International Journal of Innovative Computing, Information and Control
Volume **16**, Number **1**, February **2020**

 ISSN 1349-4198
pp. **331–348**

A NEW APPROACH TO GENERATE MULTI S-BOXES BASED ON RNA COMPUTING

Alaa Kadhim Farhan[1], Rasha Subhi Ali[2], Hassan Rashed Yassein[3]
Nadia Mohammed Ghanim Al-Saidi[4,*] and Ghassan Hameed Abdul-Majeed[5]

[1]Computer Science Department
[4]Department of Applied Sciences
University of Technology
Al Sina'a Street, Baghdad, Bagdad 10066, Iraq
[*]Corresponding author: 110030@uotechnology.edu.iq

[2]Department of Computer Techniques Engineering
AL-Nisour University College
Al-Harthiya-Al-Nisour Sq. Baghdad 10092, Iraq

[3]Department of Mathematics
College of Education
University of Al-Qadisiyah
Aj Jama'a District, Ad Diwaniyah, Al Qadisiyah 58001, Iraq

[5]College of Engineering
University of Baghdad
Al-Jadriya, Karrada, Baghdad 10071, Iraq

Received May 2019; revised September 2019

Abstract. *Many scientists have tried to design new security methods in the domains of cryptography and steganography that are inspired by biological techniques, such as Deoxyribose Nucleic Acid (DNA), Ribonucleic Acid (RNA), Messenger RNA (mRNA), and other bio-molecular methods. Encryption techniques are typically applied to preserving the confidentiality of data, and this work aims to increase the protection of data through modern cryptographic methods. Based on DNA computing, our approach increases confusion through the design of multiple S-boxes. First, a new mRNA-based S-box generates different S-boxes for each user, and a secret key recursively creates these S-boxes until obtaining the required number of S-boxes. This proposed design passes S-box test criteria effectively, including invertibility, balance, completeness, avalanche, and strict avalanche. When changing only one bit in the key used to generate the S-boxes, the results show the newly constructed S-box changes with approximately 99% differences from the original. With comparisons between our technique to other searches, we found it converges to an optimal solution faster in abstracted steps. This new method can be leveraged in block cipher algorithms, such as AES, DES, and Ghost.*
Keywords: Multi S-box, RNA, Bio-computing, DNA map rules

1. **Introduction.** Information security is an urgent need today with the continuing rapid development of information technologies, which are an indispensable part of life. Cryptography is defined as the science of protecting the privacy of information by making it unreadable for the unauthorized public from alteration, exploitation or loss, and ensuring it remains comprehensible only to the intended receiver [1].

Symmetric and asymmetric keys are the two primary approaches for cryptographic algorithms. The former is widely used and addresses privacy, integrity, and authentication

DOI: 10.24507/ijicic.16.01.331

in their design. The Advanced Encryption Standard (AES) is the modern symmetric key cipher based on a matrix power function known as S-box. As a non-linear operation, the latest version of S-box performs substitution and exhibits a good level of security, making AES a good candidate for meeting security constraint resources [2]. The previous design criteria of S-boxes were susceptible to recently invented attacks. Therefore, exploring new techniques remains an important challenge with potential and gains better performance. Several studies attempted to enhance its security by replacing the fixed S-box approach with the key-dependent S-box [3]. Additional ways for building good S-boxes for encryption exist and methods for evaluation are available based on criteria that should be passed, which are described in Section 5. Multiple methodologies were proposed during the past three decades to impact cryptographic protocols, such as chaos theory [4-8], fractals [9-11], quantum information [1], and methods based on molecular biological traits [12-18]. The unique characteristics of molecular biology have led researchers to leverage them for cryptography designs. For example, DNA cryptography represents a branch of bio-science that encompasses massive information and storage capacity based on the biology of DNA [19]. Recently, also inspired by biological techniques, new methods were adopted in the design of secure cryptosystems. Since the security of AES relies on the S-box, an enhanced design for the S-box will result in a highly secured AES system.

After Adleman's research in the field of DNA computing [20], many subsequent studies on DNA cryptography emerged with new literature proposing how to address its properties in a cryptographic system. This work especially covers the design and modification of the S-box, as described in the following. In 2008, Tran et al. [21] presented the Gray S-box for AES by constructing an added binary gray code transformation as a preprocessing step to the original AES S-box. Gray S-box also achieves the important cryptographic properties of AES S-box, including strict avalanche criterion, nonlinearity, and differential uniformity. Inspiration from DNA encryption schemes utilised biological alphabets to manipulate information employing the DNA sequence reaction was introduced in 2014 by Behnam et al. [22]. This approach autonomously created a copy of its threads as an extended encryption key. Information is converted from plaintext into several formats by mimicking the stages of protein formation from DNA sequences to generate an extended key based on the chemical properties and attributes used in the encryption mechanism. In 2015, Al-Wattar et al. [23] proposed a technique for developing a powerful $(8 * 8)$ S-box based on operations inspired by the biological DNA structure. The S-box criteria and NIST randomness tests were applied for testing their constructed S-box. In 2016, Kadhim et al. [24] proposed another new S-box based on DNA computing and mathematical operations. In 2017, Mazhar and Waleej [13] presented a cipher solution with a new symmetric key generation model based on DNA strands, nucleotides, codon base pair rules, mutation, and DNA to mRNA conversion. This solution emphasises biological processes by simulating random changes found in DNA for the key generation model. Recently, DNA computing was combined with chaos theory to construct secure encryption methods [25-27].

The physical features of DNA, such as massive parallelism, a capability to store an enormous amount of data, and its ultra-power consumption, make it intriguing for use in the design of secure cryptographic systems. Biological traits are represented as algebraic operations, such as DNA bases, DNA addition, and a DNA XOR operation.

In this work, we design secure substitution functions based on the S-box. Our proposed technique is based on RNA to construct a multi S-box for each user recursively. Each S-box is generated from the previous one with the initial S-box based on the secret key. The mRNA, in cooperation with a secure equation, creates a new S-box, and the S-box inverse. The created S-box obtains an approximate accuracy of 99% suggesting significant efficiency with our method.

The remainder of the paper is organised as follows. Section 2 discusses the biomolecular technique, DNA translation, and transcription process. The proposed method is presented in Section 3 with details about bio-computing operations and the proposed algorithms that generate the S-boxes and their inverses. Section 4 discusses the results followed by a performance analysis in Section 5 and concluded remarks in Section 6.

2. **Bimolecular Technique.** DNA is the genetic material within living organisms that carries the genetic options from one generation to its offspring. Living organisms have a distinctive DNA, which consists of many compound units known as nucleotides. Every nucleotide is comprised the three subsequent units, deoxyribose sugar, a phosphate group, and a nitrogenous base of Adenine (A), Guanine (G), Thymine (T), and Cytosine (C).

Individual cells maintain a full set of DNA, which are polymers made from monomers named deoxy ribonucleotides [5]. Discovered by James Watson in 1953, the DNA molecule consists of two single strands that form a double helix structure. The backbone of each helix strand alternates between a sugar and phosphate group [3]. The sequence of nucleotides determines the code for each gene and represents vital information, such as skin color, weight, nose shape, eye color, and hair color. Watson and Crick devised the complementary DNA structure, which is inherently employed today in DNA computing [19]. The field of cryptography takes advantage of DNA computing's massive parallelism, huge storage, and ultralow power consumption of DNA molecules [3]. The unique property of DNA encoding is also used in computations to improve security and mitigate the current flaws.

DNA-based biomolecular cryptography design is a library of one-time pads assembled secretly in the arrangement of DNA strands used to encrypt or decrypt short messages. Consisting of two chains twisted to form a double-stranded helix, simulated A and T are bonded together while C and G are in the opposite chain. Just as in information is transmitted to new cells during development and growth using complimentary pairing, this approach enables data to be replicated autonomously using a synthesising template. Simulated DNA strands are mapped to numbers, alphabetical letters, and other attributes, and are widely utilised for encoding and decoding as well as digital storing of data. Information encryption using DNA sequences is now considered the preferred approach, especially for those applications requiring a robust data encryption scheme to limit unauthorised access [22].

In DNA translation, transcription incorporates two DNA strands separated by an enzyme and a single-stranded messenger RNA that is complementary to the DNA strand. The process maps DNA sequences of A, T, C, and G to complementary RNA sequences of U, A, G, and C where both share common codons of A, G, and C. DNA includes the additional T codon, and RNA has U, which, along with the other three RNA codons, form proteins composed of amino acids that include around twenty protiens, some of them are: Leucine (Leu), Threonine (Thr), Valine (Val), and Histidine (His) [28]. The noncoded segments are called DNA sequences, which are removed by splicing with the remaining portions, called exons, encoding the information for protein synthesis as are assembled in the mRNA [29]. Figure 1 illustrates the conversion process from DNA and RNA. Tables 1, 2, and 3 represent the corresponding operations in DNA computing, and the essential methods used by biological molecules (Transcription process) as illustrated in Figure 2.

3. **Methodology.** This work presents a modern cryptographic method utilising bioinformatic techniques based on RNA with a secret equation. DNA approaches are applied to converting data into DNA codes, and subsequently into the corresponding RNA string. To generate S-box values, the process begins with the first S-box created based on a secret

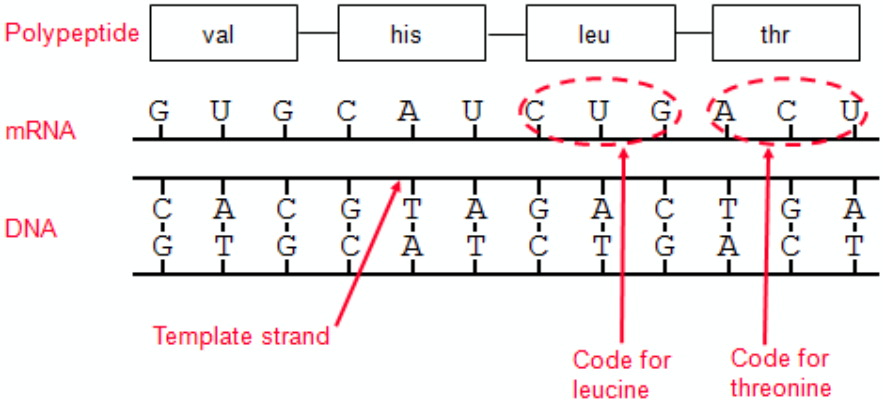

FIGURE 1. Translation process from DNA to RNA and protein

TABLE 1. Eight map rules

	1	2	3	4	5	6	7	8
0	A	A	C	C	G	G	T	T
1	C	G	A	T	A	T	C	G
2	G	C	T	A	T	A	G	C
3	T	T	G	G	C	C	A	A

TABLE 2. Addition and subtraction operations of DNA nucleotides

+	A	T	C	G		−	A	T	C	G
A	T	G	A	C		A	C	G	A	T
T	G	C	T	A		T	A	C	T	G
C	A	T	C	G		C	G	T	C	A
G	C	A	G	T		G	T	A	G	C

TABLE 3. XOR operation of DNA nucleotides

⊕	A	T	C	G
A	A	T	C	G
T	T	C	G	A
C	C	G	A	T
G	G	C	T	A

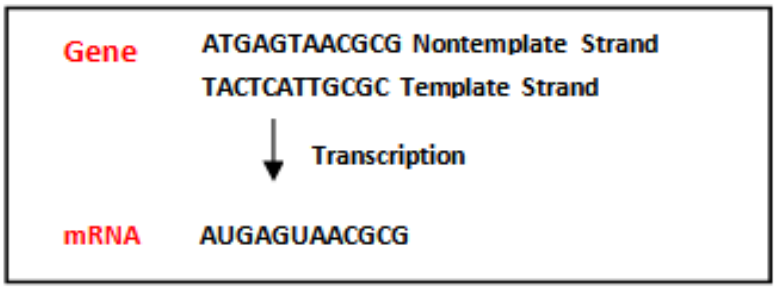

FIGURE 2. Transcription process

key to recursively generate multiple S-boxes. In other words, the first S-box from a new secret key creates the second S-box, which in turn used to generate a third S-box, until obtaining the required number of S-boxes.

As illustrated in Figure 5 and detailed in Algorithm 5, the S-box generation process includes data prepossessing by conversion to a DNA string (Algorithm 1), converting to RNA sequences (Algorithm 2), and applying the bio-computational and mathematical operations (Algorithm 3).

3.1. **Data preprocessing.** This first step is essential in the generation of new S-boxes and includes the conversion of user data into a binary string, which is then translated into DNA sequences based on Table 1. The following two conversion methods are available for generating the DNA sequences.

1) Substitute all 2 bits with one DNA code, such that 00 is replaced by A, 01 by C, 10 by G, and 11 by T.
2) Following the eight mapping rules shown in Table 1, make replacements by taking the intersection of the row and column for every 2 bits.

We apply the second method above because it provides data with high confusion properties. For example, if the secret key is ABC, then its corresponding binary string is "010000010100001001000011". By converting this string into a DNA sequence, we use the intersection of their location in the row and column of Table 1 for every 2 bits. So, the 2 bits in position 11 is 00, then (11 mod 8 = 3). The intersection between row 00 and location 3 is C. Therefore, the resulting DNA string becomes "$CACTAGTCCACG$". The steps for conversion of the user data to DNA codes are illustrated in Algorithm 1.

Algorithm 1: Conversion to DNA codes

Input: User string (S).
Output: DNA sequence.
begin
1. Read char (i)
2. Convert every character of the string to binary and add 0'S bits to the left of the bits character as C string, if the number of bits is less than 8 bits, then name it B.
3. Combine the binary strings C and B.
4. Compare every 2 bits with Table 1 by taking the intersection of the bits value with its location (mod 9) to represent the DNA codes.
end

For example, if the user string is AB, then:

The Binary (A) = "01000001", and the Binary (B) = "01000010".

By combining of these binary data C = "0100000101000010", the locations for this sequence is 12345678.

Compare every 2 bits with its location to get the DNA code based on Table 1, such as "01" in position 1 (mod 9); the intersection is C. Therefore, the DNA code for AB is "$CACTAGTC$".

3.2. **Converting to mRNA string.** This second step converts the calculated DNA string to RNA sequences by simulating the replication and transcription processes. Replication consists of the complementing and replication processes for the DNA string. Transcription is executed through exchanging A with U and T with A from the results of

the replication process. Algorithm 2, and Figure 3 illustrates this conversion to the mRNA string. Continuing with the example from Section 3.1, the DNA sequence becomes "$CACTAGTCCACG$".

The complementing (DNA) is Comp = "$GTGATCAGGTGC$". The replication(Comp) is Rep = "$CACUAGUCCACG$".

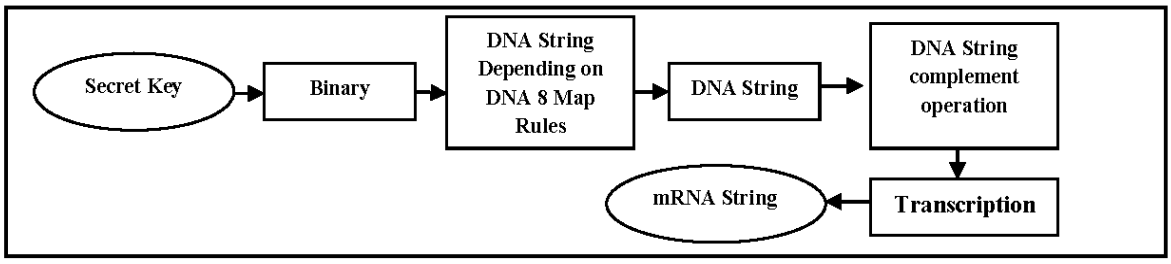

FIGURE 3. The conversion to mRNA string

Algorithm 2: Conversion to mRNA code

Input: Secret key E.
Output: mRNA codes.
begin

1. Convert the secret key string into a binary string.
2. Convert the binary string into DNA codes according to Algorithm 1 by taking the intersection of each 2 bits with its (location mod 9). If the binary string is "0110011110" then the locations are (01) 1, (10) 2, (01) 3, (11) 4 and (10) 5. The 2 bits represent a row in Table 1, and the locations (1, 2, 3, 4, and 5) represent the column number. The result of this example is "$CCAGT$".
3. Apply the replication process by complementing each DNA base pair, such that A $\Rightarrow$ T, C $\Rightarrow$ G, T $\Rightarrow$ A and G $\Rightarrow$ C, for the DNA string created in step 2. For this example, the $comp$(DNA) is "$GGTCA$".
4. Apply the transcription process on the results of the replication process by generating the complement by replacing "A" with "U". The result for the above example is istrans($comp$(DNA)) = "$CCAGU$".

end

3.3. **Bio-computing operations.** The bio-computing operation is based on Tables 4, 5, and 6, which are similar to the DNA bio-computing operations outlined in Tables 2 and 3 differing only by replacing DNA with RNA codes. The RNA string is separated into two parts of where the addition is applied and where the RNA subtraction operation is implemented. These bio-computing operations are presented in Algorithm 3 and Figure 4.

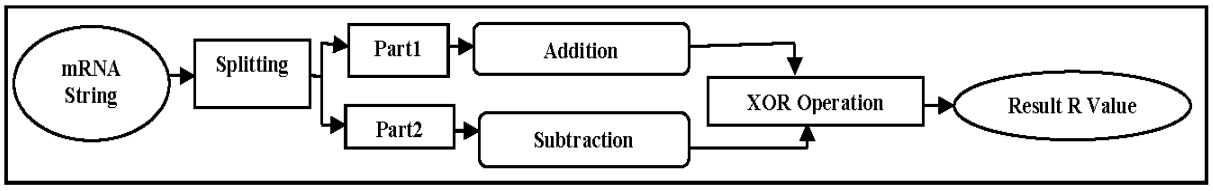

FIGURE 4. Bio-computing operation

Algorithm 3: Bio-computing operations

Input: mRNA string.
Output: mRNA string with reduced size.
begin
1. Split the mRNA string into part 1 and part 2.
2. Apply the addition operation to the mRNA part 1 following the rules provided in Table 4.
3. Apply the subtraction operation to the mRNA part 2 following the rules provided in Table 5.
4. Apply the XOR operation between the results of Steps 2 and 3 following the rules provided in Table 6.
end

3.3.1. *Addition operation.* The RNA addition operation follows the rules in Table 4. As the RNA string is separated into two parts, this operation is applied between every two characters in the RNA string (the first part of the RNA sequence) resulting from the previous step. The RNA sequence of the first part is reduced to half its original length, meaning that it is reduced to a quarter of the entire initial RNA sequence length. For the above example, the first part becomes "$CACUAG$" so that the result of the addition operation is "AUC".

Table 4. RNA addition operation

+	**A**	**U**	**C**	**G**
A	**U**	**G**	**A**	**C**
U	**G**	**C**	**U**	**A**
C	**A**	**U**	**C**	**G**
G	**C**	**A**	**G**	**U**

3.3.2. *Subtraction operation.* The subtraction operation is similar to addition except it depends on a difference table presented as the RNA subtraction rules in Table 5. This operation is also applied between every two characters of part 2. However, the resulting string is reduced in size to one quarter of its original. For the above example, the second part is "$UCCACG$", so the resulting subtraction operation is "UGA".

Table 5. RNA subtraction operation

−	**A**	**U**	**C**	**G**
A	**C**	**G**	**A**	**U**
U	**A**	**C**	**U**	**G**
C	**G**	**U**	**C**	**A**
G	**U**	**A**	**G**	**C**

3.3.3. *XOR operations.* The XOR operation is applied between the results of previous addition and subtraction steps for which the size reduces to one-quarter of the original mRNA sequence. The XOR operation is based on Table 6, and, for the above example, the result is computed as "AUC" XOR "UGA" equals "UAC".

After applying the RNA XOR operation, the XOR logic operation is applied to these results using the corresponding ASCII code characters so that the output is numeric.

TABLE 6. RNA XOR operation

$\oplus$	**A**	**U**	**C**	**G**
A	**A**	**U**	**C**	**G**
U	**U**	**C**	**G**	**A**
C	**C**	**G**	**A**	**U**
G	**G**	**C**	**U**	**A**

This number represents the first cell of the S-box. From this initial number, 255 different numbers are generated with Equation (2) depending on the secret key (i.e., the current location cell and the first generated cell). For example, if the secret key is "*asd ali*", then the following steps illustrate the XOR operation along with the mathematical steps to create cell values. The result from the RNA with XOR logic operations is 18 to represent R and is expressed in the first S-box cell after converting to hexadecimal "12" to represent S, as explained in Algorithm 4.

$$\begin{aligned}
&\mathrm{R} = cell(i) = \text{“12”} = cell(0)\\
&N1 = ascii(secret\ key\text{: }char(i\ mod\ secret\ key\text{: }length)) \qquad (1)\\
&N = (ascii(first\ character\ value\ of\ S\text{-}box) + ascii(second\ character\ value\\
&\qquad of\ S\text{-}box) + (i) + N1) \bmod 256\\
&N = ascii(char(S(1))) + ascii(char(S(2)) + (i) + N1) \bmod 256 \qquad (2)\\
&i = i + 1 = 1\\
&cell(i) = ConverttoHex(N) = \text{“}C4\text{”} = cell(1)\\
&\mathrm{R} = \mathrm{S} = cell(i) = \text{“}C4\text{”}
\end{aligned}$$

The mathematical operation depends on the result of the previous cell, current cell location, and ASCII code of the secret key.

3.3.4. *Search operation.* A search process finds repeated and nonexistent values. After computing the S-box values, the repeated values are first removed. Nonexistent values

Algorithm 4: Mathematical and logical operation

Input: mRNA string resulting from Algorithm 3.
Output: S-box with 256 different values in Hexadicemal.
begin
  1. Set count $i = 0$.
  2. Apply XOR logic operation between the ASCII codes of the input string R to obtain the first cell value.
  3. Compute the values for the S-box cells based on the following steps:
  begin
    Do
    Compute the next cell value X based on Equations (1) and (2).
    Convert the X value to Hexadicemal form, such that $\mathrm{R} = Hex(X)$.
    $i = i + 1$
    $cell(i) = \mathrm{R}$
  end
  While $i <= 255$
end

are identified by comparing the actual S-box values with $(0 \ldots 255)$, and these are then added to the created S-box. Algorithm 5 represents the construction of the S-box matrix.

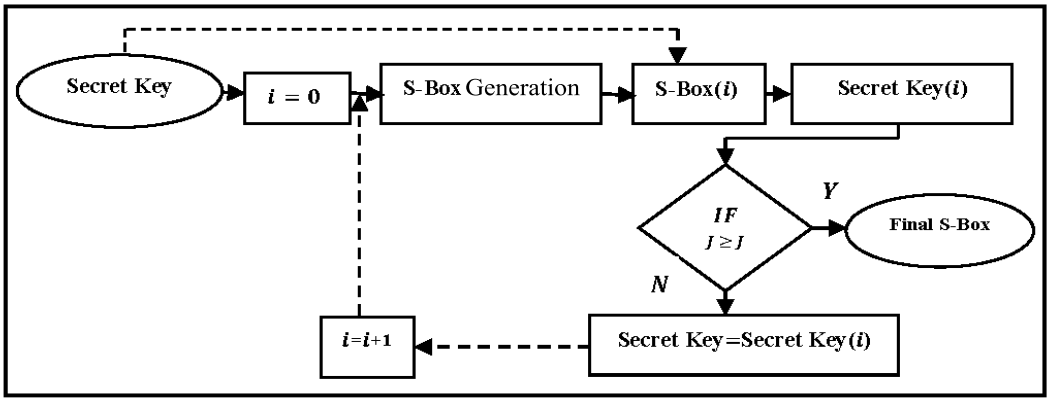

FIGURE 5. Constructing of several S-boxes

Algorithm 5: S-box construction

Input: A secret key, K and J // where J is the number of the required S-boxes and $K = 0$ //.

Output: J is the number of the required S-boxes.

Repeat step 1 to 10 until $k >= j$

begin

1. Convert the secret key to DNA codes based on Algorithm 1.

2. Convert the DNA codes to mRNA codes based on steps of Algorithm 2, and Figure 4.

3. Implement the bio-computing operations (addition, subtraction, and XOR) for the mRNA codes following Tables 4, 5 and 6 as outlined in Algorithm 3.

4. Implement the XOR logic operation between the ASCII codes of the mRNA codes that result from Step 3 to obtain the first cell value R, then convert to hexadecimal as S = Hex(R) following the steps performed in Algorithm 3.

5. Apply the following mathematical operation to this result to obtain the next cell value:

$X = (ascii(first\ character\ of\ S) + ascii(second\ character\ of\ S) + i + ascii(character\ of\ secret\ (i) \bmod length(secret\ key))) \bmod 256$

$cell(i) = Hex(X)$

R = $cell(i)$

begin

Do

Repeat step 5

$i = i + 1$

While $i >= 255$

end

6. Remove the repeated values.

7. Find the value between (0 to 255) that is not included in S-box values Y.

8. Add Y to the newly created S-box.

9. Encrypt the secret key E by entering it into the constructed S-boxes $Key = E$.

10. $k = k + 1$

end

3.4. **S-box inverse.** For the decryption process, we must calculate the S-box inverse, which is generated with Algorithm 6, just like the original S-box matrix with only a single difference in Step 11. The index of the S-box matrix is the new value saved in the generated S-box inverse matrix that is based on the new index taken from the hexadecimal number of each cell in the S-box. An example from Table A will clarify this process. Let S-box$[0, 0]$ = A7, then the two hexadecimal digits A7 represent the new index $[10, 7]$ in the S-box inverse matrix according to A = 10, and 7 = 7. Therefore, the S-box inverse $[10, 7]$ = 00, as shown in Table D where 00 represents the index of A7 in the original matrix.

Algorithm 6: S-box inverse

Input: A secret key, K and J // where J represents the number of required S-box inverses and $K = 0$//.

Output: J is the number of new S-boxes.

begin

1. Convert the secret key to DNA codes based on Algorithm 1.

2. Convert the DNA codes to mRNA codes based on Algorithm 2.

3. Apply the bio-computing operations (addition, subtraction, and XOR operation) for the mRNA codes following Tables 4, 5 and 6.

4. Implement the XOR logic operation between the ASCII codes of mRNA codes resulted from Step 3 to obtain first cell value R, then convert to hexadecimal S = Hex(R) following the step is detailed in Algorithm 4.

5. Apply the following mathematical operation to this result to obtain the next cell value:

$X = (ascii(first\ character\ of\ S) + ascii(second\ character\ of\ S) + i + ascii(character\ of\ secret\ (i) \bmod length(secret\ key))) \bmod 256$

$cell(i) = Hex(X)$

R = $cell(i)$

begin

Do

Repeat step 5

$i = i + 1$

While $i >= 255$

end

6. Remove the repeated values.

7. Find the value from (0 to 255), that is not included in the S-box values Y.

8. Add Y to the newly created S-box.

9. Encrypt the secret key using the constructed table, and label it E.

10. Set the account $i = 0$.

11. Do

a. Extract each number from the created table and separate into two digits.

b. The address for these two digits is additionally taken and combined to represent the new value included in the S-box inverse. The first number is the row and the second number is the column.

c. $i = i + 1$

12. Repeat steps (a, b, and c) until $i >= 255$.

13. $k = k + 1$

14. Repeat step 1 to 13 until $k >= j$

end

4. **Results and Discussion.** As a security requirement for the 128-bit AES cryptosystem, the goal is to use alternate arrangements that help obtain a different output. In comparison to other variant S-boxes, the simulated results show a high confusion property. The constructed S-box is entirely uncorrelated with a percentage of 99% as shown in Tables A $\rightarrow$ J which are presented in the appendix. These tables are classified into four classes. The first three Tables A, B, and C are constructed by one secret key. The next three Tables D, E, and F represent the S-box inverses for Tables A, B, and C. Tables G, and H are constructed based on the new key that is different from the previous one by only one bit, for example, "ABA". Finally, Tables I, and J represent the S-box inverses for Tables G, and H.

Using our proposed method for constructing the multi S-boxes, we found that all those based on one key contain entirely different elements. Table 7 shows that the rate of change is approximately 99%, as well as after changing one bit in the secret key, the rate of change between newly generated S-boxes approaches 99%. According to Tables A, B, C, when the key is "ABC", these S-boxes contain entirely different elements. After changing the key to "ABA", the resulting Tables G, and H have different features and different elements. The computation time for construction of the multi S-boxes with their inverses based on one key is listed in Table 7.

TABLE 7. The average differences between the created S-boxes in terms of the consumed time

Average Differences							**Consumed Time**	
	Input	**S-Box1**	**S-Box2**	**S-Box3**	**S-Box4**	**S-Box5**	**S-box time**	**S-boxinv time**
S-Box1	ABC	*X*	*99%*	*99%*			*130 Ms*	
S-Box2		*99%*	*X*	*99%*				
S-Box3		*99%*	*99%*	*X*				
S-Box4	ABA				*X*	*99%*	*143 Ms*	
S-Box5					*99%*	*X*		

5. **Performance Evaluation.** The performance of our proposed technique for constructing new S-boxes is evaluated in terms of standard statistical parameters, including balanced, completeness, avalanche, strict avalanche criteria, and invertibility. Comparisons are also made with highly-performed studies from the literature (Wang et al. [5] and Balajee and Gnanasekar [29]) to demonstrate the minimum requirements for its efficiency and functionality. The evaluation criteria are considered in the following sections.

5.1. **Balanced criteria.** An essential criterion the S-box should satisfy is the balanced distribution of the 0 and 1 values in the generated output sequence. Through this evaluation of our proposed method, we find that the generated S-boxes are balanced because they contain equal numbers of 0's and 1's as seen in Table 8, which also shows a better balance compared to the other two methods [5,29]. This test was performed using three passwords, and it was observed that the distribution of 0 and 1 values was random. On the other hand, the elements in the constructed S-boxes using the DNA cryptosystem featured an optimal distribution between 0 and 1, which was the highest among the compared methods. The recursive generation of the S-boxes based on one DNA secret key achieved a Shannon criterion of confusion and diffusion rate of 99% due to complete difference in their values, which is a requirement for a highly secured S-box design. However,

TABLE 8. Comparison between AC, SAC and BC of the proposed S-box generation technique with [5,29]

	AC			SAC			BC (3 word)						
	Min	Max	Avg	Min	Max	Avg	ABHKEF31		01234567		01234567		Avg
							No.0's	No.1's	No.0's	No.1's	No.0's	No.1's	
[29]	0.25	0.75	0.5	0.485	0.601	0.507	24	40	33	31	36	28	99.86
[5]	0.125	0.875	0.5	0.421	0.593	0.5039	31	33	28	36	27	37	99.89
The proposed method	0.25	0.875	0.56	0.376	0.597	0.505	31	33	32	32	32	32	99.99

the generated S-boxes for each user are also different with the same rate of change due to the recursive dependency of each newly constructed S-box, as shown in Table 8.

5.2. **Completeness criteria.** The newly generated S-boxes achieved the completeness criteria due to the dependency of each on the input string. If there exists at least one combination of an eight-bit input, e.g., X and X_i, that differ in just one bit, then the output of $f(X)$ and $f(X_i)$ are entirely different in at least J bits.

5.3. **Avalanche criteria (AC).** An essential criterion in block cipher is the avalanche property AC, which refers to how a tiny change in the input bits leads to a large (avalanche) change in the output. With an optimal value of 0.5, this criterion is a desirable feature for block cipher methods because of its result related to the computing of diffusion. Typically, when designing a block cipher, we must consider the avalanche result where a single modification in a single bit of input leads to an entirely different output. Table 8 shows the AC value of the proposed method compared to the Wang et al. [5] and Balajee and Gnanasekar [29] methods.

$$Avalanche\ Effect = \frac{Number\ of\ Flipped\ Bits\ in\ (output)\ Cipher\ Text}{Number\ of\ All\ Bits\ in\ (output)\ Cipher\ Text} \tag{3}$$

In designing the S-box algorithm, the elements should have a normal distribution between 0 and 1. This result depends on the password where the letters must be distributed at the beginning of the algorithm according to Equation (3). This criterion is validated for our proposed algorithm by calculating this value for the letters A through Z and determining the average ratio. Table 9 shows that the average value of the proposed method is better than the comparisons while maintaining an ideal AC value. Therefore, the avalanche criterion is achieved. Table 10 illustrates how to calculate the AC in the case of one bit difference per entry using the proposed and compared methods. This type of non-linearity in the generated S-boxes is an important criterion for the performance of a designed cryptosystem. Measured through a Min-Max non-linearity, this criterion is calculated for our proposed method by comparing to other S-box structures presented in the literature as outlined in Table 8.

An example of how to calculate the AC in the case of one bit difference per entry of the proposed method and by the compared methods is illustrated in Table 10.

The non-linearity of the generated S-Boxes is an important criterion for performance evaluation of the designed cryptosystem. This criterion is measured through Min-Max non-linearity, which are calculated for our proposed method in comparing to some S-boxes structure in the literature and as shown in Table 8.

5.4. **Strict avalanche criteria (SAC).** The transformation function (S-box) satisfies the SA criteria if every bit of its output is modified by a change of one-half once a single bit of its created output is complemented. This criterion merges the completeness and

TABLE 9. Comparisons of the avalanche effective of A...Z using S-Box1 of the proposed method with some other methods

Method Name	Actual Data	Sum of Avalanche Values of Each Input (26)	Avalanche Average
[29]	*A...Z (65...90)*	*12.625*	*0.4855*
[5]	*A...Z (65...90)*	*13.875*	*0.5333*
Proposed method	*A...Z (65...90)*	*13.250*	*0.5096*

TABLE 10. Examples for AC computation of the proposed method and methods in [5,29]

Method	Actual Data	ASCII	Hex	Binary	Replace in S-box	Binary Output	AC
[29]	*B*	*66*	*42*	*0100 0010*	*83*	*1000 0011*	*5/8=0.625*
Modify 1 bit	*C*	*67*	*43*	*0100 0011*	*56*	*0101 0110*	
[5]	*B*	*66*	*42*	*0100 0011*	*DB*	*1011 1101*	*2/8=0.25*
Modify 1 bit	*C*	*67*	*43*	*0100 0011*	*8D*	*1000 1101*	
Proposed method	*B*	*66*	*42*	*0100 0010*	*CE*	*1100 1110*	*6/8=0.75*
Modify 1 bit	*C*	*67*	*43*	*0100 0011*	*20*	*0010 0000*	

avalanche criteria. Therefore, the proposed technique also fulfills this criterion, so SAC is achieved, as shown in Table 8.

5.5. **Invertability.** The S-boxes satisfy invertibility feature, if $S(x_1) = S(x_2)$ such that $x_1 = x_2$ for all inputs x_1, and x_2. Let $x_1 =$ "AB" and $x_2 =$ "AB", i.e, $x_1 = x_2$, so the output of $S(x_1)$ is "21CE", and $S(x_2)$ is "21CE". Now, using the S-box inverse we have, S-boxinv ("21CE") = "4142" in hexadecimal corresponding to ("4142") = "0100000101000010" in binary, after converting of this value to decimal, the output is 6566, which corresponds to $char(65) = A$ and $char(66) = B$. Therefore, the output is "AB" meaning the S-box is invertable due to its ability to extract the original data.

6. **Conclusions.** DNA cryptography has been previously demonstrated as a promising approach with high computational efficiency, an essential level of security, and low storage space. A new technique for designing multi S-boxes based on mRNA was proposed by integrating biological concepts from DNA for generating an efficient secret key with large keyspace and a good security level. We found that the computational processes were extremely reduced, and the designed S-boxes met multiple criteria considered for high performance, including balanced, completeness, avalanche, and strict avalanche. The computational and biological processes are satisfied as a double security layer by using the DNA-AES cryptosystem. The recursive generation of the S-boxes based on one DNA secret key achieves the Shannon criteria of confusion and diffusion due to complete differences in their values, which attained a rate of 99% as a requirement for a highly secured

S-box design. However, the generated S-boxes for each user are also different with the same rate of change due to the recursive dependency of each newly constructed S-box.

REFERENCES

[1] W. K. Hamoudi and N. M. G. Al-Saidi, Information security-based nano- and bio-cryptography, *Multidisciplinary Perspectives in Cryptology and Information Security*, pp.179-199, 2014.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson, Upper Saddle River, 2017.

[3] A. Alabaichi, True color image encryption based on DNA sequence, 3D chaotic map, and key dependent DNA S-box of AES, *Journal of Theoretical and Applied Information Technology*, vol.96, no.2, pp.304-321, 2018.

[4] G. Jakimoski and K. Ljupco, Chaos and cryptography: Block encryption ciphers based on chaotic maps, *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol.48, no.2, pp.163-169, 2001.

[5] X. Wang et al., S-box based image encryption application using a chaotic system without equilibrium, *Applied Sciences*, vol.9, no.4, p.781, 2019.

[6] V. M. Silva-García, R. Flores-Carapia, C. Rentera-Mrquez, B. Luna-Benoso and M. Aldape-Pérez, Substitution box generation using chaos: An image encryption application, *Applied Mathematics and Computation*, vol.332, pp.123-135, 2018.

[7] H. Natiq, N. M. G. Al-Saidi, M. R. M. Said and A. Kilicman, A new hyperchaotic map and its application for image encryption, *The European Physical Journal Plus*, vol.133, no.1, 2018.

[8] Ü. Çavu, A. Zengin, I. Pehlivan and S. Kaçar, A novel approach for strong S-Box generation algorithm design based on chaotic scaled Zhongtang system, *Nonlinear Dyn.*, vol.87, pp.1081-1094, 2017.

[9] N. M. Al-Saidi and M. R. M. Said, A new approach in cryptographic systems using fractal image coding, *Journal of Mathematics and Statistics*, vol.5, no.3, pp.183-190, 2009.

[10] N. M. Al-Saidi, S. S. Al-Bundi and N. J. Al-Jawari, A hybrid of fractal image coding and fractal dimension for an efficient retrieval method, *Computational and Applied Mathematics*, vol.37, no.2, pp.996-1011, 2018.

[11] N. M. Al-Saidi and M. R. Said, Biometric identification based local iterated function systems, *The European Physical Journal Special Topics, Chaos, Cryptography and Communications*, 2014.

[12] H. Shaw, A cryptographic system based upon the principles of gene expression, *Cryptography*, vol.1, no.3, 2017.

[13] K. Mazhar and H. Waleej, Cryptography using DNA nucleotides, *International Journal of Computer Applications*, vol.168, no.7, pp.16-18, 2017.

[14] S. Marwan, S. Ahmed and N. Khaled, DNA-based cryptographic methods for data hiding in DNA media, *Biosystems*, vol.150, pp.110-118, 2016.

[15] S. Sadeg, M. Gougache, N. Mansouri and H. Drias, An encryption algorithm inspired from DNA, *IEEE International Conference on Machine and Web Intelligence (ICMWI)*, pp.344-349, 2010.

[16] E. M. S. Hossain, K. M. R. Alam, M. R. Biswas and Y. Morimoto, A DNA cryptographic technique based on dynamic DNA sequence table, *The 19th International Conference on Computer and Information Technology (ICCIT)*, pp.270-275, 2016.

[17] P. Barman and S. Banani, DNA encoded elliptic curve cryptography system for IoT security, *International Journal of Computational Intelligence & IoT*, vol.2, no.2, 2019.

[18] N. H. UbaidurRahman, C. Balamurugan and R. Mariappan, A novel DNA computing based encryption and decryption algorithm, *Procedia Computer Science*, vol.46, pp.463-475, 2015.

[19] S. Anwar, S. Paul and K. Singh, Message transmission based on DNA cryptography: Review, *International Journal of Bio-Science and Bio-Technology*, vol.6, no.5, 2014.

[20] L. M. Adleman, Molecular computation of solutions to combinatorial problems, *Science*, vol.266, no.5187, pp.1021-1024, 1994.

[21] M.-T. Tran, D. K. Bui and A. D. Duong, Gray S-box for advanced encryption standard, *2008 International Conference on Computational Intelligence and Security*, Suzhou, China, 2008.

[22] B. Behnam, T. M. Anil and D. L. Jones, Data encryption using bio molecular information, *International Journal on Cryptography and Information Security (IJCIS)*, vol.4, no.3, 2014.

[23] A. H. Al-Wattar, R. Mahmod, Z. A. Zukarnain and N. I. Udzir, A new DNA-based S-box, *Int. J. Eng. Technol.*, vol.15, pp.1-9, 2015.

[24] F. A. Kadhim, G. H. A. Majeed and R. S. Ali, Proposal new S-box depending on DNA computing and mathematical operations, *Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*, 2016.
[25] A. Jain, P. Agarwal, R. Jain and V. Singh, Chaotic image encryption technique using S-box based on DNA approach, *International Journal of Computer Applications*, vol.92, no.13, 2014.
[26] H. Wen, S. Yu and J. Lv, Breaking an image encryption algorithm based on DNA encoding and spatiotemporal chaos, *Entropy*, vol.21, no.3, 2019.
[27] A. Girdhar and K. Vijay, A RGB image encryption technique using Lorenz and Rossler chaotic system on DNA sequences, *Multimedia Tools and Applications*, vol.77, no.20, pp.27017-27039, 2018.
[28] R. S. Ali, S. N. Alsaad and N. T. Mahmood, Data encryption using zigzag and sequences of bio molecular information, *Indian Journal of Public Health Research and Development*, vol.10, no.1, 2019.
[29] M. K. Balajee and J. M. Gnanasekar, Evaluation of key dependent S-box based data security algorithm using hamming distance and balanced output, *TEM Journal*, vol.5, no.1, 2016.

Appendix

TABLE A. Values for first S-Box1 (Secret key = "ABC")

A7	E5	2E	47	34	48	37	81	98	DC	C9	F0	82	CC	7E	99
DA	1D	32	E9	FF	49	39	53	6F	DE	F9	EA	87	D0	7F	9C
09	EB	CD	1E	65	26	3D	55	9E	E2	FA	C0	60	C7	80	A0
AD	EC	30	16	06	51	3F	5D	A2	AC	FE	FC	91	FB	83	A5
0A	21	CE	20	01	23	6E	89	97	B1	02	F4	64	1B	84	A6
D8	BC	33	1C	68	25	72	5B	75	B2	9A	2C	6B	24	85	B7
E3	1F	D2	19	0C	2D	3C	8A	73	B4	D4	EF	6C	3A	86	B9
E0	F3	04	22	07	59	70	8E	71	B8	D1	54	9D	46	88	BB
DF	F5	05	EE	0D	5A	41	92	74	C2	DD	03	A3	4B	8B	BF
14	F8	D6	56	38	5E	44	61	7C	BA	AB	3B	A1	4F	8C	C4
B0	FD	0B	F7	08	62	4C	5F	A8	C6	AF	57	9B	50	8D	C8
17	29	10	2A	3E	31	78	93	A9	C1	0E	43	A4	58	8F	CA
B6	CB	11	5C	42	2F	52	66	AA	BD	AE	45	9F	6A	90	D7
E8	27	D9	2B	40	63	4A	67	D3	F1	13	7A	B5	76	94	E1
ED	28	15	00	12	35	79	69	D5	C3	B3	7D	BE	77	95	E6
1A	CF	0F	F6	18	36	4D	6D	DB	C5	E4	4E	F2	7B	96	E7

TABLE B. Values for second S-Box2 (Secret key = "!kh")

84	A8	0A	EF	E0	1B	04	25	67	54	57	03	D8	33	44	63
8E	CC	82	20	01	C9	B2	51	EE	8C	7E	A7	7C	38	86	6A
6D	FA	AF	65	07	26	36	24	9F	3D	B7	B5	E7	68	B6	6B
BE	A9	0D	ED	D7	F3	0C	56	75	8D	62	DB	1A	3B	87	89
EC	D0	88	23	B9	D5	E4	2C	F0	93	66	0B	1D	6F	0E	8A
9B	CB	B8	F9	10	29	3E	27	6E	11	BC	B4	F1	71	16	96
C2	AA	E3	FD	85	FC	45	2D	74	91	90	DE	FE	48	19	9A
BD	FB	8F	D6	13	08	41	5E	F2	98	BB	BF	30	7D	1E	A1
9C	02	E9	28	15	5C	14	0F	73	6C	99	AE	B1	76	1F	A3
C6	4A	EA	79	E2	5B	46	5F	78	47	C5	92	34	B3	2F	A5
C4	D1	C0	D3	12	2E	1C	64	49	9D	4D	95	09	06	39	BA
70	D2	E1	DA	18	60	F4	37	4F	A4	C7	22	43	5A	3C	C3
94	81	E5	2B	EB	00	4E	69	2A	7A	CA	F8	72	59	40	CD
77	D9	97	DD	F7	35	80	3F	53	7B	A2	AC	4C	61	50	DF
9E	8B	C1	31	21	05	C8	17	5D	AB	A6	FF	4B	42	52	F5
CF	DC	E6	CE	E8	B0	55	3A	83	7F	AD	32	D4	A0	58	F6

TABLE C. Values for third S-Box3 (Secret key = "Ú@m")

5A	F0	BD	A5	26	CB	2A	06	48	63	C7	D7	4B	10	80	75
91	A2	00	11	29	16	4F	41	1E	7D	51	AE	DC	15	96	77
AC	E5	81	A8	B9	CD	66	01	8F	6A	95	7B	0F	E6	9D	7E
C1	A9	8B	17	EA	2E	F3	FE	52	67	58	B1	25	4C	B7	8A
92	AA	03	A1	5C	45	21	AF	90	84	8C	F2	DE	73	BC	93
B0	19	87	CC	E9	D2	32	7A	20	3E	89	88	E4	FC	04	AB
6C	7C	94	47	2F	CE	EF	3A	65	6B	A6	8E	34	72	18	B5
E1	AD	D9	A7	EC	35	56	54	28	B2	60	02	BA	61	1B	BE
99	1F	C4	D5	31	D3	3D	3B	59	69	9E	85	36	44	2D	C8
76	DA	37	1D	BB	4E	24	82	30	86	5E	39	C2	38	49	D0
B8	22	E3	D4	1C	0B	F4	12	27	46	9C	E2	F8	8D	5F	DD
A0	B3	C5	D8	33	05	6F	83	74	43	A3	0A	42	9B	64	DF
98	F5	C6	23	C0	0C	2C	13	2B	4A	9A	0D	CA	5D	68	F1
BF	B6	0E	DB	EE	09	40	57	62	71	E7	CF	07	4D	6D	FB
D1	B4	C9	50	F6	E0	F7	1A	79	78	A4	E8	7F	97	6E	FD
9F	F9	14	08	FA	55	3F	5B	3C	C3	ED	EB	D6	53	70	FF

TABLE D. Values for first S-Box1 inverse (Secret key = "ABC")

3E	44	A4	B8	27	28	43	47	4A	02	04	2A	46	48	AB	2F
2B	2C	4E	AD	09	2E	33	0B	4F	36	0F	D4	35	11	32	16
34	14	37	54	D5	55	52	1D	1E	1B	3B	3D	B5	56	20	5C
23	5B	21	25	40	5E	5F	60	49	61	D6	B9	66	62	4B	63
4D	68	4C	BB	69	BC	D7	30	50	51	6D	D8	6A	6F	BF	D9
DA	53	6C	71	B7	72	39	BA	DB	57	58	75	3C	73	59	7A
C2	79	5A	5D	C4	42	7C	7D	45	7E	DC	C5	C6	7F	64	81
67	87	65	86	88	85	DD	DE	6B	6E	BD	DF	89	BE	E0	E1
E2	70	C0	E3	E4	E5	E6	C1	E7	74	76	E8	E9	EA	77	EB
EC	C3	78	7B	ED	EE	EF	84	80	F0	A5	CA	F1	C7	82	CC
F2	C9	83	C8	CB	F3	F4	00	8A	8B	8C	A9	93	03	AC	AA
0A	94	95	AE	96	CD	0C	F5	97	F6	99	F7	15	9C	CE	F8
B2	9B	98	9E	F9	9F	9A	D2	FA	A0	FB	1C	D0	22	24	1F
D1	A7	26	8D	A6	8E	29	FC	05	2D	01	8F	90	A8	91	08
07	FD	92	06	AF	10	FE	FF	0D	31	B1	12	13	0E	38	B6
B0	9D	CF	17	B4	18	3F	3A	19	A1	A2	D3	B3	1A	A3	41

TABLE E. Values for second S-Box2 inverse (Secret key = "!kh")

3C	12	A5	40	3F	3A	1B	1D	99	19	58	45	1F	43	5F	21
46	9D	A6	A7	A8	4B	27	4A	A9	60	9F	2A	2F	AA	AB	62
2D	AC	AD	4F	4D	31	A1	A3	36	35	54	38	69	53	3B	3E
55	6B	5A	AE	AF	5B	A4	59	6F	B0	5C	44	B1	5D	72	B2
5E	B3	B4	48	B5	B6	74	61	B7	75	B8	B9	63	64	BA	65
7B	BB	66	4E	BC	68	67	BD	6A	52	BE	7C	6C	BF	C0	80
C1	6D	C2	C3	6E	C4	C5	57	82	C6	83	70	C7	C8	C9	CA
71	CB	CC	73	CD	CE	87	76	CF	88	D0	D1	77	78	D2	79
D3	D4	7A	D5	D6	7D	00	D7	7E	D8	D9	DA	02	DB	DC	DD
01	7F	DE	DF	81	E0	E1	84	E2	97	13	85	E3	E4	E5	E6
86	E7	E8	E9	EA	9A	8A	89	EB	8B	1C	1A	8C	1E	EC	8D
90	ED	EE	EF	9E	F0	F1	F2	28	91	2E	92	08	F3	F4	03
07	94	F5	32	F6	A2	F7	95	0A	0B	3D	0C	0E	37	F8	F9
96	10	41	15	14	FA	16	42	9B	17	FB	20	47	18	9C	8E
22	23	8F	24	49	93	25	2B	4C	A0	26	FC	29	04	30	05
FD	2C	06	33	50	0D	09	34	0F	39	FE	51	11	FF	56	98

TABLE F. Values for third S-Box3 inverse (Secret key = "Ú@m")

21	72	B7	24	E5	5B	70	CD	3F	5D	BB	5A	5C	BC	2D	C2
D0	31	7A	7C	2F	D1	51	33	E6	15	7E	E7	4A	39	81	18
85	64	1A	3C	69	C3	40	8A	87	41	60	8C	6C	E8	53	46
89	48	65	4B	C6	57	C8	29	D9	B9	76	78	8F	68	95	6F
6D	71	CB	9B	D8	54	9A	36	80	E9	9C	C0	D3	DD	59	61
3E	A1	83	DF	77	5F	67	7D	A3	88	00	7F	44	DC	A9	EA
A7	D7	8D	90	EB	86	62	93	EC	98	92	96	06	ED	EE	6B
EF	9D	D6	D4	8B	F0	09	F1	9E	8E	75	B2	16	91	F2	CE
E0	22	79	7B	94	B8	99	25	B5	A5	F3	23	A4	DA	B6	82
84	01	04	F4	26	A2	E1	DE	0C	08	AC	DB	AA	E2	A8	0F
0B	34	11	AB	AE	30	A6	37	32	13	14	F5	02	17	B1	74
05	B3	97	1B	1E	F6	1D	E3	0A	42	C7	49	E4	20	F7	0D
4C	03	C9	9F	28	2B	2C	A0	F8	2E	CC	50	35	52	56	BD
F9	0E	55	58	3A	38	CF	B0	3B	27	19	3D	C1	FA	C4	FB
5E	07	BA	2A	C5	12	D2	AD	BE	45	43	BF	47	AF	4D	66
10	FC	B4	63	6A	1C	4E	6E	CA	1F	4F	FD	D5	FE	73	FF

TABLE G. Values for first S-Box4 (Secret key = "% : %")

80	B3	AC	22	F4	00	1F	23	A3	D1	B7	4B	03	5A	85	A8
87	C5	0E	C2	15	F7	55	49	70	BB	FC	2D	06	5C	89	A9
74	C9	DB	D9	E2	2E	3B	5F	76	C1	27	16	08	5D	8A	AB
8C	C8	F2	FD	1D	04	58	7F	79	C7	0A	71	1B	60	8D	AE
BD	CF	E0	2B	14	01	31	82	62	AD	C4	77	1E	63	92	B0
CE	E3	DE	29	45	E4	5B	4F	8F	D0	0D	61	24	66	93	B1
EC	FF	F8	13	2F	38	5E	41	AF	FA	2A	52	2C	69	96	B4
ED	E9	EA	CA	19	18	47	6E	95	E6	DF	59	30	6C	97	BA
D4	D7	17	E1	20	35	37	8E	98	D3	21	65	39	6F	99	BC
BF	D5	9D	36	34	25	3A	91	88	B6	CD	73	3D	72	9A	C0
F0	B9	EE	EB	F1	12	4A	44	B5	D6	4D	6B	3F	75	9B	C3
A4	D8	1C	0B	26	0F	67	64	B2	DC	53	A1	48	78	9C	CB
F3	07	CC	05	F6	FE	40	50	8B	11	33	94	4E	7B	9F	DD
AA	A7	1A	3C	10	7C	6A	7D	B8	90	68	83	51	7E	A2	E5
EF	BE	E7	0C	28	4C	6D	A0	9E	C6	42	86	54	81	A5	F5
F9	09	DA	43	46	32	56	7A	E8	D2	3E	02	57	84	A6	FB

TABLE H. Values for second S-Box5 (Secret key = "% : %")

EB	00	46	C6	78	96	1A	D0	E7	01	29	5F	E8	8C	DB	7E
51	93	18	64	50	5D	B0	63	83	D3	FA	65	88	BE	ED	87
8A	07	B1	36	53	04	80	C9	4E	DA	F0	CD	5B	C8	15	8B
25	99	B4	97	E6	62	82	6E	7F	0B	2C	91	22	CB	16	98
F4	10	49	3F	81	67	1E	D4	F2	09	2F	3B	F7	D8	17	9F
C1	13	21	02	84	A1	B2	6D	EE	77	5E	FD	08	74	1C	BD
23	06	24	38	56	31	4A	D9	1B	E4	35	9A	68	DD	34	C7
F5	A2	B7	D7	EF	6B	94	AC	EC	11	9D	6C	9E	AD	39	CC
C5	37	52	6F	86	A0	B8	05	BB	E9	32	0A	3C	AB	44	D2
2A	0F	55	71	89	70	90	A7	1F	AA	3D	6A	9C	BF	48	DE
26	D5	27	72	FB	0E	92	DC	F3	1D	A9	A3	0D	BC	4B	E1
95	40	C0	41	BA	3A	2E	A5	8F	B5	4C	DF	7D	8D	4D	EA
03	0C	C3	75	B6	A4	C2	76	FC	20	AF	7A	AE	2B	57	F1
F9	43	58	47	5C	A6	5A	E2	28	85	7C	45	73	D1	60	F6
CF	12	30	E0	8E	42	C4	7B	CA	F8	4F	14	B3	D6	66	FE
61	A8	33	E3	2D	79	9B	E5	CE	B9	59	19	54	3E	69	FF

TABLE I. Values for first S-Box4 inverse (Secret key = "% : %")

50	54	BF	C0	53	3C	C1	1C	C2	1F	A3	3B	3E	A5	21	5B
4D	9C	5A	36	44	41	B2	28	57	47	2D	C3	2B	43	C4	60
48	A8	30	70	C5	59	4B	A2	4E	35	A6	34	C6	B1	52	46
C7	64	5F	AC	49	58	39	68	56	C8	69	62	3D	C9	AF	CA
6C	76	AE	3F	7A	45	4F	67	CB	71	6A	B0	5E	AA	CC	75
7C	CD	B6	AB	CE	61	6F	CF	63	B7	D0	65	D1	D2	66	72
D3	B5	84	D4	7B	B8	D5	6B	AD	D6	6D	BA	D7	6E	77	D8
81	B3	D9	B9	02	DA	82	B4	DB	83	7F	DC	5D	7D	DD	73
00	DE	74	BD	DF	E0	BE	01	89	E1	E2	8C	03	E3	78	85
9D	79	E4	E5	BC	87	E6	E7	88	E8	E9	EA	EB	29	8E	EC
7E	BB	ED	80	0B	EE	EF	1D	F0	F1	0D	F2	20	94	F3	86
F4	F5	8B	10	F6	8A	99	A0	8D	1A	F7	91	F8	04	1E	09
F9	92	31	FA	A4	11	9E	93	13	12	37	FB	2C	A9	05	14
95	90	9F	98	08	19	9A	18	1B	32	2F	22	9B	FC	25	A7
24	38	42	15	55	FD	97	2E	8F	17	27	3A	06	07	2A	0E
0A	4A	23	0C	40	FE	4C	51	26	0F	96	FF	A1	33	5C	16

TABLE J. Values for second S-Box5 inverse (Secret key = "% : %")

10	90	35	0C	52	78	16	12	C5	94	B8	93	1C	CA	5A	19
14	97	1E	15	BE	E2	E3	E4	21	BF	60	86	E5	9A	64	89
9C	25	C3	06	26	03	0A	2A	8D	A0	09	DC	A3	4F	6B	A4
2E	56	A8	2F	E6	A6	32	18	36	E7	5B	B4	C8	A9	DF	34
1B	3B	5E	1D	E8	BD	20	3D	E9	24	66	EA	AB	EB	82	AE
41	01	28	42	CF	29	46	EC	2D	AF	6D	C2	4D	51	A5	B0
ED	0F	53	71	31	B1	EE	54	C6	EF	B9	57	B7	75	73	38
59	39	3A	CD	D5	3C	7C	95	40	5F	BC	7E	AD	CB	F0	83
62	44	63	81	45	9D	48	F1	C1	49	02	F2	D0	DB	4E	8B
69	B3	6A	11	67	0B	50	33	F3	13	B6	6F	C9	A7	C7	F4
58	55	17	BA	5C	7B	5D	79	1F	AA	99	D8	77	D7	CC	AC
61	22	65	CE	23	9B	4C	27	68	9F	4B	88	DA	F5	D1	D9
2B	05	6C	2C	6E	08	30	F6	D2	72	8E	D3	F7	B2	8F	0E
70	DD	F8	91	74	1A	DE	37	D4	76	92	E0	7A	D6	F9	BB
3E	FA	7D	3F	96	7F	43	80	C0	98	FB	00	87	E1	85	47
A2	FC	84	8A	04	07	FD	C4	9E	0D	A1	4A	8C	B5	FE	FF