# Design and analysis of DNA Binary Cryptography Algorithm for Plaintext

Hassan Al-Mahdi[#1], Osama R.Shahin[#*2], Yasser Fouad[&3], Khalid Alkhaldi[#1]
[#] Computer Science and Information, Jouf University
Faculty of Science & Arts, Saudi Arabia
[1]{hassanwesf, K.Alkhaldi}@ju.edu.sa
[*] Physic and Mathematic Dept., Faculty of Engineering,Helwan University, Egypt
[2]osama.r.shaheen@gmail.com
[&]Faculty of Computer and Information, Suez University, Egypt
[3]yasserfrb@suezuniv.edu.eg

***Abstract*—Due to the rapid growth of computer networks, more sensitive information is being exchanged over such networks. Securing information from unauthorized parties has become a critical issue in the field of securing information technology. DNA cryptography is a promising technology in the cryptographic field which enables users to encrypt data securely based on the real biological DNA strands. Although there are many DNA encryption algorithms, there is still room for security improvement. In this paper, we introduce a symmetric DNA binary cryptography algorithm to encrypt and decrypt plaintext information. the contribution of this paper are twofold: First, we introduce a mathematical algorithm to generate strong secret key from the DNA of different multiple living creatures. Second, the encryption process is implemented using another 16 keys which randomly generated from the secret key. The efficiency and confidence of the proposed algorithm is examined in terms of encryption and decryption time, avalanche effect and the resistance of the secret key against the attack.**

**Keyword-**DNA cryptography, Symmetric encryption, Block cipher, Data security.

## I. INTRODUCTION

Currently, computer networks are rapidly growth and more new services are emerged. As a result, a very large amount of information is being transmitted over such networks [1]. Usually, transmissions over different networks are not secure due to the presence of hackers who wait for a chance to gain access to private information [2] [3]. Hence, the private information needs a certain level of communication and computer protection.

Encryption is the process of converting plaintext or information into an unintelligible form for the purpose of hiding this data [4]. There are many techniques used to transforming information (i.e., plain text) into unreadable format (i.e., cipher text). Cryptography is one of such techniques which information is hidden by a secret key with a specific algorithm [5][6]. The secret key is the password used in the encryption or decryption algorithm and only those authorized parties can know it [7]. While the algorithm is the steps needed to achieve the cryptography process. Confidentiality, integrity, authentication and Non-repudiation are four main objectives behind the use of cryptography [8].

Cryptography can be symmetric or asymmetric [9][10]. Symmetric makes use of a single key for both the encryption and decryption process. Some of the encryption algorithms that use symmetric keys include: AES, Blowfish, DES, Triple DES, Serpent, and Twofish, which are modern, sophisticated and proven in today's encryption field [11]. On the other hand, asymmetric is based on two keys: the public key and the privet key, where the public key is used to encrypt messages and the private key is used to decrypt messages. Systems that use this type of encryption are PGP, DSA, Deffie-Hellman, Elgamal, RSA [12].

The cryptography is typically classified into three approaches: modern cryptography, quantum cryptography, and DNA cryptography [13]. DNA cryptography is a new technique that utilizes DNA nucleotides as an informational carrier with the aid of molecular techniques [14]. Due to the large storage capacity of Deoxyribonucleic Acid (DNA), one gram of which has a capability to store data (108 Tera Bytes), this DNA is highly susceptible to storing information above all known storage media (electric, magnetic, optical) [15]. DNA is a polymer made of monomers called deoxyribonucleotides [16]. Each molecule of nucleotide consists of three simpler molecules that are directly related to each other These molecules are: nitrogen bases, carboxylic sugar, and phosphoric acid. The nitrogenous bases are of two kinds: purines (Adenine and Guanine) and pyrimidines (Cytosine and Thymine). They are arranged as A, G, C and T [17]. The use of Deoxyribonucleic Acid DNA in the field of encryption provides two levels of protection for encrypted data. The first is the difficulty in making and using processes and biological techniques. The second level concerns the difficulty of solving and analyzing mathematical processes used in data encryption [18].

Watson and Crick in 1953 presented a model of DNA consisting of two twisted taps in the form of a helical ladder in which one of the nitrogen bases in one snail is connected to the other's nitrogenous base by hydrogen bonds. The adenine in one of the tapes is always associated with thymine in the other tape with hydrogen bonds and the cytosine in one of the tapes with the guanine in the other tape by using three hydrogen bonds [19].

Research work is being done on DNA Computing either utilizing test tubes (organically) or mimicking the tasks of DNA utilizing PCs (Pseudo or Virtual DNA processing). The first to use DNA in the calculation was Adleman [20] in 1994 to solve the problem of finding the shortest path. It was found that acid has a parallel processing property that provides very high speed if used in arithmetic calculation. In 1995, Boneh et al. [21] used DNA to break the standard data encryption system (DES). Many attempts [22] [23] [24] to build acid-based data encryption systems DNA has been used to randomize the sequence of the nitrogen bases of the acid as a key to the one-time system.

Gehani et. al., presented the primary trial of DNA based Cryptography in which a substitution technique utilizing libraries of particular one time cushions, every one of which characterizes a particular, arbitrarily created, combine shrewd mapping and a XOR conspire using atomic calculation and ordered, arbitrary key strings are utilized for encryption [14]. Other methodologies used an image encryption algorithm based on Sequence Addition Operation, this is done by DNA sequence matrix, DNA sequence addition us Logistic maps and complementarily. This algorithm was adopted again by using DNA sequence matching here the data converted into pointers according to DNA strand taken and key send to the receiver in a secure channel [25].

The contributions of our paper are as follows: most of symmetric DNA algorithms which are introduced in the literature use a secret key from single living creature. In this paper we generate a secret key from multiple DNA of living creatures. On the other hand, we introduce a mathematical method to derive the secret key. In addition, the encryption process will be conducted using another 16 keys generated from the secret key. The remainder of this paper is organised as follows: Section II introduces the proposed encryption algorithm in details. The performance of the proposed algorithm is introduced in Section III. Finally, the conclusion is drawn in Section IV.

## II. Proposed Algorithm

The proposed Symmetric DNA Binary encryption (SDB) algorithm uses both DNA (Deoxyribo Nucleic Acid) and binary system to converts the plaintext into cipher format. It is a block cipher with block size of 64 nucleotides which represent 16 alphabet characters. DNA has two long strands of nucleotides and each nucleotide is made of deoxyribose sugar, phosphate group and a nitrogenous base. Nitrogenous bases are A (Adenine), G (guanine), C (Cytosine) and T (Thymine). These 16 nucleotides involved in DNA synthesis set $S = \{A, C, G, T\}$. The SDB algorithm consists of the two sub algorithms SDB1 and SDB2 used for encrypting a plaintext message (pMsg) and decrypting a ciphertext message (cMsg) respectively as shown in Figure 1.

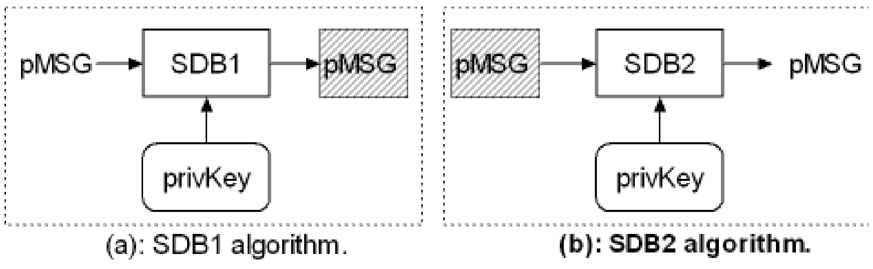


Fig. 1: The system dynamic of SDB algorithm.

Where $privKey$ denoting the main private key. The privKey generates sub private keys $PK_i (i = 1,2,3, \dots ,16)$ which are used through the encryption and decryption processes. The values of these keys will be described in section *B*. In the proposed SDB algorithm, the privKey is the only information which will be kept secret at both the sender and the receiver sides. In addition, the value of the sub keys are dynamically derived at the start of algorithms SDB1 and SDB2. In fact, the SDB algorithm encrypts a pMSG message in 16 stages using SDB1 and decrypts cMsg using the same stages but in reverse order.

### A. Main Private Key

The SDB algorithm uses a privKey as main key to encrypt and decrypt the plaintext. The value of masKey is given by the DNA sequence $e_1e_2e_3 \dots e_N$ with length $N \in \Omega$ , where $e_i \in S$ , $i = 1,2,3, \dots , N$and$\Omega = \{256, 512, 1024, \cdots\}$. The power of this algorithm comes from the fact that the $privKey$ is not taken from the DNA of one living creature. Instead, it consists of $k \in \Theta$ sequences (i.e. segments) chosen randomly from the DNA of different living creatures, where$\Theta = \left\{1, 2, 3, \dots \left\lfloor \frac{N}{2} \right\rfloor\right\}$. Note that the value of k is also chosen randomly. If $s_j$ denoting the length of segment $j (j = 1,2,3, \dots , k)$ then

$$N = \sum_{j=1}^{k} s_j$$

The length of segment $s_j$ is calculated depending of the parameters $N$ and $k$ according to the following cases:

**Case 1:**If the number of chosen segments $k$ can be represented in a form $2^m$, m is an arbitrary natural number, and then $s_j$ is given as

$$s_j = 2^{\left(log_2^N - log_2^k\right)}, j = 1,2,3,\dots,k \tag{1}$$

**Case 2:**If the number of chosen segments $k$ can not be represented in a form $2^m$ then the lenghts of $(k_{prev} - k_{dis})$ segments out of k are given as

$$s_j = 2^{\left(log_2^N - log_2^{k_{prev}}\right)}, j = 1,2,3,\dots,(k_{pre} - k_{dis}) \tag{2}$$

Where $k_{pre}$ is the nearest least number to $k$ which can be represented in the form of $2^m$ for any value of m and $k_{dis} = k - k_{pre}$. The lengths of the remaining $(k - k_{pre} + k_{dis})$ segments out of $k$ are given for $j = k_{prev} - k_{dis} + 1, k_{prev} - k_{dis} + 2, \dots, (k - k_{pre} + k_{dis})$ as follows

$$s_j = 2^{\left(log_2^N - log_2^{k_{next}}\right)}, \tag{3}$$

Where $k_{next}$ is the nearest largest number to $k$ which can be represented in the form of $2^m$ for any value ofm.

To clarify the process of selecting the lengths of chosen segments, let us give the following example:

1) Let $\mathrm{N} = 128$ and $\mathrm{k} = 7$.
2) The values of $\mathrm{k_{pre}}$, $\mathrm{k_{dis}}$ and $\mathrm{k_{next}}$ are calculated as 4, 3 and 8 repectivey.
3) Since $\mathrm{k} = 7$ then k can not be represented in form $2^{\mathrm{m}}$ for any value of m.
4) The lengths of the first $\mathrm{k_{pre}} - \mathrm{k_{dis}} = 4 - 3 = 1$ segments are caculated using (2) to give one segment of length 32 nucleotides.
5) The lengths of the remaining$k - k_{pre} + k_{dis} = 7 - 4 + 3 = 6$ segments are calculated using (3) to give six segments of length 16 nucleotides.
6) The total lengths of *7* segments is $32 + (6x16) = 128$ nucleotides which equal to $N$.

*B. Sub Private Keys*

The sub private keys $PK_i (i = 1,2,3,\dots,16)$ are derived from the privKey during the encryption and decryption processes as follows. Let $n_1$, $n_2$, $n_3$ and $n_4$ denoting the occurrence numbers of nucleotides A, C, G and T in the DNA sequence of the $privKey$, respectively, where $n_1 + n_2 + n_3 + n_4 = N$ and the $privKey$ length $N \in \{256,512,1024,\cdots\}$. Let $P_1 = \{a_1, a_2, a_3, \dots, a_{n_1}\}$ , $P_2 = \{c_1, c_2, c_3, \dots, c_{n_2}\}$ , $P_3 = \{g_1, g_2, g_3, \dots, g_{n_3}\}$ and $P_4 = \{t_1, t_2, t_3, \dots, t_{n_4}\}$ are four sets which represent the positions number of nucleotides A, C, G and T within the $privKey$, respectively. Convert each element in $P_1$, $P_2$, $P_3$ and $P_4$ into their equivalent 8 binary form. Concatenate, from right to left, all the binary elements of $P_1$ , $P_2$ , $P_3$ and $P_4$ as one binary string $\mathfrak{R}$. Divide $\mathfrak{R}$ into $r_i (i = 1,2,3,\dots,16)$ blocks with equal lengths, where the length of each block is given as$\frac{|\mathfrak{R}|}{16}$.

Finally, take $PK_i = r_i for\ i = 1,2,3,\dots,16$.

*C. Data Dependency Segment*

The concept of data dependent is used extensively in encryption and decryption algorithms which increase the unpredictability of cipher text. In this paper, we generate a Data Dependency Segment (DDS) with length 16 bits for the entir palintext. We use the DDS in both rotation and XORing processes througth encryption process. The DDS is generated as follows:

1) Convert pMSG to its equivalent binary form, where the ASCII code of each character is converted to its equivalent 8 binary bits.
2) Divide the resulted binary string into n blocks of length 128 bits as follows:

$$n = \begin{cases} \left\lfloor \frac{L}{128} \right\rfloor & if \quad L\ mod\ 128 = 0 \\ \left\lfloor \frac{L}{128} \right\rfloor + 1 & if \quad L\ mod\ 128 \neq 0 \end{cases} \tag{4}$$

   Note that, the length of the last block is less than 128 if $L\ mod\ 128 \neq 0$.

3) Take block $i$, $(i = 1,2,3,\dots,n)$ from left to right as $s_1, s_2, \dots, s_m$ segments with 8 bits each.

4) For $j = 1,2,3, \dots, m$, convert each segment $s_j$ to its equivalent decimal value$d_j$. Generate the numeric value Z as

$$Z = \sum_{j=1}^{m} j \times d_j \tag{5}$$

Where$j$ represents the weight of segment $s_j$ within its block.

5) Convert the numeric value $Z$ to its equivalent binary form $D_i$ with length 16 bits. If the length of$D_i$ is less then 16 bits then $D_i$ is padded with zeros from left. The final form of $D_i$ represent the DDS for block $i$.
6) Repeat steps 3 to 6 to generate the DDS for all the remaining blocks.
7) The final DDS for the entire plaintext, $D$, is drived as follows.
   i. **Case 1:** If $n = 1$then $D = D_1$.
   ii. **Case 2:** If $n > 1$then $D = V_{n-1}$. The value of $V_{n-1}$ is calculaed as follows. Take $V_0 = D_1$ as initial value and then apply the following recursive equations:

$$\begin{gathered} V_1 = V_0 \oplus D_2 \\ V_2 = V_1 \oplus D_3 \\ V_3 = V_2 \oplus D4 \\ \vdots \\ V_{n-2} = V_{n-3} \oplus D_{n-1} \\ V_{n-1} = V_{n-2} \oplus D_n \end{gathered}$$

*D. Encryption Process*

The encrypting procedure uses the SDB1 algorithm to convert he original information or plaintext to encrypted form or a cipher text. The SDB1 algorithm uses the following steps:

1) Generate the main key as described in section ($A$).
2) From the main key, drive the sub private keys $PK_j (j = 1,2,3, \dots, 16)$ as described in section (B).
3) Convert pMSG to its equivalent binary form, where the ASCII code of each character is converted to its equivalent 8 binary bits.
4) Divide the resulted binary string into n blocks, $(B_1, B_2, B_3, \dots, B_n)$, using (5).
5) Generate the value of $D$ that represent the DDS for the entire pMSG.
6) Concatenate value of $D$ to the right side of each block $B_i$, $i = 1,2,3, \dots, n$ *(i.e.,* $B_i = B_i + D$*).*
7) Take the 16 least significant bits (LSBs) of block $B_1$ as $S_1$ and the remaining bits as $S_2$.
8) Label the bits of $S_1$ from left to right as $b_j$ where$b_j \in (0,1), = 1,2,3, \dots, |S_1|$ and$|\times|$ denotes the length of a string $\times$.
9) Label the bits of $S_2$ from left to right as $c_j$ where $c_j \in (0,1)$ and $j = 1,2,3, \dots, |S_2|$
10) Label the bits of $PK_k$ from left to right as $d_j$ where $d_j \in (0,1)$, $k = 1,2,2, \dots, 16$ and $= 1,2,3, \dots, |PK_k|$.
11) Perform Exclusive OR of $S_2$ with the sub private keys $PK_k$ based on the bits value of $S_1$ using algorithm 1.
12) Rotate the bits of $S_2$ right or left based the bits of $S_1$ as follows. The 0 in the most significant bit (MSB) of $S_1$ means right rotation and 1 means left rotation. The decimal value of $S_1$ represents the number of rotations.
13) Reconstruct block $B_1$ as $S_2 + S_1$ again.
14) Take the 16 most significant bits (MSBs) of block$B_1$ as $S_1$ and the remaining bits as $S_2$.
15) Repeat steps 8 to 13.
16) Reconstruct block $B_1$ as $S_1 + S_2$ again.

**Algorithm 1** Exclusive OR ($\oplus$) operation

```
1: uuu.
2: for $i = 1$ to $|S_1|$ do
3:     if $S_1.b_i = 1$ then
4:         for $j = 1$ to $|S_2|$ do
5:             $S_2.c_j = S_2.c_j \oplus PK_i.d_j$ .
6:         end for
7:     end if
8: end for
```

17) For the remaining block, repeat steps from 8 to 16.
18) Repeat steps from 8 to 17 $n$ times.
19) Take the last output of each block and convert it to hexadecimal form which represents the cipher message, cMSG, of the plaintext pMSg.

*E. Decryption Process*

As aforementioned, the sub algorithm SDB2 is used to restore the plaintext, pMSG, from the cipher text, cMSG, as follows.

1) Load the main key.
2) From the main key, drive the sub private keys $PK_j(j = 1,2,3, \dots ,16)$ as described in section (B).
3) Load the cipher text cMSG.
4) Convert cMSG to its equivalent binary form, where each character is converted to its equivalent 4 binary bits.
5) Divide it into $n$ blocks, $(B_1, B_2, B_3, \dots, B_n)$, as follows.

$$n = \begin{cases} \left\lfloor \frac{L}{144} \right\rfloor & if \quad L \bmod 144 = 0 \\ \left\lfloor \frac{L}{144} \right\rfloor + 1 & if \quad L \bmod 144 \neq 0 \end{cases} \tag{6}$$

6) Take the 16 MSBs of block $B_1$ as $S_1$ and the remaining bits as $S_2$.
7) Rotate the bits of $S_2$ right or left based the bits of $S_1$ as follows. The 0 in the MSB of $S_1$ means left rotation and 1 means right rotation. The decimal value of $S_1$ represents the number of rotations.
8) Perform Exclusive OR of $S_2$ with the sub private keys $PK_k$ based on the bits value of $S_1$ using algorithm 2.

**Algorithm 2** Exclusive OR ($\oplus$) operation

```
1: uuu.
2: for $i = |S_1|$ downto 1 do
3:     if $S_1.b_i = 1$ then
4:         for $j = 1$ to $|S_2|$ do
5:             $S_2.c_j = S_2.c_j \oplus PK_i.d_j$ .
6:         end for
7:     end if
8: end for
```

9) Reconstruct the block i as $S_1 + S_2$ again.
10) Repeat steps 6 and 7.
11) Take the 16 LSBs of block $B_1$ as $S_1$ and the remaining bits as $S_2$.
12) Perform Exclusive OR of $S_2$ with the sub private keys $PK_k$ based on the bits value of $S_1$ using the following algoritm 8.
13) Reconstruct the block $B_1$ as $S_2 + S_1$ again.
14) For the remaining block, repeat steps from 6 to 13.
15) Repeat steps from 6 to 14 n times.

16) For each block, remove the DDS from right and then convert each 8 bits to their decimal value and then substitute each decimal value by its corresponding alphabet character as defined in ASCII table.
17) The final form represents the plaintext pMSG.

## III. Performance Analysis

In this section, we evaluate the performance of the proposed encryption algorithm in terms of the resistance against the attack on both main and sub keys, the encrypted time and decrypted time and Avalanche test. The proposed SDB1 and SDB2 algorithms are conducted using JAVA platform. The main key is formed from the Public DNA taken from the European Bioinformatics Institute's (EBI's) nucleotide archive. As aforementioned, the main key is taken as a combination of multiple DNA segments from different living creature which generates strong main key. Since the generation of the sub keys $PK_k$, $k = 1,2,3 \dots ,16$ depend on the main key, changing any segment in the main key will create entirely different sub keys.

### *A. Keys Force Attack*

Its highly impossible for the force attack analysis to recovery the main key for the following reasons:

1) The DNA public database has millions of nucleotide sequences.
2) The length $N$ of the main key is chosen randomly from the elements of an infinite set $\Omega = \{256,512,\cdots\}$.
3) The main key consists of multiple DNA segments taken randomly from the DNA of different living creature. These segments are concatenating in random way to form the main key. So, if the segments k are taken from the set $\Theta$ living creatures, then the number of permutation is given as

$$\frac{k!\,\Theta!}{(\Theta-k)!}.$$

As aforementioned, each sub key $PK_k$, $k = 1,2,3 \dots ,16$ consists $\Gamma = \frac{N*8}{16}$ bits which is chosen based on the positions of nucleotideA,C, G and T within the main key. If the length of the main key is $N \in \Omega$ then the number of possible combinations of $PK_k$, $k = 1,2,3 \dots ,16$ are given respectively as

$$\mathrm{PK}_{\mathrm{k}}^{\mathrm{permu}} = \Gamma! \sum_{\mathrm{N}\in(256,512,\dots)}^{\infty} \frac{(8\mathrm{N})!}{(8\mathrm{N}-\Gamma)!}$$

where $k = 1,2,3 \dots ,16$. If, for example, the length of the main key is taken from set$\Omega = \{64,128,256,512\}$ then

$$PK_k^{permu} = \Gamma! \sum_{N\in(64,128,256,512)} \frac{(8N)!}{(8N-\Gamma)!}$$

### *B. Encryption and Decryption Time*

The time necessary to converts the plaintext into the ciphertext is called encryption time. This time is subject on the message block size and the main key size, and denoted in milliseconds. On the other hand, the decryption time is necessary to convert the plaintext from cipher text. It is desirable that the decryption time must be smaller than or equal to the encryption time. In this algorithm, the encryption time, $T_{enc}$, consists of the following components: 1) The time, $T_{key}$, required to generate the main key. 2) The time, $T_{sub}$, required to generate the sub private keys $PK_k$, $k = 1,2,3 \dots ,16$. *3)* The time, $T_{sdb}$, required to encrypet the palintext. Hence, the total encryption time is given as

$$T_{enc} = T_{key} + T_{sub} + T_{sdb}. \tag{7}$$

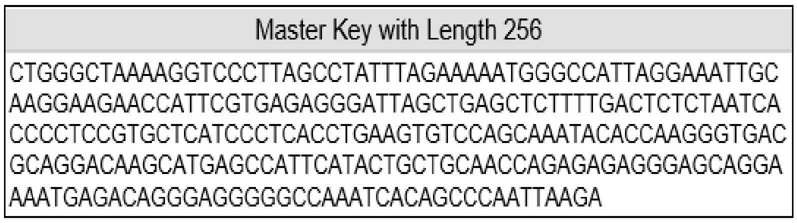
On the other hand, the decryption process is implemented by excusing the same steps of the encryption process in reverse order expect that the step of generating the main key is not executed. In such case, the decryption time, $T_{dec}$, is given as

$$T_{dec} = T_{sub} + T_{sdb}. \tag{8}$$

By comparing (7) and (8), we note that $T_{dec} < T_{enc}$.

### *C. Avalanche Effect*

The avalanche measures the correlation between ciphertext bits and both plaintext and main key bits. In other words, the avalanche test measures the change in the cipher text when some bit in plaintext or main key is changed [26]. First, we retain the main key to the constant value to all plain text as shown in Figure 2.


Master Key with Length 256

CTGGGCTAAAAGGTCCCTTAGCCTATTTAGAAAAATGGGCCATTAGGAAATTGC
AAGGAAGAACCATTCGTGAGAGGGATTAGCTGAGCTCTTTTGACTCTCTAATCA
CCCCTCCGTGCTCATCCCTCACCTGAAGTGTCCAGCAAATACACCAAGGGTGAC
GCAGGACAAGCATGAGCCATTCATACTGCTGCAACCAGAGAGAGGGAGCAGGA
AAATGAGACAGGGAGGGGGCCAAATCACAGCCCAATTAAGA

Fig. 2: Main Key with length 256.

The plaintext "Multiprogramming" is encrypted using the selected main key and then we use **o** instead of **g**, **v** instead of **r**, **A** instead of **a** and **M** instead of **m**. For the plaintext "Computer Organization", we use **N** instead of **n**. For "5555333388887777 5555333388887777", we use **1** instead of the last **7**. Finally, for "AA112233445566FE AA11223344556666", we use **5** instead of the last **6**. Table 1 illustrates the avalanche test results for 4 simple plaintext. Results show that the proposed SDB algorithm exhibits strong avalanche property.

TABLE I. The avalanche test of different simple plaintext.

| **Plaintext** | **Cipher text** | **Bits** | **Bits changed** | **Avalanche test** |
|---|---|---|---|---|
| Multiprogramming | F4596664B77B4451E 9F726B2EFF9AB75039D | 144 | – | – |
| Multiprogrammino | A5190856E884A66DB 62C1B7714A1CE5B450A | 144 | 81 | 0.4375 |
| Multiprogvamming | 3CD6CE58FDF60B503 72A70C8B5C62E9A000B | 144 | 75 | 0.4791 |
| MultiprogAmming | FA143DDD1BD67ADAB B38AE6D37A98472FB19 | 144 | 74 | 0.4861 |
| MultiprogamMing | FA143DDD1BD67ADAB B38AE6D37A98472FB19 | 144 | 76 | 0.4722 |
| Computer Organization | 03B40F5DB2CCD86CC6 87F5F81FDFE7BE B8BA2025704AF7 | 184 | – | – |
| Computer OrganizatioN | FF13D40F81F3F06832 EDBE13C65AFD2E 471AAC953B277C | 184 | 94 | 0.48913 |
| 5555333388887777 | 7448B8D0FD3C38B04358 AFC76BBE8EC17E347 448B8D0FD3C38B04358A FC76BBE8EC17E34 | 288 | – | – |
| 5555333388887775 | 24E083464183975B3702 02657E7D353FCD8D6 19225DD87526AD1FDB85 3922D80585A9D57 | 288 | 162 | 0.4375 |
| AA112233445566FE AA11223344556666 | 77B7FDA80EBCFED0F6A 24281829C6232D260753E F9FB3D73E33E3327956 ED56838677B6D | 288 | – | – |
| AA112233445566FE AA11223344556665 | 1349F64D9A89F608A2E8 699C0CEA17A1B52FB82 D10E85D6FD9AB8069C 38574CBE13BACEE | 288 | 145 | 0.496 |

## IV. CONCLUSION

In this paper, a symmetric DNA binary cryptography algorithm to encrypt and decrypt plaintext information is introduced. Mathematical algorithm is introduced to generate a secret key from the DNA of different multiple living creatures. From the main key, 16 sub keys are randomly generated. The efficiency and confidence of the proposed algorithm is examined in terms of encryption and decryption time, avalanche effect and the resistance of the secret key against the attack. The experiment test shows that the proposed encryption algorithm has very good avalanche test and private keys force attacks.

## REFERENCES

[1] Shipra Jain and Vishal Bhatnagar. Analogy of various dna based security algorithms using cryptography and steganography. In Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on, pages 285–291. IEEE, 2014.
[2] Jing Yang, Jingjing Ma, Shi Liu, and Cheng Zhang. A molecular cryptography model based on structures of dna self-assembly. Chinese science bulletin, 59(11):1192–1198, 2014.
[3] Aysha Divan and Janice Royds. Molecular Biology: A Very Short Introduction. Oxford University Press, 2016.
[4] J Thomas LeontinPhiljon and N Venkateshvara Rao. Metamorphic cryptography—a paradox between cryptography and steganography using dynamic encryption. In Recent Trends in Information Technology (ICRTIT), 2011 International Conference on, pages 217–222. IEEE, 2011.
[5] Marwa E Saleh, Abdelmgeid A Aly, and Fatma A Omara. Data security using cryptography and steganography techniques. IJACSA) International Journal of Advanced Computer Science and Applications, 7(6), 2016.
[6] Yunpeng Zhang and Liu He Bochen Fu. Research on dna cryptography.In Applied cryptography and network security. InTech, 2012.
[7] SomdipDey, JoyshreeNath, and AsokeNath. An integrated symmetric key cryptographic method-amalgamation of ttjsa algorithm, advanced caesar cipher algorithm, bit rotation and reversal method: Sja algorithm. International Journal of Modern Education and Computer Science, 4(5):1, 2012.
[8] RajdeepBhanot and Rahul Hans. A review and comparative analysis of various encryption algorithms. International Journal of Security and Its Applications, 9(4):289–306, 2015.
[9] Majid Khan and Tariq Shah. A literature review on image encryption techniques. 3D Research, 5(4):29, 2014.
[10] Sandeep Tayal, Nipin Gupta, Pankaj Gupta, Deepak Goyal, and Monika Goyal. A review paper on network security and cryptography. vol, 10:763–770, 2017.
[11] Omar Farook Mohammad, MohdShafryMohd Rahim, Subhi Rafeeq Mohammed Zeebaree, and Falah YH Ahmed. A survey and analysis of the image encryption methods. International Journal of Applied Engineering Research, 12(23):13265–13280, 2017.
[12] Prashant Kumar Arya, Mahendra Singh Aswal, and Vinod Kumar. Comparative study of asymmetric key cryptographic algorithms. International Journal of Computer Science & Communication Networks, 5(1):17–21, 2015.
[13] Yunpeng Zhang, Xin Liu, Yongqiang Ma, and Liang-Chieh Cheng. An optimized dna based encryption scheme with enforced secure key distribution. Cluster Computing, 20(4):3119–3130, 2017.
[14] Ashish Gehani, Thomas LaBean, and John Reif. Dna-based cryptography. In Aspects of Molecular Computing, pages 167–188. Springer, 2003.
[15] Sarfaraz K Niazi and Justin L Brown. Fundamentals of Modern Bioprocessing. CRC Press, 2015.
[16] AbdelkrimBerrada, Minghwa Benjamin Liang, Lawrence Jung, and Kurt Jensen. Alkaline activation for immobilization of dna taggants, October 17 2017. US Patent 9,790,538.
[17] Tilo Lajos Vittorio Ulbricht. Purines, pyrimidines and nucleotides and the chemistry of nucleic acids. Elsevier, 2016.
[18] BeenishAnam, KaziSakib, Md Hossain, Keshav Dahal, et al. Review on the advancements of dna cryptography. arXiv preprint arXiv:1010.0186, 2010.
[19] Leslie Pray. Discovery of dna structure and function: Watson and crick. Nature Education, 1(1):100, 2008.
[20] Leonard M Adleman. Molecular computation of solutions to combinatorial problems. Science, 266(5187):1021–1024, 1994.
[21] Richard J Lipton. Breaking dbs using a molecular computer dan boneh christopher dimworth. DNA based computers, 27:37, 1996.
[22] Jie Chen. A dna-based, biomolecular cryptography design. In Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on, volume 3, pages III–III. IEEE, 2003.
[23] Grasha Jacob and A Murugan. An integrated approach for the secure transmission of images based on dna sequences. arXiv preprint arXiv:1611.08252, 2016.
[24] AsishAich, Alo Sen, Satya Ranjan Dash, and Satchidananda Dehuri. A symmetric key cryptosystem using dna sequence with otp key. In Information Systems Design and Intelligent Applications, pages 207–215. Springer, 2015.
[25] Hui-Hsien Chou and Michael H Holmes. Dna sequence quality trimming and vector removal. Bioinformatics, 17(12):1093–1104, 2001.
[26] I Vergili and MD Yucel. Avalanche and bit independent property for the esembles of randomely choosen nxn s-boxes. EE dept of METU.