Received: February 8, 2024. Revised: March 1, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Unprecedented Security Analysis Results for a Novel Key Expansion Algorithm Based on Protein Sequences, Dynamic Mealy Machine, and 3D Logistic Map

Radhwan Jawad Kadhim[1]* **Hussein K. Khafaji[2]**

[1]*Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics, Baghdad, Iraq*
[2]*Computer Engineering Department-Al-Rafidain University College, Baghdad, Iraq*
* Corresponding author's Email: phd202110686@iips.edu.iq; hussain.ketan.elce@ruc.edu.iq

Abstract: The key expansion algorithm is an important part of any symmetric block cipher system since its effectiveness directly impacts the security of the entire block cipher; if it is not strong enough, the whole cryptosystem could be broken. Therefore, the round keys must be generated in a very secure way so that they cannot be attacked at all. Despite its great importance, cryptographic algorithm designers were not as interested in creating a secure round keys generation algorithm as they were in encryption itself. In this regard, designing a novel, simple, and flexible key expansion algorithm that generates round keys with secure characteristics is the aim of this research. The proposed Key Expansion algorithm is based on Dynamic Mealy Machine, 3D Logistic Map, and Protein Sequence (KE-DMM3DLMPS). The strength of the proposed KE-DMM3DLMPS algorithm is tested using flexibility, the NIST SP800-22 randomness test suite, histogram analysis, key space analysis, correlation coefficient, hamming distance, number of bit change rate, initial key sensitivity, confusion and diffusion, and differential attack. In comparison to some existing algorithms, experimental results showed that the generated round keys by the proposed KE-DMM3DLMPS algorithm passed all the NIST tests with higher randomness, a uniform and ideal distribution of the amino acids present in each round key, and a higher key space of up to 20×2^{471}. Furthermore, the KE-DMM3DLMPS avoids the linear relationship between the master secret key and the generated round keys and is capable of effectively blocking differential attacks. The proposed algorithm successfully adheres to key cryptographic principles such as irreversibility, independence, the strict avalanche effect, confusion, and diffusion. Through comprehensive testing and comparisons, the derived conclusion asserts that our algorithm stands as an efficient and secure solution. Its applicability extends to any symmetric block cryptosystem, with the primary goal of enhancing encryption and bolstering security.

Keywords: Key expansion, Symmetric block cipher, 3D logistic map, Mealy machine, Protein sequence, NIST tests, Key space, Initial key sensitivity, Irreversibility, Independence, Strict avalanche effect.

1. Introduction

Security is crucial in the storage and transmission of information across networks, ensuring that it is protected and delivered in a secure manner between different locations [1]. Therefore, ensuring secure communication is an essential prerequisite for any transactions conducted across networks. Cryptography plays a crucial role in guaranteeing the secure transmission of data by employing security measures such as authentication, data integrity, non-repudiation, access control, and confidentiality. Data confidentiality is the safeguarding of sensitive data to prevent unwanted access by external entities [2].

Cryptography offers a means of safeguarding confidential data by transforming it into incomprehensible form, which can only be deciphered by the authorized recipient to retrieve the original information. The act of transforming plain, readable information into coded, unreadable material using a specific key is referred to as the encryption process. Conversely, the act of decoding the encrypted text back into its original form is known as the decryption process [1]. The creation of a completely secure cryptographic system is

challenging due to the persistent efforts of cryptanalysts who are always attempting to compromise any existing cryptographic systems[3]. Cryptographic systems can be classified into three main types: symmetric cipher system, which uses a secret key for both encryption and decryption, asymmetric cipher system, which uses a public key for encryption and a private key for decryption and hashing function. Moreover, the symmetric cipher system can be categorized into block and stream ciphers based on the combination of message bits. Symmetric block cipher system consists of five primary components: plaintext, encryption algorithm, ciphertext, decryption algorithm, and key expansion algorithm(KEA)[4]. The key must possess sufficient strength and length to prevent it from being compromised through a brute-force attack[5]. In the present day, it is advisable to utilize a minimum of a 128-bit key for symmetric algorithms[6]. In a symmetric block cipher system, before starting the encryption or decryption process, a key called master secret key is used to derive the required number of sub-keys (round-keys) based on the specified KEA. To conceal the correlation between the round input and round output, each sub-key is shuffled with the round data. Therefore, creating a strong KEA plays a crucial role in the advancement of any symmetric block cipher system since its effectiveness directly impacts the security of the entire block cipher[7]-[9]. Therefore, a KEA must generate sub-keys with a high degree of randomness and also exhibit robust confusion and diffusion characteristics, ensuring that all derived sub-keys are independent of one another. This guarantees that the compromise of any individual sub-key does not disclose any details about the secret key or other sub-keys. The entire cryptosystem may be compromised if the KEA is weak. One of the reasons for the weakness of the KEA arises from the linear relationship between the generated sub-keys and the master secret key, making the cryptosystem potentially vulnerable to differential, linear, related-key, statistical, and slide attacks [7][10]. A strong and secure KEA enhances the entire cipher's resistance against various assaults, including the mentioned attacks and others.

In the literature, designing a strong and secure KEA has received less attention compared to encryption techniques[10], [11]. However, Harmouch and El Kouch [12] incorporated the concept of chaos into the key schedule (expansion) algorithm, resulting in the development of a new key scheduling method called CKSA, which is based on logic maps. This suggested algorithm is a one-way function that guarantees effective confusion and diffusion, as well as a good avalanche effect. But the randomization degree of this method is not sufficiently high, according to the NIST test suite. Which means that the generated round keys have weak randomness and low complexity. Wang et al.[13] proposed a key expansion algorithm based on the chaotic map and genetic algorithm. Ten out of fifteen tests have passed the randomization tests of the NIST test suite, while the rest were not explained. Poojari and H R[14] proposed a novel method that generates random numbers by utilizing the scrambling algorithm and Linear Feedback Shift Register to produce only five sub-keys for the lightweight encryption algorithms, each sub-key with a length of 16 bits. Therefore, the key length is not sufficient to resist the brute force attack. The randomness of the generated sub-keys is tested using only the NIST test suite. The extent of the linear relationship of the generated sub-keys to the master secret key was not measured. Zakaria et al. [15] enhance the RECTANGLE key expansion algorithm to augment its confusion and randomization characteristics in addition to the performance results regarding speed and throughput. However, the two designs did not pass all of the NIST tests. Garba et al. [16] proposed a simple key expansion algorithm. Only four tests out of the 15 NIST tests were used to evaluate the performance of the algorithm. Some of the sub-keys did not pass the poker, serial, and frequency tests. Therefore, any sub-key that does not pass the frequency test is not considered random, and thus the algorithm is unsafe. Alawida et al. [17] proposed a new method to generate round keys from a given secret key based on finite state machine and DNA sequence. This method is not very sensitive to single-bit changes because, when the experiment was repeated for 64 different secret keys, it was found that there are three sub-keys that are exactly similar to the secret key, with an average difference equal to zero. Xu and Liu [18] use a primitive polynomial over GF(2n) and a 2D nondegenerate exponential chaotic map to build a key expansion algorithm. The round keys are mutually independent, and the algorithm successfully satisfies the irreversibility and parallelism requirements. The NIST test suite has not assessed the randomness of the generated keys, and it is also challenging for readers who are unfamiliar with the subject to understand this key generation method to demonstrate that it is secure.

In this regard, this scientific paper proposes a novel key expansion algorithm based on Dynamic Mealy Machine, 3D Logistic Map, and Protein Sequence (KE-DMM3DLMPS), which can be used for any symmetric block cipher system. The proposed KE-DMM3DLMPS algorithm can produce strong and secure round keys and overcoming all the

problems highlighted in previous studies. The following are this study's contributions and novelties:

- Incorporate the concepts of Mealy Machine, 3D Logistic Map, and Protein Sequence to propose a novel key expansion algorithm (KE-DMM3DLMPS), which can be used for any symmetric block cipher system. The protein sequences are not utilized for the purposes of key expansion or data encryption.
- Proposing a new method for encoding amino acids called (Amino Acid Binary Encoding Rule) for the purpose of using them later in encryption or key expansion operations.
- The ability to generate a different number of round keys with the desired lengths, which allows many people interested in cryptography to use this algorithm.
- Generate round keys with a high degree of complexity and randomness by passing all of the NIST tests.
- A uniform and ideal distribution of the amino acids present in each round key, which ensures resistance to statistical analysis attacks.
- A very high key space of up to 20×2^{471}.
- Breaking the linear relationship between the master secret key and the generated round keys.
- The suggested key expansion method meets the following principles:
  - Irreversibility: no sub-key can deduce the master secret key.
  - Independence: all generated sub-keys are independent of each other.
  - Initial key sensitivity: it is very sensitive to the master secret key, satisfying the strict avalanche effect (SAC).
  - Confusion: making the relationship between the master secret key and the round keys as complicated as possible.
  - Diffusion: ensuring that even a slight change in the master secret key will have a widespread effect on all bits of the round key.
- The suggested technique is capable of effectively blocking differential attacks.
- The proposed KE-DMM3DLMPS algorithm is compared with previous studies based on the NIST test suite, histogram analysis, and key space analysis. We conclude that our proposed model outperforms the existing algorithms.

The subsequent sections of this study are structured in the following manner: Section 2 discusses the context of the suggested scheme. The design of the proposed key expansion algorithm is outlined in Section 3. Section 4 explains the assessment of experimental results for the suggested method. Section 5 focuses on comparing the proposed method with some key expansion algorithms. Section 6 presents the conclusions of this research.

Table 1. The genetic code [19]

First Letter	Second Letter: U		Second Letter: C		Second Letter: A		Second Letter: G		Third Letter
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC		UCC		UAC		UGC		C
	UUA	Leu	UCA		**UAA**	Stop	**UGA**	Stop	A
	UUG		UCG		**UAG**		UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lys	AGA	Arg	A
	AUG	Met	ACG		AAG		AGG		G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glu	GGA		A
	GUG		GCG		GAG		GGG		G

2. Context of the suggested scheme

In the following section, we include some background material on protein sequence, PAM250 matrix, 3D Logistic map and the Mealy Machine because both of these concepts are important to the suggested Key Expansion Scheme.

2.1 Protein sequence

Deoxyribose nucleic acid, or DNA, is a very big molecule that carries genetic information and features that are vital to the survival and development of every living organism [19]. DNA is typically made up of two long strands that run in opposing directions, forming a double helix. Each strand is composed of a lengthy chain of subunits. The building blocks of the subunits are termed nucleotides, and each nucleotide is composed of a phosphate group, a nitrogenous base, and either a purine or pyrimidine base. Two types of nitrogenous bases are distinguished: the pyrimidine bases, Thymine (T) and Cytosine (C), which constitute the "genetic code," and the purine bases, Adenine (A) and Guanine (G). Thymine (T) and Cytosine (C) are the pairings of Adenine (A) and Guanine (G) respectively [19, 20]. The biological system of any living organism depends on how these four bases are arranged, as this determines the type of protein molecule and drives all activity in living cells. Furthermore, distinct protein types have varied functions [21]. Transcription is the term for the complex and protracted process known as central dogma that converts DNA to RNA (ribonucleic acid), which is thought to be a step in the synthesis of

Table 2. PAM250 Matrix[22] (for only 8 amino acids)

Amino Acid	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2	-2	**0**	**0**	**-2**	0	**0**	1	**-1**	-1	**-2**	-1	**-1**	**-3**	1	**1**	1	-6	**-3**	**0**
T	1	-1	**0**	**0**	**-2**	-1	**0**	0	**-1**	0	**-2**	0	**-1**	**-3**	0	**1**	3	-5	**-3**	**0**
E	0	-1	**1**	3	**-5**	2	4	0	1	**-2**	**-3**	0	-2	**-5**	**-1**	**0**	**0**	-7	**-4**	**-2**
K	-1	3	**1**	0	**-5**	1	0	-2	0	**-2**	**-3**	5	0	**-5**	**-1**	**0**	**0**	-3	**-4**	**-2**
I	-1	**-2**	**-2**	**-2**	**-2**	**-2**	**-2**	-3	**-2**	5	**2**	**-2**	**2**	1	-2	**-1**	**0**	-5	-1	**4**
V	0	**-2**	**-2**	**-2**	**-2**	**-2**	**-2**	-1	**-2**	4	**2**	**-2**	**2**	-1	-1	**-1**	**0**	-6	-2	**4**
N	0	**0**	2	2	-4	1	1	**0**	2	**-2**	**-3**	1	**-2**	-3	0	**1**	**0**	-4	-2	-2
P	1	**0**	0	-1	-3	0	-1	**0**	0	**-2**	**-3**	-1	**-2**	-5	6	**1**	**0**	-6	-5	-1

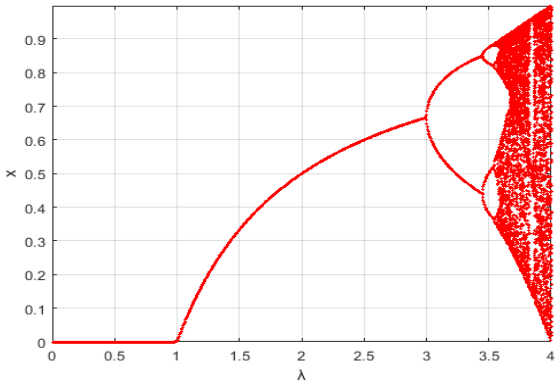

Figure. 1 Chaotic behavior of a 1D logistic map for 1000 iterations

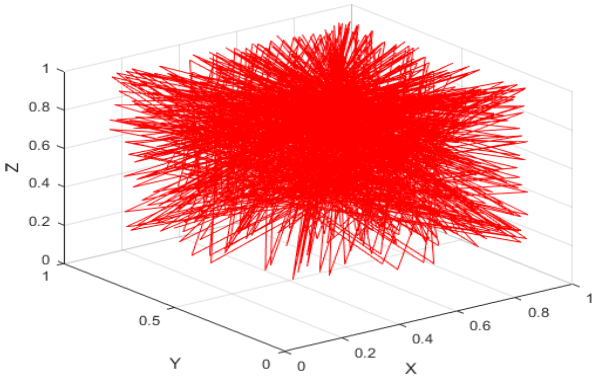

Figure. 2 Chaotic behavior of a 3D logistic map for 1000 iterations

proteins [20]. Translation is the process that converts RNA into the amino acids that make up a protein molecule. Codons are groups of three consecutive nucleotides that are extracted from RNA during translation. Each codon represents an amino acid, and the arrangement of these amino acids determines the structure and function of the resulting protein [19], [21-23]. Most of the twenty amino acids that can be produced from distinct codons can be produced from several codons, as Table 1 illustrates. Three STOP codons serve as additional markers for the protein sequence's end in addition to amino acids [24].

2.2 PAM250 scoring matrix

The PAM250 matrix, also known as the Point Accepted Mutation Version 250 matrix [22], is derived by iteratively multiplying the PAM1 matrix with itself for a total of 250 iterations. This matrix is widely employed in BLAST searches of databases. It is a widely employed tool in the field of bioinformatics, specifically for the purpose of matching amino acid sequences. Its primary function is to assign a numerical score to each alignment, facilitating the comparison and evaluation of different alignments. The intersection of amino acids inside the matrix corresponds to a distinct score that quantifies the degree of their potential interactions with other amino acids in the matrix, as explained in Table 2.

2.3 Chaotic map

Chaos maps are currently used in the area of encryption due to their benefits and ability to improve encryption system security[25]. They are dynamical systems that lack linearity and are highly sensitive to the initial conditions, which display random behavior in response to those initial conditions. These conditions include the values allocated to the system parameters, as explained below [26]. Different kinds of chaotic maps have been used by researchers, but

the one-dimensional (1D) logistic map is probably the most simple and well-known map and is defined in Eq. (1) [27, 28], as follows:

$$x_{n+1} = \lambda x_n(1 - x_n) \quad (1)$$

Here, x_n is the state variable ($0 < x_n < 1$), λ is the system parameter ($0 < \lambda < 4$), and n is the number of iterations needed to iteratively produce the state values. It was proved that the sequences generated in a 1D logistic map and at $3.56994 < \lambda \leq 4$ lead towards chaotic behavior [29] , as shown in Fig. 1.

A 1D chaotic map can be expanded into a three-dimensional (3D) chaotic map, which offers a high level of unpredictability and thus more security, as explained in Eqs. (2)-(4) [30].

$$x_{i+1} = \alpha x_i(1 - x_i) + \beta y_i^2 x_i + \sigma z_i^3 \quad (2)$$

$$y_{i+1} = \alpha y_i(1 - y_i) + \beta z_i^2 y_i + \sigma x_i^3 \quad (3)$$

$$z_{i+1} = \alpha z_i(1 - z_i) + \beta x_i^2 z_i + \sigma y_i^3 \quad (4)$$

Chaotic behavior is exhibited when the values of the parameters α, β, and σ fall within the specified ranges of $3.53 < \alpha < 3.81$, $0 < \beta < 0.022$, and $0 < \sigma < 0.015$, respectively. Additionally, the variables x_0, y_0, and z_0 are constrained to the interval [0, 1].

Fig. 2 illustrates the full chaotic nature and dynamical behavior of the three-dimensional logistic map, which exhibits a higher degree of randomness compared to its one-dimensional counterpart and can be effectively employed as a pseudo-random number generator in building the private secret tables for a mealy machine.

2.4 Mealy machine (MM)

In computation theory, the Mealy machine is defined as a finite-state machine [31]; MM can be used in a cryptographic field to achieve a more secure system [32]-[35], where its output depends on the present state and the present input of the machine [35]. It is defined by six tuples, i.e., M = (Q, I, O, δ, Ω, q_0). Such that:

- Q represents a collection of non-empty, finite states.
- I and O represent a collection of finite input and output alphabets, respectively.
- q_0 represents the initial state, where $q0 \in Q$.

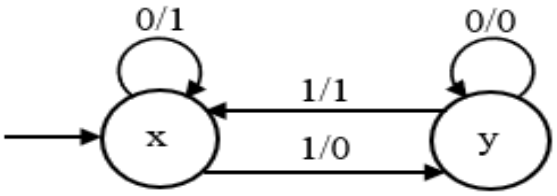

Figure. 3 Design of MM with two states

Table 3. Mealy machine state table

Input transition function δ: $Q \times I \rightarrow Q$		
Present State	**Input = 0**	**Input = 1**
	Next State	Next State
X	x	y
Y	y	x

Table 4. Mealy machine output table

Output transition function Ω: $Q \times I \rightarrow O$		
Present State	**Input = 0**	**Input = 1**
	Output	Output
x	1	0
y	0	1

- δ represents the input transition function, such that δ: $Q \times I \rightarrow Q$.
- Ω represents the output transition function, such that Ω: $Q \times I \rightarrow O$.

An example of Mealy machine is as follows:

- Q = {x, y}
- I = {0, 1}
- O= {0, 1}
- q_0 = x

As depicted in Fig. 3 as a transition diagram, the Mealy machine in this example consists of two states ('x' and 'y'), where 'x' is the initial state. The binary bits 0 and 1 are considered input and output alphabets. The Mealy machine in this example is used to convert any binary string into another binary string.

The design of the Mealy machine is based on the state and output tables, as shown in Tables 3 and 4. Assume the binary string 1011101 is the Mealy machine's input. The machine starts from the initial state 'x' and reads the first input symbol '1'; therefore, the output string is '0', and it moves to state 'y'. Then the next binary sting read is '0', so the output machine is '0', and the machine stays in the state 'y'. This procedure is iterated until all the input binary strings have been used. Finally, the Mealy machine input string '1011101' becomes '0010110'.

3. Proposed key expansion algorithm design

This section discusses the materials and methods used to design a novel, robust, and secure key

Table 5. Amino acid binary encoding rule

Amino Acid	Three letters Code	One letter code	4 bits Binary code
Cysteine	Cys	C	0000
Aspartic Acid	Asp	D	0001
Phenylalanine	Phe	F	0010
Alanine	Ala	A	**0011**
Threonine	Thr	T	
Glycine	Gly	G	0100
Glutamic Acid	Glu	E	**0101**
Lysine	Lys	K	
Histidine	His	H	0110
Leucine	Leu	L	0111
Methionine	Met	M	1000
Glutamine	Gln	Q	1001
Isoleucine	Ile	I	**1010**
Valine	Val	V	
Arginine	Arg	R	1011
Asparagine	Asn	N	**1100**
Proline	Pro	P	
Serine	Ser	S	1101
Tryptophan	Trp	W	1110
Tyrosine	Tyr	Y	1111

expansion method that has the ability to create sub-keys independent of each other, has high randomness, a large key space, and some other good security properties that enable it to be widely used in any symmetric block cryptosystem. The proposed Key Expansion algorithm is based on Dynamic Mealy Machine, 3D Logistic Map, and Protein Sequence (KE-DMM3DLMPS). Below is a detailed explanation of the design of the proposed key expansion method.

3.1 Conversion secret key to amino acid-bases

In computing domains like data encryption and key generation, DNA ideas are commonly used. This is because it is simple to translate each of the two binary numbers (00, 10, 01, 11) into one of the four nucleotides (A, C, G, and T). However, our investigation revealed that no one has ever used a protein sequence to expand the keys or encrypt data. The twenty amino acids can be represented using five bits [36]. We proposed to take the four most significant bits to represent 20 amino acids. Therefore, four amino acids will have the same binary numbering when represented by four bits. According to Table 2 in subsection 2.2, we conclude that the score of the amino acid Alanine (A) is similar to the score of the amino acid Threonine (T) in eleven positions, the score of the amino acid Glutamic Acid (E) is similar to the score of the amino acid Lysine (K) in ten positions, the score of the amino acid Isoleucine (I) is similar to the score of the amino acid Valine (V) in thirteen positions, and the score of the amino acid Asparagine (N) is similar to the score of the amino acid Proline (P) in seven positions when each of them interacts with all amino acids. So, we will combine those amino acids that have a high degree of similarity.

In Table 5, we proposed the Amino Acid Binary Encoding Rule (AABER) for 20 amino acids to facilitate the processes taking place on the protein, such as Amino Acid exclusive-OR (AA-XOR), which can be used during the suggested key expansion algorithm. Assume '2f34e9a3' is the secret key that must be changed into amino acid bases. First, this secret key must be converted to binary form. Next, the binary value of the secret key '0010 1111 0011 0100 1110 1001 1010 0011' must be pre-processed to count the repeated number of binary codes for the merged amino acids. If the number of binary codes for the merged amino acids is greater than one, then those amino acids are distributed equally among those binary codes in order to give equal shares to each of those amino acids. In this example, the number of binary codes (0011) for the merged amino acids (A and T) is two (Count_AT=2). After the pre-processing step, each of the 4 bits must be converted to an amino acid, according to Table 5, taking into account the number of merged amino acids. So, the binary secret key for this example becomes as follows: FYAGWQIT.

3.2 Proposed protein operations

The twenty amino acids of a protein sequence can be represented using four bits, as we proposed in Section 3.1. There are in total 16! (20,922,789,888,000) different ways to map these four bits to the amino acids (also called amino acid binary encoding rules); therefore, it is very difficult for an attacker to guess the correct binary coding rule for amino acids. For example, Table 5 is one of the amino acid binary encoding rules out of 16!.

Besides these amino acids binary encoding rules, various protein operations can be carried out on these twenty amino acids, such as Amino Acid (AA) addition, AA subtraction, AA-XOR, and AA complementary operation. The process of key expansion holds significant importance in encryption systems, with XOR playing a pivotal role in augmenting both the speed and security of this process. So, we will use the AA-XOR operation in this research.

In order to apply the AA-XOR operation to amino acids, the amino acids are first converted to the binary sequence according to the rule in Table 5, and then the usual XOR operations are performed (i.e., 0 xor 1 = 1, 1 xor 1 = 0). Next, we pre-process the binary sequence to count the number of repeated binary codes for the merged amino acids. If the number of binary codes for the merged amino acids is greater than one, then those amino acids are distributed equally among those binary codes in order to give equal shares to each of those amino acids. Finally, we convert the binary sequence into an amino acid sequence using the rule specified in Table 5. For instance, if the two protein sequences are "CHAA" and "RDYY", then the XOR operation of those two sequences is "RLNP".

3.3 3D logistic map sequence generation

We can use the Chaos map system that was previously explained in sub-section 2.3 to generate the pseudorandom sequence, which can be exploited as a secret key to build the mealy machine transition tables. There is a need to use chaotic systems in the cryptographic field because they naturally have nonlinearity and random behavior, which makes it possible to create pseudorandom sequences. The subsequent procedure delineates the process of sequence generation:

Step 1: Initial Values Generation

The initial values of x_0, y_0, and z_0 in Eqs. (2)-(4) are obtained from a 256-bit master key MK, as shown by the proposed Eqs. (5)-(7), where $MK = \{K_1, K_2, \dots, K_{32}\}$, and each K represents an 8-bit binary number.

$$X_0 = mod(\, K_1 \oplus K_2 \oplus \dots \oplus K_{10} + \sum_{i=1}^{32} K_i \,/\, 2^{12}, 1) \quad (5)$$

$$Y_0 = mod\,(\, K_{11} \oplus K_{12} \oplus \dots \oplus K_{21} + \sum_{i=0}^{15} K_{2i+1} \,/\, 2^{12}, 1) \quad (6)$$

$$Z_0 = mod\,(\, K_{22} \oplus K_{23} \oplus \dots \oplus K_{32} + \sum_{i=1}^{16} K_{2i} \,/\, 2^{12}, 1) \quad (7)$$

where the symbol $\oplus$ denotes the exclusive OR (XOR) operator. According to Eqs. (5)-(7), the initial values of X_0, Y_0, and Z_0 will be set within the interval [0, 1]. For example, if the master key is: MK ='895389AD00493BFEDF5A293B1E876B25C6127E1C26C0FBE228F57CB0D7476053', then X_0, Y_0, and Z_0 are equal to (0.97949, 0.46558, and 0.50049), respectively.

Step 2: 3D Logistic Map Sequence Generation

Firstly, iterate the Eqs. (2)-(4) for n times, using the initial values (X_0, Y_0, and Z_0) that were obtained from step 1 and with (α = 3.80, β = 0.021, and σ = 0.013). For each iteration, we can obtain three decimal sequences (X_i, Y_i, and Z_i) that lie between 0 and 1.

Secondly, convert the second and third sequences (Y_i and Z_i) of the 3D Logistic Map into integer sequences (IY_i and IZ_i)between 1 and 20 in order to use them in building state and output tables, as shown in the proposed Eqs. (8) and (9):

$$IY_i = mod\,(\lfloor\,((Y_i + 100) \times 10^{10})\,\rfloor, 20) + 1 \quad (8)$$

$$IZ_i = mod\,(\lfloor\,((Z_i + 100) \times 10^{10})\rfloor, 20) + 1 \quad (9)$$

where $\lfloor x \rfloor$ = maximum $\{a \in Z;\, x \geq a\}$.

Thirdly, after applying Eqs. (8) and (9), all the decimal sequences (Y_i and Z_i) of the 3D Logistic Map will be converted to integer sequences (IY_i and IZ_i), but duplicate values will appear. To obtain the desired randomness, all repeated values are removed and replaced with the remaining values within the period from 1 to 20.

3.4 Design a dynamic mealy machine based on a 3D logistic map and protein sequences (DMM3DLMPS)

The main objective of this sub-section is to allocate values to the six tuples of the proposed mealy machine. Since the input and the output of the mealy machine are protein sequences, the tuples *I* and *O* are sets of amino acid bases. Given that there are twenty distinct amino acid bases and each state produces a certain amino acid base, the machine is characterized by twenty states, namely 1, 2, 3 ,4 ,5 ,6 ,7 ,8 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20, where any state of those twenty states can be assigned to the initial state. The input transition function (δ: $Q \times I \rightarrow Q$) and the output transition function (Ω: $Q \times I \rightarrow O$) are generated randomly based on the 3D logistic map, and the results are stored in the state table and the output table, respectively. Therefore, the values that were assigned to the six tuples can be summarized as follows:

- Q = {1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,20}
- I = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}
- O= {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}
- δ = Randomly generated
- Ω = Randomly generated

Table 6. A section of a generated mealy machine state table

Input transition function δ: $Q \times I \rightarrow Q$																				
Present State	**Input**																			
	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
	Next State																			
1	20	12	11	2	14	15	6	18	5	19	4	13	8	3	7	16	9	10	17	1
2	1	20	12	11	2	14	15	6	18	5	19	4	13	8	3	7	16	9	10	17
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	19	4	13	8	3	7	16	9	10	17	1	20	12	11	2	14	15	6	18	5
20	9	10	17	1	20	12	11	2	14	15	6	18	5	19	4	13	8	3	7	16

Table 7. A Section of a generated mealy machine output table

Output transition function Ω: $Q \times I \rightarrow O$																				
Present State	**Input**																			
	D	E	A	C	F	G	H	I	L	Y	M	K	N	Q	R	S	P	T	W	V
	Output																			
1	Y	L	K	N	D	M	Q	W	T	G	I	S	A	H	V	C	P	E	R	F
2	I	S	A	H	V	C	P	E	R	F	Y	L	K	N	D	M	Q	W	T	G
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	C	P	E	R	F	Y	L	K	N	D	M	Q	W	T	G	I	S	A	H	V
20	M	Q	W	T	G	I	S	A	H	V	C	P	E	R	F	Y	L	K	N	D

- q_0 = Assigned by the user (q_0 = x | x ∈ Q)

The design of the DMM3DLMPS is primarily expressed by means of two transition tables, which are referred to as the secret state table (SST) and the secret output table (SOT). The SST store the input transition function δ: Q × I → Q as shown in Table 6 and the SOT store the output transition function Ω: Q × I → O as shown in Table 7. The values of the 'next states' in Table 6 are allocated at random from the set {1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,20} based on the decimal numbers generated from the second dimension of the 3D logistic map (IY_n) and left shift as explained in Algorithm 1. The values of the 'Output' in Table 7 are allocated at random from the set {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y} based on the decimal numbers generated from the third dimension of the 3D logistic map (IZ_n) and left shift as explained in Algorithm 2.

Algorithm 1. Proposed Secret State Table Generation Algorithm

Input: $\boldsymbol{IY_n}$ // The second dimension of the 3D logistic map
Output: $\boldsymbol{SST}$ // $\boldsymbol{SST}$ is a 20 × 20 double matrix
Step1. $\boldsymbol{SST}$ = zeros (20) // Initialize a 20 × 20 empty matrix
Step2. Performing a left shift rotation and filling the *SST* matrix:
for $\boldsymbol{k}$ from **1** to **20** do
$\boldsymbol{R_E} = \boldsymbol{Left_{Shift}}\ (\boldsymbol{IY_n}\ , -(\boldsymbol{IY_n}\ [\boldsymbol{k}]))$
$\boldsymbol{SST}\ [\boldsymbol{k}\ , :] = \boldsymbol{R_E}$
end $\boldsymbol{k}$

For example, if $\boldsymbol{IY_n}$ = [1,20,12,11,2,14,15,6,18,5,19,4,13,8,3,7,16,9,10,17], then a fragment of SST is presented in Table 6.

Algorithm 2. Proposed Secret Output Table Generation Algorithm

Input: The third dimension of the 3D logistic map ($\boldsymbol{IZ_n}$), amino acids symbols ($\boldsymbol{AAS}$)
Output: $\boldsymbol{SOT}$ // $\boldsymbol{SOT}$ is a 20 × 20 cell matrix
Step1. $\boldsymbol{Sh_{AA}} = \boldsymbol{AAS}\ (\boldsymbol{IZ_n}\ (\mathbf{1:20}))$ // Shuffling of ($\boldsymbol{AAS}$) based on the third dimension of the 3D logistic map
Step2. $\boldsymbol{SOT}$ = cell (20, 20) // Initialize a 20 × 20 empty matrix
Step3. Performing a left shift rotation and filling the SOT matrix:
for $\boldsymbol{j}$ from **1** to **20** do
$\boldsymbol{R_V} = \boldsymbol{Left_{Shift}}\ (\boldsymbol{Sh_{AA}}\ , -(\boldsymbol{IZ_n}\ [\boldsymbol{j}]))$
$\boldsymbol{SOT}\ [\boldsymbol{j}\ , :] = \boldsymbol{R_V}$
end $\boldsymbol{j}$

For example, if $\boldsymbol{IZ_n}$ = [20,10,9,12,3,11,14,19,17,6,8,16,1,7,18,2,13,4,15,5] and $\boldsymbol{AAS}$ = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}, then a section of **SOT** is presented in Table 7.

3.5 Generating of round keys

In the proposed KE-DMM3DLMPS algorithm, we create a mealy machine that consists of 20 states. Each state has 20 bidirectional transitions, which are linked to other states and the state itself, and these transitions are symbols for the amino acids of the protein, as discussed in Section 3.4. The transitions between these states are controlled by the secret key. Before starting the key expansion process, as shown in Algorithm 3, the proposed algorithm distributes a copy of the master secret key (MSK) to all 20 states, such that each state holds a copy of MSK, as shown in Step 1. The master secret key is in the form of a protein sequence, as explained in Section 3.1. Each state, in turn, subsequently carries out two logical processes on its secret key, namely: AA-XOR and amino acid-left shift rotation (AA-LSR), and these two logical processes depend on the addresses of the twenty amino acids (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y). In Step 2, the user must select the starting state ($Actv_{st}$) from which they wish to start. After that, the algorithm picks up the first amino acid from the secret key found in the starting state ($Actv_{aa}$), as explained in step 3. In step 4, the proposed key expansion algorithm starts by performing an AA-XOR process between the $Actv_{aa}$ and all the amino acids found in the key of this state ($SK_{st}\{Actv_{st}\}$). After completing the AA-XOR process, we apply the AA-LSR process to all the bases in that key to eliminate any possible redundancy between the round keys. For each amino acid base, there is a fixed number. For A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y, the fixed numbers are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20, respectively. The number of left-shift rotations is determined by multiplying the fixed number with the state number. The updated secret key, which is the result of the AA-LSR process, becomes the secret key in this state and serves as the first-round key, controlling the KE-DMM3DLMPS, as demonstrated in the example below.

For the second-round key, the KE-DMM3DLMPS algorithm examines the second amino acid in the updated secret key in the starting state, known as the Nxt_{aa}, to determine the new state to move to. The mealy machine (MM) takes the SST, SOT, Nxt_{aa}, and the $Actv_{st}$ as input to determine the output of the second amino acid (O_{aa}) and the new state (N_{st}). The KE-DMM3DLMPS algorithm then does an AA-XOR operation on the output of the second amino acid and all the amino acids of the key in the new state. After completing the AA-XOR process, the proposed algorithm applies the AA-LSR process to all the bases of the XORing key. The updated secret key, which is the result of the AA-LSR process, becomes the secret key in this state and serves as the second-round key. The proposed algorithm will continue with this procedure based on the size of the entered secret key, such that in each state there are N protein base updates. The number of rounds is calculated according to Eq. (10) because each base needs 4 bits to be represented in binary.

$$Number\ of\ rounds = \frac{key\ length\ in\ bit}{4} \tag{10}$$

For example, if the size of the secret key is 256 bits, then the KE-DMM3DLMPS algorithm iterates 64 times to generate 64 different round keys, each of them with a length of 256 bits, and this is considered to be one of the main advantages of our proposed algorithm.

Algorithm 3. Proposed KE-DMM3DLMPS Algorithm

Input: Master Secret Key ($\boldsymbol{MSK}$), $\boldsymbol{SST}$, $\boldsymbol{SOT}$, and $\boldsymbol{AABER}$
Output: KA // Keys Array, which stores multiple round keys
Step1. $SK_{st} = Distribute_{keys}$ ($\boldsymbol{MSK}$)
Step2. $Actv_{st} = Starting_{state}$
Step3. $Actv_{aa} = SK_{st}\{Actv_{st}\}$ (1)
Step4.
for i_{rounds} from **1** to **length ($\boldsymbol{MSK}$)** do
$\boldsymbol{X_k} = Amino\_Acid_{XORing}\ (Actv_{aa}\ ,\ SK_{st}\{Actv_{st}\})$
$\boldsymbol{R_k} = Amino\_Acid_{LSR}\ (X_k\ ,\ Actv_{st})$
$\boldsymbol{SK_{st}}\{Actv_{st}\} = R_k$ // Update the secret key
$\boldsymbol{KA}\ \{i_{rounds}\} = SK_{st}\{Actv_{st}\}$// Store the keys
If (i_{rounds} + 1) <= length (MSK)
$Nxt_{aa} = SK_{st}\{Actv_{st}\}\ (i_{rounds} + 1)$
$[O_{aa}\ ,\ N_{st}] = MM\ (SST, SOT, Nxt_{aa}\ ,\ Actv_{st})$
$Actv_{aa} = O_{aa}$
$Actv_{st} = N_{st}$
else
break
end if
end i_{rounds}

To illustrate the process of key expansion step by step, we will take the following example: Given the master secret key (MSK), SST, SOT, and AABER as Table 5.

- MSK = 'KRMATY' (24 bits)
- SST, and SOT: the same in Table 6 and 7, respectively

Step 1: States 1 to 20 have a copy of the secret key: MSK = 'KRMATY'.
Step 2: $Actv_{st}$ = 8.
Step 3: $Actv_{aa}$ = 'K'.
Step 4:

Round 1:

- X_k = ('KKKKKK' **XOR** 'KRMATY') = 'CWSHHI'
- R_k = 'HICWSH'
- SK_{st}{8} = 'HICWSH' // Update the secret key
- KA {1} = 'HICWSH' // Store the key in the array
- Nxt_{aa} = ' I '
- ['Q', 8] = MM (SST, SOT, 'I', 8)
- $Actv_{aa}$ = 'Q'
- $Actv_{st}$ = 8

Round 2:

- X_k = ('QQQQQQ' **XOR** 'HICWSH') = 'YAQLGY'
- R_k = 'GYYAQL'
- SK_{st}{8} = 'GYYAQL'
- KA {2} = 'GYYAQL'
- Nxt_{aa} = 'Y'
- ['T',14] = MM (SST, SOT, 'Y', 8)
- $Actv_{aa}$ = 'T'
- $Actv_{st}$ = 14

Round 3:

- X_k = ('TTTTTT' **XOR** 'KRMATY') = 'HMRCCN'
- R_k = 'RCCNHM'
- SK_{st} {14} = 'RCCNHM'
- KA {3} = 'RCCNHM'
- Nxt_{aa} = 'N'
- ['F',17] = MM (SST, SOT, 'N', 14)
- $Actv_{aa}$ = 'F'
- $Actv_{st}$ = 17

Round 4:

- X_k = ('FFFFFF' **XOR** 'KRMATY') = 'LQIDDS'
- R_k = 'LQIDDS'
- SK_{st}{17} = 'LQIDDS'
- KA {4} = 'LQIDDS'
- Nxt_{aa} = 'D'
- ['H',10] = MM (SST, SOT, 'D', 17)
- $Actv_{aa}$ = 'H'
- $Actv_{st}$ = 10

Round 5:

- X_k = ('HHHHHH' **XOR** 'KRMATY') = 'ASWEKQ'
- R_k = 'ASWEKQ'
- SK_{st} {10} ='ASWEKQ'
- KA {5} = 'ASWEKQ'
- Nxt_{aa} = 'Q'
- ['F', 10] = MM (SST, SOT, 'Q', 10)
- $Actv_{aa}$ = 'F'
- $Actv_{st}$ = 10

Round 6:

- X_k = ('FFFFFF' **XOR** ' ASWEKQ') = 'DYNLLR'
- R_k = 'NLLRDY'
- SK_{st} {10} = 'NLLRDY'
- KA {6} = 'NLLRDY'

Therefore, the six round keys that are generated from the proposed key expansion algorithm are: KA = {'HICWSH', 'GYYAQL', 'RCCNHM', 'LQIDDS', 'ASWEKQ', 'NLLRDY'}. The larger the master secret key, the more round keys there are, and the greater the complexity between those round keys.

4. Assessment of experimental results for the suggested method

The suggested KE-DMM3DLMPS algorithm is tested in this section using Flexibility, the NIST Statistical Test Suite, Histogram Analysis, Key Space Analysis, Correlation Coefficient, Hamming Distance, Number of Bit Change Rate, Initial Key Sensitivity, Confusion and Diffusion, and Differential Attack. The aim of this test is to demonstrate the strength and security of the proposed key expansion algorithm.

4.1 Flexibility

The proposed KE-DMM3DLMPS algorithm is highly flexible: (1) it can produce keys of different lengths that are divisible by 8, such as 8, 16, 24, ..., 128, 136, ..., 1024 bits, or longer as desired by the user. (2) According to point (1), it can generate a different number of round keys by specifying the input size for the master secret key as described in Table 8. The user can also specify the number of sub-keys he needs for encryption.

4.2 NIST SP 800-22 test suite

The National Institute of Standards and Technology [37] proposed the NIST SP 800-22 test suite as a standard test to measure the randomness of key expansion algorithms and any cryptographic system associated with random numbers. It consists of several tests to discover the different properties of

Table 8. The relationship between the key size and the number of rounds

Key Length	Number of rounds
64-bit	16
128-bit	32
256-bit	64
512-bit	128
1024-bit	256

Table 9. NIST test suite results of the proposed KE-DMM3DLMPS algorithm

Statistical Test	P-value	Result
Frequency	0.875291	Pass
Block Frequency	0.654822	Pass
Cumulative Sums	0.804396	Pass
Runs	0.447751	Pass
Longest Runs of Ones	0.571849	Pass
Rank	0.436898	Pass
Spectral DFT	0.483807	Pass
Non-Overlapping Templates	0.460300	Pass
Overlapping Templates	0.999988	Pass
Universal	0.996711	Pass
Approximate Entropy	0.753984	Pass
Random Excursions	0.341179	Pass
Random Excursions Variant	0.675457	Pass
Serial	0.595208	Pass
Linear Complexity	0.861094	Pass

random sequences. Some of the NIST test suite should be applied to sequences of length greater than or equal to one million bits. For small sequences, the NIST test suite gives misleading results. In order to obtain enough bits to perform all the NIST test suite properly, a master secret key with a length of 256 bits (64 amino acids) is used, and then the proposed KE-DMM3DLMPS algorithm is repeated 150 times to generate 9601 rounds with a total length of 2457856 bits.

The obtained round keys must be converted into binary form using Table 5, and these binary sequences are considered input to the NIST test suite, which consists of 15 tests. Each test has a p-value that compares with a fixed level of significance (σ), where the value of the σ is to be at least 0.01. If the p-value obtained from each test is greater than σ, then the test is successful; otherwise, the test is a failure. That is, the p-value reflects the result of the test. A larger p-value indicates a higher level of randomness in the tested sequence. Based on the results obtained and shown in Table 9, the proposed KE-DMM3DLMPS algorithm passed all tests successfully with no obvious statistical defects. Which means that the generated round keys are very random and complex.

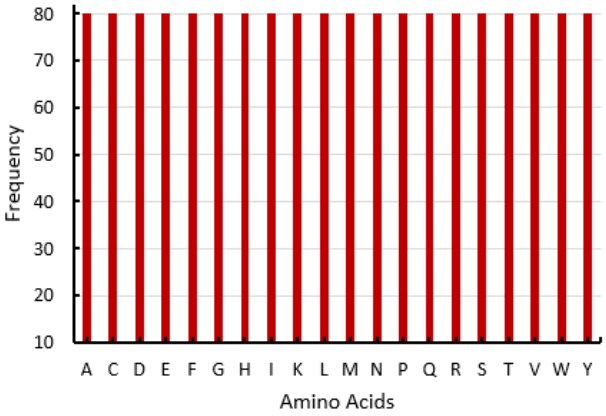

Figure. 4 Histogram of Amino Acid for 160 Bits

4.3 Histogram analysis (Statistical attacks)

The histogram is a graphical representation that illustrates the frequency distribution of the amino acids for several round keys in the propoesd algorithm. The uniform distribution of amino acids for several round keys makes the algorithm resistant to statistical attacks, making it very difficult for an attacker to know any information about the secret key. The high frequency of some amino acids leads to the disclosure of information. So, as a good proposed key expansion system, the histogram of amino acids should be evenly distributed. In this study, a histogram of the round keys is displayed by counting the number of each amino acid (count $_{aa}$), where the optimal value for each amino acid is calculated as shown in Eq. (11).

$$Optimal_{value} = L_{SK} \times \frac{No._{Rounds}}{20} \quad (11)$$

Where L_{SK} is the number of amino acids in the secret key and $No._{Rounds}$ is the number of generated rounds.

We take a master secret key with a length of 160 bits (i.e., 40 amino acids), as follows: key = 'TGQWISAHVCTYEGRYLKNDMEHRNCFVPKAFMLPIDQWS'. The optimal value for this key should be 80 to ensure a uniform distribution of amino acids for 40 rounds. Fig. 4 displays a histogram of the amino acids in the above-mentioned secret key for 40 rounds; therefore, we conclude from this figure that each amino acid has an average count $_{aa}$ equal to 80, which indicates that the amino acids of the round keys are completely evenly

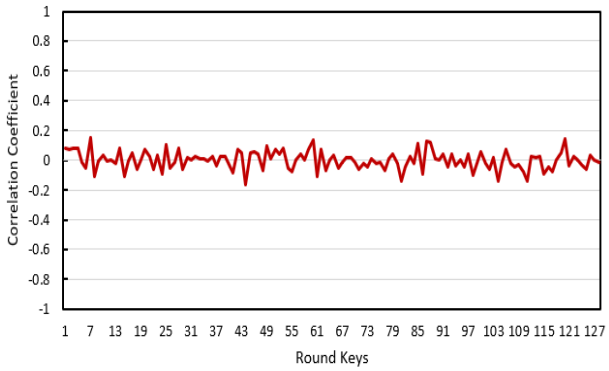

Figure. 5 Correlation coefficient for 128 round keys (256-bits secret key)

distributed, and therefore the proposed KE-DMM3DLMPS algorithm is efficiently resistant to statistical analysis attacks.

4.4 Key space analysis

From the cryptanalysis point of view, the key space should be greater than 2^{100} in order to render brute-force attacks futile [27]. The key space of the suggested KE-DMM3DLMPS algorithm includes: (1) the initial values of the state variables (x_0, y_0, and z_0) and the control parameters (α, β, and σ) of the 3D logistic map that is used to generate the output and the state tables. When the computer's precision is about 10^{-15}, then the key space of the initial values is $(10^{15})^6 = 10^{90} \approx 2^{299}$. (2) the amino acid binary encoding rules, where there are, in total, 16! $\approx 2^{44}$ different rule to map the four bits to the amino acids. (3) the master secret key, which is considered an input to the Mealy machine, is in the form of a protein sequence and of different sizes according to the user's desire. Assuming that the length of the master secret key is 128 bits, the key space of the master secret key is 2^{128}. (4) the starting state of the mealy machine, because there are 20 states, so the key space is 20. Thus, the proposed key expansion method has a key space of a total of 20×2^{471}, indicating a significant level of security against brute force assaults.

4.5 Correlation coefficient assessment

The correlation coefficient is a statistical metric that exhibits the linear relationship between the master secret key and the generated round keys. Thus, an important feature of the key expansion algorithm is to break this linear relationship and make the algorithm resist all kinds of statistical attacks. If the coefficient value is 0 or near zero, there is no linear relationship between the master secret key and the generated round key. To check the KE-DMM3DLMPS algorithm's correlation coefficient, we use a 256-bit master secret key, shown below, and iterate KE-DMM3DLMPS twice to generate 128 round keys. We then change these keys from protein sequences to binary sequences.

Key =
'EKELKIFFAEVKTRTYLGTLIKSTDKDLLVLS
DPSIKEMEEIVVIGAAKAKGDHDWVDIDLYDD
'

The correlation coefficient (CC) between the master secret key and each round key can be calculated through Eqs. (12)-(15):

$$M(x) = \frac{1}{LK} \sum_{j=1}^{LK} x_j \tag{12}$$

$$V(x) = \frac{1}{LK} \sum_{j=1}^{LK} (x_j - M(x))^2 \tag{13}$$

$$C(x,y) = \frac{1}{LK} \sum_{j=1}^{LK} ((x_j - M(x)) \times (y_j - M(y))) \tag{14}$$

$$CC_{xy} = \frac{C(x,y)}{\sqrt{V(x)} \times \sqrt{V(y)}}, \quad -1 \le CC_{xy} \le 1 \tag{15}$$

where x and y are the binary bits of the master secret key and each round key, respectively; $M(x)$ is the mean; $V(x)$ is the variance; $C(x,y)$ is the covariance; and LK is the length of the secret key in bits.

From the correlation coefficient results in Fig. 5, we find there is no correlation between the master secret key and the generated round keys because all values of the correlation coefficients are near zero.

4.6 Hamming distance (HD)

Table 10 lists the generated round keys of the proposed KE-DMM3DLMPS algorithm and their hamming distances to the master secret key (MSK) with a size of 128 bits, concluding that the HD for each round key and the average hamming distances (AHDs) of all round keys are close to the optimal value. To further assess the effectiveness of the suggested key expansion technique, we produced 3,840 round keys with lengths of 128-bit, 256-bit, 512-bit, and 1024-bit. The AHDs between each master secret key and its corresponding round keys, as displayed in Table 11, indicate that the AHDs closely approximate the optimal values of 64, 128, 256, and 512. Therefore, we could deduce that the suggested key expansion method meets the principle of irreversibility, i.e., no sub-key can deduce the master secret key.

Table 10. The 128-bit round keys and their hamming distance

Round	MSK and Round Keys	HD	NBCR
MSK	'AMLRFWGQVSCQLIFRESLWPHNFQCGHMYRC'		
1	'PTQRMVQHALWCMHPENFLQVTNISWVYIHWL'	72	56.2500
2	'GNCQKWGRFTICEYLTHPQYSHGQCRLEDWYE'	53	41.4063
3	'CVASMQIYHVMLPERISAFEDTYANDMYILFS'	66	51.5625
4	'DGLKSHPILQAVNSGDYVTWHDMTLHSQYGEI'	67	52.3438
5	'WKNAFYMKVFIMTSNWPDKWIGTMLDISQAET'	63	49.2188
6	'ICGDWVLRSHEVTYFHPTGFRDESQCHSGNRF'	59	46.0938
7	'IRTHAQTVSDNMAVILPCDTQRPKFNQEKVND'	66	51.5625
8	'WLQTSGKDVWCFMDYHLFIYSLWGMCATFCSR'	68	53.1250
9	'LYDCTMYHCWSNGPETGHAWCKEHSVLQWGTN'	65	50.7813
10	'DALPRYTAQRNTMCLDWKADNEMTGKNCSPIM'	60	46.8750
11	'GMWLCGPESAFHMACVPGRHCENYTPAMSYLQ'	62	48.4375
12	'SFTWGKAYHRVKNMIRDNTIYWVHQFRHTCYI'	56	43.7500
13	'HCGVIPKCADFPRANFVMKIWEPINLVFYENM'	57	44.5313
14	'FVYTMHDLGNELYPMTWNYHCRKMLEDRVITE'	58	45.3125
15	'FQVRTFLVIGWYPKRYLCIVAKLGMENIYRWE'	62	48.4375
16	'GYDWFVIKDTSQPFYGEQRNVGARDVFTEYLK'	63	49.2188
17	'FRGVWATLMYHLGNWVDYGACPKWLHTPRSVH'	62	48.4375
18	'MLSNTMWSHCIPKYNPWEHSGYWCRFDHPNIF'	57	44.5313
19	'WRHTSGMYFINWFTLGSCNTMPEAGFWYEHDS'	70	54.6875
20	'DQWIRQNYHPVINECFLHQRDFAMPRSANPHE'	66	51.5625
21	'ANCIPAECMWQKHYIKERMCFYEWVLDMKIQL'	65	50.7813
22	'YPLDEIARNPEVMHWADLGHQKRLCQERAVGD'	62	48.4375
23	'MNYKPGMTFEPWTLEYQGKVRPKLIYEARLQC'	60	46.8750
24	'VLHNQWKLYIEWMYFENCKQPAWQFSNERAFC'	66	51.5625
25	'INDMENHTFLWMHCKYSFNEIYVPLEAVHLFC'	64	50
26	'KGMWSDKRPHTMNCQHFEWCVFKWMRQNADCN'	75	58.5938
27	'CVIQMYRVWMERFKICSHVCELFRNHEKDQGF'	66	51.5625
28	'FHNDTRFYWGQNMCAGEVDCKEFDNYAMSRCM'	70	54.6875
29	'ADQFAGNPKSQGEYHIPAFLHMCKFRMGKPEL'	66	51.5625
30	'PRKENQVTCHVRMEKFCRNYWSEVHQWPAKHG'	76	59.3750
31	'NWPSCEAWQCKARFPNIGWNKYRADGKFTSLR'	69	53.9063
32	'FVGQEYPQLMFCWYLVRHNFQPGHSACPISLC'	52	40.6250
	Average	63.84	49.8779

Table 11. The average hamming distances and NBCRs of 3840 round keys

Length of MSK	Average HDs	Average NBCRs (%)
128-bit	63.9789	49.9591
256-bit	128.0799	50.0312
512-bit	255.9901	49.9981
1024-bit	512.1607	50.0157

4.7 The number of bit change rate (NBCR)

The number of bit change rate can be used to measure how sensitive the initial key is to the key expansion algorithm [38]. The NBCR of two sequences of keys, K1 and K2, can be determined using Eq. (16):

$$\text{NBCR} = \frac{\text{HD (K1,K2)}}{\text{Len}} \times 100\ \% \tag{16}$$

where Len is the length of K1 or K2, and HD (K1, K2) calculates the Hamming distance between K1 and K2. The optimal NBCR value is 50%, which can be obtained when two keys are fully independent. Tables 10 and 11 reveal that all of the average NBCRs between the round keys and the master secret key are near the optimal value of 50%. This demonstrates the independence between the master secret key and the round keys.

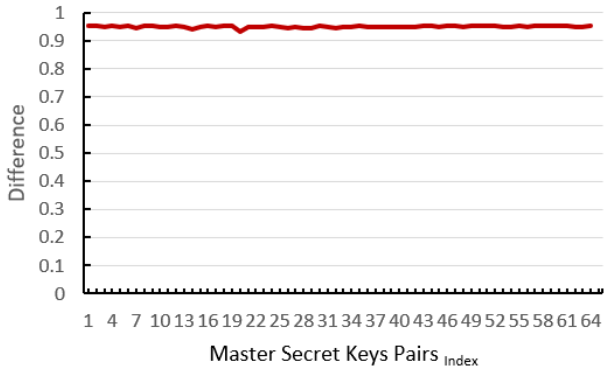

Figure. 6 Key Sensitivity Analysis of KE-DMM3DLMPS

Table 12. The Results of the Initial Key Sensitivity

Length of MSK	Average HDs	Average NBCRs (%)
128-bit	64.1464	50.1144
256-bit	128.1120	50.0438
512-bit	255.9828	49.9966
1024-bit	512.1748	50.0171

4.8 Sensitivity of the initial key

The most important feature of any secure key expansion algorithm is key sensitivity, as it requires high sensitivity of the secret key. This means that a very small change between the two master secret keys can cause a significant difference between their two round keys.

To test the key sensitivity of the proposed KE-DMM3DLMPS algorithm, a master secret key with a length of 256 bits is determined, and then one bit is randomly chosen from this key and changed for the purpose of creating two master secret keys that differ in only one bit, as follows, where the change in bit is in blue color (M = 1000, Q = 1001):

Key₁ =
VLNQYQNKSAPHAMTSRCERVRDHWFMGYC
GSMY**M**WTDKWNATFEQTCKIGHPIDFIGPEL
VAC

Key₂ =
VLNQYQNKSAPHAMTSRCERVRDHWFMGYC
GSMY**Q**WTDKWNATFEQTCKIGHPIDFIGPELV
AC

After that, each of the two master secret keys above is entered into the KE-DMM3DLMPS algorithm in order to obtain their respective round keys. To calculate the difference between the round keys generated for each of the two keys above, we calculate the difference (pairwise distance) between the amino acids for each sub-key pair ($subkey_{i\,(key1)}$ and $subkey_{i\,(key2)}$). If the amino acids are completely different, then the difference value (Ω) is equal to one, and if they are similar, then the Ω value is equal to zero. After calculating the differences between all sub-key pairs of the two keys above for 64 round keys, it was found that the average of all Ω values is equal to 0.9514. This value is close to one, which indicates high sensitivity to changing one bit of the master secret key. We conducted the experiment 64 times, utilizing various pairs of master secret keys, and Fig. 6 displays the results. We conclude that the proposed method is highly sensitive because all values are close to one.

To further evaluate the initial key sensitivity of the proposed algorithm, we change one bit from the 128-bit, 256-bit, 512-bit, and 1024-bit master secret keys (MSK_i) to obtain four new master secret keys ($MSK_i^{'}$), and generate 3840 round keys for each of them (MSK_i and $MSK_i^{'}$), then calculate the Hamming distance and the NBCR between each pair of round keys (i.e., between $round_j^{128}$ and $round_j^{128'}$ and so on). From what is shown in Table 12, we can deduce that the average NBCRs and HDs are getting closer to their optimal values. This implies that a change of one bit from the master secret key yields good results and that the suggested algorithm is very sensitive to the master secret key, satisfying the strict avalanche effect (SAC).

4.9 Analysis of confusion and diffusion

Confusion and diffusion are two significant evaluation criteria in the key expansion algorithm. Confusion aims to make the relationship between the master secret key and the round keys as complicated as possible, while diffusion ensures that even a slight change in the master secret key will have a widespread effect on all bits of the round key [7].

A perfect outcome is that a single-bit change in the master secret key will lead to a 50% alteration in the bits of the round key [38]. In this context, a specific master secret key(MSK) is chosen, and then only one bit of that key is changed to obtain a new master secret key ($MSK^{'}$) with a difference of only one bit, and we generate 3840 round keys for each. Each round key generated from the MSK is then compared with the corresponding round key generated from the $MSK^{'}$ bit by bit to obtain the number of flipped bits (NFB). From Fig. 7, it is apparent that the NFBs are centered around 64, 128, 256, and 512 bits. This suggests that the proposed key expansion technique is very sensitive to confusion and diffusion.

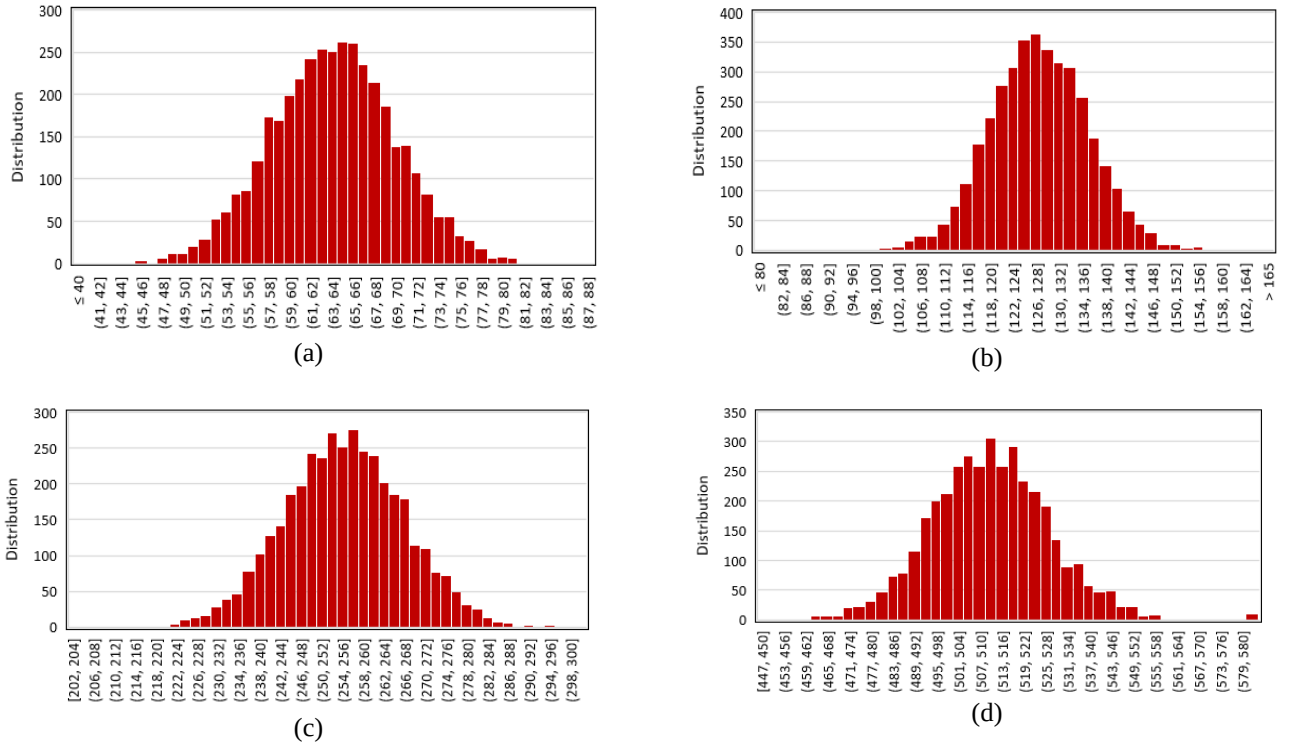

Figure. 7 Number of different bits distribution: (a)128-bit master secret key, (b)256-bit master secret key, (c)512-bit master secret key, and (d)1024-bit master secret key

Table 13. Comparison of the NIST test suite results

Statistical Test	**P-value** in this research	**P-value** in Ref. [12]	**P-value** in Ref. [13]	**P-value** in Ref. [14]	**P-value** in Ref. [17]
Frequency	0.875291	0.382115	0.576150	0.739918	0.7542
Block Frequency	0.654822	0.082010	0.859684	0.179120	0.6523
Cumulative Sums	0.804396	0.650549	0.586368	0.534146	0.8564
Runs	0.447751	0.197981	0.695002	0.350485	0.3265
Longest Runs of Ones	0.571849	0.498313	0.296950	0.179120	0.1542
Rank	0.436898	0.363593	0.693720	0.035174	0.6589
Spectral DFT	0.483807	0.015816	0.123812	0.213309	0.5421
Non-Overlapping Templates	0.460300	0.163643	Not defined	0.122325	0.3265
Overlapping Templates	0.999988	0.060112	Not defined	0.430102	0.6953
Universal	0.996711	0.401192	Not defined	0.122325	0.4526
Approximate Entropy	0.753984	0.076131	1.000000	0.350485	0.3574
Random Excursions	0.341179	0.095397	Not defined	0.911413	0.1594
Random Excursions Variant	0.675457	0.524892	Not defined	0.534146	0.6532
Serial	0.595208	0.843974	0.498531	0.035174	0.3254
Linear Complexity	0.861094	0.001046	0.919689	0.739918	0.9658

4.10 Differential attack analysis

Differential attacks are concerned with attacking the master secret key, where attackers try to find out whether a particular modification to the master key might lead to a specific difference in the key output of each round [12]. As demonstrated in Sections 4.7 and 4.8 through experimentation, a 1-bit change to the master secret key will get NBCR very close to the optimal value of 50%. Hence, we can deduce that our suggested technique is capable of effectively blocking differential attacks.

Table 14. Comparison of the key space

Research	Initial key size	key space
Ref. [13]	128-bits	2^{128}
Ref. [14]	64-bit	2^{16}
Ref. [15]	128-bits	2^{128}
Ref. [16]	128-bits	2^{64}
Ref. [17]	128-bits	2^{128}
Ref. [18]	128-bits	$2^{128} \times 10^{45}$
KE-DMM3DLMPS	128-bits	20×2^{471}

5. Comparison results

After evaluating the performance of the proposed KE-DMM3DLMPS algorithm through several different tests and proving the strength and security of our proposed algorithm, in this section an evaluation between KE-DMM3DLMPS and some other key expansion algorithms is presented. The comparison is based on the NIST test suite, histogram analysis, and key space analysis. Table 13 shows the NIST test suite results of the compared key expansion algorithms. The KE-DMM3DLMPS passed all tests successfully and with almost the best p-values compared to the results of the previous studies that were evaluated with the NIST test suite. The KE-DMM3DLMPS achieved a higher P-value, which means that the generated round keys are very random.

Furthermore, no one of the previous studies used histogram analysis except Ref. [17], but the result is not in an ideal uniform distribution in spite of the fact that it used DNA sequences while the proposed KE-DMM3DLMPS algorithm depended on protein sequences. As displayed previously in Fig. 4, the amino acids of the round keys are completely evenly distributed, thus the histogram of the proposed KE-DMM3DLMPS algorithm is in an ideal uniform distribution. This ensures strict resistance against statistical analysis attacks.

Finally, Table 14 shows the comparison of the proposed KE-DMM3DLMPS algorithm and the previous studies based on the key space, indicating that the suggested algorithm has a larger key space and a significant level of security against brute force assaults.

6. Conclusions

A novel and secure key expansion method was proposed based on Dynamic Mealy Machine, 3D Logistic Map, and Protein Sequence. The proposed KE-DMM3DLMPS algorithm was introduced to solve the problem of sub-key randomness and some related key attacks. Through the results of analysis and comparisons of the proposed method with some related studies for the purpose of evaluating the performance of the proposed method, we can conclude the following points:

- The ability of KE-DMM3DLMPS to generate a different number of round keys with the desired lengths.
- The generated round keys have a higher degree of randomness by passing all of the NIST tests.
- A uniform and ideal distribution of the amino acids present in each round key, which ensures strict resistance against statistical analysis attacks.
- A higher key space of up to 20×2^{471}, indicating a significant level of security against brute force assaults.
- The ability of KE-DMM3DLMPS to break the linear relationship between the master secret key and the generated round keys is evidenced by the fact that all correlation coefficients exhibit values in close proximity to zero.
- The KE-DMM3DLMPS satisfies irreversibility and independence by having ideal values of average hamming distances and NBCRs, respectively.
- The proposed key expansion algorithm also manages to maintain its sensitivity to a one-bit change in the master secret key at 95 percent, satisfying the strict avalanche effect (SAC).
- The KE-DMM3DLMPS ensures Confusion and Diffusion.
- It is capable of effectively blocking differential attacks.

Conflicts of Interest

The authors assert that they have no conflict of interest.

Author Contributions

Radhwan Jawad Kadhim made significant contributions to the conceptualization, methodology, program development, formal analysis, validation, resource management, data curation, and original draft preparation. Hussein K. Khafaji Provided oversight, reviewed, and made corrections to the work.

References

[1] M. Agrawal and P. Mishra, "A comparative survey on symmetric key encryption techniques", *Intern. J. Comput. Sci. Eng.*, Vol. 4, No. 5, pp. 877-882, 2012.

[2] M. Chakraborty and M. Singh, *Introduction to*

Network Security Technologies. *The" Essence" of Network Security: An End-to-End Panorama*, 2021. doi: 10.1007/978-981-15-9317-8_1.

[3] S. Jamel, T. Herawan, and M. M. Deris, "A cryptographic algorithm based on hybrid cubes", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 6019 LNCS, No. PART 4, pp. 175-187, 2010, doi: 10.1007/978-3-642-12189-0_16.

[4] M. Stamp, *Information Security: Principles and Practice.* John Wiley & Sons, 2011.

[5] L. R. Knudsen and M. Robshaw, *The block cipher companion*, Springer Science & Business Media, 2011.

[6] J. N. B. Salameh, "A new technique for sub-key generation in block ciphers", *World Appl. Sci. J.*, Vol. 19, No. 11, pp. 1630-1639, 2012, doi: 10.5829/idosi.wasj.2012.19.11.1871.

[7] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, "Cryptographic Strength Evaluation of Key Schedule Algorithms", *Secur. Commun. Networks*, Vol. 2020, 2020, doi: 10.1155/2020/3189601.

[8] G. T. Cayabyab, A. M. Sison, and R. P. Medina, "A secure key scheduling operation for international data encryption algorithm using serpent key schedule operation", *ACM Int. Conf. Proceeding Ser.*, No. 2021, pp. 63-67, 2019, doi: 10.1145/3366650.3366659.

[9] M. Al-Muhammed, "A novel key expansion technique using diffusion", *Comput. Fraud Secur.*, Vol. 2018, No. 3, pp. 12-20, 2018, doi: 10.1016/S1361-3723(18)30025-3.

[10] I. Sultan, M. Y. Lone, M. Nazish, and M. T. Banday, "A Secure Key Expansion Algorithm for Present", *IEEE Sens. J.*, Vol. 23, No. 20, pp. 25367-25376, 2023, doi: 10.1109/JSEN.2023.3267386.

[11] A. Dmukh, D. Trifonov, and A. Chookhno, "Modification of the key schedule of the 2-GOST block cipher and its implementation on FPGA", *J. Comput. Virol. Hacking Tech.*, Vol. 18, No. 1, pp. 49-59, 2022, doi: 10.1007/s11416-021-00406-x.

[12] Y. Harmouch and R. El Kouch, "The benefit of using chaos in key schedule algorithm", *J. Inf. Secur. Appl.*, Vol. 45, pp. 143-155, 2019, doi: 10.1016/j.jisa.2019.02.001.

[13] J. Wang, B. W. Pan, Q. R. Wang, and Q. Ding, "A chaotic key expansion algorithm based on genetic algorithm", *J. Inf. Hiding Multimed. Signal Process.*, Vol. 10, No. 2, pp. 289-299, 2019.

[14] A. Poojari and H. R. Nagesh, "FPGA implementation of random number generator using LFSR and scrambling algorithm for lightweight cryptography", *Int. J. Appl. Sci. Eng.*, Vol. 18, No. 6, pp. 1-9, 2021, doi: 10.6703/IJASE.202112_18(6).001.

[15] A. A. Zakaria, A. H. Azni, F. Ridzuan, N. H. Zakaria, and M. Daud, *Modifications of Key Schedule Algorithm on RECTANGLE Block Cipher*, Vol. 1347. Springer Singapore, 2021. doi: 10.1007/978-981-33-6835-4_13.

[16] S. G. Garba, A. A. Obiniyi, M. A. Ibrahim, and B. I. E. Ahmad, "On the Key Schedule of Lightweight Block Cipher", In: *Proc. of 5th Int. Conf. Inf. Technol. Educ. Dev. Chang. Narrat. Through Build. a Secur. Soc. with Disruptive Technol. ITED 2022*, pp. 1-6, 2022, doi: 10.1109/ITED56637.2022.10051257.

[17] M. Alawida, J. Sen Teh, and W. H. Alshoura, "A New Image Encryption Algorithm Based on DNA State Machine for UAV Data Encryption", *Drones*, Vol. 7, No. 1, 2023, doi: 10.3390/drones7010038.

[18] D. Xu and H. Liu, "A Strong Key Expansion Algorithm Based on Nondegenerate 2D Chaotic Map Over GF(2n)", *Int. J. Bifurc. Chaos*, Vol. 33, No. 15, 2023, doi: 10.1142/S0218127423501778.

[19] R. Jiang, X. Zhang, and M. Q. Zhang, *Basics of bioinformatics: Lecture notes of the graduate summer school on bioinformatics of China*, Vol. 9783642389. 2013. doi: 10.1007/978-3-642-38951-1.

[20] B. Alberts, *Molecular biology of the cell*, WW Norton & Company, 2017.

[21] E. Keedwell and A. Narayanan, "Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems", *Intell. Bioinforma. Appl. Artif. Intell. Tech. to Bioinforma. Probl.*, pp. 1-280, 2005, doi: 10.1002/0470015721.

[22] J. Pevsner, *Bioinformatics and Functional Genomics*, John Wiley & Sons, 2005. doi: 10.1002/047145916x.

[23] M. Morange, "The Central Dogma of molecular biology", *Resonance*, Vol. 14, No. 3, pp. 236-247, 2009, doi: 10.1007/s12045-009-0024-6.

[24] G. Edited, I. C. Gray, and M. R. Barnes, *Bioinformatics for geneticists*, John Wiley & Sons, 2003.

[25] P. N. Lone, D. singh, and U. H. Mir, "Image encryption using DNA coding and three-dimensional chaotic systems", *Multimed. Tools Appl.*, Vol. 81, No. 4, pp. 5669-5693, 2022, doi: 10.1007/s11042-021-11802-2.

[26] C. Liu and Q. Ding, "A Color Image Encryption Scheme Based on a Novel 3D Chaotic

Mapping", *Complexity*, Vol. 2020, 2020, doi: 10.1155/2020/3837209.

[27] X. Chai, Z. Gan, K. Yuan, Y. Chen, and X. Liu, "A novel image encryption scheme based on DNA sequence operations and chaotic systems", *Neural Comput. Appl.*, Vol. 31, No. 1, pp. 219-237, 2019, doi: 10.1007/s00521-017-2993-9.

[28] K. Singh and K. Kaur, "Image Encryption using Chaotic Maps and DNA Addition Operation and Noise Effects on it", *Int. J. Comput. Appl.*, Vol. 23, No. 6, pp. 17-24, 2011, doi: 10.5120/2892-3779.

[29] S. Patel, Bharath K P, and Rajesh Kumar M, "Symmetric keys image encryption and decryption using 3D chaotic maps with DNA encoding technique", *Multimed. Tools Appl.*, Vol. 79, No. 43-44, pp. 31739-31757, 2020, doi: 10.1007/s11042-020-09551-9.

[30] P. N. Khade and P. M. Narnaware, "3D Chaotic Functions for Image Encryption", *Int. J. Comput. Sci. Issues (IJCSI)*, Vol. 9, No. 3, pp. 323-328, 2012.

[31] P. Garapati and S. Musala, "Moore and Mealy Negative Edge detector A VHDL Example for Finite State Machine", In: *Proc. of 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020*, pp. 1159-1161, 2020, doi: 10.1109/ICCSP48568.2020.9182310.

[32] A. Bhowmik, S. Karforma, and J. Dey, "Symmetric key and artificial neural network with mealy machine: A neoteric model of cryptosystem for cloud security", *Mach. Learn. Tech. Anal. Cloud Secur.*, pp. 81-101, 2021, doi: 10.1002/9781119764113.ch5.

[33] B. B. Kodada and D. A. D'Mello, "Symmetric Key Cryptosystem based on Sequential State Machine", *IOP Conf. Ser. Mater. Sci. Eng.*, Vol. 1187, No. 1, p. 012026, 2021, doi: 10.1088/1757-899x/1187/1/012026.

[34] P. Pavithran, S. Mathew, S. Namasudra, and P. Lorenz, "A novel cryptosystem based on DNA cryptography and randomly generated mealy machine", *Comput. Secur.*, Vol. 104, p. 102160, 2021, doi: 10.1016/j.cose.2020.102160.

[35] P. Pavithran, S. Mathew, S. Namasudra, and A. Singh, "Enhancing randomness of the ciphertext generated by DNA-based cryptosystem and finite state machine", *Cluster Comput.*, Vol. 26, No. 2, pp. 1035-1051, 2023, doi: 10.1007/s10586-022-03653-9.

[36] R. J. Kadhim and H. K. Khafaji, "Unprecedented Security Analysis Results for a Novel Steganography Approach Based on Protein Sequences", *Int. J. Intell. Eng. Syst.*, Vol. 16, No. 2, pp. 464-476, 2023, doi: 10.22266/ijies2023.0430.37.

[37] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", *Nist Spec. Publ.*, Vol. 22, pp. 1/1--G/1, 2010.

[38] M. Zhao and H. Liu, "Construction of a Nondegenerate 2D Chaotic Map with Application to Irreversible Parallel Key Expansion Algorithm", *Int. J. Bifurc. Chaos*, Vol. 32, No. 6, 2022, doi: 10.1142/S021812742250081X.