DNA Sequences Compression by GP² R and Selective Encryption Using Modified RSA Technique

SYED MAHAMUD HOSSEIN¹, DEBASHIS DE², (Senior Member, IEEE), PRADEEP KUMAR DAS MOHAPATRA³, SANKAR PRASAD MONDAL⁴, ALI AHMADIAN⁵.⁶, FERIAL GHAEMI⁷, AND NORAZAK SENU⁶

¹Department of Computer Science, Vidyasagar University, Midnapore 721102, India
²Department of Computer Science and Engineering, Maulana Abdul Kalam Azad University of Technology, Nadia 741249, India
³Department of Microbiology, Raiganj University, Raiganj 733134, India
⁴Department of Applied Science, Maulana Abdul Kalam Azad University of Technology, Nadia 741249, India
⁵Institute for Mathematical Research (INSPERM), Universiti Putra Malaysia (UPM), Selangor 43400, Malaysia
⁶Institute of Industry Revolution 4.0, National University of Malaysia, Selangor 43600, Malaysia
⁷Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia (UKM), Selangor 43600, Malaysia

Corresponding authors: Ali Ahmadian (ahmadian.hosseini@gmail.com) and Norazak Senu (norazak@upm.edu.my)

This work was supported in part by the Ministry of Education, Malaysia, under Grant FRGS/1/2018/STG06/UPM/02/2, and in part by the Universiti Putra Malaysia.

ABSTRACT Humans, by nature, have always been fascinated by the possibility of being able to acquire more information in minimum possible time and space. The effective lossless compression method, effective data structure, and DNA (Deoxyribonucleic Acid) data searching are quite essential as they provide a stimulus to easy accessibility and communication. The proposed algorithm is a new Lossless Compression algorithm, which compresses data, based on two tiers. Firstly, it searches for the exact Genetic Palindrome(GP), Palindrome(P) and Reverse(R) [GP²R] and the substring is reported, which is replaced by the corresponding ASCII character creating a Library file. By using the ASCII code, the Library file acts as a signature as well as provides the security of data. Secondly, modified RSA technique is proposed for the selection encryption purpose. This selection encryption of the modified RSA technique is an approach to lessen computational resources for greatly sized DNA facts. The experimental work shows 44% to 45% original sequence is encrypted where above 95% of the original file is damaged by using this method. This technique can find out the 3.851273 bits per base of the compression rate. The O(n) is the complexity of this algorithm. The running time is a few seconds of this algorithm. This is a hybrid approach to the compression & encryption process. For reducing the compression rate, the first pass output is again compressed by the second pass but it is lossy, This experiment is performed on benchmark DNA order.

INDEX TERMS Reverse, genetic palindrome, palindrome, compression, ratio, rate, encryption, speed, genetic palindrome, palindrome, and reverse (GP²R).

I. INTRODUCTION

The amount of DNA being taken from organisms and order is increasing exponentially [1]. This gives in two questions - a place for storing and safe transmission. The hard question of place for storing while useful to the workplace is depending on the size of each base. The DNA order size vary from Megabyte (MB) to Terabyte (TB) annually [2]–[8]. The DNA contains some logical organization [9], hence data structure for storing, accessing and efficient processing tasks is challenging [10], [11]. The DNA database requires an efficient compression algorithm for storing. The available compression methods [12], [13] cannot be applied to biological data aptly because the DNA sequences have some specialty [14]. The DNA data of a living organism are nonrandom, so the two bits encoding techniques cannot be applied directly and has some limitations [14]. Huffman’s lossless compression technique both of the static and the adaptive model are not well applicable due to the presence of very less number of different characters in a DNA sequence [10], [14]. The phenomenal characteristics of genomic data have conation, so there occurs many repetitions within the DNA...
sequence [10]. Some specific structures [10], [15], [16] are present in DNA order, to which researchers have kept in mind and proposed several DNA compression algorithms. It is not an easy task to find out the exact GP²R match position in a long DNA sequence. The offered algorithm has three parts: i) firstly, finding all the exact GP²R and ii) secondly, encoding GP²R and the non-match regions iii) finally encrypt the compressed file, library file or in both by using modified RSA technique. The basic principle of this algorithm is quick and sensitive homology searching [17], as our exact GP²R search engine. The substring technique creates an online library file and the ASCII characters are placed on the source file. Also developed another algorithm of one to one character matching, sequence orientation change and measurement size of files, etc.

In accordance with its function, DNA shows different properties from other kinds of facts. The compression algorithms for text files utilize short repeated patterns and contextual likeness to get compression. These techniques cannot be successfully applied to DNA. The compression diminishes the file size and the process of encryption makes certain the safety of one text record which is to be sent over some unreliable network like the internet. Many algorithms have undergone growth. Each of which has its own forces and feeblenesses. The data compression tries to minimize storing space and encrypting a nucleotide genome order from unauthorized uses. If the genome order is sent from source to destination and unauthorized user can access the order and make changes in it by putting forward some west information, the structure of the genome will change and this will lead to loss of unused power. The order is compressed with the help of encryption, keeping in mind that it does not lead to data loss.

Now a day’s sending over the wireless of DNA/RNA/PROTEIN order is very essential. The computational price and a small place for storing things is the demand for mass. Information safety is a hard question to keep safe the facts from hackers. This offered technique keeps safe the facts from unauthorized users. Also, make a comparison of the selection encryption technique with the RSA algorithm. This algorithm also applies to artificial facts. The process is mentioned in Fig. 1.

The offered algorithm consists of three forms: i) discovering all exact GP²R substring ii) encode exact match of GP²R region & unmatched region iii) Encrypt the compressed file, library file or in both using modified RSA technique.

This proposed compression method provided two-tier safety i) the data are compressed and encrypt the order, produces two separate text records one at a time and each text record has in it ASCII code of 256 different characters ii) applying selective encryption of modified RSA technique.

This is a hybrid model of compression and security techniques This model was proposed in a strong, hybrid cryptosystem. If selection encryption is applied before compression, we observed very low-level safety because DNA order has only four symbols, any unauthorized user can decrypt the sequence by trial and error method and also select options are less. The compressed output text has more symbols than the input text. After compression the selection encryption techniques are applied easily, getting very high-level security and also increase the selection option.

This is a two-pass method, in first pass use GP²R technique and in second pass use modified RSA’s techniques, where first pass output uses the input of the second pass and finally getting the ultimate result at the end.

**FIGURE 1.** Compression encryption process.

### II. EXISTING COMPRESSION ALGORITHMS

All genome compression method used redundancy within the order but differs greatly in the way they do so. In general compression algorithms can be put in order into Naive Bit manipulation, Dictionary-based, statistical and referential algorithms. Most of the compression method used today, including the DNA compression falls into two groups. First is
the statistical method, which compresses facts by giving
another in place of a more having general approval special
sign to a small sign. Second is designed on a dictionary that
compress facts by replacing long orders with small informa-
tion to a similar order in a dictionary.

The statistical compression technique like the statistical
process, CTW and arithmetic coding compresses the DNA
sequences well. But the Huffman technique is inapplicable on
the DNA sequence. Both the algorithms Lempel-Ziv'78 and
Lempel-Ziv'77 works on this principle. In GS compress,
LZ77 design with reverse complement is introduced as a
dictionary-based design. Rivals et al. [18] developed Cfact,
is another compression algorithm, which using a suffix tree
data structure and searches the longest matching repeat.
Based on approximate string matching property, Sadeh pro-
posed a lossy compression technique. The limitation of CTW,
arithmetic coding is low decompression speed but the com-
pression rate is good.

Biocompress [19] specially design for DNA
compression was offered by Grumbach and Tahi. Based on
order-2 finite context arithmetic encoder the Biocompress is
modified as Biocompress-2 [11]. Sliding window-based algo-
rithm offered by Lempel and Ziv [20], is familiar as LZ77.
The compression algorithm like Cfact [21], off line [22],
DNASC [23] and B2DNR [24], etc are based on the common
characteristic of sequence repetitions. Shibata et al. Proposed
the Boyer-Moore algorithm based on compressed pattern
matching. In the 1st phase of DNA compress [25], [26]
use patternHunter tool which finds out the highest score
of complimentary palindrome and approximate repeats and
encodes in the 2nd phase.

The popular DNA compression algorithm is
GenCompress-I [25] based only on replacement opera-
tion. The GenCompress-2, the modified version of
GenCompress-I is based on deletion and insertion operation
in the subsequence and the compression performance is the
same in both the cases.

GENBIT compress tool [27] is designed by Rajeswari and
Apparao specially for DNA sequence compression based on
binary bit coding. They also introduced another DNA com-
pression algorithm called HUFFBIT [28]. It produced better
results than GENBIT. DNABIT Compress tool(DBC) [29]
was also designed by them. It used binary bit ‘in the
bit-preprocessing stage’ of DNA smaller part repeats and
reverse.

III. EXISTING SELECTION ENCRYPTION ALGORITHMS

The RSA, DES, 3DES, and AES are popular encryption
algorithms discussed in this paper. The use of the net and
network is growing quickly. So there are more requirements to
secure the facts sent over different networks using different
services. To provide the network safety and facts different
encryption methods are used [30]. In this paper, a survey of
the currently existing works on the encryption techniques has
been done. Every method has its own importance and works
in its own special characteristic way. For data storage and

retrieval we have used the compression-encryption algorithm
parallelly and enhance the security level.

The idea of selective encryption with a purpose of prob-
abilistically selective encryption algorithm was proposed by
Ren et al. [31].

Kala [32] proposed Quadrature Mirror Filters and lossless
compression technique for wireless ad hoc network based on
selection encryption.

Singh and Samaddar [33] have used the selection [34]
encryption technique in RSA based on a singular cubic curve
for the text-based Documents.

IV. MOTIVATION & CONTRIBUTION

The main purpose of this work is to analyze the good effect
of executing the method of compression and encryption. First
the sequence is grouped into three/four bases, replaced by
special sign and as a result, we get compression encryption
simultaneously. This process reduces the complexity of the
standard procedure. The secure private key is generated by a
group of nucleotide bases which is replaced by a single ASCII
code and a private key is known only who is transmitting the
sequence. The decompression process is known to everyone
but the private key is not known to everyone, except the
transmitting user, the process is known as cryptanalysis, and
it is very reliable. This selection encryption efficiently first
searches the sensitive region of the DNA sequence. By using
Lavenshtein distance (LD) we have calculated the effective-
ness of the process.

V. PROPOSED TECHNIQUE OF GENETIC PALINDROME,
PALINDROME AND REVERSE

A. METHODOLOGY OF GP^2 R TECHNIQUE

Consider a string S consisting of four symbols g,t,c & a
and the size is b byte. The process is graphically shown in
Fig. 2.

The substring tta is the Genetic palindrome (GP) of aat
shown twice in the string, the substring ata is the palin-
drome(P) of ata shown twice in the string and gta is the
reverse(R) of atg shown twice in the string. After compression
the string is s= &c#@&g#@&g#c@

The string has 31 characters and requires 31 bytes for
storing, after compression required 15 bytes which is less than
input file size and corresponding Library file are aat—&, ata–
# and atg–@. We can take special character in some case this
case is also considered. i.e N

Searching for exact Reverse, genetic palindrome, palin-
drome, encoding analysis and decoding procedure, details
discussed in this paper [35].
B. FLOWCHART
C. SELECTIVE ENCRYPTION BY USING MODIFIED RSA TECHNIQUE AND COMPRESSION, DECOMPRESSSION, ENCRYPTION & DECRYPTION ALGORITHM

The selection encryption can be applied to the compressed text in the following ways i) chose a single character ii) chose numeric number iii) chose the pattern.

| 1st pass compression algorithm based on GP^2R |
|-----------------------------------------------|
| INITIALIZATION OF INPUTS:                    |
| 1. Cellular & artificial order use as source file |
| 2. Compressed file and library file is the target file |

ESTIMATED OUTPUT:
Compressed file*COM and library file *LIB

Step 1: enter the input file name

Step 2: Repeat for i=0 to i<1-4.
fcom[i]=finp[i].
Repeat for i=0 to i<1-4
flib[i]=finp[i].

Step 3: Do then if ch==127 || ch==128 || ch==129 || ch==141 || ch==143 || ch==144 || ch==157 || ch==160
If (ch==’a’ || ch==’t’ || ch==’g’ || ch==’c’) ||
(ch+72==’a’ || ch+72==’t’ || ch+72==’g’ || ch+72==’c’) ||
(ch+144==’a’ || ch+144==’t’ || ch+144==’g’) ||
(ch+144==’c’ || ch+144==’c’)
If t==4 then break.

Step 4: While !feof(inp) then if a!=’a’&& a!=’t’&& a!=’g’&& a!=’c’) &&
(b!=’a’&& b!=’t’&& b!=’g’&&
b!=’c’) &&
(c!=’a’&& c!=’t’&& c!=’g’&& c!=’c’).
else if(a!=’a’&& a!=’t’&& a!=’g’&& a!=’c’) &&
(b!=’a’&& b!=’t’&& b!=’g’&& b!=’c’) &&
(c==’a’|| c==’t’|| c==’g’|| c==’c’).
else if(a!=’a’&& a!=’t’&& a!=’g’&& a!=’c’) &&
(b!=’a’&& b!=’t’&& b!=’g’&& b!=’c’) &&
(c==’a’|| c==’t’|| c==’g’|| c==’c’) &&
flag==0).
Step 5: Repeat for i=0 to i<3. then fclose(lib).
Repeat for i=0 to i<3 then rs[i]=s[3-1-i],rs[3]y=NULL.
Repeat for i=0 to i<3 then
if(s[i]==’a’) then gs[i]=’a’.
else if(s[i]==’t’) then gs[i]=’t’.
else if(s[i]==’g’) then gs[i]=’g’.
else if(s[i]==’c’) then gs[i]=’c’.
else then if feof(inp) putc(a,com).putc(b,com).putc(c,com).break;
else then fclose(com).

Step 6: While !feof(inp) then if a!=’a’&& a!=’t’&& a!=’g’&& a!=’c’
If feof(inp) then b=getc(inp)
If feof(inp) then c=getc(inp)

Step 7: If strcmp(s,s1)==0 then putc(ch,com).
else if strcmp(rs,s1)==0 then putc(ch+72,com).
else if strcmp(gs,s1)==0 then
putc(ch+144,com).
Else if feof(inp) then break;
Else then putc(a,com);

Step 8: While !feof(com) then a=getc(com)
If feof(com) then break.
While !feof(inp) then do (m==255) and break.

Step 9: While !feof(inp) then fclose(inp) and fclose(com)
ntime needed for execution is=%2.3f",difftime(end,beg)
and remove(finp)
End for loop
End while loop

Step 10: End

| 2nd pass compression algorithm based on GP^2R |
|-----------------------------------------------|
| INITIALIZATION OF INPUTS:                    |
| 1. Compressed file used as a source file |
| 2. 2nd pass compressed file is the target file |

ESTIMATED OUTPUT:
The 2nd passed compressed file is *COM and library file *LIB

Step 1: enter the input file name

Step 2: Repeat for i=0 to i<1-4.
fcom[i]=finp[i].
Repeat for i=0 to i<1-4
flib[i]=finp[i].

Step 3: Do then if ch==127 || ch==128 || ch==129 || ch==141 || ch==143 || ch==144 || ch==157 || ch==160
If (ch==’a’ || ch==’t’ || ch==’g’ || ch==’c’)
(ch+72==’a’ || ch+72==’t’ || ch+72==’g’)
End while loop
Step 4: While !feof(inp) then if (a!=’a’ & & a!=’t’ & & a!=’g’ & & a!=’c’) & & (b! =’a’ & & b!=’t’ & & b!=’g’ & & b!=’c’) & & (c!=’a’ & & c!=’t’ & & c!=’g’ & & c!=’c’) & & (d!=’a’ & & d!=’t’ & & d!=’g’ & & d!=’c’).  
else if(a!=’a’ & & a!=’t’ & & a!=’g’ & & a!=’c’) & & (b!=’a’ & & b!=’t’ & & b!=’g’ & & b!=’c’) & & (c!=’a’ & & c!=’t’ & & c!=’g’ & & c!=’c’).  
else if((a==’a’ || a==’t’ && a==’g’ && a==’c’ && b==’a’ && b==’t’ && b==’g’ && b==’c’ && c==’a’ || c==’t’ || c==’g’ || c==’c’). 
else if((a==’a’ || a==’t’ && a==’g’ && a==’c’ && b==’a’ && b==’t’ && b==’g’ && b==’c’ && c==’a’ || c==’t’ || c==’g’ || c==’c’ && c==’c’). 
else ifflag==0). 

Step 5: Repeat for i=0 to i<3, thenfclose(lib).  
Repeat for i=0 to i<3 then rs[i]=s[3-1-i], rs[3]=NULL.  
Repeat for i=0 to i<3 then if(s[i]==‘a’) then gs[i]=‘t’.  
else if(s[i]==‘t’) then gs[i]=‘a’.  
else if(s[i]==‘g’) then gs[i]=‘c’.  
else if(s[i]==‘c’) then gs[i]=‘g’.  
else then iffeof(inp) putc(a,com),putc(b,com),putc(c,com).  
break;  
else then fclose(com).  

Step 6: While !feof(inp) then if a!=’a’ & & a!=’t’ & & a!=’g’ & & a!=’c’ and continue 
If feof(inp) then b=getc(inp) and break.  
If feof(inp) then c=getc(inp) 

Step 7: If strlen(s,s1)==0 then putc(ch,com).  
else if strlen(rs,s1)==0 then putc(ch+72,com).  
else if strlen(gs,s1)==0 then putc(ch+144,com).  
  Else if feof(inp) then break;  
  Else then putc(a,com);  

Step 8: While(!feof(com)) then a=getc(com);  
If (feof(com)) then break.  
While (!feof(inp)) then  
Do if (m==255) then break.  
While !feof(inp).  
Time needed for execution is=%2.3f”,dfttime(end,beg).  

Step 9: End

1st pass Decompression algorithm based on GP²R

INITIALIZATION OF INPUTS:
- Initializing characters: (a,t,c,g)
- Decompressed file is ip=fopen(vbx,”r”).

ESTIMATED OUTPUT:
- output file is op=fopen(vbx,”w”).

Step 1: Enter the name of the compression file:

Step 2: Repeat for i=0 to i<7 then finp[i]=fcom[i].  
Repeat for i=0 to i<7 then flib[i]=fcom[i].

Step 3: While !feof(com) then  
Do if eof(com) then goto swar1.
While !feof(lib) then if feof(lib) then goto swar1
Ifch==’a’ ||ch==’t’ || ch==’g’ || ch==’c’ then 
putc(ch,inp) and gotoswar;

Step 4: If (ch==ch5) then gotoswar  
else if(ch==ch5+72) then gotoswar.  
  else if(ch==ch5+144) then
  if(ch==’a’) then putc(‘t’,inp)
  else if(ch==’t’) then putc(‘a’,inp)
  else if(ch==’g’) then putc(‘c’,inp)
  else if(ch==’c’) then putc(‘g’,inp)

Step 5: if(ch2==’a’) then putc(‘t’,inp)
else if(ch2==’t’) then putc(‘a’,inp)
else if(ch2==’g’) then putc(‘c’,inp)
else if(ch2==’c’) then putc(‘g’,inp).

Step 6: 
if(ch3==’a’) then putc(‘t’,inp).
else if(ch3==’t’) then putc(‘a’,inp).
else if(ch3==’g’) then putc(‘c’,inp).
else if(ch3==’c’) then putc(‘g’,inp) and gotoswar.  
else if(ch==’a’ ||ch==’t’ || ch==’g’ && ch==’c’) then 
  if(iefocom) and goto swar1.
else if(chi==’a’ &&ch!=’t’ &&ch!=’g’ &&ch!=’c’) then continue

Step 7: While !feof(com) then swar1
While !feof(inp) then if (m==255) then break.
Print “decompression complete successfully”.

Step 8: End.
2nd pass Decompression algorithm based on GP²R

INITIALIZATION OF INPUTS

- Initializing characters: (a,t,c,g)
- Decompressed file is ip=fopen(vbx,“r”).

ESTIMATED OUTPUT:

- output file is op=fopen(vbx,“w”).

Step 1:
Enter the name of the compression file:

Step 2:
If(argc!=2) then exit(0);
Repeat for i=0 to i<l-8 then fibl[i]=fcom[i];
Repeat for i=0 to i<l-5 then fnp[i]=fcom[i];
Repeat for i=0 to i<l-8 then fib[i]=fcom[i];

Step 3:
While (!feof(lib)) then if(feof(lib)) then break.
Repeat for i=0 to i<k-1 then ch=fgetc(lib);
Repeat for i=0 to i<k-5 then ch1=fgetc(lib1);
Repeat for i=0 to i<k-1 then ch2=fgetc(lib2);

Step 4:
if(i==k) then store[k+]=xxx.
While (!feof(lib)) then if(feof(lib)) then break.
Repeat for i=0 to i<k-5 then ch=fgetc(lib);
Repeat for i=0 to i<k-1 then
if(i==i) then xx=replace[j]+72 |
else then continue.

Step 5:
if(i==k-2) then x2[k+]=store[i].
if(s5==0) then s6=(k-1)/2)-1;
else then s6=((k-2)/2)-1;
do then if(feof(com)) then goto swar1;
while (!feof(lib)) then if(feof(lib)) goto swar1;
Repeat for i=0 to i<k-4 then if(ch=replace[i] |
ch=replace[i]+72 | ch=replace[i]+144 | ch=’a’ | |
ch=’t’ | ch=’g’ | ch=’c’) then goto swar
Else then continue.

Step 6:
if(ch=’a’ | ch=’t’ | ch=’g’ | ch=’c’) then break
Repeat for i=0 to i<k3-1 then if(ch=x2[i]) goto swar.
Else continue.

Step 7:
If(ch=ch5) then Repeat for i=0 to i<k-2-1
if(ch=x1[i]) then break;
else then continue.
else if(ch=store[s+s6+1]) then break.
else then continue;

Step 8:
while(!feof(com)) then swar1w
while(!feof(inp)) then
if(m==255) then break.
Decompression complete.

Step 9:
End.

Algorithm for “encryption” using modified RSA algorithm

Step 1:
if(b==0) then return(a) and end of if block
else return(b,a%b);
End of else block

Step 2:
longint result = 1.
while (e > 0)
then if ((e & 1) == 1) result = (result * b) and e >>= 1
b = (b * b) % m
End of while loop.

Step 3:
if(!(fp1=fopen(argv[?],“r”))) then print “Enter a valid path!” and exit end of if block.
if(!(fp3=fopen(“C:\crystloc.tmp”,“w”))) then print “Could not create temporary file!” and exit end of if block.
if(!(fp2=fopen(“C:\Crypt.txt”,“w”))) then print “Could not create output file!” and exit end of if block.

Step 4:
switch(opt)
case 1:while(c!=EOF) then if(c>’0’&&c<’9’) else
and end of if else block and while loop and break.
case 2:while(sw1[i]!‘0’i++) and end of while block
do if(c==sw1[?]!
if((strcmp(s,sw1)) then i=i+strlen(sw1) and end of if block.
elsei++;;
else
if(c==‘n’) i++
i++;while(c!=EOF);
break; end of case 2.
Step 5: Default and exit.

Step 6: 

if(!(fp3=fopen("C:\cryoff.tmp","w"))) then print “Could not create temporary file!” and exit. End of if block.
if (tracker==0) then print “Enter a number to generate keys.”.
do while((gcd(m,e)!=1)
while(=((+m)%e)!=0) and x++ then d=(+m)/e and end of while loop.

Step 7: End.

Algorithm for “decryption” using modified RSA technique

Step 1:
if(b==0) then return(a)
else return(b,a%b);
End of else block.

Step 2:
unsigned long int result = 1
while (e > 0) then
if (((e & 1) == 1)then result = (result * b) % m; and e
\gg= 1
b = (b * b) % m;
End of while loop.

Step 3:
If(!(fp3=fopen("C:\cryloc.tmp","r"))) then print “Required temporary file not found!” and exit.
End of if block.

Step 4:
i=0 while(c!=EOF) then fscanf(fp3,"%d",&arr[i])
if(c==EOF) then arr[i]=1 and i++ end of while loop.

Step 5:
switch(opt)
case 1: if(!(fp1=fopen("C:\Crypt.txt","r"))) then print “Encrypted file not found! and exit End of if block.
if(!(fp3=fopen("C:\Output.txt","w"))) then print “Output file could not be created!” and exit. End of if block.
while(1) then if(i==arr[j])j++ end of while loop
if(c==EOF) break and end of if block and case 1

case 2: if(!(fp1=fopen("C:\Crypt.txt","r"))) then print “Encrypted file not found!” and exit. End of if block.
if(!(fp3=fopen("C:\cryoff.tmp","r"))) then print “Required temporary file not found!” and exit. End of if block.
while(c!=EOF)

VI. RESULTS & DISCUSSION OF GENETIC PALINDROME, PALINDROME & REVERSE TECHNIQUE

This algorithm of genetic palindrome, palindrome & reverse tested on standard benchmark data used in [11]. For testing purposes use two sets (data set-1 & 2) of data, they come under different sources. The data set-3 of some real DNA sequences taken from Gen Bank. Encryption ratio (ER): This criterion measures the ratio between the size of an encrypted part and the whole data size.

For Cellular sequences, the results are presented graphically in Fig. 4 for the data set-1 and 7 for data set-2. The Fig. 4 & 7 considers only compressed file, shows that the compression rate is dependent on file size. The minimum average compression rate is 3.58077 bits/base for data set-1 and 3.58072 bits/base for the data set-2 where word size is 3 and sequence orientation is complement. The compression rate is increased when the word size increases. Also, compression time increases when the word size increases from 3 to 4. So, word size 3 base compressions are better than word size 4 or 5. The nature of the graph is heterogeneous in nature because sequences come under from different species as shown in Fig. 4 & 7 for both the data sets. The result shows that the increase in file size decreases the compression rate. The disk utilization and encryption rate are shown in Fig. 5 for data set-I and 8 for data set-II where it is shown that both changes are parallel with file size.

For Artificial data, the results are presented graphically in Fig. 4 for the data set-1 and 7 for data set-2. The Fig. 4 & 7 considers only the compressed file, which shows that the compression rate is dependent on file size as well as word size. The minimum average compression rate is 3.60244 bits/base for data set-1 and 3.60539 bits/ base for the data set-2, where word size is 3 and sequence orientation is a compliment. The nature of the graph is homogeneous in nature because sequences are randomly generated as shown in Fig. 4 & 7. The cellular sequences versus artificial data, getting distinct fig. with naked eye shows two different graph characteristic. Where observed that cellular sequences have structure and nonrandom data, whereas random data is unstructured. Also observed that library file is constant in size.
FIGURE 3. Step by step design process.

FIGURE 4. File size versus compression rate of cellular & artificial sequence of data set-1.

FIGURE 5. File size versus disk utilization and encryption rate of data set-1.

FIGURE 6. Cellular & artificial sequence compression and decompression time of data set-1.

FIGURE 7. File size versus compression rate of cellular & artificial sequence of data set-2.

The Fig. 6 & 7 shows the compression and decompression time of cellular & artificial sequences of both the data set. It is shown that cellular sequence compression and decompression timeless than artificial sequences.

If sequences encrypt by three/four-character secret keys (Genetic palindrome, palindrome & reverse technique, sequences compressed by the sub-sequence/word of different
Figure 8. File size versus disk utilization & encryption rate of cellular & artificial sequence of data set-2.

Figure 9. Cellular & artificial sequence compression and decompression time of data set-2.

Figure 10. File size versus % effect on actual text of data set-1.

Figure 11. Encryption versus compression rate of data set-1.

Figure 12. Encryption versus effect on actual text of data set-1.

Figure 13. File size versus percentage effect on actual text of data set-2.

size), calculate the percentage of encryption and percentage of modification of the actual text. From this, it is observed that an average 44%-45% for both the data set-I & II of encryption on the actual text will be modified 95% (for both the data set) on the actual file.

The entropy is increased two to three times before and after compression. As a result, both the compressed file and library file increased the randomness, so, the attacker can not attack the sequence easily. The Fig. 10, 11 & 12 for data set-1 and 13, 14 & 15 shows the encryption is in increasing order. The Fig. 15 for data set-I & 16 for the data set-2 shows that before and after compression, the entropy of compressed file and library file and entropy is increased in both the cases.

We test the programme on real DNA sequences, result shown in Figure 18. The real data is collected from National Center for Biotechnology Information site.
(www.ncbi.nlm.nih.gov/genbank). Also this technique overcome the lossy compression. This result is presented in Fig. 18. It is shown that the compression rate and ratio are the same in all the data sets.

Now, after first pass compression, these two sets of DNA orders are converted into simple text files of another size and find out the result on it’s as a percentage of encryption and percentage of effect on actual file by changing the level. □

In selection encryption, the file is encrypted based on the basic principle of selection i.e selection of a single character, double character, and alphanumeric character. Find out the
result on data set in the original file, compressed file and library file. The result shows that RSA exhibits the highest avalanche effect. The Avalanche effect gives us the extent of the diffusion of the message. One bit of change in the plaintext brings about the significant change in bits of the ciphertext. The data is presented in Fig. 19 to 22 for both the data set for the original file, Fig. 23 to 28 for compressed file and Fig. 29 to 31 for the library file. It was shown that the decryption time is always less than the encryption time and independent of file size for both the data set. The encryption rate & avalanche effect increased with file size in both the data set. If consider the highest level of Hamming
distance, the effect on the original file is highest on the basis of top-level interchanging. On the other at a lower level Hamming Distance the effect is proportional. The encryption is increased with respect to output text effectiveness. The percentage of effect on actual file in increased when input file size is increased and vice-versa.

Now using an appropriate selection of character decode the encrypted text and get back the original text as getting in our previous experiment. But if decrypt without applying an appropriate selection of character value or entered an incorrect key the message will be different.

Table 1 shown the encryption & decryption throughput. The result showing the decryption throughput is less than encryption throughput. Also observed that data set-2 throughput is better than data set-1. Also observed that selection encryption applied on compressed file is better than original file encryption.

The encryption & decryption time is also graphically presented for both the data set is presented in Fig. 19 & 21 for source file, 23 & 26 for compressed file and 29 for library file. It is observed that the encryption time is always greater than decryption time. The encryption and decryption time is independent of file size. This is the basic principle for all

FIGURE 26. Time in encryption & decryption vs. file size of data set-2 1 of compressed file.

FIGURE 27. File size versus avalanche effect of data set-2 1 of compressed file.

FIGURE 28. File size versus encryption rate of data set-2 1 of compressed file.

FIGURE 29. Time in encryption & decryption vs. file size of library file.

FIGURE 30. File size versus avalanche effect of library file.

FIGURE 31. File size versus encryption rate of library file.
Researchers S. M. Hossein et al. describe DNA Sequences Compression by GP² R and Selective Encryption. They present the throughput of the process in Table 1 and the compression rate using the 2nd pass compression technique in Table 2. The compression algorithms include AES, DES, RSA, and DNA. These techniques are highly effective for security purposes.

Also, the compression score is calculated as 2.32453 for data set-1 and 3.88363 for data set-2. The algorithm is compression friendly because of no impact on data compression efficiency.

The second pass result is shown in Table 2. It is shown that data is lost in the second pass.

| Techniques                  | Data set-I          | Data set-II          |
|-----------------------------|---------------------|----------------------|
|                             | Encryption throughput (byte/sec) | Decryption throughput (byte/sec) | Encryption throughput (byte/sec) | Decryption throughput (byte/sec) |
| GP²R                        | 3773.024           | 196881.2             | 3564.75               | 159113.2            |
| Selection encryption on original file | 33748.15           | 50528.9             | 8362.282              | 12351.55            |
| Selection encryption on compressed file | 24944.41           | 39426.04            | 5220.515              | 8308.944            |
| Selection encryption on library file       | 360                 | 577.2414            | Independent of data set |
VII. CONCLUSION

The results show that the compression rate & ratio varies from each other due to the data set comes from different sources. This algorithm is very helpful for storing the DNA database. Our algorithm stores the DNA sequence as a record in the database without maintaining them as files. By utilizing the decrypt decompression algorithm instantly we can get the original sequence at the client end without any error. This algorithm is user-friendly.

The experimental results show that the reverse, palindrome and genetic palindrome matching patterns are similar in all kinds of sources. A major part is played by the lookup table in finding the regularities and similarities of DNA order.

The output text file has an ASCII symbol and non matched c,t,g & a, providing information safety. It is very useful for data transmission and provides data protection. This process protects the particular source of the DNA sequence. Here we can get better security than a static lookup table (LUT).

Internal of genetic palindrome, palindrome & reverse is the key idea of our algorithm. This algorithm act as a DNA sequence compression model that brings out the real features of the DNA order. The output of our experiments also shows that our process is better than other standard processes such as the RAY compression ratio [36]. By using our method the regularities in DNA sequence like crossover and mutation are detected. This algorithm fails to attain a high compression rate and ratio than other standard methods but proves that information security is very high. The encryption performance depends on speed, our algorithm shows that the modified RSA algorithm speed is better than the RSA algorithm. The limitation of this research work, if nonmatch base pair and ASCII code again compressed, we can not provide corresponding ASCII code, because this compression is one pass. But using other orientations shows no meaningful changes.

Important observations are:

i) Genetic palindrome, Palindrome & reverse subsequence size vary from 2 to 5 and no match found in case the subsequence size becoming more than six.

ii) The substring length three is highly repeated than 4 or 5 bases long substring. So, 3 bases substring is more compressible than other substring length.

iii) The cellular DNA sequence is more compressible than other orientations.

iv) It is observed that the cellular DNA sequence compression rate & ratio are distinguishably different because this data set is collected from different species. The artificial data generated by the random string generation process, compression rate & ratio are similar both the data set.

v) We have developed a second pass compression algorithm for reducing compression rate and ratio but the second pass algorithm is lossy, we cannot consider this algorithm in our work.

vi) Our algorithm works more efficiently on a short pattern than a long pattern.

vii) The compression-encryption output file efficiency increases and is user-friendly.

viii) Here using GP²R technique we can get more than four characters in the output file, in that situation we can apply Huffman’s technique easily.

ix) Modified RSA technique also tests other compression methods such as repeat, reverse, repeat-reverse.. etc

x) This algorithm also compares with some real DNA sequence taken from Gen Bank but compression rate and ratio are the same as benchmark data.

Our method provides higher information security than other standard methods. The first pass process produces two separate files, each having more than four symbols. If two files are transmitted one by one, then the decrypt of the file by unauthorized persons is very hard. Also, the first-pass output file contains 256 symbols, so the selection option increases and getting very strong safety.

From these tables & graphs, it is observed that cellular DNA sequences have a logical organization, structure, systematic and nonrandom whereas artificial data are random and unstructured. Also observed that the cellular DNA sequence compression ratio follows the equation as (1-Output/2*Input) whereas the artificial data, the compression ratio is followed by the equation as (1-Output/input); where the output size is a number of bits. Also, the same table shows the average compression gain of the sequences, observed that lower the compression rate, compression gain is high.

REFERENCES

[1] A. Jahaan, T. N. Ravi, and S. P. Arokiaraj. “Bit DNA squeezer (BDNAS): A unique techniques for data compression,” Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol., vol. 2, no. 4, pp. 512–517, 2017.

[2] T. Matsumoto, K. Sadakane, and H. Imai. “Biological sequence compression algorithms,” Genome Inform., vol. 11, pp. 43–52, Jan. 2000.

[3] I. Karsch-Mizrachi, Y. Nakamura, and G. Cochrane. “The international nucleotide sequence database collaboration,” Nucleic Acids Res., vol. 40, no. 1, pp. 33–37, 2012.

[4] S. Deorowicz and S. Grabowski. “Robust relative compression of genomes with random access,” Bioinformatics, vol. 27, no. 21, pp. 2979–2986, Nov. 2011.

[5] C. Brooksbank, G. Cameron, and J. Thornton. “The European bioinformatics institute’s data resources,” Nucleic Acids Res., vol. 38, no. 1, pp. 17–25, 2010.

[6] M. Shumway, G. Cochrane, and H. Sugawara. “Archiving next generation sequencing data,” Nucleic Acids Res., vol. 38, pp. 870–871, Jan. 2010.

[7] M. Kapushesky, I. Emam, E. Holloway, P. Karnson, A. Zorin, J. Malone, G. Rustici, E. Williams, H. Parkinson, and A. Brazma. “Gene expression atlas at the European bioinformatics institute,” Nucleic Acids Res., vol. 38, no. 1, pp. D690–D698, Jan. 2010.

[8] G. Korodi, L. Tabus, J. Rissanen, and J. Astola. “DNA sequence compression–based on the normalized maximum likelihood model,” IEEE Signal Process. Mag., vol. 24, no. 1, pp. 47–53, Jan. 2007.

[9] H. Deepak. “State of the art: DNA compression algorithms,” Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 3, no. 10, pp. 397–400, 2013.

[10] N. Bakr and A. Sharawi. “DNA lossless compression algorithms: Review,” Amer. J. Bioinf. Res., vol. 3, pp. 72–81, Nov. 2013.

[11] S. Grumbach and F. Tahi. “A new challenge for compression algorithms: Genetic sequences,” Inf. Process. Manage., vol. 30, no. 6, pp. 875–886, Nov. 1994.

[12] X. Chen, S. Kwong, and M. Li. “A compression algorithm for DNA sequences and its applications in genome comparison,” Genome Inform., vol. 10, pp. 52–61, Feb. 1999.
(2013). *International Nucleotide Sequence Database Collaboration.* [Online]. Available: http://www.insdc.org

R. Giancarlo, D. Scatutto, and F. Uuto, “Textual data compression in computational biology: A synopsis,” *Bioinformatics*, vol. 25, no. 13, pp. 1575–1586, Jul. 2009.

O. Nalbantoglu, D. Russell, and K. Sayood, “Data compression concepts and algorithms and their applications to bioinformatics,” *Entropy*, vol. 12, no. 1, pp. 34–52, Dec. 2009, doi: 10.3390/e12010034.

B. Ma, J. Tromp, and M. Li, “PatternHunter: Faster and more sensitive homology search,” *Bioinformatics*, vol. 18, no. 3, pp. 440–445, Mar. 2002.

E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange, “A guaranteed compression scheme for repetitive DNA sequences,” LIFL Lille I Univ., Lille, France, Tech. Rep. IT-285, 1995.

S. Grumbach and F. Tahi, “Compression of DNA sequences,” in *Proc. IEEE Symp. Data Compress. Conf. (DCC)*, Snowbird, UT, USA, Mar./Apr. 1993, pp. 340–350.

J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977.

E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange, “A guaranteed compression scheme for repetitive DNA sequences,” in *Proc. Data Compress. Conf. (DCC)*, 1996, p. 453.

A. Apostolico and S. Lonardi, “Compression of biological sequences by greedy off-line textual substitution,” in *Proc. Data Compress. Conf. (DCC)*, 2000, pp. 143–152.

K. N. Mishra, D. A. Aagarwal, D. E. Abdelhadi, and D. P. C. Srivastava, “An efficient horizontal and vertical method for online DNA sequence compression,” *Int. J. Comput. Appl.*, vol. 3, no. 1, pp. 39–46, 2010.

S. Roy and S. Khatta, “Compression algorithm for all specified bases in nucleic acid sequences,” *Int. J. Comput. Appl.*, vol. 75, no. 4, pp. 29–34, 2013.

X. Chen, S. Kwong, and M. Li, “A compression algorithm for DNA sequences and its applications in genome compression,” in *10th Workshop Genome Inform. (GIW)*, vol. 10, K. Asai, S. Miyano, and T. Takagi, Eds., Tokyo, Japan, 1999, pp. 51–61.

X. Chen, S. Kwong, and M. Li, “A compression algorithm for DNA sequences,” *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 4, pp. 61–66, Jul. 2001.

P. R. Rajeswari, and A. Apparao, “Genbit compress tool (GBC): A Java-based tool to compress DNA sequences and compute compression ratio (BIITS/BASE) of genomes,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, no. 3, pp. 181–191, 2010.

D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

T. Matsumoto, K. Sadakane, H. Imai, and T. Okazaki, “Nucleotide archival format (NAF) enables efficient lossless reference-free compression of DNA sequences,” *Bioinformatics*, vol. 35, no. 19, pp. 3826–3828, Oct. 2019.

K. Kryukov, M. T. Ueda, S. Nakagawa, and T. Imanishi, “Nucleotide archival format (NAF) enables efficient lossless reference-free compression of DNA sequences,” *Bioinformatics*, vol. 35, no. 19, pp. 3826–3828, Oct. 2019.

SYED MAHAMUD HOSSEIN received the master’s degree in computer applications from Swami Ramanand Teerth Marathwada University, Nanded, in 1998, and the M.Eng. degree in information technology from the West Bengal University of Technology, Kolkata, in 2005.

DEBASHIS DE (Senior Member, IEEE) received the M.Tech. degree in radio physics and electronics, in 2002, and the Ph.D. degree in engineering from Jadavpur University, in 2005. He worked as a Research and Development Engineer with Teletronics. He is currently a Professor with the Department of Computer Science and Engineering, West Bengal University of Technology, India, and an Adjunct Research Fellow with the University of Western Australia, Australia. His research interests include location management and power consumption control in mobile networks and low power nano device design for mobile application and disaster management. He was awarded the prestigious Boyscast Fellowship from the Department of Science and Technology, Govt. of India, to work at Herriot-Watt University, U.K. He is also awarded the Endeavour Fellowship Award, from 2008 to 2009 by DEST Australia to work in the University of Western Australia. He received the Young Scientist Award in 2005 at New Delhi and in 2011 at Istanbul by the International Union of Radio Science, Belgium.

SYED MAHAMUD HOSSEIN received the master’s degree in computer applications from Swami Ramanand Teerth Marathwada University, Nanded, in 1998, and the M.Eng. degree in information technology from the West Bengal University of Technology, Kolkata, in 2005.

DEBASHIS DE (Senior Member, IEEE) received the M.Tech. degree in radio physics and electronics, in 2002, and the Ph.D. degree in engineering from Jadavpur University, in 2005. He worked as a Research and Development Engineer with Teletronics. He is currently a Professor with the Department of Computer Science and Engineering, West Bengal University of Technology, India, and an Adjunct Research Fellow with the University of Western Australia, Australia. His research interests include location management and power consumption control in mobile networks and low power nano device design for mobile application and disaster management. He was awarded the prestigious Boyscast Fellowship from the Department of Science and Technology, Govt. of India, to work at Herriot-Watt University, U.K. He is also awarded the Endeavour Fellowship Award, from 2008 to 2009 by DEST Australia to work in the University of Western Australia. He received the Young Scientist Award in 2005 at New Delhi and in 2011 at Istanbul by the International Union of Radio Science, Belgium.

PRADEEP KUMAR DAS MOHAPATRA received the Ph.D. degree from Vidyasagar University, in 2008. He is currently working as an Assistant Professor with the Department of Microbiology, Vidyasagar University. He has significantly contributed in the field of enzyme biotechnology, environmental, and industrial microbiology. He has worked on microbial nutrition, fermentation, probiotics, tissue engineering, and bioinformatics. He has published more than 41 original research articles in reputed national and international journals, one book chapter, three popular articles, and 55 abstracts. He also attended nine training programmes on microbial research and covered 24 national and international conferences. He has received research funds from UGC, Govt. of India and Biostadt India Ltd. He has received the Junior Scientists of the Year Award, in 2008 and the Environmentalist of the Year Award, in 2009, from NESA, New Delhi. He is nominated as a Fellow of the Society for Applied Biotechnology, in 2011. He is also serving as an Editorial Board Member of two journals and reviewers of many journals of Elsevier, Blackwell, Springer, and Taylor & Francis. He is also associated with many academic and research societies, such as AFB (France), NESA (New Delhi), BRSI (Trivandrum), FBVM (Kolkata), and SAB (Dhaward). He is currently acting as the General Secretary of Biotech Research Society of India, Vidyasagar University Unit.
SANKAR PRASAD MONDAL was an Assistant Professor with the Department of Mathematics, Midnapore College (Autonomous) and with the National Institute of Technology, Agartala, for five years. He is currently an Assistant Professor with the Department of Natural Science, Maulana Abul Kalam Azad University of Technology, Haringhata, India. He is having five years of research experience in the field of operations research, differential equation, fuzzy sets, mathematical biology, and fuzzy differential equation.

ALI AHMADIAN received the Ph.D. degree in applied mathematics from the Universiti Putra Malaysia, Selangor, Malaysia, in 2013. He is currently a Fellow Researcher with Universiti Putra Malaysia. He has published more than 70 peer-reviewed scientific publications. He is a Reviewer for 70 international journals. He was involved in several national and international projects related to the applications of fuzzy systems in the real-world systems. His current research interests include artificial intelligence, fuzzy fractional calculus, interval-valued functions, numerical simulations, and mathematical modeling.

FERIAL GHAEMI received the Ph.D. degree in nanotechnology from the Institute for Advanced Technology, Universiti Putra Malaysia, in 2015. She joined the Institute for Tropical Forestry and Forest Products, in 2016, where she is currently a Postdoctoral Research Fellow. She is a University Lecturer with the National University of Malaysia. She has published more than 30 research works in prestigious CIJ journals. Her main research interests include the synthesis of different types of nanomaterials with applications arising in drug delivery, polymer composites, and microextraction techniques.

NORAZAK SENU is currently an Associate Professor with the Institute for Mathematical Research, Universiti Putra Malaysia. As his main interests are working on different types of differential equations and modeling real-world systems using such equations, he published more than 100 articles in the peer-reviewed international journals. He obtained several prizes for his research works from the Ministry of Education, Malaysia, and achieved a number of governmental grants to support his scientific works.

***