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

Hassan Al-Mahdi[1]
Computer Science & Information Dept.,
College of Science and Arts,
Jouf University, KSA

Meshrif Alruily[2]
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.

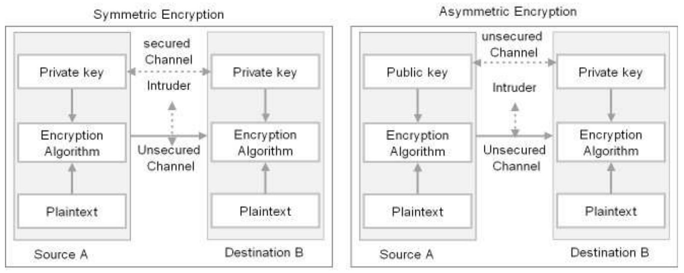

Fig. 1. General Construction of Symmetric and Asymmetric Encryption Algorithms.

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.

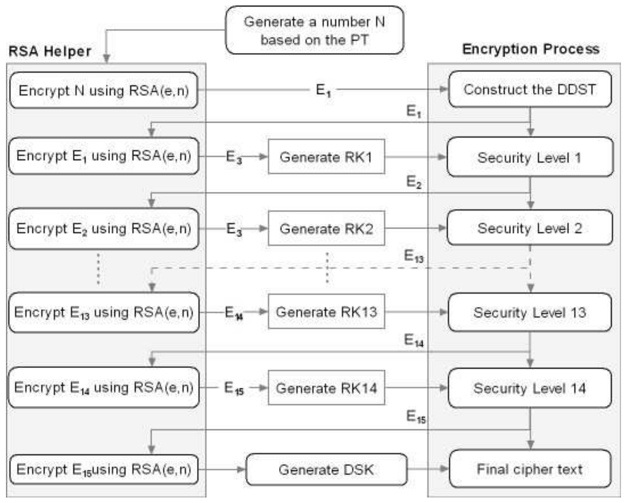

Fig. 2. Block diagram of the RSA helper function.

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 $RK_i, i = 1, 2, 3, \ldots, 14$, 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 I illustrates the PST for a certain session. On the other hand, the values of e, d and n are generated using asymmetric cryptosystem RSA with a 1024 bit key. The value of e is kept secret at the receiver. 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, $e =$ ”1393980256209590861” 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=”Organiz” and chunk3=”ation”.

TABLE I. PUBLIC DNA SEQUENCES TABLE.

space → CCAG	! → CACT	” → TCGA	# → GTAC
\$ → CACA	% → GATG	& → TTGC	′ → ACAT
(→ GCTG	) → CGTG	∗ → ATGG	+ → TGTA
, → AAAT	- → GGCC	. → TGGG	/ → TCCT
0 → CGCT	1 → TCAC	2 → GAGG	3 → CTAC
4 → CCTC	5 → CCTT	6 → AAAG	7 → GGGT
8 → TTGT	9 → TAAT	: → AGGG	; → GTTT
¡ → GTGT	= → CAAG	¿ → AACA	? → CTTG
@ → CAAA	A → TGTT	B → CAAC	C → TTAA
D → GAAA	E → CCTG	F → TGAG	G → ACCC
H → CCCC	I → GGAT	J → TGGT	K → CAGA
L → CTTC	M → ATAC	N → CCAA	O → GGCA
P → TGAA	Q → CTGG	R → GGGC	S → GCTA
T → CCCG	U → GGAA	V → AGAC	W → ACTG
X → GCAT	Y → ACCT	Z → TCTT	[→ CGTT
\ → TGGC	] → CTAT	ˆ → AGGA	_ → AGAA
‘ → ACGG	a → CTCT	b → GGTG	c → GGAG
d → TAAA	e → GCCA	f → GACC	g → GTGA
h → TGCT	i → ATAT	j → GAGA	k → CAGT
l → AATT	m → TTGG	n → GTAG	o → TCTC
p → TTTG	q → TTCC	r → GTCT	s → AGTT
t → ACAC	u → GCAA	v → TTCT	w → TCAA
x → GGTC	y → TCTG	z → AAGA	{ → GCTT
|→ GTGC	} → CCCA	˜ → ATGT	

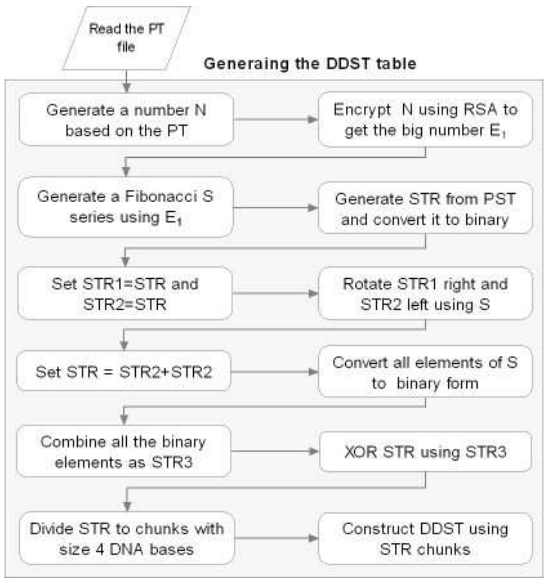
Read the PT file
Generaing the DDST table
Generate a number N based on the PT
Encrypt N using RSA to get the big number E1
Generate a Fibonacci S series using E1
Generate STR from PST and convert it to binary
Set STR1=STR and STR2=STR
Rotate STR1 right and STR2 left using S
Set STR = STR2+STR2
Convert all elements of S to binary form
Combine all the binary elements as STR3
XOR STR using STR3
Divide STR to chunks with size 4 DNA bases
Construct DDST using STR chunks

Fig. 3. Block Diagram of Generating the Dynamic DNA Sequence Table.

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 79 114 103 97 110 105 122” and chunk3 = ”97 116 105 111 110”.
3) For each chunk, convert each ASCII code number to binary format and combine all binary values as one binary string as follows:

chunk1 = 10000111101111110110111100001110
1011110100110010111110010
chunk2=10000010011111110010110011111000
0111011101101001111010
chunk3=11000011110100110100111011111011
10

4) Convert each chunk to its corresponding decimal value: chunk1=38209605565494002, chunk2=18365789048124666 and chunk3=26283243502.

5) Set $N = chunk1 + chunk2 + chunk3 = 3578783238063992861$.
6) Encrypt the value of N using the 64 bit RSA cryptography algorithm with the public keys e and n to give the value $E_1 = 484564171844271401$. Actually, using the RSA cryptography system with 1024 bit will generate huge numbers. Thus, for simplicity, we use RSA with 64 bit in this example.
7) Use algorithm 1 to generate Fibonacci series with input E_1 to get $S = \{48, 45, 93, 138, 231, 369, 600, \ldots, 580804687053\}$. In fact, changing one bit in a PT will cause large changes on the elements of a Fibonacci series even if the values e and n of are fixed.
8) Traverse the PST from the first element to the last and concatenate all their corresponding 4 DNA bases into one string STR.
9) Convert the DNA sequence STR into binary format using substitutions A = 00, C = 01, G = 10 and T= 11. Set $STR1 = STR$ and $STR2 = STR$.
10) For each element $d_i \in S$, rotate $STR1$ right d_i times if d_i is even or left if d_i is odd.
11) For each element $d_i \in S$, rotate $STR2$ left d_i times if d_i is even or right if d_i is odd.
12) Reconstruct STR as $STR1 + STR2$.
13) Convert each element in S into binary string using algorithm 2 and then combine all binary strings as $STR3$.
14) Set $STR = STR \oplus STR3$, where $\oplus$ denoting the XOR operation.
15) Convert STR to DNA sequence using substitutions 00 = A, 01 = C, 10 = G and 11 = T.
16) Divide STR into chunks with size 4 bases each and remove duplication (if any).
17) Pick the first four DNA bases form STR, i.e. GATC, and assign it to the first alphabet character in the PST. Pick the second four DNA bases, i.e. ATAA, and assign it to the second alphabet character in the PST (i.e., $=ATAA). The substitution process continues until it reaches the last alphabet character in the PST. By the end of the substitution process, the DDST table is constructed, as shown in Table II. The DDST table is created during the encryption process and is deleted when the encryption process is completed.

```
Algorithm 1 Generating Fibonacci Series
1: Input: Big integer number $\Psi$ with digits $d_1d_2d_3 \ldots d_w$, $w \geq 5$.
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.

→ GATC	! → ATAA	” → ATAT	# → CTCG
$ → CACA	% → GGAT	& → GCGA	’ → CTAA
(→ CATG	) → TCGC	* → GGTG	+ → CTCT
, → TAAC	- → GGCA	. → GAAG	/ → AAGA
0 → GATG	1 → TCTA	2 → CGCG	3 → ACTC
4 → ACAA	5 → TATC	6 → AAGG	7 → TGCC
8 → TTGT	9 → GGGT	: → GACG	; → GCAT
¡ → TACG	= → ACAC	¿ → TAGA	? → AGTA
@ → CGAA	A → GTGA	B → AACT	C → TTCG
D → AGCG	E → CCTT	F → ACTT	G → AACA
H → CTGA	I → CGTC	J → TGTC	K → AGAT
L → CTGT	M → GTCG	N → CACC	O → AGGG
P → GGTA	Q → ATAC	R → CTCC	S → CTTA
T → CGAC	U → CCCG	V → ACGA	W → TTAT
X → AGGT	Y → GGTC	Z → AGGA	[→ GACA
\ → TGAT	] → GTTA	ˆ → GAGT	_ → CCTC
‘ → AGAA	a → GTGC	b → AAAC	c → CAAT
d → CGCC	e → CGGC	f → CACG	g → GCAA
h → AGCA	i → AATC	j → CGTA	k → GTAG
l → TTTC	m → CTGC	n → GGCT	o → GGGG
p → TGGC	q → TGGG	r → ACCG	s → ATGT
t → GGGC	u → GCCG	v → CCAG	w → CTGG
x → ATAG	y → GAGA	z → TGCT	{ → CTCA
|→ TGAC	} → TCAT	˜ → CCGA	

```
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 $RK_i, i = 1, 2, 3, \ldots, 14$ are generated using the DDST table. The round keys must be generated in ascending order starting from RK_1 to RK_{14}. For RK_i, to generate the round key RK_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_i 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_i 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 $RK_i = STR$. 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 sequence from the DDST table. The DNA sequences are given as Block1 = ”TTCGGGGGCTGCTG-GCGCCGGGGCCGGCACCGGATCAGGGACCG-GCAAGTGCGGCTAATCTGCT” and Block2 = ”GTGCGGGCAATCGGGGGGCT”.
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 = 1111011010101010011110011110100 1
1001011010101001011010010001011010001101
0010101000010110100100001011100110100111
0000110111100111
Block2 = 1011100110101001000011011010101 0
10100111
```

5) User E_1 as input and generate the round key RK_1 as described in section II-C.
6) Divide RK_1 into a number of chunks $C_1, C_2, \ldots C_L$ of equal size 64 bits, where L denotes the number of chunks.
7) For all $j = 1, 2, 3, \ldots, L$, set $Block1 = Block1 \oplus C_j$ as follows:

$$\begin{aligned} Block1 &= Block1 \oplus C_1 \\ Block1 &= Block1 \oplus C_2 \\ Block1 &= Block1 \oplus C_3 \\ &\vdots \\ Block1 &= Block1 \oplus C_L \end{aligned}$$

8) Repeat step 7 to perform the XOR operation on Block2 and chunks $C_1, C_2, \ldots C_L$.
9) For the remaining round keys $RK_i, i = 2, 3, \ldots, 14$, repeat steps 5 to 8 to get:

```
Block1 = 0101100101101110001000001100 01110
0110100110100101011110101010110110001101
0010101000010110100100001011100110100111
0000110111100111
Block2 = 0001011001101101010101011100110 1
01011000
```

10) Convert both Block1 and Block2 to DNA sequence using substitutions 00 = A, 01 = C, 10 = G, 11 = T.
11) Set DSK=E_{16}. The value of E_{16} is obtained during the generation process of the round key RK_{14}.
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.

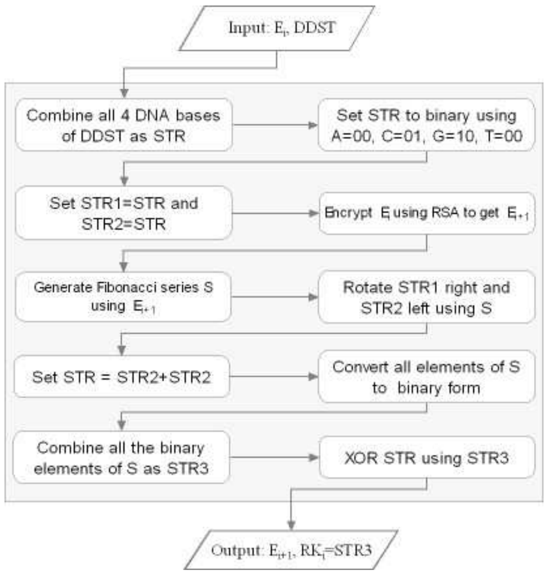
Input: E_i, DDST
Combine all 4 DNA bases of DDST as STR
Set STR to binary using A=00, C=01, G=10, T=00
Set STR1=STR and STR2=STR
Encrypt E_i using RSA to get E_{i+1}
Generate Fibonacci series S using E_{i+1}
Rotate STR1 right and STR2 left using S
Set STR = STR2+STR2
Convert all elements of S to binary form
Combine all the binary elements of S as STR3
XOR STR using STR3
Output: E_{i+1}, RK_i=STR3

Fig. 4. Block Diagram of Generating Round key RK_i.

14) Set X to the length of SDK and convert it to 16 bits binary format.
15) Convert X to DNA sequence using substitutions 00 = A, 01 = C, 10 = G, 11 = T.
16) Finally, set Block1 followed by Block 2 as a sandwich between X and DSK which represents the final CT as follows:

```
ciphertext = AAAAACTTCCGCCGTGAGACG
ATGCGGCGGCCCTGGGGTCGATCAGGGAC
CGGCAAGTGCGGCTAATCTGCTACCGCGT
CCCCCTATCCCGATTCAGAGAGATAATTA
CCGATTCACACCGGA
```

17) The sender sends the CT to the receiver over a public communication channel.

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 and 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 E_{16} =DSK.
8) For $i = 14, 13, 12 \ldots, 1$, follow the steps below:
    a) Decrypt E_{i+2} using RSA with secret key e to get the number E_{i+1}.

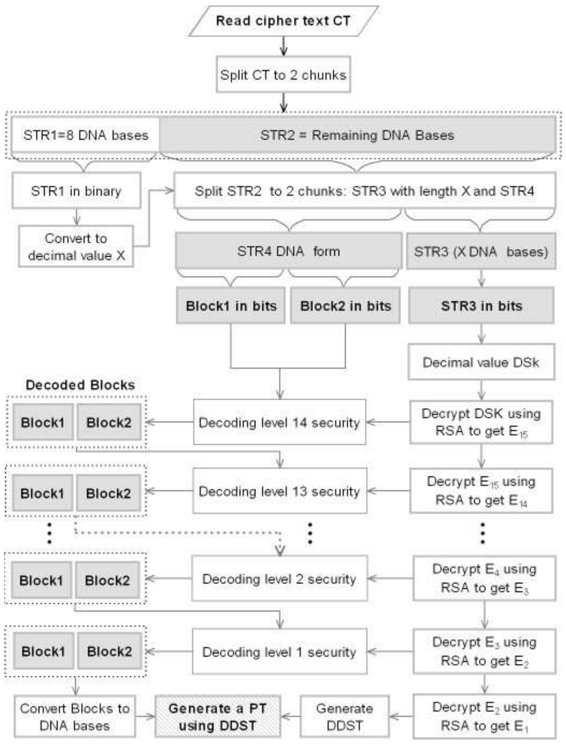
Read cipher text CT
Split CT to 2 chunks
STR1=8 DNA bases
STR2 = Remaining DNA Bases
STR1 in binary
Split STR2 to 2 chunks: STR3 with length X and STR4
Convert to decimal value X
STR4 DNA form
STR3 (X DNA bases)
Block1 in bits
Block2 in bits
STR3 in bits
Decimal value DSk
Decoded Blocks
Block1
Block2
Decoding level 14 security
Decrypt DSK using RSA to get E15
Decoding level 13 security
Decrypt E15 using RSA to get E14
Decoding level 2 security
Decrypt E4 using RSA to get E3
Decoding level 1 security
Decrypt E3 using RSA to get E2
Convert Blocks to DNA bases
Generate a PT using DDST
Generate DDST
Decrypt E2 using RSA to get E1

Fig. 5. Block Diagram of Decryption Process.

b) Generate the round key RK_i as described in section II-C.

c) Divide RK_i into chunks $C_1, C_2, \ldots C_L$ of equal size 64 bits, where L denotes the number of chunks.

d) For all $j = L, L-1, L-2, \ldots, 1$, set $Block1 = Block1 \oplus C_j$ as follows:

$$\begin{aligned} Block1 &= Block1 \oplus C_L \\ Block1 &= Block1 \oplus C_{L-1} \\ Block1 &= Block1 \oplus C_{L-2} \\ &\vdots \\ Block1 &= Block1 \oplus C_1 \end{aligned}$$

e) Repeat step (d) to perform the XOR operation on Block2 and chunks $C_1, C_2, \ldots C_L$.

9) Decrypt E_2 using RSA with secret key e to get the number E_1.

10) Use E_1 to generate the DDST table as illustrated in section II-B.

11) Starting from left to right, replace each four DNA bases in **str4** with its corresponding alphabet character from the DDST table.

12) The resulting string represents the PT.

III. Performance Evaluation

The proposed asymmetric DNA encryption algorithm based on the RSA cryptography system is conducted in JAVA platform. 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,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.

Plaintext	No. of Matched	Randomization Degree
Computer Organization	2	98%
Multiprogramming	0	100%
5555333388887777	0	100%
AA112233445566FE	1	99%
Aljouf university	0	100%

TABLE IV. The DDST table randomization with one bit difference.

Bit index	PT changes	No. of unchanged 4 DNA values	Randomization Degree
3	**c**omputer Organization	0	100%
13	Co**E**puter Organization	0	100%
27	Comp**U**ter Organization	0	100%
35	Compu**V**er Organization	1	98.94%
47	Comput**m**r Organization	0	100%
53	Computer Organiz**A**tion	0	100%
99	Computer Organizat**y**on	1	98.94%
156	computer Organizatio**N**	0	100%

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]. TThis 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 $B_{changed}$ and the total number of bits in the cipher text is B_{total}. In such cases, the Aavalanche Eeffect (AE) is given as [26], [27]:

$$AE = \frac{B_{changed}}{B_{total}} \times 100\%$$

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 Org**Q**nization" 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.

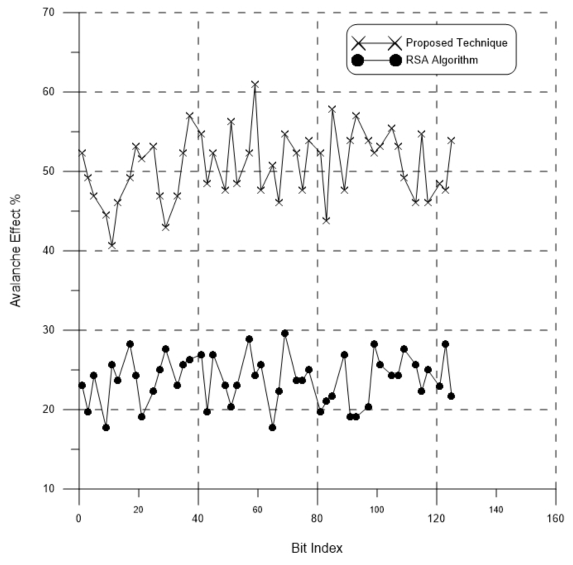

Fig. 6. Comparison of Avalanche Test For the Proposed Technique and RSA Algorithm

TABLE V. ROUND KEYS AVALANCHE EFFECT.

Round key index	No. of bits changed	Avalanche test
1	390.0	51.32%
2	383.0	50.4%
3	388.0	51.21%
4	391.0	51.45%
5	389.0	51.19%
6	387.0	50.93%
7	390.0	51.32%
8	379.0	49.87%
9	379.0	49.87%
10	395.0	51.98%
11	386.0	50.79%
12	393.0	51.72%
13	384.0	50.53%
14	373.0	50.21%

TABLE VI. COMPARISON OF THE NUMBER OF BITS CHANGED.

Encryption Algorithm	Average No. of bits
Proposed Technique	64.895
RSA Algorithm	36.212

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="AA112233445566F**D**" 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="AA112233445566F**A**" and a new CT is generated. These processes are repeated until the bit number 125 in PT is flipped to get the new PT="**Q**A112233445566FE" 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 RAS 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

[1] P. Langley *et al.*, "Selection of relevant features in machine learning," in *Proceedings of the AAAI Fall symposium on relevance*, vol. 184, 1994, pp. 245–271.

[2] K. Javed, S. Maruf, and H. A. Babri, "A two-stage markov blanket based feature selection algorithm for text classification," *Neurocomputing*, vol. 157, pp. 91–104, 2015.

[3] N. Azizi, N. Farah, M. T. Khadir, and M. Sellami, "Arabic handwritten word recognition using classifiers selection and features extraction/selection," *Recent Advances in Intelligent Information Systems*, pp. 735–742, 2009.

[4] N. Azizi, Y. Tlili-Guiassa, and N. Zemmal, "A computer-aided diagnosis system for breast cancer combining features complementarily and new scheme of svm classifiers fusion," *International Journal Of Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, pp. 45–58, 2013.

[5] L. Zhang, F. Xiang, J. Pu, and Z. Zhang, "Application of improved hu moments in object recognition," in *IEEE International Conference on Automation and Logistics*. IEEE, 2012, pp. 554–558.

[6] S. Das, S. Das, B. Bandyopadhyay, and S. Sanyal, "Steganography and steganalysis: different approaches," *arXiv preprint arXiv:1111.3758*, 2011.

[7] T. Mandge and V. Choudhary, "A dna encryption technique based on matrix manipulation and secure key generation scheme," in *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*. IEEE, 2013, pp. 47–52.

[8] F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems," in *International Workshop on Multiple Classifier Systems*. Springer, 2001, pp. 78–87.

[9] P. Mahajan and A. Sachdeva, "A study of encryption algorithms aes, des and rsa for security," *Global Journal of Computer Science and Technology*, 2013.

[10] J. Zhang, D. Fang, and H. Ren, "Image encryption algorithm based on dna encoding and chaotic maps," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[11] I. Peterson, "Hiding in dna," *Proceedings of Muse*, vol. 22, 2001.

[12] J. D. Watson *et al.*, "Molecular biology of the gene." *Molecular biology of the gene.*, no. 2nd edn, 1970.

[13] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson, "Energieumwandlung: Mitochondrien und choroplasten," *Molekularbiologie der Zelle (Original: Molecular biology ofthe cell, Third edition). Jaenicke, L.(ed.). Weinheim: VCH Ver-lagsgesellschaft mbH*, pp. 771–851, 1995.

[14] D. Nelson and M. Cox, "Lehninger principles of biochemistry, (worth, new york, 2000)," *Google Scholar*.

[15] A. Leier, C. Richter, W. Banzhaf, and H. Rauhe, "Cryptography with dna binary strands," *Biosystems*, vol. 57, no. 1, pp. 13–22, 2000.

[16] S. T. Amin, M. Saeb, and S. El-Gindi, "A dna-based implementation of yaea encryption algorithm." in *Computational Intelligence*, 2006, pp. 120–125.

[17] H. Shiu, K.-L. Ng, J.-F. Fang, R. C. Lee, and C.-H. Huang, "Data hiding methods based upon dna sequences," *Information Sciences*, vol. 180, no. 11, pp. 2196–2208, 2010.

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

[19] B. A. Mitras and A. Abo, "Proposed steganography approach using dna properties," *international journal of information technology and business management*, vol. 14, no. 1, pp. 96–102, 2013.

[20] C. Guo, C.-C. Chang, and Z.-H. Wang, "A new data hiding scheme based on dna sequence," *Int. J. Innov. Comput. Inf. Control*, vol. 8, no. 1, pp. 139–149, 2012.

[21] D. S. Chouhan and R. Mahajan, "An architectural framework for encryption & generation of digital signature using dna cryptography," in *International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2014, pp. 743–748.

[22] E. M. S. Hossain, K. M. R. Alam, M. R. Biswas, and Y. Morimoto, "A dna cryptographic technique based on dynamic dna sequence table," in *Computer and Information Technology (ICCIT), 2016 19th International Conference on*. IEEE, 2016, pp. 270–275.

[23] K. Sajisha and S. Mathew, "An encryption based on dna cryptography and steganography," in *Electronics, Communication and Aerospace Technology (ICECA), 2017 International conference of*, vol. 2. IEEE, 2017, pp. 162–167.

[24] H. Al-Mahdi, O. Shahin, Y. Fouad, and K. Alkhaldi, "Design and analysis of dna binary cryptography algorithm for plaintext," *International Journal of Engineering and Technology*, vol. 10, pp. 699–706, 2018.

[25] 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.

[26] F. H. Nejad, S. Sabah, and A. J. Jam, "Analysis of avalanche effect on advance encryption standard by using dynamic s-box depends on rounds keys," in *2014 International Conference on Computational Science and Technology (ICCST)*, Aug 2014, pp. 1–5.

[27] S. Ramanujam and M. Karuppiah, "Designing an algorithm with high avalanche effect," *IJCSNS International Journal of Computer Science and Network Security*, vol. 11, no. 1, pp. 106–111, 2011.