Design and Analysis of DNA Encryption and Decryption Technique based on Asymmetric Cryptography System

Hassan Al-Mahdi
Computer Science & Information Dept.,
College of Science and Arts,
Jouf University, KSA

Meshrif Alruily
Department of Computer Science,
College of Computer and Information Sciences,
Jouf University, KSA

Osama R. Shahin∗†3, Khalid Alkhaldi∗4
∗†Computer Science & Information Dept.,
College of Science and Arts,
Jouf University, KSA
†Physics and Mathematics Dept,
Faculty of Engineering,
Helwan University, Egypt

Abstract—Security of sensitive information at the time of transmission over public channels is one of the critical issues in digital society. The DNA-based cryptography technique is a new paradigm in the cryptography field that is used to protect data during transmission. In this paper we introduce the asymmetric DNA cryptography technique for encrypting and decrypting plain-texts. This technique is based on the concept of data dependency, dynamic encoding and asymmetric cryptosystem (i.e. RSA algorithm). The asymmetric cryptosystem is used solely to initiate the encryption and decryption processes that are completely conducted using DNA computing. The basic idea is to create a dynamic DNA table based on the plaintext, using multi-level security, data dependency and generating 14 dynamic round keys. The proposed technique is implemented using the JAVA platform and its efficiency is examined in terms of avalanche property. The evaluation process proves that the proposed technique outperforms the RSA algorithm in terms of avalanche property.

Keywords—DNA cryptography; asymmetric encryption; block cipher; data dependency; dynamic encoding

I. INTRODUCTION

Information is a treasured commodity in today’s societies. As the world becomes ever more connected, the need for effective and intensive information security grows exponentially and is essential for protecting information against unauthorized access and for preserving information privacy. Moreover, the number of intruders is said to be directly proportional to the advances in information technology [1], [2]. The most common techniques used in computer security fields are steganography and cryptography [3], [4]. The primary task of these techniques is to maintain the security and confidentiality of information [5].

Cryptography is a method of encrypting and decrypting text by blocking confidential data in an incomprehensible way to the intruder [6], [7], [8]. Different cryptography procedures [7] have been created, such as the substitution algorithm, which depends on supplanting one letter with another, and can be generally classified according to the type of encryption key into symmetric and asymmetric encryption. The RSA algorithm is considered a strong asymmetric encryption algorithm.

In symmetric encryption, the same key is used for both encryption and decryption. Therefore, it is important to identify a safe way to transfer the key between the sender and recipient. Asymmetric encryption uses the key pair concept; it uses a different key for encryption and decryption. The key usually specifies the private key and the other key, known as the public key. The private key is kept private by the owner and the public key is shared between the approved recipients or is made available to everyone. Encrypted data can only be encrypted with the recipient’s public key using the corresponding private key [9], [10]. The general construction of the encryption algorithms is illustrated below in Fig. 1. For maximum protection and robust security with high capacity, new methods of hiding data were suggested by the researchers based on DNA [11], [12].

Recently, research has been carried out on DNA-based data hiding schemes. Most use biological properties of DNA sequences. First, however, some basic knowledge should be introduced [13], [14]. DNA is a nucleic acid consisting of genetic information that is used in the development and work of living creatures and some viruses. It consists of the most complex organic molecules. DNA stores genetic information as a symbol of four chemical bases: adenine (A), guanine (G), cytosine (C) and thymine (T). The information required to build and maintain a living organism is determined by the sequence of the rules above. However, like every data storage device, DNA requires protection through a secure algorithm. This has led to the field of new research based on DNA
computing.

Leier et al. [15] proposed a robust scheme using a special key sequence, known as a primer, to decode sequential DNA. In addition, the generic DNA sequence is used as a reference, which defines the receiver. Thus, a specific primer and an encrypted sequence are sent to the receiver. Without specific prefixes and sequences, binary data cannot be decrypted correctly. In [11], Peterson proposed a method to hide data in DNA sequences by replacing three consecutive DNA bases as one letter. For example, "B" = AAC", "E = CCG", etc. There are 64 symbols that can be encoded. However, the repetitions of the letters "E" and "I" that appear in English text are very high. Therefore, an attacker could use this property to break the encrypted message.

The proposed DNA coding technique in [16] is based on a symmetric key where key sequences are attained from the genetic database and left as they are on both ends: sender and recipient. The plain text is firstly converted to binary format and then to DNA format using the DNA substitution. In [17], three test techniques based on DNA were proposed. These methods are: insertion method, complementary pair method and replacement method. For these three methods, a DNA reference sequence is chosen and the secret message is incorporated into it to obtain a pseudo DNA sequence that is sent to the receiver ...

The system presented in [18] proposes a key block encoding inspired by three-phase DNA. These are: initial, repetition and final stages. It includes a step that mimics the idea of the original biological molecules of transcription, i.e. transfer from DNA to messenger RNA, which then translates from mRNA to amino acids. During design, it follows expert recommendations in coding and focuses on "confusion" and "propagation", which are basic properties of encoded text.

Another data hiding technique [19] was developed mainly through two phases. In the first phase, plain text is encrypted using the RSA encryption algorithm whereas, in the second phase, the encrypted message is encrypted using the complementary characters while preserving the index of each hidden letter of the message in the DNA sequence. The strength of this algorithm is the use of the RSA algorithm, which is considered one of the most powerful asymmetric encryption techniques.

A new way of data hiding is suggested in [20] based on the replacement of the repeated characters of the DNA reference sequence by placing an injection scheme between a complementary base and two secret bits in the message. This algorithm reduces the rate of modification by substituting only consecutive DNA characters by expanding zero. However, the modification rate can be very high if the DNA sequence contains many repetitive characters.

Tushar and Vijay [7] designed 4*4 DNA encryption technologies to manipulate matrices using a main generation system, making data extremely secure. Apart from features that provide a good security layer, restrictions include large encrypted text along with security that only depends on the key.

The proposed technology in [21] relies on the DNA and RSA encryption system, and is able to provide an architectural framework for encrypting and generating digital signatures for all characters, simple text data, and text files. Here, the whole process consists of four steps. These are: main generation, data processing before and after, DNA and signature generation.

The technique proposed in [22] is the concept of a dynamic DNA sequence table that assigns random ASCII characters in the DNA sequence at the beginning. It then applies a limited number of duplicates to dynamically change the ASCII position in the sequence table based on a mathematical string. However, use of the one-time pad (OTP) board makes the technology more efficient because the normal OTP plaintext and the key must be equal in size so the safe transfer of the key is more difficult.

A new hybrid method combining cryptography and steganography is proposed in [23]. This achieves multi-layer security of the system based on DNA encryption. The methods of concealment adopted here do not expand the reference DNA sequence, and the embedded data can be extracted without the need for a real DNA reference sequence. Recently, Hassan Mahdi et al. [24] provided a symmetric binary DNA encryption algorithm to encrypt and decrypt plain text information. The contribution of this paper is twofold: firstly, we provide a mathematical algorithm to generate a strong secret key of the DNA of multiple living organisms. Secondly, encryption is performed using 16 other keys randomly generated from the secret key.

The contributions of this paper are as follows: most of the DNA algorithms that are introduced in the literature are symmetric, which send a secret key over a secure channel. In this paper asymmetric algorithms are introduced with public and private keys. The proposed algorithm is better suited to the plaintext data. In addition, the encryption process is conducted using multi-level security via generating a dynamic coding table, data dependency and multiple dynamic round keys. The remainder of this paper is organized as follows: Section 1 contains an introduction and related works. Section 2 introduces the proposed asymmetric cryptography technique in detail. The performance of the proposed algorithm is introduced in Section 3. Finally, the conclusion is drawn in Section 4.

II. PROPOSED ALGORITHM

The introduced asymmetric cryptography technique constructs the public key \( pubKey = (n, e, PST) \) for encryption and the private key \( privKey = (n, d, PST) \) for decryption. The parameters \( e, d \) and \( n \) are generated using the well known RSA cryptography algorithm. Anyone can use the public key to encrypt the plaintext (PT) while the parameter \( e \) is kept secret. The parameter \( PST \), denoting the public DNA Sequence Table, consists of 24*4 size matrix, as used in [25], [22]. This table fulfills all the alphabet characters: uppercase, lowercase, numbers, and special characters. The encryption of PT and decryption of cipher text (CT) processes are given, respectively, as \( CT = Encrypt(PT, pubKey) \) and \( PT = Decrypt(CT, privKey) \). The proposed asymmetric cryptography technique consists of the following five stages:

1) Construction of DNA public and private keys.
2) Construction of a dynamic DNA sequence table.
3) Generating 14 round keys.
4) Encryption process.
5) Decryption process.
The encryption process at the sender consists of 14 security levels. On the other hand, the RSA cryptography system is not used to encrypt a PT; rather, it is used as a helper function to generate the DNA Dynamic Sequence Table (DDST), the round keys $R_{k,i}, i = 1, 2, 3, \ldots, 14$, and the Start Decryption Key (SDK) during the encryption process, as shown in Fig. 2. The SDK is combined with the CT and is used to initiate the decryption process.

A. Public and Private Keys Generation

In this paper, a receiver constructs the PST by generating a long single-stranded DNA string $S$ which is chosen randomly from the DNA of different living creatures. The string $S$ is divided into chunks with 4 DNA bases. Each chunk is randomly assigned to an alphabet character with no duplication. The PST table is generated with each session and, hence, the DNA sequences and the assignment of alphabets are different from session to session. Table 1 illustrates the PST defined in Table I, and $e$ is the public key used as a helper function for all subsequent processes. For simplicity, in this paper we will use 64 bit RSA cryptography for all further examples.

B. Dynamic DNA Sequence Table

The first step of the encryption process is to generate the DDST table. The generation process of the DDST table depends on the plaintext, the public DNA sequence table and the RSA public keys, which are denoted by the quadruplets: $(PT, PST, e, n)$. The concept of data dependent is introduced here through using the parameter $PT$ which increases the unpredictability of DDST table. For all subsequent processes, we use PST defined in Table I, and $n = 8076924410049049481$. As shown in Fig. 3, the steps of creating the DDST table are as follows:

1) Divide the PT into a number of chunks of equal size of 8 characters. For example, the PT "Computer Organization" is divided into chunk1="Computer", chunk2="Organization" and chunk3="ation".

2) Traverse each chunk and replace each character by its ASCII code, which gives chunk1 = "67 111 109 112 117 116 101 114", chunk2 = "32 79 116 105 114 103 97 110 105 122" and chunk3 = "97 116 110 105 111 110".

3) For each chunk, convert each ASCII code number to binary format and combine all binary values as one binary string as follows:

<table>
<thead>
<tr>
<th>chunk1</th>
<th>chunk2</th>
<th>chunk3</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000011110111100110100110000110</td>
<td>10000010111111001010011100001110</td>
<td>10101110110110011010011111010001</td>
</tr>
</tbody>
</table>

4) Convert each chunk to its corresponding decimal value: chunk1=3820960565494002, chunk2=18365789048124666 and chunk3=26283243502.
Generating Fibonacci Series

1: Input: Big integer number $\Psi$ with digits $d_1,d_2,d_3,\ldots,d_w$, $w \geq 5$.
2: Traverse $\Psi$ from left to right.
3: Set $n_1 = d_1d_2$.
4: Set $n_2 = d_3d_4$.
5: Set $S[0] = n_1$, $S[1] = n_2$
6: Set $L = \sum_{i=5}^{w} d_i$
7: for $j = 2$ to $L$ do
8: Set $n_3 = n_1 + n_2$
9: Set $S[j] = n_3$
10: Set $n_1 = n_2, n_2 = n_3$
11: end for
12: Output: Fibonacci series $S$.

Table II: Dynamic DNA Sequence Table

<table>
<thead>
<tr>
<th>String</th>
<th>ATAA</th>
<th>ATGC</th>
<th>TCTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>GACA</td>
<td>GGGT</td>
<td>CGAC</td>
<td></td>
</tr>
<tr>
<td>CGCA</td>
<td>CTGC</td>
<td>TGTG</td>
<td></td>
</tr>
<tr>
<td>AGCA</td>
<td>GCAG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCGA</td>
<td>GTCC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTGA</td>
<td>TCGA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GATG</td>
<td>GTGT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TATC</td>
<td>GTCA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAC</td>
<td>TGGG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATA</td>
<td>TCGT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTCA</td>
<td>GTA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Algorithm 2 Generating Binary String

1: Input: Set of integer number $S = [d_1,d_2,d_3,\ldots,d_w]$.
2: For all $d_i \in S$, convert $d_i$ to binary bits $b_i$.
3: Set string STR=$b_1+b_2+b_3+\ldots+b_w$.
4: Output: Binary string STR.

C. Generating Round Keys

As shown in Fig. 4, the round keys $R\overline{K}_i, i = 1,2,3,\ldots,14$ are generated using the DDST table. The round keys must be generated in ascending order starting from $R\overline{K}_1$ to $R\overline{K}_{14}$. For $R\overline{K}_i$, to generate the round key $R\overline{K}_i$, perform the following steps:

1) Traverse the DDST table from the first element to the last and concatenate all their corresponding 4 DNA bases into one string STR.
2) Convert the DNA sequence STR into binary format using substitutions $A = 00$, $C = 01$, $G = 10$ and $T = 11$. Set STR1 = STR and STR2 = STR. 
3) Encrypt the value of $E_i$ using RSA cryptography algorithm to get $E_{i+1}$.
4) Use algorithm 1 to generate Fibonacci series $S$ with input $E_{i+1}$.
5) For each element $d_j \in S$, rotate STR1 right $d_j$ times if $d_j$ is even or left if $d_j$ is odd, where $j = 1,2,3,\ldots,S.length$.
6) For each element $d_j \in S$, rotate STR2 left $d_j$ times if $d_j$ is even or right if $d_j$ is odd.
7) Reconstruct STR as STR1+$STR2$.
8) Convert each element in S into binary string using algorithm 2 and then combine all binary strings as STR3.
9) Set STR = STR $\oplus$ STR3.
10) Set $R\overline{K}_i = STR$. Use the value of $E_{i+1}$ as input to the next round key $i+1$ generation process.

D. The Encryption Process

The receiver constructs the public keys (i.e., $e$, $n$, PST) and sends these keys on a public channel keeping the private key

(i.e., d) secret. Any sender can use the public keys to encrypt its PT. To clarify the encryption process, we assume that PT = “Computer Organization”. The encryption process passes through the following steps:

1) Read the PT file and divide it into blocks with size 16 alphabet characters each. These blocks are as follows:
   Block1 = "Computer Organiz" and Block2 = "ation". The length of the last block may be less than 16 alphabet characters.
2) Generate the DDST table as described in section II-B.
3) Convert each block to DNA sequence by substituting each character with its corresponding DNA base alphabet characters.
The length of the last block may be less than 16
   alphabet characters each. These blocks are as follows:
   Block1 = "Computer Organiz" and Block2 = "ation".
4) Convert the DNA sequence of Block1 and Block2 to 2-bit binary format (A = 00, T = 01, C = 10, G = 11) as follows:
   Block1 = 1111011010101010011110110101
   100110101010100101101000101000101100100110
   0010100100001011010010001011001110110011101
   1001101111011111111
   Block2 = 1011101010101001010000101101001110100111
   5) User E1 as input and generate the round key RK1 as described in section II-C.
6) Divide RK1 into a number of chunks C1, C2, ..., CL of equal size 64 bits, where L denotes the number of chunks.
7) For all j = 1, 2, 3, ..., L, set Block1 = Block1 ⊕ Cj as follows:
   \[
   Block1 = Block1 \oplus C_1 \\
   Block1 = Block1 \oplus C_2 \\
   \vdots \\
   Block1 = Block1 \oplus C_L
   \]
8) Repeat step 7 to perform the XOR operation on Block2 and chunks C1, C2, ..., CL.
9) For the remaining round keys RKi, i = 2, 3, ..., 14, repeat steps 5 to 8 to get:
   \[
   Block1 = 01011001101101000010001100011110
   01101001101010101011011010110011101001101
   0010100100001011010010001011011101110110011
   0001101101111101
   Block2 = 000110101011010110101011101101010101101000
   1010100100
   \]
10) Convert both Block1 and Block2 to DNA sequence using substitutions 00 = A, 01 = C, 10 = G, 11 = T.
11) Set DSK = E16. The value of E16 is obtained during the generation process of the round key RK14.
12) Convert the numeric value of SDK to binary format. If the number of bits in SDK is odd, attach "0" to the left.
13) Convert SDK to DNA sequence using substitutions 00 = A, 01 = C, 10 = G, 11 = T.

E. Decryption Process

As illustrated in Fig. 5 below, the decryption process includes the following steps for decrypting the received CT to PT. In fact, the process of executing the encryption steps in reverse order represents the decryption process.

1) Read a CT file as DNA string sequence str.
2) Convert the DNA sequence of CT into its equivalent binary form using substitutions A = 00, C = 01, G = 10, T = 1.
3) Take the first 16 bits of str as str1 and the remaining bits as str2.
4) Convert str1 to its corresponding decimal value X.
5) Starting from the right of str2, take X bits as str3 and the remaining bits as str4.
6) Convert str3 to its corresponding decimal value. This decimal value represents the start decryption key SDK.
7) Set E16 = DSK.
8) For i = 14, 13, 12, ..., 1, follow the steps below:
   a) Decrypt Ei+2 using RSA with secret key e to get the number Ei+1.
The proposed asymmetric DNA encryption algorithm based on
the RSA cryptography system is conducted in JAVA plat-
form. The public DNA sequence table PST is generated using
the European Nucleotide Archive which provides a very large
collection of nucleotide sequences. The proposed technique is
evaluated in terms of avalanche test, execution time and plain
text size.

A. Randomization of the DDST Table

The proposed technique maximizes the secrecy of CT
through generating a DDST table and 14 round keys based on
public key and PT. If a DDST table can be detected from the
PST table, it has poor randomization. This may be sufficient
for making predictions about the input. However, it is very
difficult to predict the input from the DDST table if it has high
randomization. The DDST table has very high randomization
if there is no alphabet character has the 4 DNA bases value
in both DDAT and PST. Table III shows that the DDST table
exhibits a high degree of randomization at different plaintexts.
At first, it is assumed that the plain text is given as PT= "Computer
Organization". The value of the public keys e and n are given as:
"1393980256209590861" and "8076924410049049481",
respectively. On the other hand, the public DNA sequence
table is given in Table I. Table IV shows the DDST table
randomization degree when encrypting the same PT many
times with flipping a single bit every time.

### TABLE III. THE DDST TABLE RANDOMIZATION COMPARED TO THE
PST.

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>No. of Matched</th>
<th>Randomization Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Organization</td>
<td>2</td>
<td>99%</td>
</tr>
<tr>
<td>Multiprogramming</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>5555333558887777</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Aljouf University</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

### TABLE IV. THE DDST TABLE RANDOMIZATION WITH ONE BIT
DIFFERENCE.

<table>
<thead>
<tr>
<th>Bit index</th>
<th>PT changes</th>
<th>No. of unchanged 4 DNA values</th>
<th>Randomization Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>computer Organization</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>13</td>
<td>Computer Organization</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>27</td>
<td>Computer Organization</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>35</td>
<td>Computer Organization</td>
<td>1</td>
<td>98.94%</td>
</tr>
<tr>
<td>47</td>
<td>Computer Organization</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>53</td>
<td>Computer Organization</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>99</td>
<td>Computer Organization</td>
<td>1</td>
<td>98.94%</td>
</tr>
<tr>
<td>156</td>
<td>computer Organization</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

B. Avalanche Property

Avalanche property quantifies the effect on a CT when
input PT is changed slightly (for example, flipping a single
bit) [26], [24]. This change must cause a significant change
in the CT (e.g., 50% of output bits flip). If the number of bits
is changed in a cipher text, due to changing one bit is \( B_{\text{changed}} \)
and the total number of bits in the cipher text is \( B_{\text{total}} \). In such
cases, the Aavalanche Effect (AE) is given as \([26], [27]:\)

\[
AE = \frac{B_{\text{changed}}}{B_{\text{total}}} \times 100\%
\]

Firstly, Table V shows the avalanche effect of the 14
round keys, which are generated during encrypting the two
plaintexts PT1= "Computer Organization" and PT2= "Computer
Organization" with one bit difference. On average, the
avalanche effect on round keys is 50.81%.

Secondly, we investigated the avalanche effect on the cipher
text CT when changing one bit in the input plaintext PT.
Fig. 6. Comparison of Avalanche Test For the Proposed Technique and RSA Algorithm

TABLE V. ROUND KEYS AVALANCHE EFFECT.

<table>
<thead>
<tr>
<th>Round key index</th>
<th>No. of bits changed</th>
<th>Avalanche test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>390.0</td>
<td>51.72%</td>
</tr>
<tr>
<td>2</td>
<td>383.0</td>
<td>50.4%</td>
</tr>
<tr>
<td>3</td>
<td>388.0</td>
<td>51.21%</td>
</tr>
<tr>
<td>4</td>
<td>391.0</td>
<td>51.45%</td>
</tr>
<tr>
<td>5</td>
<td>389.0</td>
<td>51.19%</td>
</tr>
<tr>
<td>6</td>
<td>387.0</td>
<td>50.93%</td>
</tr>
<tr>
<td>7</td>
<td>390.0</td>
<td>51.32%</td>
</tr>
<tr>
<td>8</td>
<td>379.0</td>
<td>49.27%</td>
</tr>
<tr>
<td>9</td>
<td>379.0</td>
<td>49.87%</td>
</tr>
<tr>
<td>10</td>
<td>395.0</td>
<td>51.96%</td>
</tr>
<tr>
<td>11</td>
<td>386.0</td>
<td>50.79%</td>
</tr>
<tr>
<td>12</td>
<td>393.0</td>
<td>51.72%</td>
</tr>
<tr>
<td>13</td>
<td>384.0</td>
<td>50.53%</td>
</tr>
<tr>
<td>14</td>
<td>375.0</td>
<td>50.21%</td>
</tr>
</tbody>
</table>

TABLE VI. COMPARISON OF THE NUMBER OF BITS CHANGED.

<table>
<thead>
<tr>
<th>Encryption Algorithm</th>
<th>Average No. of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Technique</td>
<td>64.895</td>
</tr>
<tr>
<td>RSA Algorithm</td>
<td>30.212</td>
</tr>
</tbody>
</table>

Since the proposed technique is asymmetric cryptography, the obtained results will be compared with the RSA cryptography system. In such cases, we set PT="AA112233445566FE", e = "3199192709" and N = "8076924410049049481"; PST is given in Table I. Firstly, a CT is generated from PT using the proposed technique and RSA algorithm. Secondly, the first bit in PT is flipped to get the new PT="AA112233445566FD" and a new CT is generated, where flipping E (01000101) yields D (01000100). Thirdly, the third bit in PT is flipped to get the new PT="AA112233445566FA" and a new CT is generated. These processes are repeated until the bit number 125 in PT is flipped to get the new PT="QA112233445566FE" and a new CT is generated. Every time the avalanche effect on the CT is calculated. After 48 rounds of executing the two algorithms, there are 48-bits flipped. Table VI shows the average number of bits changed when flipping one bit from the plaintext PT. From the obtained results, we note that the proposed technique outperforms the RSA algorithm in term of the number of bit changed.

Fig. 6, below illustrates the avalanche effect on the cipher text versus the index of the bit flipped in the PT. The figure shows that the proposed algorithm exhibits strong avalanche property compared to the RSA algorithm. From this figure we note that the proposed algorithm has high avalanche test at all indices of the flipped bits with an average of 52.6% compared to the RSA algorithm with an average of 23.8%.

IV. Conclusion

In this paper, the asymmetric DNA cryptography technique based on data dependency, dynamic encoding table, dynamic round keys and the help of asymmetric cryptosystems, is introduced. The performance of this technique is tested in terms of the avalanche effect. Although the proposed encryption technique is not superior to the popular asymmetric algorithms in terms of execution time, it has strong avalanche property. Since the proposed technique generates the dynamic DNA sequence table and round keys based on the plaintext, it is impossible for attackers to detect the plaintext from the cipher text. The experiment test shows that the proposed encryption algorithm has very good avalanche property.

ACKNOWLEDGMENT

This research was supported by Research Deanship, Jouf University, KSA, on grant number 242/39.

REFERENCES


