. F. Al-Husainy, H. A. A. Al-Sewadi, and A. M. Sayed, "Using the 3D Protein Structure as Key to Encrypt Images," *Journal of Information and Organizational Sciences*, vol. 47, no. 2, Dec. 2023, https://doi.org/10.31341/jios.47.2.5.

[19] L. Yang, S. Bi, M. G. R. Faes, M. Broggi, and M. Beer, "Bayesian inversion for imprecise probabilistic models using a novel entropy-based uncertainty quantification metric," *Mechanical Systems and Signal Processing*, vol. 162, Jan. 2022, Art. no. 107954, https://doi.org/10.1016/j.ymssp.2021.107954.

[20] M. Alawida, A. Samsudin, N. Alajarmeh, J. S. Teh, M. Ahmad, and W. H. Alshoura, "A Novel Hash Function Based on a Chaotic Sponge and DNA Sequence," *IEEE Access*, vol. 9, pp. 17882–17897, 2021, https://doi.org/10.1109/ACCESS.2021.3049881.