

Securing Image Messages Using Secure Hash Algorithm 3, Chaos Scheme, and DNA Encoding

Amer Sharif*, Dian Rachmawati, Wilbert

Dept. of Computer Science, Faculty of Computer Science and Information Technology,
Universitas Sumatera Utara, Medan, Indonesia

Abstract—Security is an essential aspect to consider during data transmission, especially images. Threats that may occur during image transmission include images being stolen by third parties. One way to secure images is through encryption-decryption processes using cryptographic algorithms. One of the algorithms developed for image security involves combining a chaos scheme, DNA encoding, and hashing. Chaos scheme refers to a system sensitive to initial conditions, resulting in behavior that is difficult to predict or appears random. DNA encoding is the process of converting bits into a DNA sequence. Hashing is a mathematical function which takes variable inputs and converts them into a binary sequence with fixed length. In this research, security enhancement is achieved by replacing the hashing algorithm with Secure Hash Algorithm (SHA) 3 Keccak. This study successfully implemented cryptographic algorithms into a website that can simulate image encryption-decryption processes in about 15 seconds per process. The effectiveness of the algorithm used has also been tested in abstracting images through Mean Squared Error (MSE) and Peak-Signal-to-Noise-Ratio (PSNR) evaluations. Obtained MSE values of 0 and PSNR values of infinity indicated that the original images and decrypted images are identical.

Keywords—Image encryption; image decryption; chaos scheme; DNA encoding; secure hash algorithm 3 Keccak

I. INTRODUCTION

The advancement of digital technology has led to an increased demand for data, especially images. Despite being a routine activity, the transmission of images between parties involves significant security risks, such as data theft by third parties. Therefore, it is crucial to develop methods for securing images during transmission.

Image encryption is an effective method for securing images. Encryption involves converting plaintext into ciphertext, making the data unreadable without a decryption key. This process ensures that only the intended recipient, who possesses the key, can convert the ciphertext back into readable plaintext, thus securing the image data during transmission.

This research explores the use of symmetric key cryptography, combining a chaos scheme and DNA encoding, alongside the Secure Hash Algorithm (SHA) 3 hashing algorithm, specifically Keccak. Commonly used hashing algorithms like SHA-1 and SHA-2 have known vulnerabilities, prompting the National Institute of Standards and Technology (NIST) to seek a more secure alternative,

resulting in the adoption of the Keccak algorithm as the SHA-3 standard.

The chaos scheme generates semi-random numbers without repeating periods and is highly sensitive to initial parameter changes. This sensitivity ensures significant differences in output with minor changes in input, enhancing encryption robustness. Despite producing semi-random values, the chaos system is deterministic, meaning it can reproduce the same values with the same initial parameters, which is crucial for reliable encryption and decryption processes.

DNA encoding utilizes the properties of DNA sequences, where the bases A (Adenine) pair with T (Thymine) and C (Cytosine) pair with G (Guanine). By representing these DNA bases in a binary format, plaintext can be encoded into a secure format. This method, combined with the chaos scheme, has shown promise in previous studies, such as those by Belazi et al. [1], Wang and Su [15], and Zhong et al. [16].

Previous research indicates that the SHA-3 algorithm offers enhanced security compared to SHA-2 [10]. By integrating the chaos scheme, DNA encoding, and the SHA-3 algorithm, this study aims to develop a more secure image encryption method, potentially achieving higher security levels than existing techniques. The system developed may later be considered as an alternative means of image security in communication applications.

This study is organized next by enlisting previous studies in image encryption utilizing chaos and DNA algorithms and studies involving hash functions, followed by a literature review of the principal theoretical concepts. Following that is the problem analysis and system design, continued with implementation and system testing. Discussion and final conclusion with some considerations for future work are provided in the end.

II. PREVIOUS STUDIES

Several studies related to this approach have been conducted. Belazi et al. [1] obtained a cryptographic algorithm which met security criteria such as a large key space, resistance to statistical attacks, and a linear run time, enabling efficient execution. Alrubiae et al. [2] leveraged double-dynamic DNA sequence encryption and a chaotic 2D logistic map to produce a novel image encryption algorithm which resisted common attacks. Hussain et al. [4] developed substitution boxes by the use of chaotic logistic maps in linear fractional transformation. Arif et al. [5] used S-boxes, a hash

*Corresponding Author, Email: - amer.sharif@usu.ac.id

function, and chaos scheme to produce a secure cryptographic algorithm for images. Suryadi et al. [8] utilized a transposition scheme based on a chaos function, namely Arnold's cat map function, as a form of transposition resulting in a large key space with very low sensitivity. Munir [11] developed a simple image encryption system using a pseudo-one-time pad built upon a chaos function. Wang and Su [15] utilized the Piecewise Linear Chaotic Map (PWLCM) and Logistic Map to generate all parameters the encryption algorithm needed, combined with DNA encoding. Zhong et al. [16] combined chaos theory and a 2D Chebyshev-Sine map to achieve image encryption resistant to differential and selective plaintext attacks. Pittalia [10] compared the effectiveness of hash functions in cryptography, while Patil and Karule [7] specifically discussed the design and implementation of the Keccak hash function for cryptography.

III. LITERATURE REVIEW

A. Cryptography

Cryptography is a method for securing data using code algorithms, hashes, and signatures. It can protect data at rest, in transit, or in use. For example, cryptography can secure data stored on a hard drive, data being transferred between parties through electronic communication, or data being processed during calculations [12].

Cryptography has four main objectives: confidentiality, ensuring that information is accessible only to authorized parties; integrity, ensuring that information is not manipulated; authentication, confirming the authenticity of information and the identity of users; and non-repudiation, preventing users from denying their previous actions [14].

Cryptography generally is divided into two categories: symmetric cryptography, which utilize the same key for encryption, i.e. converting the readable plaintext/image into unreadable ciphertext/image, and decryption, i.e. converting the unreadable ciphertext/image back into the readable plaintext/image. On the other hand, asymmetric cryptography utilizes a set of different keys (called public and private keys) for encryption and decryption [14]. This research employs symmetric cryptography.

B. Secure Hash Algorithm

Secure Hash Algorithm (SHA) or often known as *one-way functions*, are mathematical functions which accept variable-length input and convert it into a binary string with a fixed length of 128 upto 512 bits. SHA are employed to ensure data integrity and authenticity [6] [14].

There are two criteria which serve as a standard for the security of an SHA algorithm:

- 1) *Preimage resistance*: Preimage resistance means for an arbitrary hash value, it is very difficult to obtain the input which produced that particular hash value.
- 2) *Collision resistance*: Collision resistance means it is very difficult to obtain two inputs which will produce an identical hash value.

Keccak is a cryptographic hash function and the basis for the SHA-3 (Secure Hash Algorithm 3) standard, which was

selected by the National Institute of Standards and Technology (NIST) in 2012. Unlike its predecessors, SHA-1 and SHA-2, Keccak employs a unique structure and process, enhancing its security and efficiency [7] [10].

Keccak operates through a series of absorbing and squeezing phases, as indicated in Fig. 1 from [7], where the input data is absorbed into a state array and then permuted. This process involves transforming blocks of 1152, 1088, 832, or 576 bits to generate hash values of 224, 256, 384, and 512 bits. These permutations ensure that even the slightest change in the input data results in a significantly different hash output, making Keccak highly resistant to collisions and attacks [10].

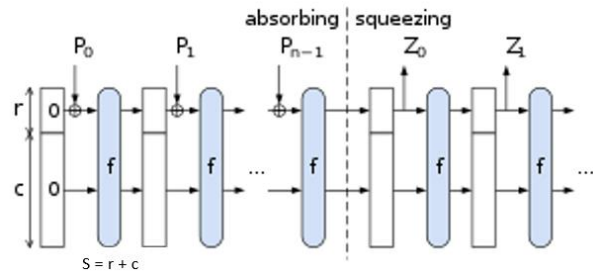


Fig. 1. Absorbing and squeezing phases in Keccak from [7].

C. Chaotic Scheme

Chaos in many contexts often refers to phenomena in which the systems are highly sensitive to initial conditions, resulting in difficult-to-predict behaviour or seemingly random behaviour [11].

In cryptography, a map refers to mathematical functions which link each element from a set to another corresponding element in another set. Maps are utilized to convert data from one form to another.

Chaotic maps are maps which exhibit chaotic properties or uncertainties by producing hard-to-predict numbers. Chaotic maps serve to make cryptographic systems more resistant to attacks due to the difficulty in describing or predicting the chaotic behavioral pattern [5].

D. Logistic-Chebyshev Map

The Logistic-Chebyshev map is a combination of two one-dimensional chaotic maps: the logistic map and the Chebyshev map. By merging these two maps, the Logistic-Chebyshev map is expected to produce more complex and varied results compared to a single chaotic map [3]. This enhanced complexity can be beneficial in various applications, such as in the development of security systems or data scrambling, where chaotic properties are leveraged to generate random number sequences with the desired level of complexity. The following is the Logistic-Chebyshev equation from [3]:

$$X_{n+1} = \left(\alpha X_n (1 - X_n) + \frac{(4-\alpha)}{4} \cos(b \times \cos^{-1}(X_n)) \right) \bmod 1 \quad (1)$$

$$X_n \in (0, 1)$$

$$\alpha \in (0, 4]$$

$$b \in \mathbb{N}$$

E. Sine-Chebyshev Map

The Sine-Chebyshev map is a one-dimensional chaotic system that combines the sine map and the Chebyshev map. The sine map, with its simple mathematical structure, and the Chebyshev map, a polynomial map, both exhibit chaotic behavior and high sensitivity to initial conditions. By integrating these two maps, the Sine-Chebyshev map enhances the unpredictability and complexity of the generated sequences, making it particularly useful for applications in cryptography, random number generation, and other fields requiring high levels of complexity and sensitivity. This synergy creates a more robust and versatile chaotic system. The following is the Sine-Chebyshev equation from [16]:

$$X_{n+1} = \left(\alpha \sin(\pi X_n) + \frac{(4-\alpha)}{4} \cos(b \times \cos^{-1}(X_n)) \right) \bmod 1$$
$$(2)$$
$$X_n \in (0, 1)$$
$$\alpha \in (0, 4]$$
$$b \in \mathbb{N}$$

F. DNA Encoding

Deoxyribonucleic Acid (DNA) carries the genetic information of all living organisms, characterized by its four constituent bases: A (Adenine), C (Cytosine), G (Guanine), and T (Thymine). These bases pair uniquely within DNA sequences, where A always pairs with T and C always pairs with G. These pairings form the fundamental structure of DNA, essential for encoding genetic instructions in living cells. In computational applications, these bases can be represented in binary form according to specific encoding rules, facilitating the transformation of plaintext data into a binary format known as DNA encoding. This process is crucial in fields such as cryptography and data security, where the inherent properties

of DNA bases can be leveraged to encode information securely and efficiently [13].

DNA decoding, conversely, involves reversing the encoding process to retrieve the original plaintext from its binary representation using the DNA base pairs. This decoding mechanism ensures that the encoded information can be accurately reconstructed into its original form, preserving data integrity and facilitating its practical application in various scientific and technological domains. Overall, DNA encoding and decoding techniques offer a promising approach for secure data transmission and storage, harnessing the natural properties of DNA for innovative computational solutions in information security. Table I indicates sample DNA encoding for two-bit pairs.

TABLE I DNA ENCODING

Rule Code	1	2	3	4	5	6	7	8
A	00	00	11	11	10	01	10	01
T	11	11	00	00	01	10	01	10
C	10	01	10	01	00	00	11	11
G	01	10	01	10	11	11	00	00

G. S-box

S-box, or Substitution Box, is a component in symmetric cryptography which substitutes values from input data blocks with corresponding outputs based on a certain rule or substitution table. The main purpose of the S-box is to increase the algorithm's security level by complicating statistical analysis and cryptanalysis attacks.

This research uses S-boxes from the research [4], namely S-box 3, S-box 4, S-box 5, dan S-box 6. S-box 3 is given in Table II below as an example.

TABLE II S-Box 3

105	197	63	16	136	75	70	74	220	96	100	125	167	98	108	148
242	5	254	93	13	78	253	45	144	12	35	196	226	179	230	44
123	204	15	41	176	0	165	64	11	217	163	59	56	62	134	140
235	250	49	77	131	252	239	157	244	214	129	248	177	113	10	152
103	231	51	130	139	32	73	7	219	33	200	156	146	192	232	191
233	202	187	23	241	246	216	158	31	161	17	94	53	9	206	117
249	89	127	24	195	46	43	162	80	6	48	209	54	119	149	65
92	102	212	135	36	203	28	126	27	132	210	172	85	145	22	224
34	188	50	25	67	225	88	182	84	81	69	240	228	104	143	72
8	86	60	3	171	205	238	55	61	245	79	83	222	58	147	142
121	37	124	4	87	183	1	199	243	166	180	118	114	91	29	184
169	189	110	101	47	170	251	40	186	18	97	229	155	174	236	153
247	150	106	237	168	193	66	2	112	215	14	120	201	21	213	38
95	52	198	57	109	208	178	255	218	211	30	227	190	76	90	82
173	99	42	39	185	194	159	111	138	137	116	181	141	154	160	175
115	68	26	122	20	221	164	71	107	207	234	128	133	151	223	19

H. Mean Squared Error (MSE)

Mean Squared Error is a parameter used to compare the level of pixel difference between two images. A high MSE value indicates that the level of pixel difference between the images is relatively large, meaning the difference between the two images is quite significant. MSE may be obtained from the following equation from [9]:

$$MSE = \frac{1}{XY} \sum_{m=1}^m \sum_{n=1}^n (O(m,n) - R(m,n))^2 \quad (3)$$

I. Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is a parameter used to compare two images. The PSNR value ranges from $[0, \infty)$. The higher the PSNR value, the more similar the compared images are. Conversely, the lower the PSNR value, the more different the compared images are [9]. PSNR is calculated using the following equation from [9]:

$$PSNR = 10 \times \log_{10} \frac{(2^n - 1)^2}{MSE} \quad (4)$$

n = bit in the image

MSE = Mean Squared Error value

IV. ANALYSIS AND DESIGN

A. Problem Analysis

The primary issue to be addressed in this research is the development of a system which may secure images by means of symmetric cryptography leveraging the Keccak hash function, chaos scheme and DNA encoding.

Functional requirements analysis:

- 1) System will secure image using the methods mentioned
- 2) System will secure image with the maximum dimension of 512x512 pixels
- 3) System will accept input from local file and database
- 4) System will accept input of secret key to encrypt and decrypt the image
- 5) System will perform registration and authentication of users
- 6) System will be able to send encrypted images to the receiver

Non-functional requirements analysis:

- 1) System will provide a user-interface which is easy to understand for novice users
- 2) System will validate every user input during encryption and decryption and provide error message for invalid input.

B. Flowchart

The flowcharts for the activities in this research are described in the following figures.

Fig. 2 describes the proposed algorithm implementation, indicating the important stages in the encryption process starting with the plain image and the secret key: key generation, image decomposition and block permutation, substitution, DNA encoding, complementing, DNA decoding, bitwise operation.

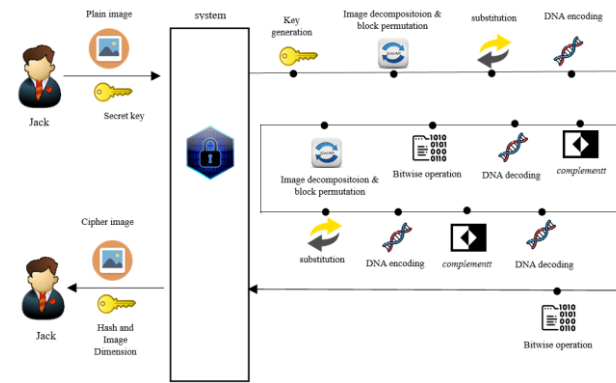


Fig. 2. Encryption process.

Fig. 3 describes the proposed algorithm for the decryption process, which essentially has the same stages as the encryption process but only in reverse order.

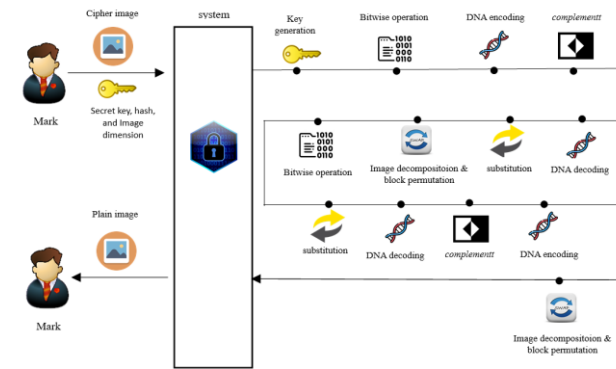


Fig. 3. Decryption process.

C. Proposed Algorithm

1) Key generation

- a) Calculate the hash value of plain image h .
- b) Divide h into 32 bytes.
- c) Calculate the following, from [4]:

$$X'_0 = \frac{1}{3} \left(X_0 + 2 \times \frac{1}{256} \times \text{bin2dec}(h_9 \oplus \dots \oplus h_{16}) \right) \quad (5)$$

$$\alpha'_0 = \frac{1}{3} \left(\alpha_0 + 2 \times \frac{1}{64} \times \text{bin2dec}(h_1 \oplus \dots \oplus h_8) \right) \quad (6)$$

$$X'_1 = \frac{1}{3} \left(X_1 + 2 \times \frac{1}{256} \times \text{bin2dec}(h_1 \oplus \dots \oplus h_{32}) \right) \quad (7)$$

$$\alpha'_1 = \frac{1}{3} \left(\alpha_1 + 2 \times \frac{1}{64} \times (\sum_{i=1}^{32} h_1(i) \bmod 256) \right) \quad (8)$$

d) Calculate Logistic-Chebyshev Map by iteration > 500 times to hide the transient effect. Then, calculate it $M \times N$ times and put it in matrix. M and N is the image dimension.

e) Map the matrix with 1 if the value > 0.5 and 0 if the value ≤ 0.5 .

f) Calculate Sine-Chebyshev Map by iteration > 500 times to hide the transient effect. Then, calculate it $M \times N$ times and put it in matrix.

g) Replace each matrix element by calculating

$$Z_1(i, j) = (Y \times 10^{15}) \bmod 256 \quad (9)$$

2) Image processing and decomposition

a) Divide image into 64 blocks.

b) Permute these blocks by the following calculation using prime p and $h = 1..64$

$$b_i = (p \times h) \bmod 64 \quad (10)$$

c) Reconstruct image based on the calculation.

3) Substitution

a) Get S-box-3, S-box-4, S-box-5, S-box-6 from [4]

b) Divide the image into 4 parts, substitute each part with each S-box, respectively.

4) DNA encoding

a) Select a DNA rule from the DNA RULE in Table III below.

b) Encode the image using the selected DNA rule, and the image size will be $M \times 4N$.

5) Complement

Using the Logistic Chebyshev matrix, replace the image pixel with the complement if the logistic Chebyshev matrix in the corresponding index is 1.

6) DNA decoding

a) Select a DNA rule from the DNA RULE below.

b) Decode the image using the DNA RULE, restoring the image size back to $M \times N$.

7) Bitwise operation

Using the Sine Chebyshev matrix, XOR the image with the matrix.

8) Calculate MSE and PSNR

a) The MSE parameter is used to determine if the image was successfully encrypted.

b) The PSNR parameter is used to determine if the image was successfully decrypted.

TABLE III DNA RULE

Rule								
Base	1	2	3	4	5	6	7	8
A	00	00	01	01	10	10	11	11
T	11	11	10	10	01	01	00	00
C	01	10	00	11	00	11	01	10
G	10	01	11	00	11	00	10	01

V. IMPLEMENTATION AND TESTING

A. Implementation

In this research, the application was built as a website using Python and Laravel PHP Framework. User data and image data were stored in a MySQL database. The software used for development is Visual Studio Code. This application has five pages: Login, Register, Home, Encryption, Decryption. These pages are indicated in Fig. 4 to Fig. 8. Encrypted images may be stored into the database for access by another user, as indicated in Fig. 7.

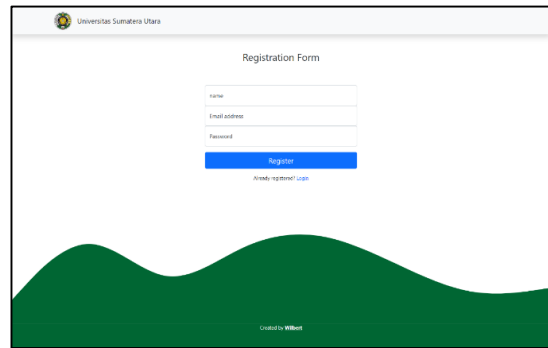


Fig. 4. Registration page.

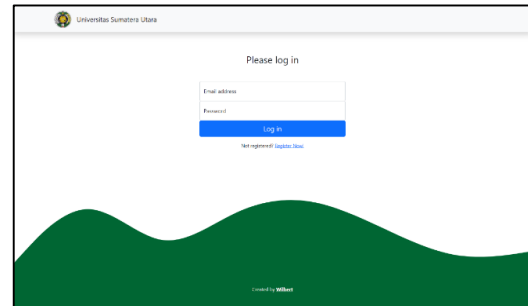


Fig. 5. Login page.

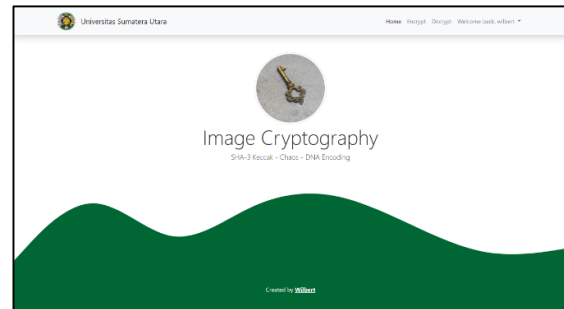


Fig. 6. Home page.

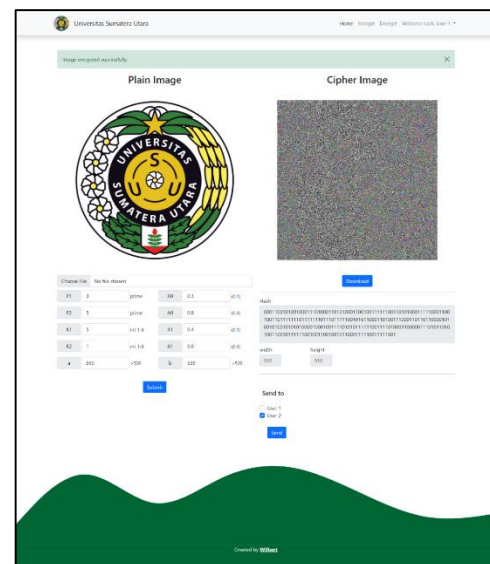


Fig. 7. Encryption page with a sample image.

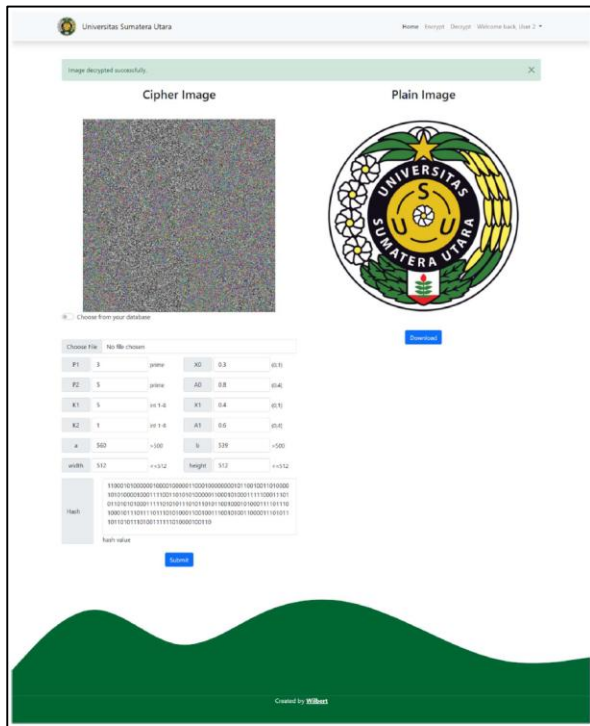


Fig. 8. Decryption page with a sample image.

B. System Testing

Testing consisted of the following steps:

- 1) Encrypt 5 test images of various sizes with .png extension and different secret keys.
- 2) Measure the MSE between the original/plain image and the obtained scrambled/cipher image.
- 3) Decrypt the cipher image back to plain image.
- 4) Measure the PSNR between the restored image to the original image

Testing was carried out on a laptop with the following specification:

- AMD Ryzen 5 5600H CPU
- GeForce RTX 3050 4GB graphics card
- 16 GB RAM

Required for encryption are the secret keys, which include two prime numbers for image decomposition and block permutation, two random integers between 1 to 8 for DNA encoding and decoding, two random fractions and two random real numbers between 0 and 4 for the chaotic maps. The number of iteration (> 500) for the Logistic-Chebyshev and Sine-Chebyshev mappings are also required. All these values are input manually into the system. The hash value of the selected plain image is calculated initially by the system as well as the dimensions of the image. Parameter for the first test image are indicated in Table IV.

TABLE IV SECRET KEYS FOR FIRST TEST IMAGE

Secret keys	Value	Remarks	Secret keys	Value	Remarks
P1	17	prime	X1	0.7	real (0, 1)
P2	23	prime	A1	3.4	real (0, 4]
K1	8	int 1 – 8	a	765	> 500
K2	2	int 1 – 8	b	594	> 500
X0	0.5	real (0, 1)	Width	400	pixels
A0	2.5	real (0, 4]	Height	400	pixels
Hash	10011001010010111011011101101011010111110011 11100111010001101001101000011011000110101110 10100000010000001110110101110111110100110010 11100001011000101000110010010011000001001001 01111111100011110100101110011001110100011010 100001111001101110001111100110000001				

Plain image, encrypted image and decrypted image for the first test image are shown in Table V. Visual inspection showed that the encrypted image is totally different from the plain image, while the decrypted image is virtually the same as the plain image.

TABLE V FIRST TEST IMAGE

Plain image	Encrypted Image	Decrypted Image

Processing time for the encryption and decryption processes is indicated in Table VI, which averaged around 15 seconds.

TABLE VI PROCESSING TIME OF FIRST TEST IMAGE

Encryption (seconds)	Decryption (seconds)
15.332939624786377	15.12040400505066




The MSE and PSNR values of the first test image are indicated in Table VII and these values supported the visual inspection of Table V. MSE between plain image and encrypted image is relatively high (105.85) with a low PSNR (27 dB) indicating a high difference between the two images. MSE of 0 and PSNR of infinity between decrypted image and plain image indicates the two images are the same.

TABLE VII MSE AND PSNR OF FIRST TEST IMAGE

	Plain image and cipher image	Plain image and decrypted image
MSE	105.85469563802083	0
PSNR (dB)	27.883702330949063	∞

Plain image, encrypted image and decrypted image for the second test image are shown in Table VIII. As the case with the first test image, visual inspection showed that the encrypted image is totally different from the plain image, while the decrypted image is virtually the same as the plain image.

TABLE VIII SECOND TEST IMAGE

Plain image	Encrypted Image	Decrypted Image
		

Secret keys used for the encryption and decryption processes are indicated in Table IX. Just as the first test image, secret keys are two prime numbers for image decomposition and block permutation, two integers between 1 and 8 for DNA encoding and decoding, two random fractions and two random real numbers between 0 and 4 for the chaotic maps. The number of iteration (> 500) for the Logistic-Chebyshev and Sine-Chebyshev mappings are also required.

TABLE IX SECRET KEYS FOR SECOND TEST IMAGE

Secret keys	Value	Remarks	Secret keys	Value	Remarks
P1	17	prime	X1	0.12	real (0, 1)
P2	23	prime	A1	2.24	real (0, 4]
K1	2	int 1 – 8	a	599	> 500
K2	8	int 1 – 8	b	652	> 500
X0	0.55	real (0, 1)	Width	383	pixels
A0	3.572	real (0, 4]	Height	512	pixels
Hash	10101100110111100011001011100010011011001001 011010100010110000010111100111111011101101010 00001100110000100010011001110000101101110110 0101001001101011111001011101011010011111110 1110111101100101011101101011011000100110100 100100011010011011010100111010000111				

MSE and PSNR values of the second test image are indicated in Table X. Here the high MSE value of 150.35 and low PSNR value of 27.90 dB between the plain image and encrypted image showed high difference between the two, while low MSE of 0 and high PSNR value of infinity between the decrypted image and plain image, indicated that both images are similar, which again supported the visual inspection result in Table VII.

Processing times for the second test image are indicated in Table XI. Here the processing times are slightly below 15 seconds (14.88 seconds for encryption and 14.97 for decryption).

Table XII shows the result of the third test image. As the case with the second test image, visual inspection showed that the encrypted image is totally different from the plain image, while the decrypted image is virtually the same as the plain image.

Secret keys used for the encryption and decryption of third test image are indicated in Table XIII. Just as the second test image, secret keys consisted of two prime numbers for image decomposition and block permutation, two integers between 1 and 8 for DNA encoding and decoding, two random fractions and two random real numbers between 0 and 4 for the chaotic maps. The number of iteration (> 500) for the Logistic-Chebyshev and Sine-Chebyshev mappings are also required.

TABLE X MSE AND PSNR OF SECOND TEST IMAGE

	Plain image and cipher image	Plain image and decrypted image
MSE	105.35418192545573	0
PSNR (dB)	27.904285817106654	∞

TABLE XI PROCESSING TIME OF SECOND TEST IMAGE

Encryption (seconds)	Decryption (seconds)
14.887351989746094	14.975344181060791

TABLE XII THIRD TEST IMAGE




Plain image	Encrypted Image	Decrypted Image
		

TABLE XIII SECRET KEYS FOR THIRD TEST IMAGE

Secret keys	Value	Remarks	Secret keys	Value	Remarks
P1	97	prime	X1	0.758	real (0, 1)
P2	11	prime	A1	1.657	real (0, 4]
K1	5	int 1 – 8	a	1124	> 500
K2	7	int 1 – 8	b	752	> 500
X0	0.5	real (0, 1)	Width	512	pixels
A0	3.21	real (0, 4]	Height	342	pixels
Hash	11101001111000011001110110111101001110011000 1010111111100010100001000001000001000011011 01001000110100101000000000001100010000100011 01000010001011111111011100101110101110001 0010101010101101110110110110011100100011011 1100100110000001110110010111011110				

MSE and PSNR values of the third test image are indicated in Table XIV. Here the high MSE value of 150.53 and low PSNR value of 27.89 dB between the plain image and encrypted image showed the high difference between the two, while low MSE of 0 and high PSNR value of infinity between decrypted image and plain image, indicated that both images are similar, which again supported the visual inspection result in Table XII.

Processing times for the third test image are indicated in Table XV. Here the encryption processing time was 14.83 seconds, and decryption was 15.19 seconds.

Table XVI shows the result of the fourth test image. As the case with the previous test images, Table XVII contains the secret keys, while Table XVIII shows the MSE and PSNR values between the plain image and cipher image, and the plain image and the decrypted image. Table XIX indicate the processing times for encryption and decryption.

Table XX shows the result of the fifth test image. As the case with the previous test images, Table XXI contains the secret keys, while Table XXII shows the MSE and PSNR values between the plain image and cipher image, and the plain image and the decrypted image. Table XXIII indicate the processing times for encryption and decryption.

TABLE XIV MSE AND PSNR OF THIRD TEST IMAGE

	Plain image and cipher image	Plain image and decrypted image
MSE	105.53628412882487	0
PSNR (dB)	27.89678562013698	∞

TABLE XV PROCESSING TIME OF THIRD TEST IMAGE

Encryption (seconds)	Decryption (seconds)
14.82814908027649	15.187212944030762

TABLE XVI FOURTH TEST IMAGE




Plain image	Encrypted Image	Decrypted Image
		

TABLE XVII SECRET KEYS FOR FOURTH TEST IMAGE

Secret keys	Value	Remarks	Secret keys	Value	Remarks
P1	17	prime	X1	0.01	real (0, 1)
P2	37	prime	A1	2.2	real (0, 4]
K1	2	int 1 – 8	a	582	> 500
K2	1	int 1 – 8	b	725	> 500
X0	0.99	real (0, 1)	Width	512	pixels
A0	4	real (0, 4]	Height	394	pixels
Hash	11011000011101100010001110011111010110111100 00100010011110110110111111011100100010101000 00010010010000111110101110100100111011110110 0110001110011101011000011000000001010010110 00110000010000010110000110011001001001110110 000000111111010000001101110110011010				

TABLE XVIII MSE AND PSNR OF FOURTH TEST IMAGE

	Plain image and cipher image	Plain image and decrypted image
MSE	105.44661966959636	0
PSNR (dB)	27.900476988632636	∞

TABLE XIX PROCESSING TIME OF FOURTH TEST IMAGE

Encryption (seconds)	Decryption (seconds)
15.099057912826538	15.182414054870605

TABLE XX FIFTH TEST IMAGE




Plain image	Encrypted Image	Decrypted Image
		

TABLE XXI SECRET KEYS FOR FIFTH TEST IMAGE

Secret keys	Value	Remarks	Secret keys	Value	Remarks
P1	11	prime	X1	0.01	real (0, 1)
P2	37	prime	A1	4	real (0, 4]
K1	1	int 1 – 8	a	501	> 500
K2	8	int 1 – 8	b	501	> 500
X0	0.99	real (0, 1)	Width	235	pixels
A0	4	real (0, 4]	Height	305	pixels
Hash	000000100100000011001101000010011111100100101 001010000000101010110011011010001101001101100 100100101111011000011001100001101011000010110 111101001010100110000010010110010100011000000 000000100101011111011101101101101010010110100 0000010011011111001011110010000				

TABLE XXII MSE AND PSNR OF FIFTH TEST IMAGE

	Plain image and cipher image	Plain image and decrypted image
MSE	105.21969985961914	0
PSNR (dB)	27.90983302234912	∞

TABLE XXIII PROCESSING TIME OF FIFTH TEST IMAGE

Encryption (seconds)	Decryption (seconds)
14.929418087005615	17.71390175819397

VI. DISCUSSION

Several additional observations maybe obtained from the implementation of the cryptosystem, as follows:

- User and image data were stored in the system using a MySQL Database
- All input images sizes were padded to achieve a dimension of 512x512 pixel.
- High MSE values (around 150) and low PSNR values (around 27 dB) between the plain images and the encrypted images indicated the plain images were successfully converted into abstract images.

- MSE values of 0 and high PSNR values (infinity) between the plain images and the decrypted images indicated that the encrypted images were successfully restored to images identical to the original plain images.
- Processing times for encryption and decryption were around 15 seconds on the testing laptop.

VI. CONCLUSION

Based on the results obtained from the above system testing, it may be concluded that:

- 1) The algorithms used were capable of performing image encryption and decryption effectively, building on previous studies mentioned at the beginning of this study. This system has successfully integrated the SHA-3 Keccak hash algorithm, Chaos Scheme, and DNA Encoding.
- 2) The system was implemented into a website using Python, PHP, and a MySQL database.
- 3) The evaluation of MSE and PSNR values showed that the algorithm successfully abstracted the images and restored them to their original form.
- 4) The processing time required for each encryption and decryption operation is still quite high for the specified laptop used in system testing.
- 5) This study has contributed to the field of cryptography an alternate approach to image security using the aforementioned algorithms.

VII. FUTURE WORK

Limitations of the present implementation include a symmetric cryptographic approach (which may introduce a key distribution problem), while also limiting images to be encrypted having a maximum dimension of 512x512 pixels. This system is also run on a stand-alone PC-based platform. Secret keys required for the encryption and decryption were input manually, resulting in a not very large key space. Several approaches for future work maybe suggested: firstly, secret keys maybe generated by the system on-demand, therefore ascertaining better randomness and larger key space; secondly, to address key distribution problem faced by symmetric cryptosystem by combining the present algorithms with a public key algorithm such as Extended RSA to achieve a more secure hybrid cryptosystem; thirdly, improving so that images to be encrypted are not limited to a dimension of 512x512 pixels; fourth, to port the cyptosystem to a mobile platform such as Android or iOS, and lastly integrating the present system into a communication system such as an e-mail or instant messaging application.

REFERENCES

- [1] A. Belazi, M. Talha, S. Kharbech and W. Xiang, "Novel Medical Image Encryption Scheme Based on Chaos and DNA Encoding," in *IEEE Access*, vol. 7, pp. 36667-36681, 2019, doi: 10.1109/ACCESS.2019.2906292.
- [2] Alrubaie, A.H., Khodher, M.A.A. & Abdulameer, A.T. "Image encryption based on 2DNA encoding and chaotic 2D logistic map". *J. Eng. Appl. Sci.* **70**, 60 (2023). <https://doi.org/10.1186/s44147-023-00228-2>.
- [3] B. Abd-El-Atty, M. Amin, A. Abd-El-Latif, H. Ugail and I. Mehmood, "An Efficient Cryptosystem based on the Logistic-Chebyshev Map," 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Island of Ulkulhas, Maldives, 2019, pp. 1-6, doi: 10.1109/SKIMA47702.2019.8982535.
- [4] I. Hussain, T. Shah, M. A. Gonda *et al.*, "An efficient approach for the construction of LFT S-boxes using chaotic logistic map", *Nonlinear Dyn* **71**, 133–140 (2013). <https://doi.org/10.1007/s11071-012-0646-1>.
- [5] J. Arif *et al.*, "A Novel Chaotic Permutation-Substitution Image Encryption Scheme Based on Logistic Map and Random Substitution," in *IEEE Access*, vol. 10, pp. 12966-12982, 2022, doi: 10.1109/ACCESS.2022.3146792.
- [6] J.-Philippe Aumasson, *Serious Cryptography*, San Francisco: No Starch Press, Inc., 2018.
- [7] M. A. Patil and P. T. Karule, "Design and implementation of keccak hash function for cryptography," 2015 International Conference on Communications and Signal Processing (ICCS), Melmaruvathur, India, 2015, pp. 0875-0878, doi: 10.1109/ICCS.2015.7322620.
- [8] M. T. Suryadi, Z. Rustam, W. Widhianto, "Implementasi Algoritma Enkripsi Citra Digital Menggunakan Skema Transposisi Berbasis Fungsi Chaos", *Prosiding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT)*, vol. 8, pp. 376-380, Depok, 2014.
- [9] N. Mahendiran, C. Deepa, "A Comprehensive Analysis on Image Encryption and Compression Techniques with the Assessment of Performance Evaluation Metrics", *SN COMPUT. SCI.* **2**, 29 (2021). <https://doi.org/10.1007/s42979-020-00397-4>.
- [10] P. P. Pittalia, "A Comparative Study of Hash Algorithms in Cryptography". *International Journal of Computer Science and Information Technology*, Vol. 8, Issue. 6, pp.147-152, 2019.
- [11] R. Munir, "Algoritma Enkripsi Citra dengan Pseudo One-Time Pad yang Menggunakan Sistem Chaos", *Konferensi Nasional Informatika (KNIF)*, pp. 12-16, 2011.
- [12] S. Kromodimoeljo, *Teori dan Aplikasi Kriptografi*, SPK IT Consulting, 2009.
- [13] T. A. S. Yusri and D. Rudhistiar, "Enkripsi Citra Digital Berbasis Kombinasi Arnold Cat Map Termodifikasi dan DNA Encoding", *Mnemonic*, vol. 5, no. 2, pp. 173-177, Aug. 2022.
- [14] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th Edition, Pearson, 2016.
- [15] X. Wang, Y. Su, "Image encryption based on compressed sensing and DNA encoding", in *Signal Processing: Image Communication*, Volume 95, 2021, 116246, ISSN 0923-5965, <https://doi.org/10.1016/j.image.2021.116246>.
- [16] Y. Zhong, H. Liu, R. Lan, T. Wang, X. Sun and X. Luo, "2D Chebyshev-Sine Map for Image Encryption," 2018 7th International Conference on Digital Home (ICDH), Guilin, China, 2018, pp. 1-8, doi: 10.1109/ICDH.2018.000.