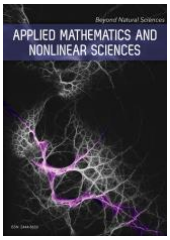
sciendo Applied Mathematics and Nonlinear Sciences, 8(2) (2023) 1643-1652



Applied Mathematics and Nonlinear Sciences

https://www.sciendo.com

Data Information Security Algorithm Based on Chaos and Hash Function

Hui Liu†

School of Internet, Henan Mechanical and Electrical Vocational College, Zhengzhou, Henan, 451191, China

Submission Info

Communicated by Juan Luis García Guirao
Received April 16, 2022
Accepted October 10, 2022
Available online April 19, 2023

Abstract

Chaotic systems are characterized by unidirectional, diffusive and initial value sensitivity of hash. Academia can use it to optimize algorithms for mathematical and computer encryption keys. This paper focuses on a hash function mixed chaotic system with a key. Then the state value and chaotic mapping relationship of the chaotic system are modified, and hash conclusions are obtained. Then the optimal design of messy technology with key hash is introduced briefly. A chaotic dynamic model with improved dynamic parameters is proposed to prevent chaos from affecting the speed and security of the algorithm. The results show that this method can effectively resist the attack of forging and peer keys. Moreover, the computation required by this algorithm is almost negligible.

Keywords: Hash function; Chaos; Dynamic parameters; Key security algorithm; Hash function with key
AMS 2020 codes: 34H10

†Corresponding author.
Email address: wagkathleen@126.com

ISSN 2444-8656
https://doi.org/10.2478/amns.2023.1.00142

1 Introduction

The traditional encryption algorithm and chaotic key scheme have better hybrid performance, and the critical mechanism is similar. The proposed method has similar chaotic properties and parameter sensitivity to chaotic systems. Conventional encryption methods have the same propagation property as chaotic systems. For example, its trajectory distribution and initial sensitivity are similar to chaotic systems. With the increasing trust in recent years, hashing methods have been used more and more. It is instrumental in analyzing the chaotic mapping of the hash function. Hash function with a key (KHF) is also known as Message Code (MAC). Its function is to carry on the authenticity of information and the identification of information sources. In the early 1970s, information verification emerged and gradually formed a set of norms [1]. In this way, the construction technology and technology of KH have significantly been developed. After repeated iterations, the DES algorithm block encryption theory can get the corresponding hash. The effectiveness and security of packet encryption depend on the underlying encryption algorithm. The only drawback is that it takes a lot of computational effort, so finding a fast and reliable encryption algorithm isn't easy. The critical hash function implemented by a chaotic technique is analyzed in detail in this paper. Firstly, the pseudo-code or peer key recovery attack is carried out. Then this paper discusses the causes and countermeasures of network attacks from the perspectives of system structure and internal compression function [2]. Finally, chaos and hash problems in chaotic systems are summarized. At last, the chaotic parameters of this method are modified, and an anti-existence fraud method with a key hash is given.

2 A type of attack on the key hash function

In this paper, packet P is divided into s blocks. The MAC value of $P = P_0 P_1 \cdots P_{s-1}$ obtained by the compression iteration method of s is denoted by G. The initial key for an authorized consumer is R. $|P_i|$ is the length of P_i bits. P_i represents a bit-string join operation. ^ is an XOR operation.

2.1 Chaotic neural network with KHF

Algorithm $|P_i| = 256, |R| = 128, |G| = 128.$

1) R is divided into four 32-bit $R = (r_1, r_2, r_3, r_4)$, $R_0 = (r_1 / 2^{32}, r_2 / 2^{32}, r_3 / 2^{32}, r_4 / 2^{32})$.

2) $i = 0, \cdots, s-1$ calculate $\eta_i = CNN(R_i, P_i); R_{i+1} = \eta_i$.

CNN is a compressed function. Firstly, four decimal points R_i are used to iterate the single-direction coupled mirror image grid in four directions [3]. Each iteration 30 times to get a system state value. There are ten of them. The first eight 4-dimensional states are assigned to U_{2i}. The last two are assigned to Θ_i and E_i. U_{2i}^l, S_i^l and E_i^l represent their l row.

P_i is divided into two bytes. $\left|\varepsilon_i^{j,l}\right| = 8,\ j = 0, \cdots, 7;\ l = 0,1,2,3,\ i$ is the i piece of information.

$$P_i = (\varepsilon_{i,}^{\ 00} // \mathrm{L}\ // \varepsilon_i^{0,3} // \varepsilon_i^{1,0} // \mathrm{L}\ // \varepsilon_i^{1,3} // \mathrm{L}\ // \varepsilon_i^{7,3})$$

$$U_i^j = z^\tau(U_1 \varepsilon_i^j, 1/3), U_i = (U_i^0, \mathrm{L}\ , U_i^7)$$

$$c_i^l = z^\tau(\mathrm{mod}(U_{2i}^l U_i + \Theta_i^l, 1), E_i^l), \eta_i = (c_i^0, \mathrm{L}\ , c_i^3)$$

z is a piecewise linear chaotic map. τ is the number of repeats. $U_1 = (1/2^8, 1/2^{16}, 1/2^{24}, 1/2^{32})$.

$G = G(c_{n-1})$. The G function converts the decimal value of the four 32-bit c_{n-1}vectors to a 128-bit hash value.

Hypothesis 1. The above method has the problem of selective information fraud.

Proof. Suppose the attacker wants to fabricate MAC code for message $P^{(1)} = P_0//P_1$. The attacker sends $P^{(2)} = P_0$ to the legal user of the corresponding MAC code $G^{(2)}$. The attacker obtains R_1 through the inverse G operation [4]. Then attacker $CNN(R_1, P_1) = \eta_1$ performs the operation of packet $P^{(1)}$ MAC code $G^{(1)} = G(\eta_1)$.

Message $G^{(1)}$ is pre-selected by the attacker, that is, the attack that chooses the message. The computational complexity is about 1 KHF of operations.

2.2 KHF application in double chaotic system

$|P_i| = N, |G| = N$ is an algorithm. $N = 128j,\ j = 1, 2, \cdots$.

1) $R = (\beta_1(0), \beta_2(0)), 0 < \beta_1(0), \beta_2(0) < 1; G_0$ is the pre-agreed vector.

2) $i = 0, \cdots, s-1$ calculate $G_{i+1} = DDC(R, P_i, G_i)$.

   DCC is $\boldsymbol{R}$ strong compression function: the sequence $\boldsymbol{r} = (\boldsymbol{r}(\boldsymbol{1}), \boldsymbol{r}(\boldsymbol{2}), \cdots, \boldsymbol{r}(\boldsymbol{N}))$ of the first A generated $\boldsymbol{N}$ by interference at the initial iteration point. $\boldsymbol{r}(\boldsymbol{i}) = \boldsymbol{h}(\boldsymbol{Z}(\boldsymbol{\beta_1}(\boldsymbol{i}), \boldsymbol{m_1}), \boldsymbol{Z}(\boldsymbol{\beta_2}(\boldsymbol{i}), \boldsymbol{m_2}), \boldsymbol{m_2}), \boldsymbol{i} = \boldsymbol{1}, \cdots, \boldsymbol{N}, \boldsymbol{Z}$ is a segment by segment nonlinear chaotic graph. $\boldsymbol{m_1}, \boldsymbol{m_2}, \boldsymbol{m_3}$ is the three interference $\boldsymbol{m}$ sequences. $\boldsymbol{h}$ outputs 0 or 1 as a comparison of the first two parameters [5]. If the two parameters are equal, then the corresponding bit of $\boldsymbol{m_3}$ is output.

   $\boldsymbol{G_{i+1}} = \boldsymbol{P_i} \oplus \boldsymbol{G_i} \oplus \boldsymbol{r}$. $\oplus$ is an XOR flag.

3) $\boldsymbol{G} = \boldsymbol{G_s}$

Proposition 2. The above method has an equivalent key recovery.

Proof. The attacker sends messages $P^{(1)} = P_0$ and $P^{(2)} = P_0||P_1$ separately to a legitimate user. Get the corresponding values for $G^{(1)}$ and $G^{(2)}$ of the MAC. Since $G^{(1)} = P_0 \oplus G_0 \oplus r$ and $G^{(2)} = P_1 \oplus G_1 \oplus r = P_1 \oplus (P_0 \oplus G_0 \oplus r) \oplus r = P_1 \oplus G^{(1)} \oplus r$, the attacker can tell that $r = P_1 \oplus G^{(1)} \oplus G^{(2)}$ and $G_0 = P_0 \oplus G^{(1)} \oplus r, (r, G_0)$ are not the original key R. However, the attacker can use $G_{i+1} = P_i \oplus G_i \oplus r$iteration method to calculate the MAC value of any piece of information. (r, G_0) is an equivalent key. The computational complexity of this algorithm is about 1 KHF operation.

2.3 Chaotic dynamic KHF based on S-Box

How it works: $|P_i| = 4 \times 8n, |G| = 4 \times 8n, n$ is an integer.

1) $R=(\alpha,\beta_0,m)$ is chaos parameter, initial value of iteration, initial iteration number. In this paper, the chaotic mapping of $G_\alpha^{m+j}(\beta_0)(j=1,2,3,\cdots)$ can obtain the 16x16 S table and the initial value G_0 of $4n$ bytes.

2) $i=0,\cdots,s-1$ calculate $G_{i+1}=\eta(S,P_i,G_i)$.

η is a compressed function. It consists of the following three loop functions.

Z function: First divide G_i into four $8n$ bits and distribute them among the four registers of $\gamma,\delta,\eta,\gamma$. Then run the operation**:**

$$\begin{cases}\gamma=z'(\gamma,\delta,\eta,\gamma,P_{i0})=((\gamma+P_{i0})\oplus\bar{\delta}\oplus\eta+\gamma)<<<1,\\ \delta=z'(\delta,\gamma,\eta,\gamma,P_{i1})=((\delta+P_{i1})\oplus\bar{\gamma}\oplus\eta+\gamma)<<<1,\\ \eta=z'(\eta,\delta,\gamma,\gamma,P_{i2})=((\eta+P_{i2})\oplus\bar{\delta}\oplus\gamma+\gamma)<<<1,\\ \gamma=z'(\gamma,\delta,\eta,\gamma,P_{i3})=((\gamma+P_{i3})\oplus\bar{\delta}\oplus\eta+\gamma)<<<1,\end{cases}\tag{1}$$

$P_{ij}(j=1,2,3,4)$ is j $8n$ bits in the packet in segment i. $<<<,+,\oplus,^-$ stands for left cyclic bit shift, modular 2^{8n} addition, XOR operation, and logical non-operation.

S: $\gamma,\delta,\eta,\gamma$ 4-bit number β is obtained from the end of A in order. Arrange the $\gamma,\delta,\eta,\gamma$ in the order shown in Table 1. In this paper, $\gamma,\delta,\eta,\gamma$ is separated from 1, 2, 3 and 4. See Table 1 for $z_{s1}^0z_{s2}^0z_{s3}^0z_{s4}^0$ shown in $Z(*)$ and the lower marker $s1,s2,s3,s4$ of $z_{s1}^0z_{s2}^0z_{s3}^0z_{s4}^0$ for an alternative order. If $s1s2s3s4=3124$, it is converted to CADB. This bit tuple is 4 bits higher than a line tag [6]. The 4-bit smaller is the column tag. Replace the table with the new byte S box. Finally, this paper concatenates each byte of n to $\gamma,\delta,\eta,\gamma$.

Table 1. Function Table

TT1	F(*)	TT1	F(*)
0	$z_1^0z_2^0z_3^0z_4^0$	1000	$z_3^0z_1^0z_2^0z_4^0$
1	$z_1^0z_4^0z_2^0z_3^0$	1001	$z_3^0z_2^0z_4^0z_1^0$
10	$z_1^0z_2^0z_4^0z_3^0$	1010	$z_3^0z_4^0z_1^0z_2^0$
11	$z_1^0z_3^0z_4^0z_2^0$	1011	$z_3^0z_1^0z_4^0z_2^0$
100	$z_2^0z_3^0z_1^0z_4^0$	1100	$z_4^0z_3^0z_2^0z_1^0$
101	$z_2^0z_4^0z_1^0z_3^0$	1101	$z_4^0z_2^0z_1^0z_3^0$
110	$z_2^0z_1^0z_3^0z_4^0$	1110	$z_4^0z_1^0z_2^0z_3^0$
111	$z_2^0z_3^0z_4^0z_1^0$	1111	$z_4^0z_2^0z_3^0z_1^0$

$\boldsymbol{S}$ function: In this paper getting a 4-bit $\boldsymbol{SS_1}$ starting from the end of $\boldsymbol{\gamma,\delta,\eta,\gamma}$. Then look for the $\boldsymbol{z_{s1}^0z_{s2}^0z_{s3}^0z_{s4}^0}$ function from Table 1. Perform $\boldsymbol{z_{s4},z_{s3},z_{s2},z_{s1}}$ functions in order according to $\boldsymbol{s1,s2,s3,s4}$ from the inside out. Here $\boldsymbol{z_1,z_2,z_3,z_4}$ is defined as

$$\begin{cases} \gamma = z_1(\gamma,\delta,\eta,\gamma) = (\gamma + \overline{\delta} \oplus \eta \oplus \gamma) <<< 3, \\ \delta = z_2(\gamma,\delta,\eta,\gamma) = (\delta + \gamma \oplus \overline{\eta} \oplus \gamma) <<< 1, \\ \eta = z_3(\gamma,\delta,\eta,\gamma) = (\eta + \gamma \oplus \delta \oplus \overline{\gamma}) <<< 2, \\ \gamma = z_4(\gamma,\delta,\eta,\gamma) = (\gamma + \gamma \oplus \delta \oplus \overline{\eta}) <<< 3, \end{cases} \tag{2}$$

3) $\boldsymbol{G = G_s}$.

Proposition 3. All the above methods have the same essential recovery attack ability.

Proof. This paper assumes that $n = 4$. This means that the length of the message block is 128 bits.

Start by traversing all values from 0 to 127 in i packet bytes at point $i = (i = 15,14,\cdots,1,0)$. The other 15 bits are denoted by zeros [7]. The $\{G_1^0,\cdots,G_1^{127}\}$value of MAC in this paper records 128 packets to obtain the original images of their D functions. Image $S^{-1}(G_1^j),\ j = 0,1,\cdots,127$ represents.

Solving the inverse function of S: Walk through 16 different combinations in the query table of a function. Add each function $z_{s1}z_{s2}z_{s3}z_{s4}$ in terms of $z_{s1}^{-1}, z_{s2}^{-1}, z_{s3}^{-1}$ and $_{s4}^{-1}$ from the outside to the inside to obtain the input value $\gamma,\delta,\eta,\gamma$ of function S. Then splitting the last bits of $\gamma,\delta,\eta,\gamma$ to form four bits of β. You can get a composite function by looking for $Z(\beta)$. If it's $s1s2s3s4 = s1's2's3's4'$, then the $\gamma,\delta,\eta,\gamma$ set will be saved. Otherwise, discard it.

The change rule of monitoring $S^{-1}(G_1^j)$ refers to the corresponding information of $S^{-1}(G_1^j)$ when the first byte of A changes. And I'm going to set it to P_{0i}^j. And then $0\beta(7z - P_{0i}^j + 1)$ is the initial value below 7 bits the i TH byte G_{0i}. That's when $G_{0i} + P_{0i}^j$ starts to advance [8]. Its conversion from 0x7f to 0x80. The bit-shift operation of the Z function changes causing one byte of the S function to become two bytes.

2.4 KHF based on chaotic dynamic parameters

Algorithm: $|P_i| = 512, |G| = 128$ or 160.

1) The order of $\boldsymbol{R = (\alpha, \beta_0, s)}$ is chaos parameter, iteration initial value and bit movement vector. $\boldsymbol{\beta}$ is a constant between zero and one. Chaotic map $\boldsymbol{G_\alpha^m(\beta_0)}$ is made from the first step of iteration $\boldsymbol{m + 1}$. $\boldsymbol{m}$ is the number of first repetitions. The current value of the output is processed in each loop to produce a 32 bit number. A total of $\boldsymbol{y}$: $\boldsymbol{S_0, S_1, \cdots, S_y}$ is produced.

2) $\boldsymbol{i = 0,\cdots,s - 1}$ calculate $\boldsymbol{G_{i+1} = \eta(G_i, P_i)}$.

   $\boldsymbol{\eta}$ is a compressed function. This method can use either the MD5 compression function or the SHA-1 encoding function. In MD5, $\boldsymbol{y = 67, S_0 - S_3}$ is its starting variable, and the other 64 constants are used in the original position. In SHA-1, $\boldsymbol{y = 84, S_0 - S_4}$ is its starting variable, and the other 80 constants are used for the original function.

3) $\boldsymbol{G = G_s}$

3 Research on security characteristics of chaos and hash

3.1 Statistical study of disorder and diffusion characteristics

Confusion and extension are introduced in Shannon's information theory. In the critical system, the plaintext must be completely scattered and chaotic in the ciphertext. Same with K-HF. It requires that the plaintext is independent of the corresponding hash value and that the ciphertext is highly sensitive [9]. Each bit in a binary hash expression is only 0 or 1, so the best hash function should be the initial value. Each bit change results in a 50% probability change in each bit of the hash result. Generally, confusion and extension of the following four statistical variables are classified as statistical analyses in this paper.

The average of the variant bits

$$\bar{\delta} = \frac{1}{N}\sum_{i=1}^{N}\delta_i \tag{3}$$

Mean probability of variation

$$P_{MD5} = (\bar{\delta}/128)\times 100\% \tag{4}$$

$$P_{SHA1} = (\bar{\delta}/160)\times 100\% \tag{5}$$

The variation of δ

$$\Delta\delta = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\delta_i - \bar{\delta})^2} \tag{6}$$

The variation of P

$$\Delta P_{MD5} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\delta_i/128 - P_{MD5})^2}\times 100\% \tag{7}$$

$$\Delta P_{SHA1} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\delta_i/160 - P_{SHA1})^2}\times 100\% \tag{8}$$

N is the amount of data. δ_i is the number of bits changed on trial i. In this paper, the plaintext is randomly selected as a hash test. Convert a 1-bit value in plain text to other hashes and compare their differences [10]. This is a K-HF method based on the iterative formula of MD5/SHA1. Its initial sensitivity is similar to that of MD5/SHA-1. The results of the comparison of N are shown in Tables 2 and 3.

Table 2. Initial sensitivity test of chaos algorithm based on MD5

The number of iterations	267	533	1067	2133	total average
B	66.90	66.69	66.74	66.87	66.80
ΔB	6.25	5.82	5.92	5.98	5.99
PMD5/%	52.26	52.10	52.13	52.24	52.19
ΔPMD5/%	4.89	4.55	4.63	4.66	4.68
Bmax	83	85	89	86	85.94
Bmin	53	48	50	42	48.18

Table 3. Initial sensitivity test based on Chaotic SHA1

The number of iterations	267	533	1067	2133	total average
B	83.517	83.539	83.188	83.190	83.358
ΔB	7.078	6.810	6.621	6.474	6.746
P_{SHA1}/%	52.198	52.211	51.992	51.994	52.099
ΔP_{SHA1}/%	4.424	4.256	4.139	4.046	4.216
$B_{\max}$	105	104	103	106	104.69
$B_{\min}$	64	61	64	63	62.76

3.2 Key sensitivity test

The paper investigates the hash values of randomly selected keys $R(\alpha, \beta_0, s)$ and R. Then the plain text is left unchanged, and the 1-bit data is randomly changed to R and the hash value is compared again. The results of N comparisons are shown in Tables 4 and 5.

Table 4. Key Sensitive tests of chaos and hash function algorithms

The number of iterations	267	533	1067	2133	total average
B	66.308	66.898	66.822	66.573	66.650
ΔB	5.632	5.910	6.020	5.847	5.852
P_{MD5}/%	51.803	52.264	52.204	52.010	52.070
ΔP_{MD5}/%	4.400	4.617	4.703	4.568	4.572

Table 5. Key sensitivity test of Chaotic SHA1 algorithm

The number of iterations	267	533	1067	2133	total average
B	83.679	83.404	83.475	83.210	83.442
ΔB	6.319	6.816	6.228	6.713	6.519
P_{SHA1}/%	52.299	52.127	52.172	52.006	52.151
ΔP_{SHA1}/%	3.949	4.259	3.893	4.195	4.074

Simulation data shows that the new key's average mutation times and mutation probability are close to half of the optimal solution when the key is changed slightly. At the same time, it is thoroughly and evenly used in the critical space. $\Delta\delta, \Delta P$ represents the stability of the scattering and scattering properties, and the closer it is to 0, the more stable it is. All of the Δ here are very small. This method has better plaintext dispersion and spurious performance [11]. Due to the extreme sensitivity and uniform diffusion of the initial value and the key, it is difficult for the attacker to obtain valid data. This way, the existing key and differential linear attacks are effectively resisted [12].

3.3 Comparison of Operation rates

In this paper, the effectiveness of mixing and hashing methods is experimentally discussed. This paper compares this method with the hybrid hashing method in other kinds of literature [4], [5] and

[6]. Each set of data was averaged over 100 tests. This paper uses five methods to summarize random data of arbitrary length in a 32-bit environment. The initial value of chaos and hash is 40, the initial value of the reference is 100, and the size of the hash abstract is 128. The respective operation times are shown in Table 6.

Table 6. Comparison of the efficiency of Chaotic hash algorithms

Text length /bit	50	100	200	1000	1×10^4
Chaos and hash function algorithms /ms	0.0081	0.0086	0.0083	0.0102	0.0173
Chaotic-sha1 algorithm /ms	0.0099	0.0103	0.0102	0.0127	0.0312
Literature [4] Algorithm /ms	0.1523	0.2267	0.6771	2.6024	19.3165
Literature [5] Algorithm /ms	29.29	86.58	502.66	16827.03	1295831.78
Literature [6] algorithm /ms	0.2121	0.2675	0.7384	2.0985	12.2549

Table 6 shows this method is more effective than the other three methods. The methods proposed in the literature [4] and literature [5] add packets to the chaotic iteration, so the processing speed is slow. In the S-Box hash of the literature [6], although a similar method is used to avoid the direct addition of information packets to the chaotic iteration, the process of S-Box construction is cumbersome, and the iterative method is not very good. Therefore, there is no significant advantage in performance. Judging from the number of iterations, if starting with 100 iteration values, the S-Box hashing requires 2148 iterations. The two algorithms presented in this paper require only 168 iterations and 185 iterations under optimal conditions. Because the new algorithm has a 512-bit data packet, the text lengths in the table are compared with 50 bits, 100 bits, 200 bits, 1000 bits and 1×10^4 bits, respectively. The results show that a single iterator takes a deficient proportion of the time in a single packet. The efficiency of this method is similar to that of MD5/SHA-1 as the text increases.

4 Conclusion

This paper presents a method of constructing a unidirectional hash using chaotic dynamics parameters. It takes the existing hash method as the center and uses the dynamic parameters generated by chaos to replace the known static parameters for hash operation. In this paper, the simulation and theoretical study of the method prove the effectiveness of the technique. Each bit change in the initial data changes is 50% of the variables in the hash value. In this way, the avalanche performance of plaintext is the best. Since any 1-bit key change over 40 initial iterations results in almost 50% of the hash value changes, it is susceptible. This method adopts a mature hashing method. It has faster computational efficiency and better security.

References

[1] Amine, Z., Naima, H. S., & Ali-Pacha, A. (2022). A secure hash function based on sponge construction and chaos-maps. International Journal of Computational Science and Engineering, 25(3), 285-297.
[2] Jabeen, T., Ashraf, H., & Ullah, A. (2021). A survey on healthcare data security in wireless body area networks. Journal of ambient intelligence and humanized computing, 12(10), 9841-9854.
[3] Mei, D. (2022). What does students' experience of e-portfolios suggest. Applied Mathematics and Nonlinear Sciences, 7(2), 15-20.
[4] Rahaman, H., Kamrul Hasan, M., Ali, A. & Shamsul Alam, M. (2021). Implicit Methods for Numerical Solution of Singular Initial Value Problems. Applied Mathematics and Nonlinear Sciences, 6(1), 1-8.
[5] Liu, H., Wang, X., & Kadir, A. (2021). Constructing chaos-based hash function via parallel impulse perturbation. Soft Computing, 25(16), 11077-11086.

[6] Chai, X., Fu, X., Gan, Z., Zhang, Y., Lu, Y., & Chen, Y. (2020). An efficient chaos-based image compression and encryption scheme using block compressive sensing and elementary cellular automata. Neural Computing and Applications, 32(9), 4961-4988.
[7] AbdElminaam, D., & Abdullah Ibrahim, M. (2022). FHE-Chaos NHCP: Developing a Novel Secure Framework for Cloud Computing Environment. Journal of Computing and Communication, 1(1), 12-26.
[8] Aouissaoui, I., Bakir, T., & Sakly, A. (2021). Robustly correlated key-medical image for DNA-chaos based encryption. IET Image Processing, 15(12), 2770-2786.
[9] Huang, R., Liao, X., Dong, A., & Sun, S. (2020). Cryptanalysis and security enhancement for a chaos-based color image encryption algorithm. Multimedia Tools and Applications, 79(37), 27483-27509.
[10] Kaur, G., Singh, K., & Gill, H. S. (2021). Chaos-based joint speech encryption scheme using SHA-1. Multimedia tools and applications, 80(7), 10927-10947.
[11] Abdoun, N., El Assad, S., Deforges, O., Assaf, R., & Khalil, M. (2020). Design and security analysis of two robust keyed hash functions based on chaotic neural networks. Journal of Ambient Intelligence and Humanized Computing, 11(5), 2137-2161.
[12] Mishra, D., Obaidat, M. S., Rana, S., Dharminder, D., Mishra, A., & Sadoun, B. (2020). Chaos-based content distribution framework for digital rights management system. IEEE Systems Journal, 15(1), 570-576.

This page is intentionally left blank