Chapter 5

DNA SEQUENCES COMPRESSION USING GP^{2}R AND SELECTIVE ENCRYPTION USING MODIFIED RSA TECHNIQUE

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)[GP^2R] and the substring is reported, which is replaced by the corresponding ASCII character creating a Library file . By using ASCII code, Library file acts as a signature as well as provided the security of data. Secondly, modified RSA technique is proposed for the selection encryption purpose. This selection encryption of 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 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 few second of this algorithm. This is a hybrid approach of compression & encryption process. For reducing 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.

Keyword : Reverse, genetic palindrome, palindrome, compression, ratio, rate, encryption, speed.

Abbreviation of GP^2R : Genetic Palindrome(GP), Palindrome(P) and Reverse(R)

1 Introduction

The amount of DNA being taken from organisms and order is increasing exponentially [12]. This gives in two questions- place for storing and safe transmission. The hard question of place for storing while useful to work place is depending on the size of each base.The DNA order size vary from Megabyte (MB) to Terabyte(TB) annually [29, 82,19,83-86]. The DNA contains some logical organization [7], hence data structure for storing, accessing and efficient processing task is challenging [8,48]. The DNA database requires an efficient compression algorithm for storing. The available compression methods [70,68,73] cannot be applied on biological data aptly because the DNA sequences have some speciality [4]. The DNA data of a living organism are non random, so the two bits encoding techniques cannot

be applied directly and has some limitations [4]. 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 [8,4]. The phenomenal characteristics of genomic data have conation, so there occurs many repetitions within the DNA sequence [8]. Some specific structures[8,87,9] 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 GP^2R match position in a long DNA sequence. The offered algorithm has three parts: i) firstly, finding all the exact GP^2R and ii) secondly, encoding GP^2R and the non-match regions iii) finally encrypt the compressed file, library file or in both by using modified RSA technique. The basic principal of this algorithm is quick and sensitive homology searching [88], as our exact GP^2R 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 kind of facts. The compression algorithms for text file utilize short repeated pattern 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 their 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 is able to access the order and make changes in it by putting forward some west information, the structure of the genome will totally 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 wireless of DNA/RNA/PROTEIN order is very essential. The computational price and a small place for storing things is the demand of mass. The information safety is a hard question to keep safe the facts from hackers. This offered technique keep safe the facts from unauthorised user. Also make a comparison of the selection encryption technique with the RSA algorithm. This algorithm also applies on artificial facts.

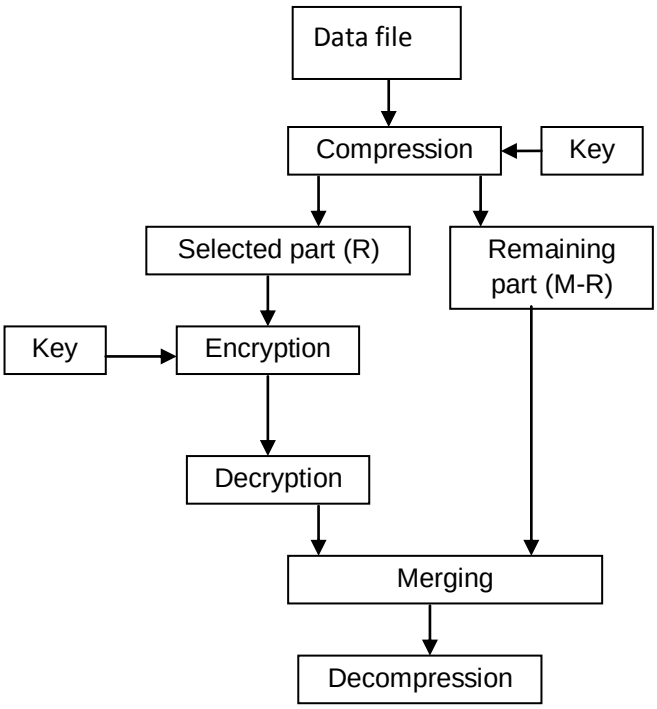

Fig. 5.1 compression encryption process

The offered algorithm consist of three forms: i) discovering all exact GP^2R substring ii) encode exact match of GP^2R region & unmatch 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 record 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. A small part is selected from the complete message is the principal of selective encryption process and remaining part of the message is in clear where the safety is not put at risk. By using proposed selection encryption in RSA increase the speed of encryption. The R part selection is important from the view point of security and R depends on medium type. In selection encryption technique a part of the bit stream (R) is used and not the whole bit stream (M). Here M is the plain text message and selective part is R. So, M-R is the unencrypted part. It is observed that the security level increases if R increases, also time of encryption increases. The R-part is selectively encrypted using RSA technique based on single or multiple or alphanumeric character selection. If decrypt without applying key value or entered a wrong key the text will be of different manner.

This is a two pass method, in first pass use GP^2R technique and in second pass use modified RSA's techniques, where first pass output uses the input of second pass and finally getting the ultimate result at the end.

2 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 statistical method, which compress 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 by small information to similar order in a dictionary.

The statistical compression technique like 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-Ziv78 and Lempel-Ziv77 works on this principle. In GS compress, LZ77 design with reverse complement is introduced as a dictionary based design. E. Rivals et al.[89] developed Cfact, is an another compression algorithm, which using suffix tree data structure and searches the longest matching repeat. Based on approximate string matching property, Sadeh proposed lossy compression technique.The limitation of CTW, arithmetic coding is low decompression speed but the compression rate is good.

Biocompress [55] specially design for DNA compression was offered by Grumbach and Tahi. Based on order-2 finite context artithmetic encoder the Biocompress is modified as Biocompress-2[48]. Sliding window based algorithm offered by Lempel and Ziv[90], is familiar as LZ77.The compression algorithm like Cfact[91], off line[56], DNASC[21] and B2DNR[92] etc are based on the common characteristic of sequence repititions. Shibata et al. Proposed Boyer-Moore algorithm based on compressed pattern matching. In the 1^{st} phase of DNA compress [70-71] use patternHunter tool which finds out the highest score of complimentary palindrome and approximate repeats and encode in the 2^{nd} phase.

The popular DNA compression algorithm is GenCompress-1[93] based only on replacement operation. The GenCompress-2, the modified version of GenCompress-I is based on deletion

and insertion operation in the subsequence and the compression performance is same in both the cases.

GENBIT compress tool[94] is designed by Rajeswari and Apparao specially for DNA sequence compression based on binary bit coding. They also introduced another DNA compression algorithm called HUFFBIT[49]. It produced better results than GENBIT. DNABIT Compress tool(DBC)[76] was also designed by them. It used binary bit 'in the bit-preprocessing stage' of DNA smaller part repeats and reverse.

3 Existing Selection Encryption Algorithms

The RSA, DES, 3DES and AES are popular encryption algorithm 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 [95]. 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 on its own special characteristic way. For data storage and retrieval we have used compression-encryption algorithm parallely and enhance the security level.

The idea of selective encryption with a purpose of probabilistically selective encryption algorithm was proposed by Yonglin Ren, Azzedine Boukerche, Lynda Mokdad[96].

S.Kala[97] proposed Quadrature Mirror Filters and lossless compression technique for wireless ad hoc network based on selection encryption.

Kalpana singh and Shefalika ghosh samaddar[98] have used the selection[99] encryption technique in RSA based on singular cubic curve for the text based Documents.

4 **Motivation & contribution**

The main purpose behind 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 over the standard procedure. The secure private key is generated by a group of nucleotide bases which is replaced by a single ASCII code and 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 every one, 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 effectiveness of the process.

5 Proposed technique of Genetic Palindrome, Palindrome and Reverse

5.1 Methodology of GP^{2}R technique

Consider a string S consisting of four symbols g,t,c and a

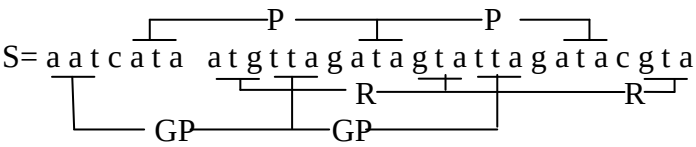

Fig.5.2 procedure of process

The substring tta is the Genetic palindrome (GP) of aat shown twice in the string, the substring ata is the palindrome(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#c@&ag#@&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--@

5.2 Searching process

Searching for exact Reverse, genetic palindrome, palindrome, encoding analysis and decoding procedure, details discussed in this paper[100].

5.3 Selective Encryption by using modified RSA technique

If selection encryption is applied before compression, we observed very low level safety because DNA order has only four symbol, any unauhorised user can decrypt the sequence by trail and error method and also selection options are less. The compressed output text have more symbols than input text. After compression the selection encryption techniques are applied easily, getting very high level security and also increase the selection option.

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

5.4 Compression ,decompression ,encryption & decryption algorithm

1^{st} 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:
Repeatfor i=0 to i<l-4.
fcom[i]=finp[i].
Repeatfor i=0 to i<l-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')
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').
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]=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 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

2^{nd} pass compression algorithm based on GP^2R

INITIALIZATION OF INPUTS:
1.Compressed file use as a source file
2. 2^{nd} pass compressed file is the target file

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

Step 1:
enter the input file name

Step 2:
Repeatfori=0 to i<l-4.
fcom[i]=finp[i].
Repeatfori=0 to i<l-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')
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').
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. 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 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' and continue
    If feof(inp) then b=getc(inp) and break.
  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
Doif(m==255) then break.
While !feof(inp).
Time needed for execution is=%2.3f",difftime(end,beg).

Step 9:
End
```

1st pass Decompression algorithm based on GP^2R

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<l-7 then finp[i]=fcom[i].
Repeat for i=0 to i<l-7 then flib[i]=fcom[i].

Step 3:
While !feof(com) then
Do if feof(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(ch1=='a') then putc('t',inp)
else if(ch1=='t') then putc('a',inp)
else if(ch1=='g') then putc('c',inp)
else if(ch1=='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(feof(com) and goto swar1 .
else if(ch!='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.

2^{nd} pass Decompression algorithm based on GP^2R

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 flib1[i]=fcom[i];
Repeat for i=0 to i<l-5 then finp[i]=fcom[i];
Repeat for i=0 to i<l-8 then flib[i]=fcom[i];

Step 3:
While (!feof(lib1)) then if(feof(lib1)) then break.
Repeat for i=0 to i<5 then ch1=fgetc(lib1);
Repeat for i=0 to i<k then print replace[i]
Repeat for i=33 to i<255 then
Repeat for j=0 to j<k-4 then if(i==127 || i==128 || i==129 || i==141 || i==143 || i==144 || i==157 || i==160) then break
else if(xxx==replace[j] || xxx==replace[j]+72 || xxx==replace[j]+144 || ch=='a' || ch=='t' || ch=='g' || ch=='c') then break.
Else then continue;

Step 4:
if(j==k) then store[k1++]=xxx.
While (!feof(lib)) then if(feof(lib)) then break.
Repeat for i=0 to i<5 then ch=fgetc(lib);
Repeat for i=0 to i<k1 then
Repeat for j=0 to j<k2-1 then
if(i==127 || i==128 || i==129 || i==141 || i==143 || i==144 || i==157 || i==160) then continue.
if(store[i]==x1[j] || store[i]==x1[j]+50) then break
else then continue.

Step 5:
if(j==k2-1) then x2[k3++]=store[i].
if(s5==0) then s6=((k1-1)/2)-1;
else then s6=((k1-2)/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<k2-1
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[0],"r"))) then print "Enter a valid path!" and exit end of if block .
if(!(fp3=fopen("C:\\cryloc.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[0])
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(((1+x*m)%e)!=0) and  x++ then d=(1+x*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 >>= 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)
 while(sw1[i]!='\0') then rep[i]=RSA(sw1[i]) i++;  andend of while loop
if(!(fp3=fopen("C:\\Output.txt","w"))) then print "Output file could not be created! and exit.
End of if block
while(c!=EOF)
  if(i==arr[j]) then i=i+strlen(sw1)-2  and end of if block
elsei++  and end of if block and while loop..
 break; end of case 2.
 Default and exit.

Step 6:
if (tracker==0) then   print "Enter private key:  "  and   return C  and end of if block.

End.
```

6 Results & discussion of Genetic Palindrome, Palindrome & Reverse technique

This algorithm of genetic palindrome, palindrome & reverse tested on standard benchmark data used in [48]. For testing purpose use two sets (data set-1 & 2) of data they comes under different sources.

The compression & encryption result are presented in different table. The merge process, 2nd pass result, throughput result[101] and comparative result are shown in different Table also.

Table 5.1 compression rate, ratio, space utilization and encryption rate of data set-1

			Cellular DNA sequences							Artificial DNA sequences						
						Compression Time							Compression Time			
sequence orientation	Sequence Name	Sequence Size	Reduce file size Byte(C)	Compression ratio	Compression rate(bits /base)	Encode Time	Decode Time	Percentage of disk utilization	% Encryption ratio	Reduce file size Byte(C)	Compression ratio	Compression rate(bits /base)	Encode Time	Decode Time	Percentage of disk utilization	% Encryption ratio
Normal Orientation	MTPACGA	100314	44784	-0.78575	3.57151	28.24	0.219	55.36	44.64	45220	-0.80314	3.60628	27.91	0.219	54.9	45.08
	MPOMTCG	186608	83484	-0.78951	3.57901	55.98	0.494	55.26	44.74	84468	-0.8106	3.6212	54.67	0.494	54.7	45.26
	CHNTXX	155844	69900	-0.7941	3.5882	45.16	0.384	55.15	44.85	70204	-0.8019	3.60381	46.59	0.384	55	45.05
	CHMPXX	121024	53886	-0.781	3.562	32.08	0.274	55.47	44.53	54484	-0.80077	3.60153	32.03	0.329	55	45.02
	HUMGHCSA	66495	29717	-0.78762	3.57525	16.75	0.109	55.31	44.69	29945	-0.80134	3.60268	16.59	0.109	55	45.03
	HUMHBB	73308	32996	-0.8004	3.60081	18.51	0.109	54.99	45.01	32926	-0.79658	3.59317	18.57	0.109	55.1	44.91
	HUMHDABCD	58864	26470	-0.79872	3.59744	13.18	0.109	55.03	44.97	26486	-0.79981	3.59962	14.01	0.054	55	45
	HUMDYSTROP	38770	17396	-0.79479	3.58958	7.3	-0.001	55.13	44.87	17440	-0.79933	3.59866	7.47	-0.001	55	44.98
	HUMHPRTB	56737	25503	-0.79798	3.59596	13.62	0.054	55.05	44.95	25557	-0.80179	3.60357	13.68	0.109	55	45.04
	VACCG	191737	85123	-0.77583	3.55166	50.49	0.494	55.6	44.4	86121	-0.79665	3.5933	55.93	0.494	55.1	44.92
	HEHCMVCG	229354	102550	-0.7885	3.577	68.24	0.659	55.29	44.71	103304	-0.80165	3.6033	68.51	0.604	55	45.04
	Average				3.58077			55.24				3.60246			54.96	
Reverse Orientation	MTPACGA	100314	44692	-0.78208	3.56417	23.95	0.219	55.45	44.55	45230	-0.80354	3.60707	27.91	0.219	54.9	45.09
	MPOMTCG	186608	83780	-0.79585	3.5917	52.3	0.494	55.1	44.9	84132	-0.8034	3.60679	54.72	0.494	54.9	45.08
	CHNTXX	155844	70090	-0.79898	3.59796	41.64	0.384	55.03	44.97	70178	-0.80124	3.60247	43.73	0.384	55	45.03
	CHMPXX	121024	53700	-0.77485	3.54971	29.83	0.274	55.63	44.37	54312	-0.79508	3.59016	32.08	0.274	55.1	44.88
	HUMGHCSA	66495	29783	-0.79159	3.58319	16.53	0.109	55.21	44.79	29951	-0.8017	3.6034	16.09	0.109	55	45.04
	HUMHBB	73308	32770	-0.78807	3.57614	18.29	0.109	55.3	44.7	33074	-0.80466	3.60932	18.73	0.109	54.9	45.12
	HUMHDABCD	58864	26260	-0.78445	3.5689	14.01	0.109	55.39	44.61	26518	-0.80198	3.60397	13.9	0.109	55	45.05
	HUMDYSTROP	38770	17404	-0.79562	3.59123	7.41	-0.001	55.11	44.89	17420	-0.79727	3.59453	7.52	0.054	55.1	44.93
	HUMHPRTB	56737	25445	-0.79389	3.58778	12.91	0.109	55.15	44.85	25655	-0.8087	3.61739	12.85	0.164	54.8	45.22
	VACCG	191737	85817	-0.79031	3.58061	52.3	0.494	55.24	44.76	86283	-0.80003	3.60006	56.53	0.494	55	45
	HEHCMVCG	229354	102002	-0.77894	3.55789	69.83	0.604	55.53	44.47	103238	-0.8005	3.601	69.39	0.659	55	45.01
	Average				3.57721			55.28				3.60329			54.95	
Complement Orientation	MTPACGA	100314	44784	-0.78575	3.57151	29.17	0.274	55.36	44.64	45220	-0.80314	3.60628	27.08	0.219	54.9	45.08
	MPOMTCG	186608	83484	-0.78951	3.57901	56.93	0.494	55.26	44.74	84468	-0.8106	3.6212	53.35	0.494	54.7	45.26
	CHNTXX	155844	69900	-0.7941	3.5882	44.94	0.384	55.15	44.85	70204	-0.8019	3.60381	45.27	0.384	55	45.05
	CHMPXX	121024	53886	-0.781	3.562	31.97	0.274	55.47	44.53	54484	-0.80077	3.60153	31.97	0.274	55	45.02
	HUMGHCSA	66495	29717	-0.78762	3.57525	16.59	0.109	55.31	44.69	29945	-0.80134	3.60268	16.59	0.109	55	45.03
	HUMHBB	73308	32996	-0.8004	3.60081	18.35	0.109	54.99	45.01	32926	-0.79658	3.59317	18.62	0.109	55.1	44.91
	HUMHDABCD	58864	26470	-0.79872	3.59744	13.02	0.109	55.03	44.97	26484	-0.79967	3.59935	14.01	0.109	55	44.99
	HUMDYSTROP	38770	17396	-0.79479	3.58958	7.14	-0.001	55.13	44.87	17440	-0.79933	3.59866	7.41	0.054	55	44.98
	HUMHPRTB	56737	25503	-0.79798	3.59596	5.62	0.054	55.05	44.95	25557	-0.80179	3.60357	13.62	0.054	55	45.04
	VACCG	191737	85123	-0.77583	3.55166	50.6	0.494	55.6	44.4	86121	-0.79665	3.5933	55.98	0.494	55.1	44.92
	HEHCMVCG	229354	102550	-0.7885	3.577	68.24	0.604	55.29	44.71	103304	-0.80165	3.6033	68.57	0.604	55	45.04
	Average				3.58077			55.24				3.60244			54.96	
Reverse Complement Orientation	MTPACGA	100314	44692	-0.78208	3.56417	24.065	0.164	55.45	44.55	45230	-0.80354	3.60707	28.13	0.219	54.9	45.09
	MPOMTCG	186608	83780	-0.79585	3.5917	52.41	0.494	55.1	44.9	84132	-0.8034	3.60679	54.78	0.494	54.9	45.08
	CHNTXX	155844	70090	-0.79898	3.59796	41.64	0.384	55.03	44.97	70178	-0.80124	3.60247	43.95	0.384	55	45.03
	CHMPXX	121024	53700	-0.77485	3.54971	29.89	0.274	55.63	44.37	54312	-0.79508	3.59016	27.14	0.329	55.1	44.88
	HUMGHCSA	66495	29783	-0.79159	3.58319	16.59	0.109	55.21	44.79	29951	-0.8017	3.6034	16.09	0.109	55	45.04
	HUMHBB	73308	32770	-0.78807	3.57614	18.35	0.164	55.3	44.7	33074	-0.80466	3.60932	18.9	0.164	54.9	45.12
	HUMHDABCD	58864	26260	-0.78445	3.5689	13.95	0.109	55.39	44.61	26518	-0.80198	3.60397	13.95	0.109	55	45.05
	HUMDYSTROP	38770	17404	-0.79562	3.59123	7.47	-0.001	55.11	44.89	17420	-0.79727	3.59453	7.52	-0.001	55.1	44.93
	HUMHPRTB	56737	25445	-0.79389	3.58778	12.91	0.109	55.15	44.85	25655	-0.8087	3.61739	12.91	0.054	54.8	45.22
	VACCG	191737	85817	-0.79031	3.58061	52.52	0.494	55.24	44.76	86283	-0.80003	3.60006	56.7	0.494	55	45
	HEHCMVCG	229354	102002	-0.77894	3.55789	69.94	0.604	55.53	44.47	103238	-0.8005	3.601	69.83	0.604	55	45.01
	Average				3.57721		0.219	55.28				3.60329	43425.3		54.95	

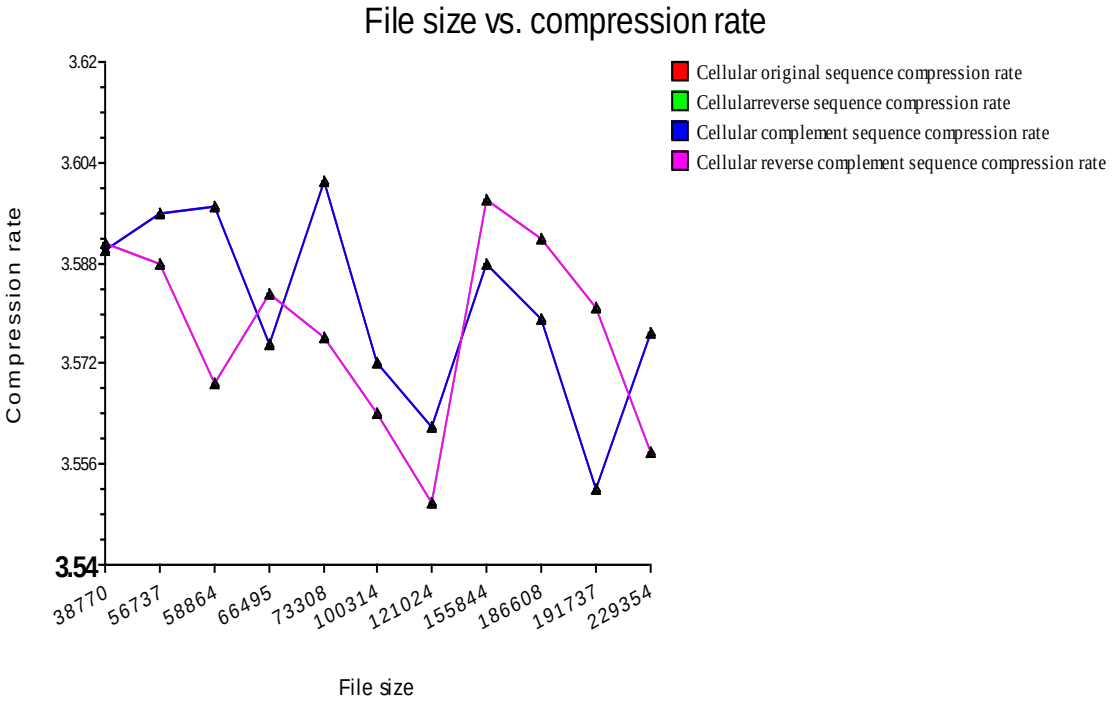

Fig. 5.3 file size versus compression rate of cellular sequence

Fig. 5.4 file size versus compression rate of artificial sequence

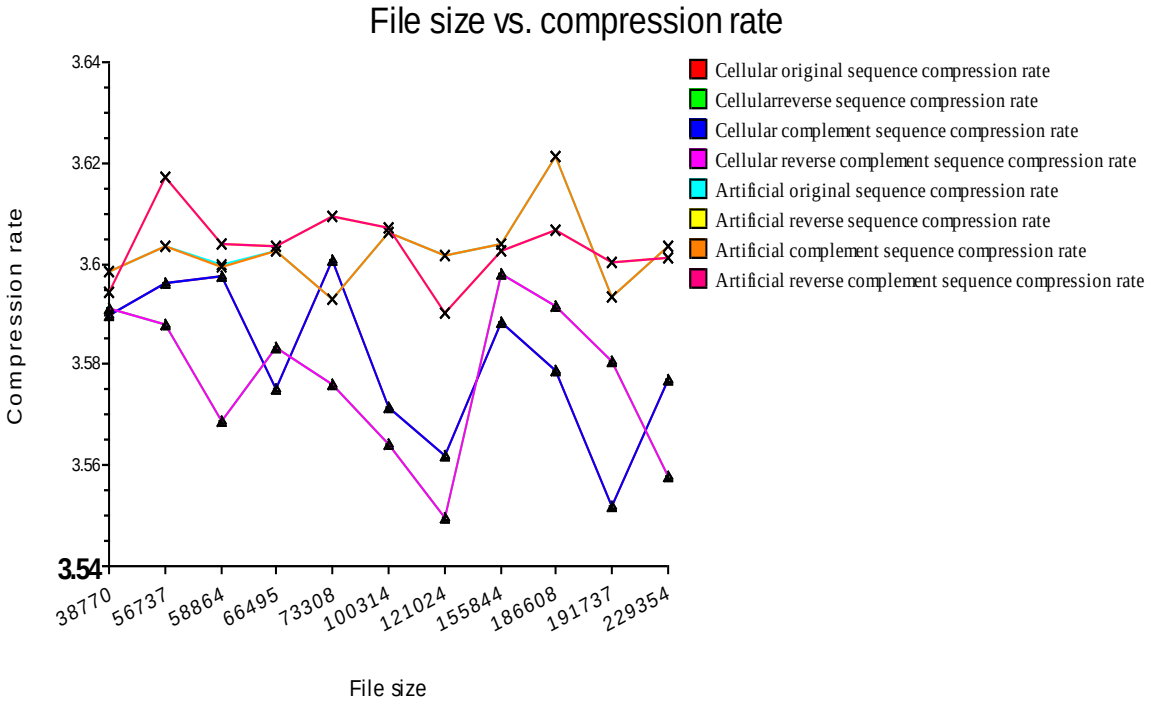

Fig. 5.5 file size versus compression rate of cellular & artificial sequence

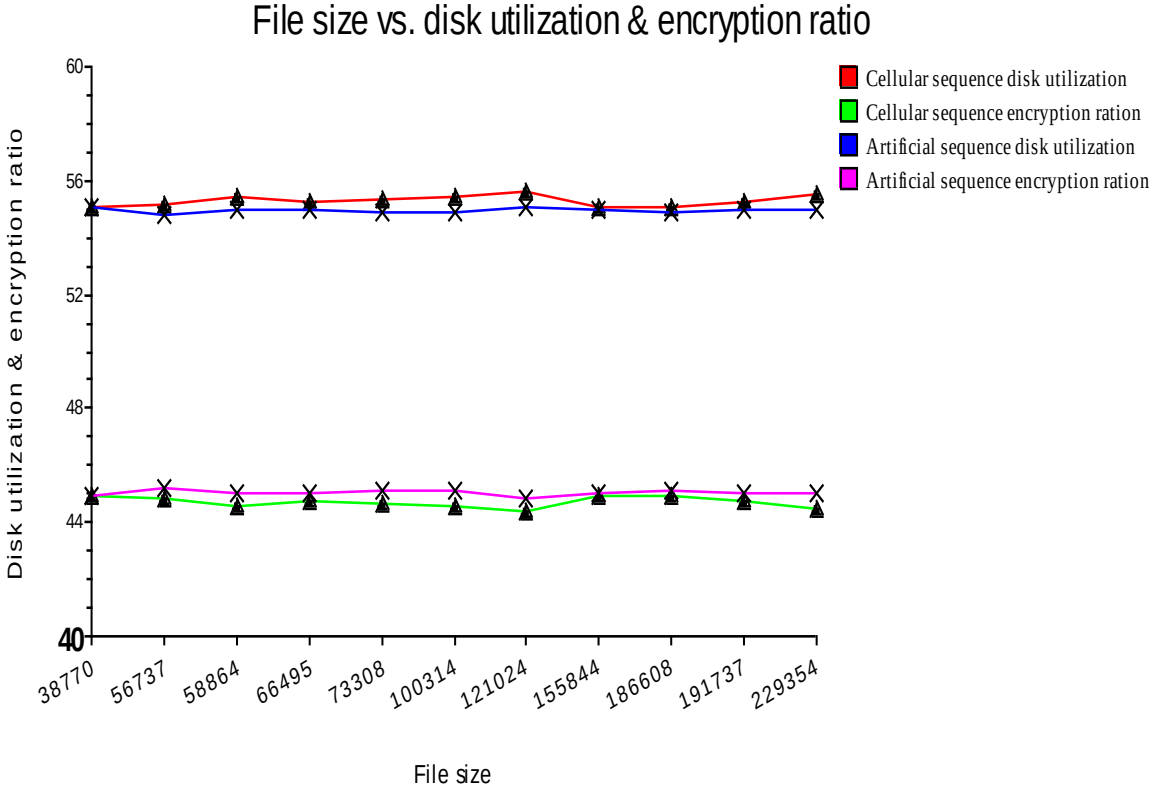

Fig. 5.6 file size versus disk utilization and encryption rate

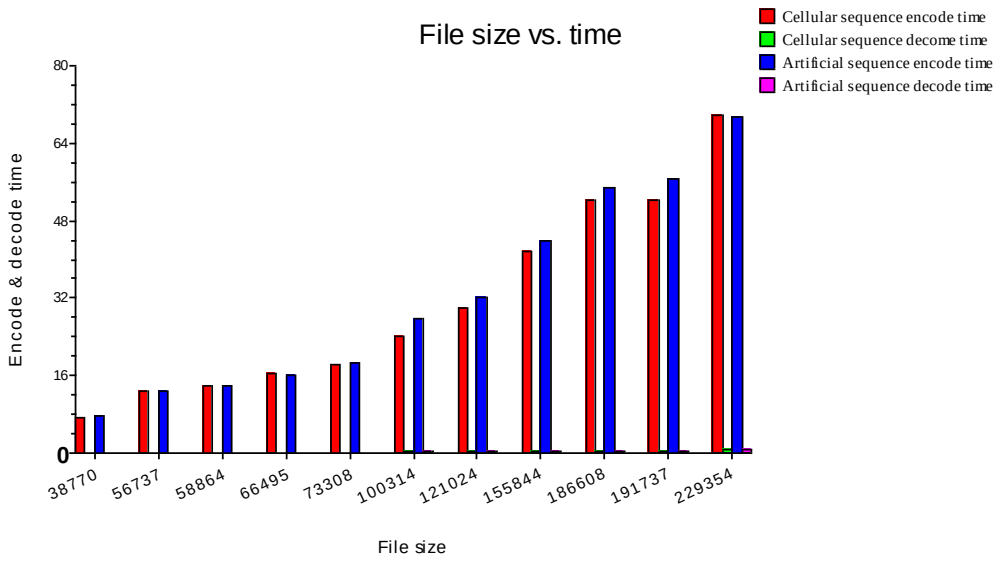

Fig. 5.7 cellular & artificial sequence compression and decompression time

Table 5.2 compression rate, ratio, space utilization and encryption rate of data set-2

			Cellular DNA sequences								Artificial DNA sequences					
Sequence orientation	Sequence Name	Sequence Size	Reduce file size Byte(C)	Compression ratio	Compression rate(bits /base)	Compression Time Encode Time	Compression Time Decode Time	Percentage of disk utilization	% Encryption ratio	Reduce file size Byte(C)	Compression ratio	Compression rate(bits /base) Encode Time	Compression Time Encode Time	Compression Time Decode Time	Percentage of disk utilization	% Encryption ratio
Normal Orientation	atatsgs	9647	4329	-0.79496	3.58992	2.12	0.044	55.13	44.87	4323	-0.79247	3.58495	2.16	0.044	55.19	44.81
	atef1a23	6022	2702	-0.79475	3.58951	1.02	0.001	55.13	44.87	2724	-0.80937	3.61873	1.12	0.044	54.77	45.23
	atrdnaf	10014	4472	-0.7863	3.5726	2.39	0.044	55.34	44.66	4502	-0.79828	3.59656	2.44	0.001	55.04	44.96
	atrdnai	5287	2367	-0.79081	3.58162	0.8	0.001	55.23	44.77	2387	-0.80594	3.61188	1.81	0.044	54.85	45.15
	celk07e12	58949	26483	-0.79701	3.59402	17.5	0.154	55.07	44.93	26645	-0.808	3.61601	17.99	0.209	54.8	45.2
	hsg6pdgen	52173	23195	-0.77831	3.55663	15.8	0.154	55.54	44.46	23449	-0.79779	3.59558	16.45	0.154	55.06	44.94
	mmzp3g	10833	4779	-0.76461	3.52922	2.56	0.044	55.88	44.12	4883	-0.80301	3.60602	2.6	0.044	54.92	45.08
	xlxfg512	19338	8780	-0.81611	3.63223	5.58	0.044	54.6	45.4	8744	-0.80867	3.61733	5.35	0.044	54.78	45.22
	Average			28.6457	3.58072			55.24			28.8471	3.60588			54.93	
Reverse Orientation	atatsgs	9647	4327	-0.79413	3.58827	2.12	0.044	55.15	44.85	4363	-0.80906	3.61812	2.27	0.044	54.77	45.23
	atef1a23	6022	2694	-0.78944	3.57888	0.96	0.044	55.26	44.74	2734	-0.81601	3.63202	1.12	0.044	54.6	45.4
	atrdnaf	10014	4464	-0.7831	3.56621	2.506	0.001	55.42	44.58	4538	-0.81266	3.62532	2.38	0.044	54.68	45.32
	atrdnai	5287	2363	-0.78778	3.57556	0.968	0.044	55.31	44.69	2373	-0.79535	3.59069	1.86	0.044	55.12	44.88
	celk07e12	58949	26699	-0.81167	3.62334	17.67	0.154	54.71	45.29	26519	-0.79945	3.59891	17.88	0.154	55.01	44.99
	hsg6pdgen	52173	23351	-0.79027	3.58055	16.24	0.154	55.24	44.76	23477	-0.79993	3.59987	16.01	0.209	55	45
	mmzp3g	10833	4853	-0.79193	3.58386	2.67	0.001	55.2	44.8	4851	-0.79119	3.58239	2.66	0.044	55.22	44.78
	xlxfg512	19338	8578	-0.77433	3.54866	5.19	0.044	55.64	44.36	8692	-0.79791	3.59582	5.57	0.044	55.05	44.95
	Average			28.6453	3.58067			55.24			28.8431	3.60539			54.93	
Complement Orientation	atatsgs	9647	4329	-0.79496	3.58992	2.06	0.044	55.13	44.87	4323	-0.79247	3.58495	2.16	0.001	55.19	44.81
	atef1a23	6022	2702	-0.79475	3.58951	1.02	0.044	55.13	44.87	2724	-0.80937	3.61873	1.06	0.001	54.77	45.23
	atrdnaf	10014	4472	-0.7863	3.5726	2.34	0.044	55.34	44.66	4502	-0.79828	3.59656	2.49	0.044	55.04	44.96
	atrdnai	5287	2367	-0.79081	3.58162	0.74	0.001	55.23	44.77	2387	-0.80594	3.61188	1.86	0.001	54.85	45.15
	celk07e12	58949	26483	-0.79701	3.59402	17.45	0.209	55.07	44.93	26645	-0.808	3.61601	17.93	1.154	54.8	45.2
	hsg6pdgen	52173	23195	-0.77831	3.55663	15.85	0.154	55.54	44.46	23449	-0.79779	3.59558	16.45	0.154	55.06	44.94
	mmzp3g	10833	4779	-0.76461	3.52922	2.5	0.044	55.88	44.12	4883	-0.80301	3.60602	2.66	0.001	54.92	45.08
	xlxfg512	19338	8780	-0.81611	3.63223	5.52	0.044	54.6	45.4	8744	-0.80867	3.61733	5.29	0.044	54.78	45.22
	Average			28.6457	3.58072			55.24			28.8471	3.60588			54.93	
Reverse Complement Orientation	atatsgs	9647	4327	-0.79413	3.58827	2.12	0.044	55.15	44.85	4363	-0.80906	3.61812	2.27	0.044	54.77	45.23
	atef1a23	6022	2694	-0.78944	3.57888	1.02	0.044	55.26	44.74	2734	-0.81601	3.63202	1.17	0.001	54.6	45.4
	atrdnaf	10014	4464	-0.7831	3.56621	2.39	0.044	55.42	44.58	4538	-0.81266	3.62532	2.49	0.044	54.68	45.32
	atrdnai	5287	2363	-0.78778	3.57556	0.96	0.044	55.31	44.69	2373	-0.79535	3.59069	1.86	0.001	55.12	44.88
	celk07e12	58949	26699	-0.81167	3.62334	17.72	0.209	54.71	45.29	26519	-0.79945	3.59891	18.88	0.154	55.01	44.99
	hsg6pdgen	52173	23351	-0.79027	3.58055	16.24	0.209	55.24	44.76	23477	-0.79993	3.59987	16.06	0.154	55	45
	mmzp3g	10833	4853	-0.79193	3.58386	2.67	0.044	55.2	44.8	4851	-0.79119	3.58239	2.6	0.044	55.22	44.78
	xlxfg512	19338	8578	-0.77433	3.54866	5.25	0.044	55.64	44.36	8692	-0.79791	3.59582	5.68	0.044	55.05	44.95
	Average			28.6453	3.58067			55.24			28.8431	3.60539			54.93	

Fig. 5.8 file size versus compression rate of cellular sequence

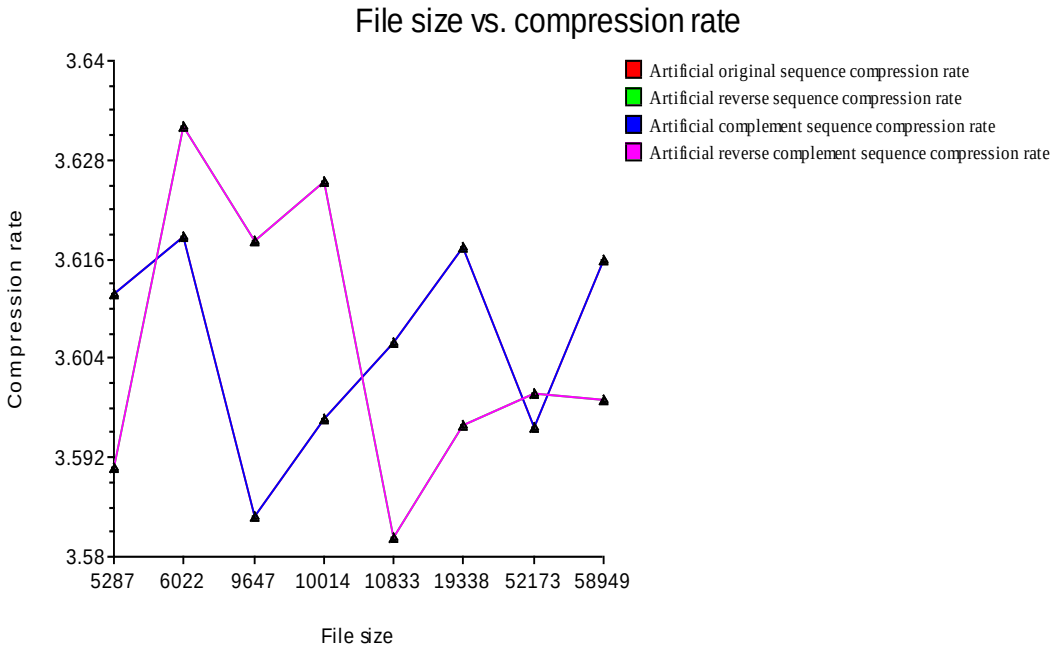

Fig. 5.9 file size versus compression rate of artificial sequence

Fig.5.10 file size versus compression rate of cellular & artificial sequence

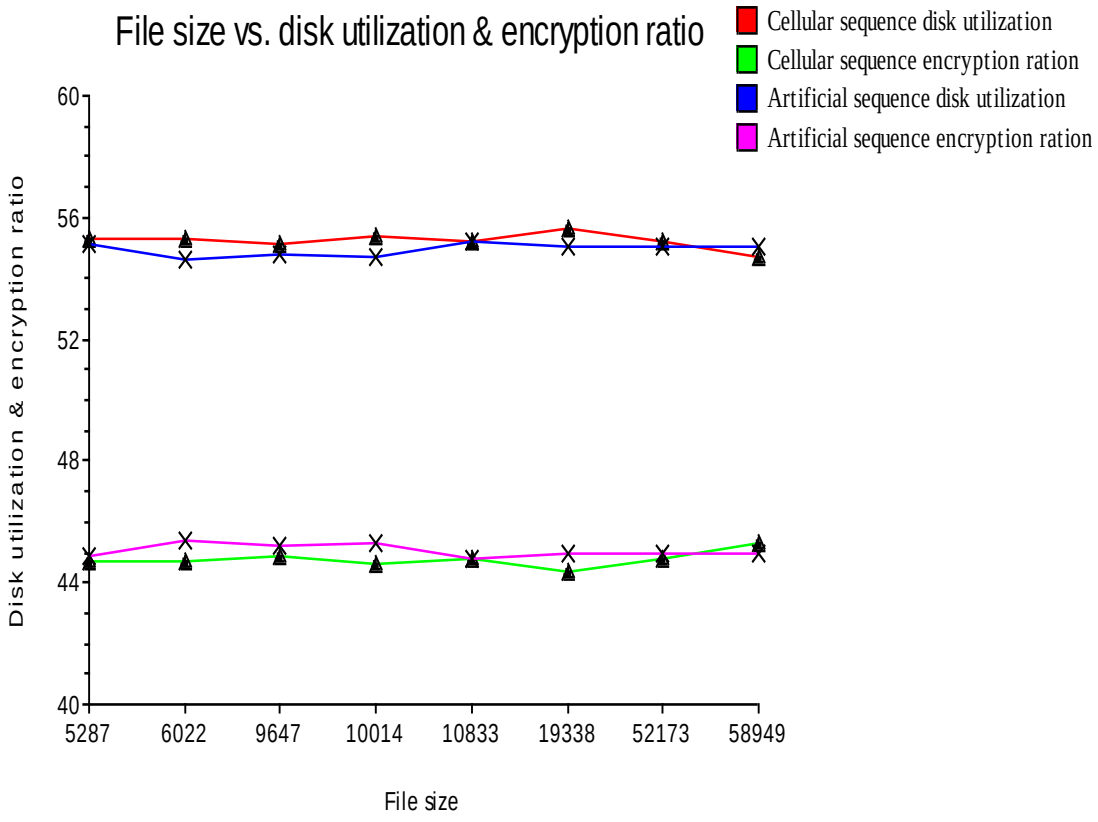

Fig. 5.11 file size versus disk utilization & encryption rate of cellular & artificial sequence

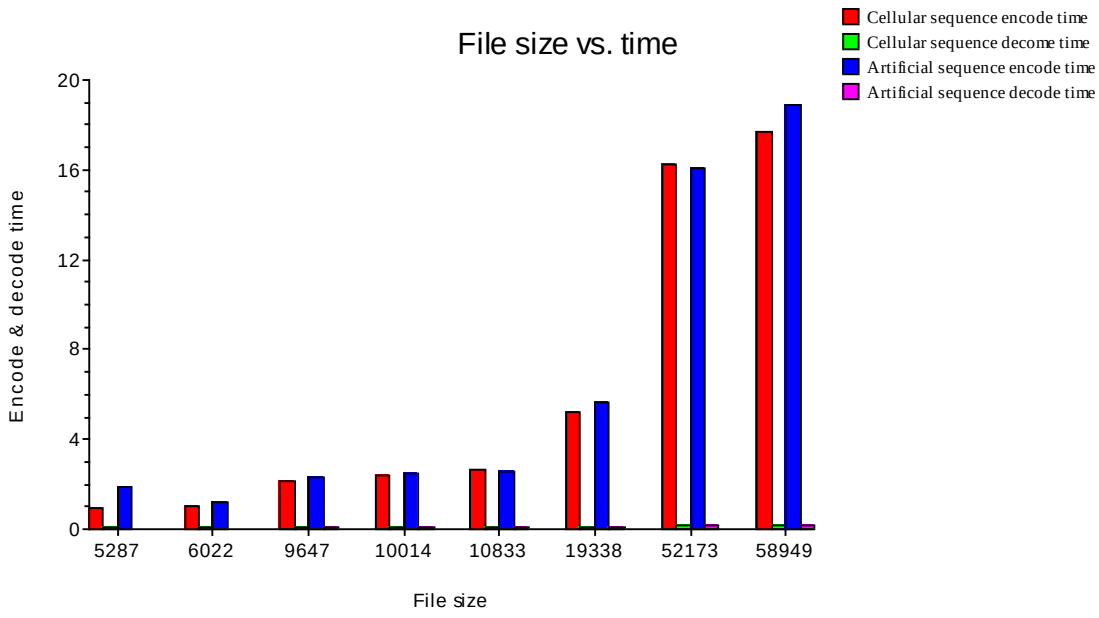

Fig. 5.12 cellular & artificial sequence compression and decompression time

Table 5.3 the percentage encryption ratio and percentage effect on actual text using compression technique

Data set	Sequence Name	Sequence Size	Compress file size	Library file size	Lavenstein Distance	% of Encryption ratio	%effect on actual text	Entropy before compression	Entropy after compression	
									Compressed file	Library file
Data set-I	MTPACGA	100314	44784	192	95388	44.64	95.09	1.87918	5.05224	3.29185
	MPOMTCG	186608	83484	192	178360	44.74	95.58	1.98324	4.98572	3.29122
	CHNTXX	155844	69900	192	148992	44.85	95.6	1.95680	4.96660	3.29185
	CHMPXX	121024	53886	192	114926	44.53	94.96	1.86621	4.77975	3.28840
	HUMGHCSA	66495	29717	192	63655	44.69	95.73	1.99957	5.03721	3.28965
	HUMHBB	73308	32996	192	69996	45.01	95.48	1.96758	4.96318	3.29216
	HUMHDABCD	58864	26470	192	56257	44.97	95.57	1.99739	4.93849	3.29185
	HUMDYSTROP	38770	17396	192	36977	44.87	95.38	1.94688	4.88918	3.29248
	HUMHPRTB	56737	25503	192	54230	44.95	95.58	1.97046	5.00871	3.29091
	VACCG	191737	85123	192	183982	44.4	95.96	1.91895	4.91566	3.29122
	HEHCMVCG	**229354**	102550	192	219769	44.71	95.82	1.98509	5.05224	3.29122
	Average							1.95194	4.96263	3.29116
Data set-II	atatsgs	9647	4329	192	9220	44.87	95.57	1.93269	4.88406	3.28840
	atef1a23	6022	2702	180	5738	44.87	95.28	1.97207	4.94303	3.28406
	atrdnaf	10014	4472	192	9556	44.66	95.43	1.99287	4.98463	3.28713
	atrdnai	5287	2367	192	5056	44.77	95.63	1.99159	4.89239	3.27092
	celk07e12	58949	26483	192	56336	44.93	95.57	1.96998	4.89481	3.28934
	hsg6pdgen	52173	23195	192	49906	44.46	95.65	1.99773	4.94733	3.28713
	mmzp3g	10833	4779	192	10412	44.12	96.11	1.99465	4.95437	3.28840
	xlxfg512	19338	8780	186	18412	45.4	95.21	1.99220	4.95625	3.28240
	Average							1.98047	4.93210	3.28472

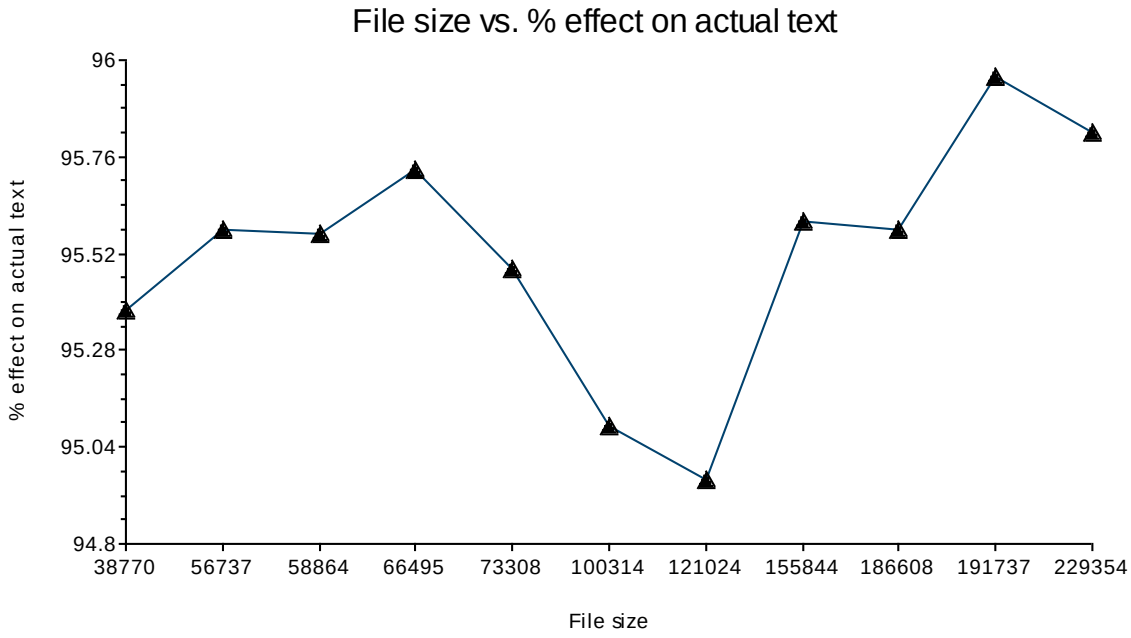

Fig. 5.13 file size versus % effect on actual text of data set-1

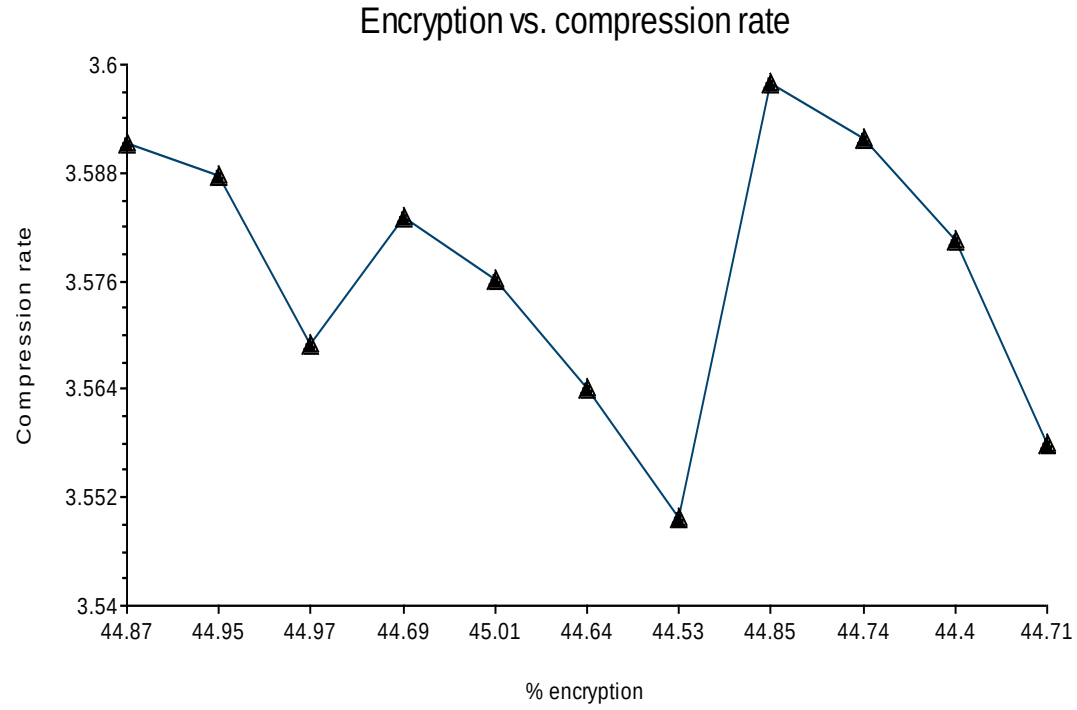

Fig. 5.14 encryption versus compression rate of data set-1

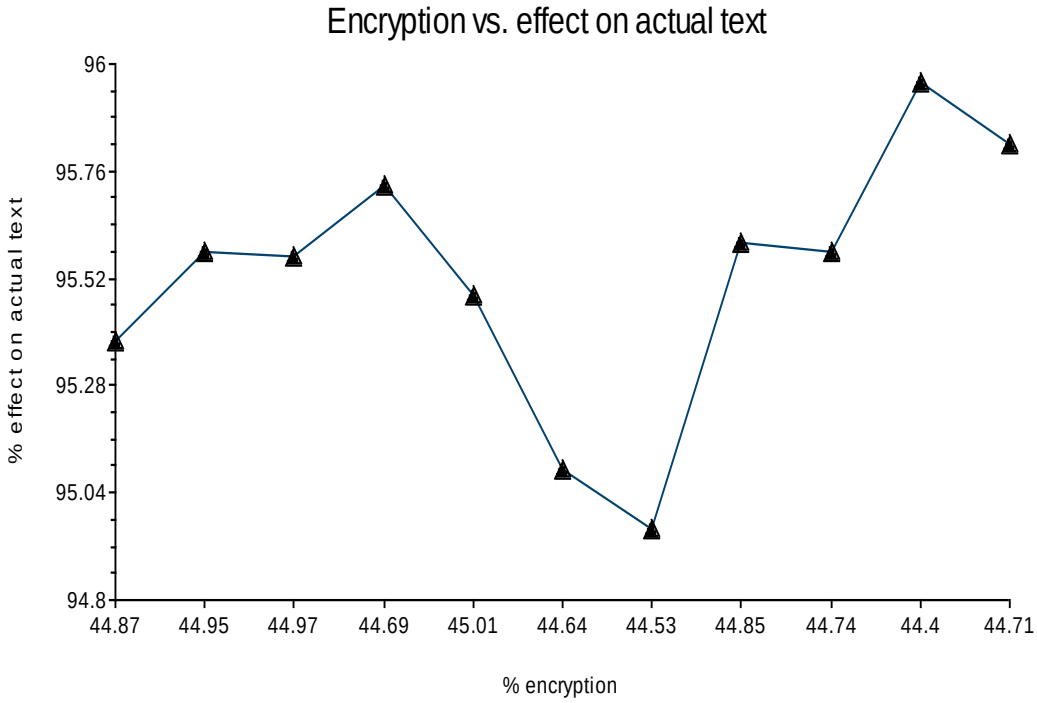

Fig. 5.15 encryption versus effect on actual text of data set-1

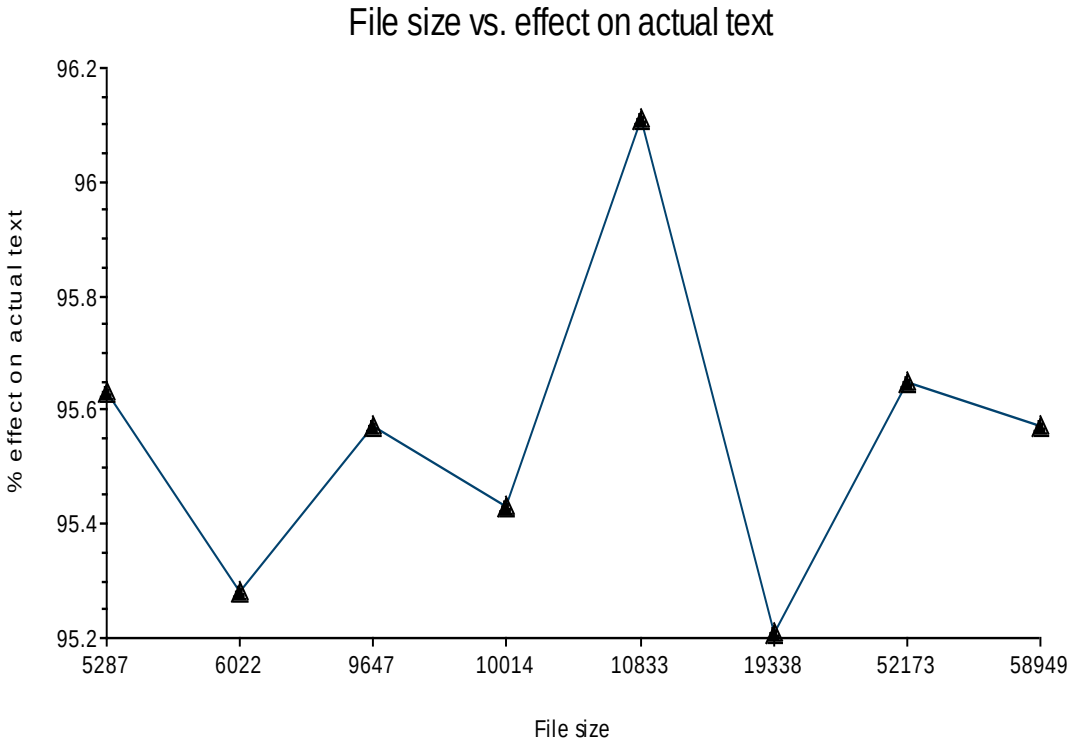

Fig. 5.16 file size versus percentage effect on actual text of data set-2

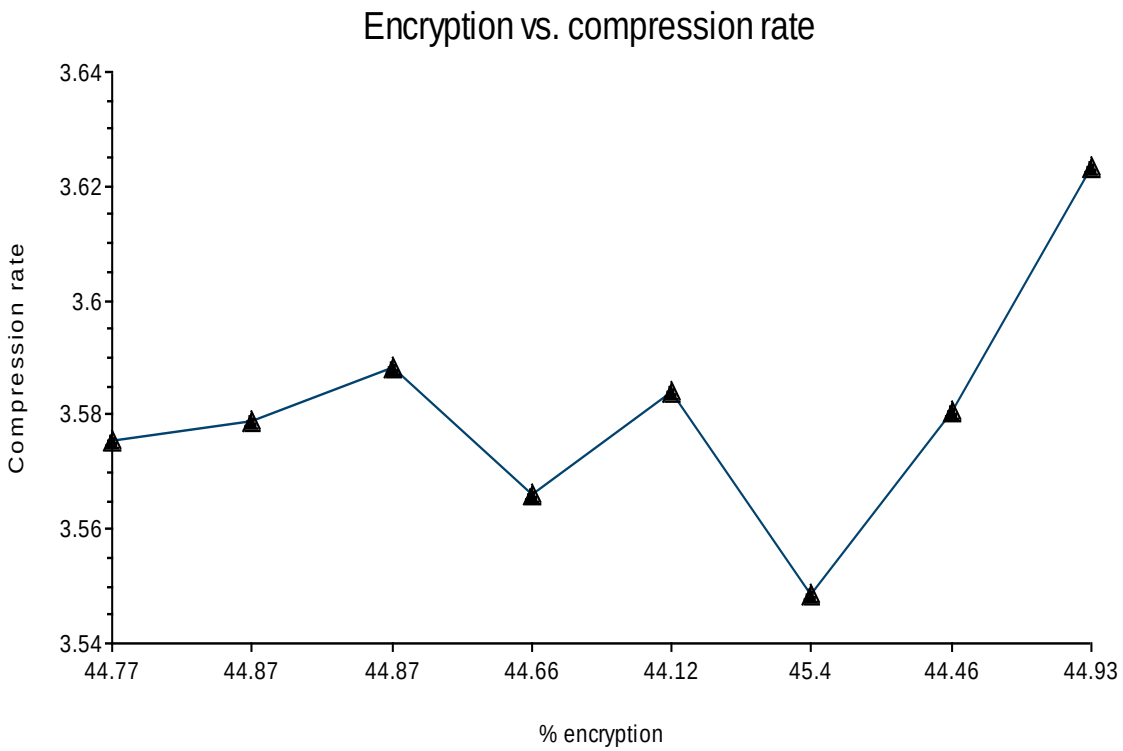

Fig. 5.17 encryption versus compression rate of data set-2

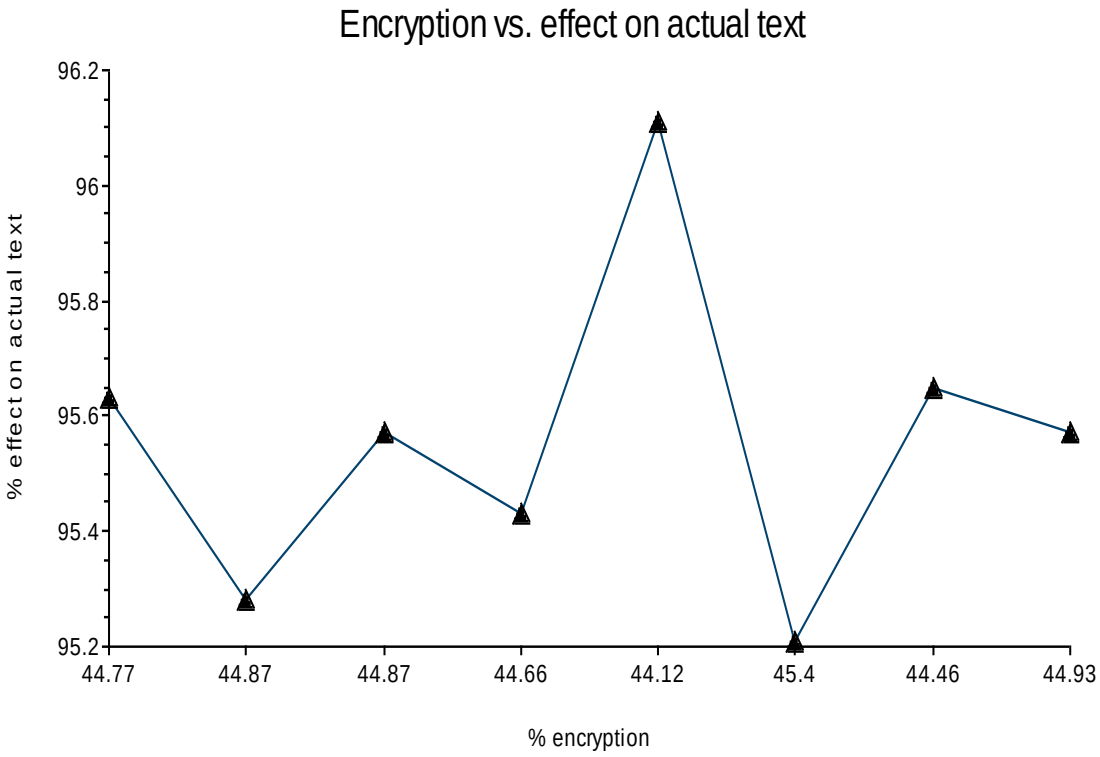

Fig. 5.18 encryption versus effect on actual text of data set-2

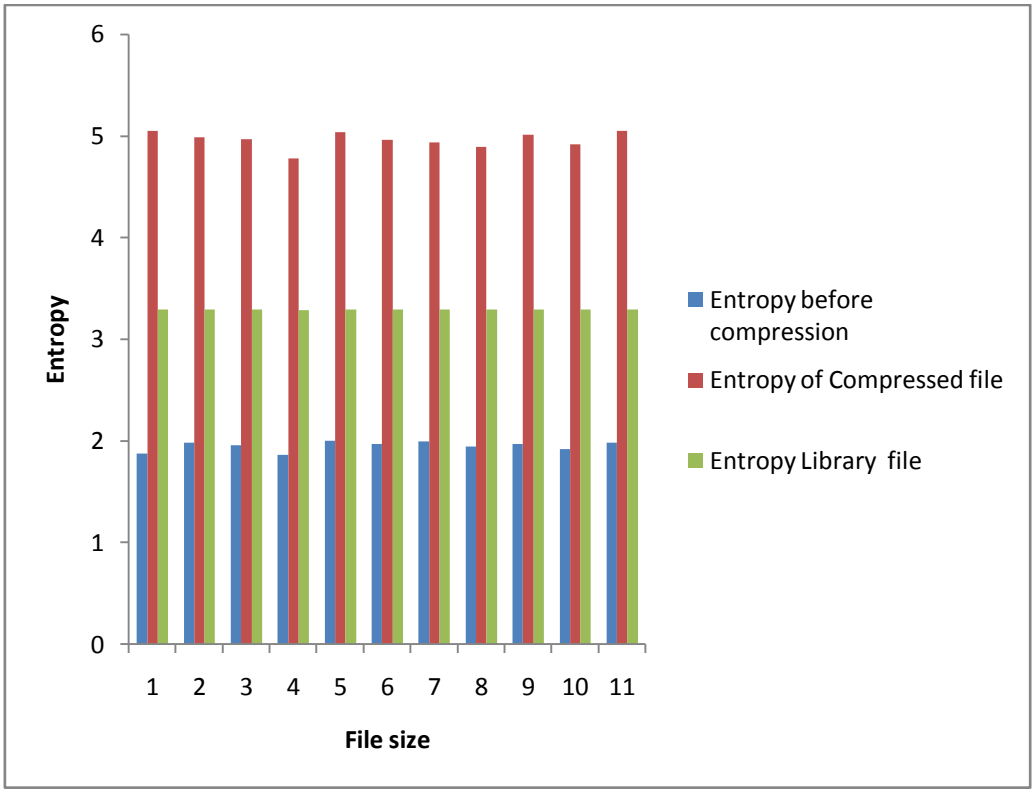

Fig. 5.19 entropy versus file size before and after encryption of data set-2

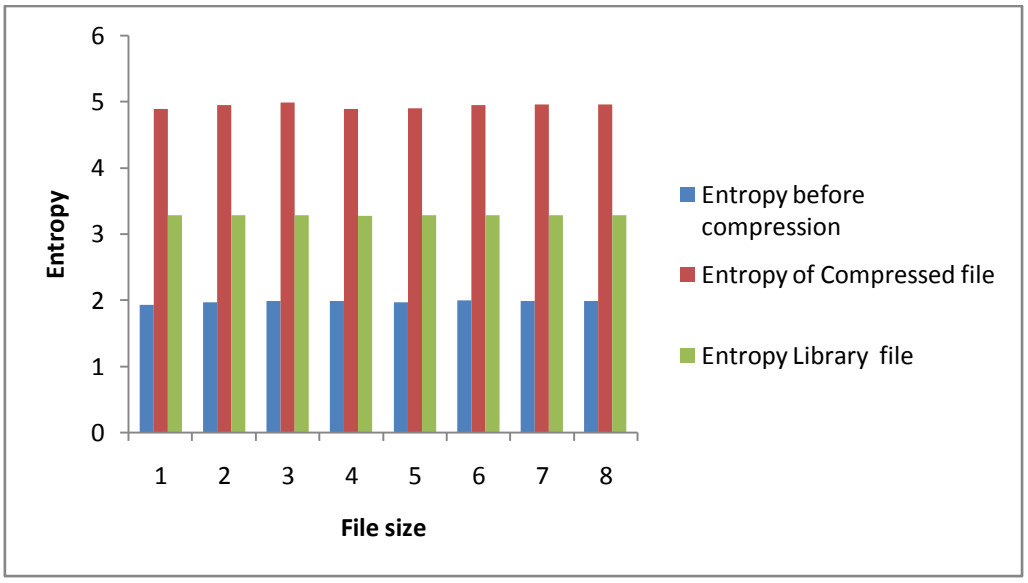

Fig.5.20 entropy versus file size before and after encryption of data set-2

Table 5.4 apply encryption algorithm on source file and calculate encryption & decryption time

Sequence Orientation	Sequences name	Sequence size	RSA		DES		AES		Selection encryption					
			Encode Time	Decode Time	Encode Time	Decode Time	Encode Time	Decode Time	Choice selection	Encode Time	Decode Time	Hamming distance	% Avalanche effect	Encryption rate (byte/sec)
Data set-I	MTPACGA	100314	5.96	5.94	2.85	2.77	2.52	2.54	S	3.7	2.97	35804	35.69	27111.89
									D	4.4	2.63	23098	23.03	22798.64
									M	4.6	3.54	5067	5.051	21807.39
	MPOMTCG	186608	5.87	5.19	5.04	4.91	4.71	4.68	S	4.3	3.92	54264	29.08	43397.21
									D	4.5	2.6	28092	15.05	41468.44
									M	5.5	3	8343	4.471	33928.73
	CHNTXX	155844	7.67	6.56	4.24	4.14	3.91	3.91	S	3.8	2.64	49037	31.47	41011.58
									D	4.2	2.16	30706	19.7	37105.71
									M	4.3	2.17	2547	1.634	36242.79
	CHMPXX	121024	7.54	6.68	3.37	3.27	3.04	3.04	S	3.8	3.02	43263	35.75	31848.42
									D	3.6	3.26	26496	21.89	33617.78
									M	3.9	2.45	5025	4.152	31031.79
	HUMGHCSA	66495	7.85	6.94	2.05	1.92	1.72	1.69	S	2.2	1.83	17311	26.03	30225
									D	4.2	2.14	6994	10.52	15832.14
									M	4.3	1.92	3336	5.017	15463.95
	HUMHBB	73308	7.40	6.13	2.26	2.09	1.93	1.86	S	2.3	1.94	22309	30.43	31873.04
									D	3.0	1.83	11900	16.23	24436
									M	3.1	2.17	4326	5.901	23647.74
	HUMHDABCD	58864	7.52	7.39	1.86	1.71	1.53	1.48	S	2.1	1.97	15906	27.02	28030.48
									D	2.4	1.82	5242	8.905	24526.67
									M	2.7	1.92	2241	3.807	21801.48
	HUMDYSTROP	38770	7.18	5.81	1.2	1.08	0.87	0.85	S	2.9	2.39	12597	32.49	13368.97
									D	3.3	1.91	7020	18.11	11748.48
									M	3.1	2.13	2373	6.121	12506.45
	HUMHPRTB	56737	8.32	7.49	1.58	1.46	1.25	1.23	S	2.8	2.05	18168	32.02	20263.21
									D	3.0	2.08	8782	15.48	18912.33
									M	3.0	2.04	2973	5.24	18912.33
	VACCG	191737	7.97	6.76	4.34	4.24	4.01	4.01	S	3.1	1.94	63921	33.34	61850.65
									D	3.4	1.84	46224	24.11	56393.24
									M	3.2	1.75	12177	6.351	59917.81
	HEHCMVCG	229354	7.28	7.27	5.13	5.01	4.80	4.78	S	3.2	2.02	66192	28.86	71673.13
									D	2.9	1.84	19172	8.359	79087.59
									M	2.9	2.05	8745	3.813	79087.59
Data set-II	atatsgs	9647	6.66	5.89	0.55	0.45	0.22	0.22	S	2.4	1.89	3180	32.96	4019.583
									D	2.5	1.69	1924	19.94	3858.8
									M	2.6	1.81	570	5.909	3710.385
	atef1a23	6022	5.04	5.03	0.86	0.41	0.53	0.18	S	2.4	2	1974	32.78	2509.167
									D	2.5	1.83	936	15.54	2408.8
									M	2.7	1.65	300	4.982	2230.37
	atrdnaf	10014	6.37	5.77	0.62	0.52	0.29	0.29	S	2.6	1.8	2919	29.15	3851.538
									D	2.8	1.83	1116	11.14	3576.429
									M	2.6	1.87	471	4.703	3851.538
	atrdnai	5287	6.28	5.81	0.53	0.38	0.20	0.15	S	2.2	1.35	1496	28.3	2403.182
									D	2.7	1.74	644	12.18	1958.148
									M	2.4	1.75	264	4.993	2202.917
	celk07e12	58949	7.06	5.46	1.61	1.49	1.28	1.26	S	2.6	1.8	18657	31.65	22672.69
									D	2.6	1.61	10044	17.04	22672.69
									M	2.6	1.64	4350	7.379	22672.69
	hsg6pdgen	52173	6.57	5.62	1.44	1.34	1.11	1.11	S	2.5	1.6	14028	26.89	20869.2
									D	2.6	1.77	4534	8.69	20066.54
									M	2.8	1.72	1968	3.772	18633.21
	mmzp3g	10833	6.54	5.79	0.69	0.51	0.36	0.28	S	2.8	1.55	2998	27.67	3868.929
									D	2.6	1.87	1086	10.02	4166.538
									M	2.8	1.94	528	4.874	3868.929
	xlxfg512	19338	6.74	5.71	0.79	0.65	0.46	0.42	S	2.4	1.66	5366	27.75	8057.5
									D	2.5	1.73	2276	11.77	7735.2
									M	2.6	1.74	822	4.251	7437.692

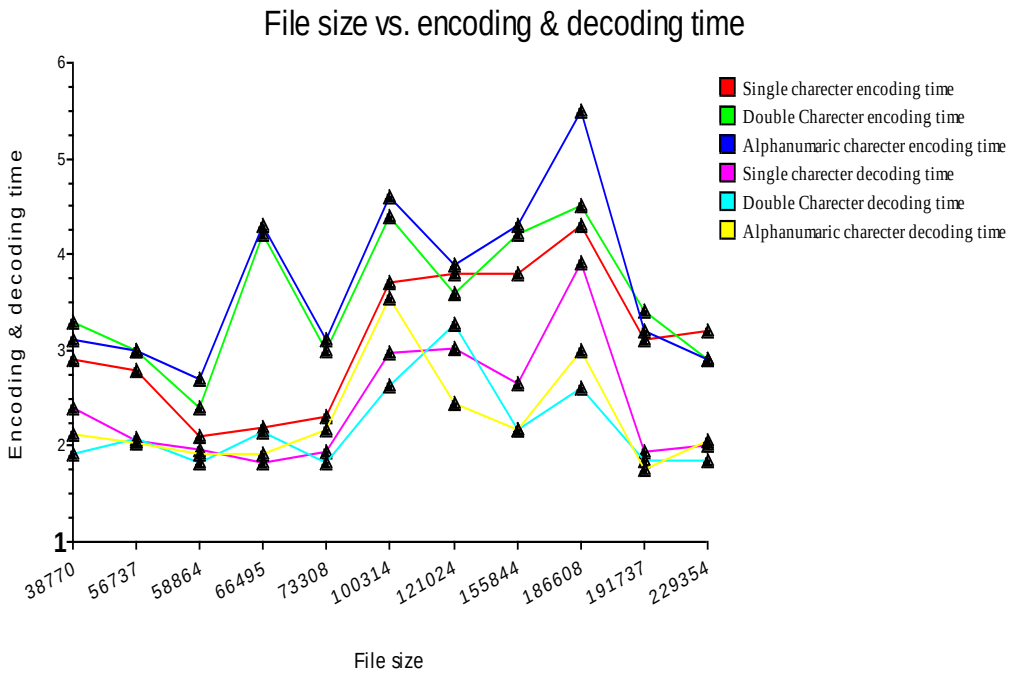

Fig. 5.21 time in encryption & decryption vs. File size before compression of data set-1

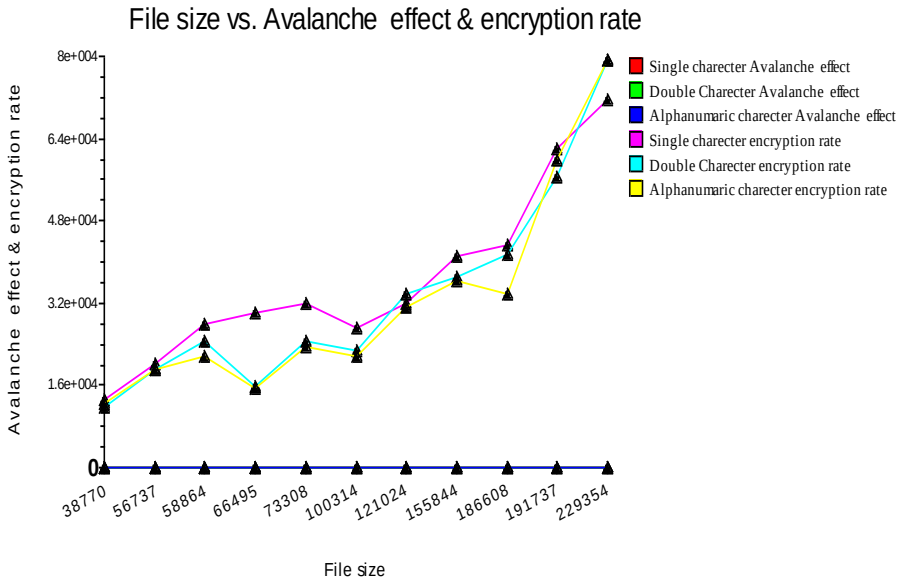

Fig.5.22 file size versus avalanche effect & encryption rate before compression of data set-1

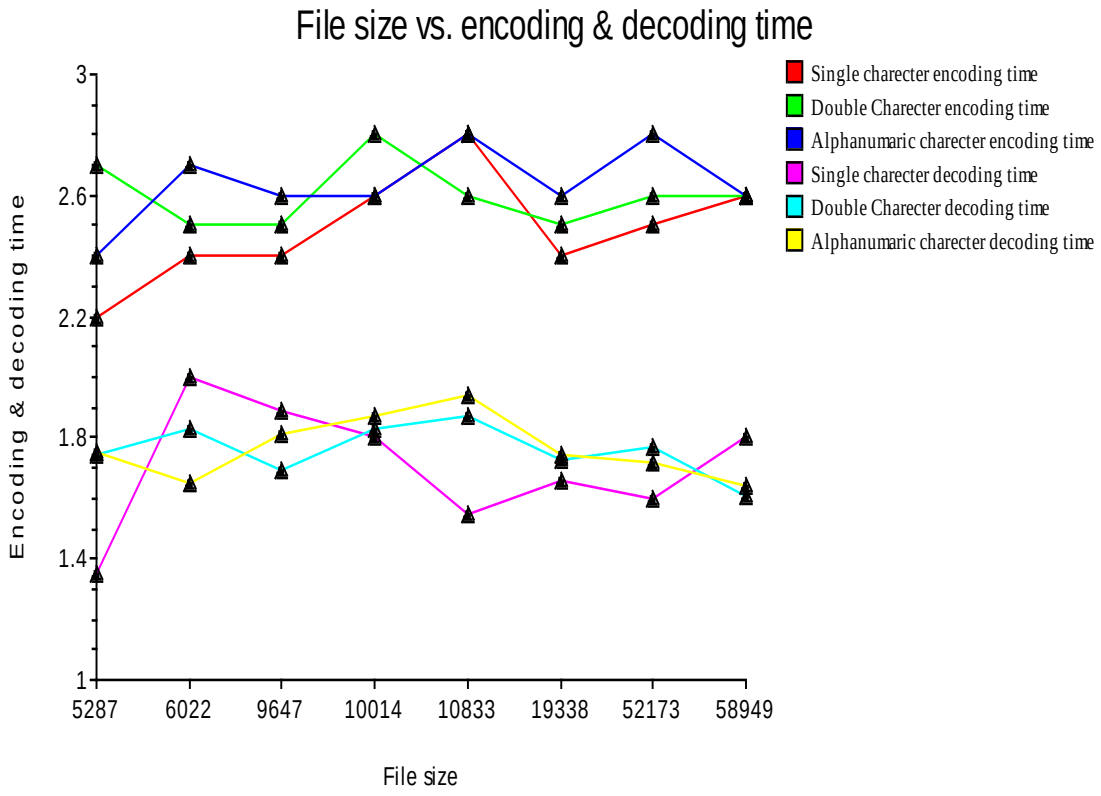

Fig.5.23 time in encryption & decryption vs. file size time before compression of data set-2

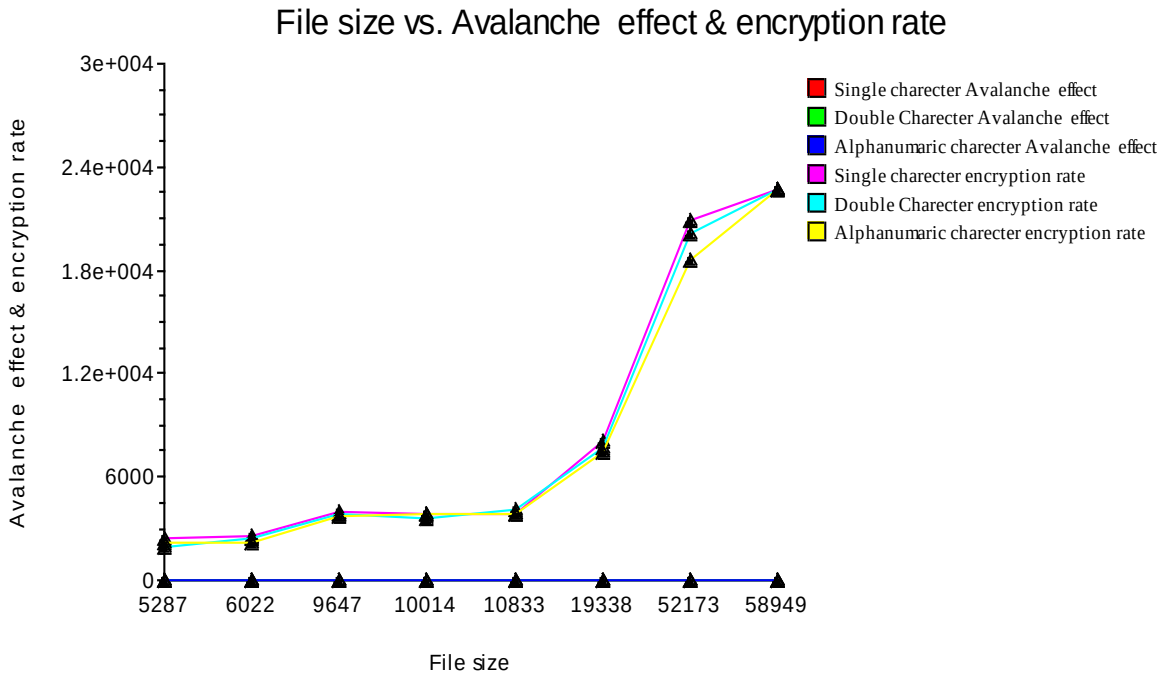

Fig.5.24 file size versus avalanche effect & encryption rate before compression of data set-2

Table 5.5 apply encryption algorithm on compressed file and calculate encryption & decryption time

Sequence Orientation	Sequences name	Compressed file size	RSA		DES		AES		Selection encryption					
			Encode Time	Decode Time	Encode Time	Decode Time	Encode Time	Decode Time	Choice selection	Encode Time	Decode Time	Hamming distance	Avalanche effect	Encryption rate (byte/sec)
Data set-I	MTPACGA	44784	5.37	3.20	2.38	1.75	1.49	0.96	S	2.99	2	6391	14.27	7052.59
									D	2.49	1.84	734	1.639	8481.81
									A	2.43	1.51	36	0.08	8662.28
	MPOMTCG	83484	5.17	3.20	2.65	2.57	1.76	1.78	S	2.29	1.51	9896	11.85	17142.51
									D	2.26	1.46	1346	1.612	17428.81
									A	2.43	1.33	69	0.083	16147.78
	CHNTXX	69900	5.20	2.93	2.36	2.27	1.47	1.48	S	1.78	1.38	6643	9.504	18492.06
									D	1.98	1.31	1504	2.152	16642.86
									A	2.45	1.31	225	0.322	13442.31
	CHMPXX	53886	4.60	2.75	2.03	1.95	1.14	1.16	S	2.02	1.07	7192	13.35	12560.84
									D	1.83	1.29	426	0.791	13852.44
									A	2.17	1.21	39	0.072	11714.35
	HUMGHCSA	29717	5.18	2.75	1.53	1.46	0.64	0.67	S	1.9	1.28	3365	11.32	7355.69
									D	1.92	1.05	482	1.622	7301.47
									A	2.44	1.29	129	0.434	5736.87
	HUMHBB	32996	4.43	2.65	1.6	1.51	0.71	0.72	S	1.71	1.31	3312	10.04	9064.83
									D	1.66	1.21	572	1.734	9347.30
									A	2.09	1.25	63	0.191	7448.30
	HUMHDABCD	26470	4.76	2.62	1.47	1.41	0.58	0.62	S	1.78	1.22	3195	12.07	7002.64
									D	2.24	1.23	692	2.614	5560.92
									A	2.03	1.18	54	0.204	6141.53
	HUMDYSTROP	17396	4.37	3.00	1.27	1.18	0.38	0.39	S	1.9	1.41	2267	13.03	4305.94
									D	1.75	1.31	200	1.15	4688.94
									A	2.06	1.27	18	0.103	3980.77
	HUMHPRTB	25503	4.81	2.75	1.45	1.38	0.56	0.59	S	1.89	1.3	2380	9.332	6359.85
									D	2.06	1.14	232	0.91	5835.92
									A	2.27	1.28	9	0.035	5302.07
	VACCG	85123	4.56	2.75	2.7	2.61	1.81	1.82	S	1.93	1.29	8763	10.29	20761.71
									D	2.11	1.16	1670	1.962	19000.67
									A	2.15	1.26	330	0.388	18667.32
	HEHCMVCG	102550	4.23	2.78	3.05	2.97	2.16	2.18	S	1.94	1.29	12405	12.1	24890.78
									D	1.83	1.25	2136	2.083	26362.47
									A	1.99	1.31	117	0.114	24243.5
Data set-II	atatsgs	4329	4.03	2.70	1	0.92	0.11	0.13	S	1.53	1.17	489	11.3	1332
									D	1.86	1.14	42	0.97	1095.94
									A	1.9	1.27	9	0.208	1074.19
	atef1a23	2702	4.25	2.57	0.97	0.86	0.08	0.07	S	1.64	1.14	264	9.771	776.43
									D	1.67	1.14	16	0.592	763.27
									A	2	1.21	15	0.555	635.76
	atrdnaf	4472	4.42	2.56	1.02	0.91	0.13	0.12	S	1.79	1.29	544	12.16	1173.75
									D	1.87	1.21	58	1.297	1123.61
									A	2.08	1.14	21	0.47	1011.76
	atrdnai	2367	4.04	2.43	0.96	0.89	0.07	0.10	S	1.82	1.14	283	11.96	613.21
									D	1.89	1.08	26	1.098	590.27
									A	1.9	1.09	74	3.126	585.89
	celk07e12	26483	5.01	2.67	1.46	1.37	0.57	0.58	S	1.63	1.14	2782	10.5	7654.04
									D	1.81	1.13	392	1.48	6896.61
									A	2.36	1.26	38	0.143	5286.02
	hsg6pdgen	23195	5.21	2.76	1.42	1.29	0.53	0.50	S	1.76	1.09	2720	11.73	6218.49
									D	2	1.3	160	0.69	5457.64
									A	2.45	1.06	66	0.285	4452.01
	mmzp3g	4779	3.93	2.68	12.89	12.8	012	012	S	1.66	1.26	478	10	1353.82
									D	1.73	1.15	52	1.088	1302.18
									M	1.85	1.07	24	0.502	1216.03
	xlxfg512	8780	4.20	2.46	1.13	1.04	0.24	0.25	S	1.51	1.08	1122	12.78	2743.75
									D	1.62	1.12	288	3.28	2544.92
									A	1.98	1.16	33	0.376	2090.47

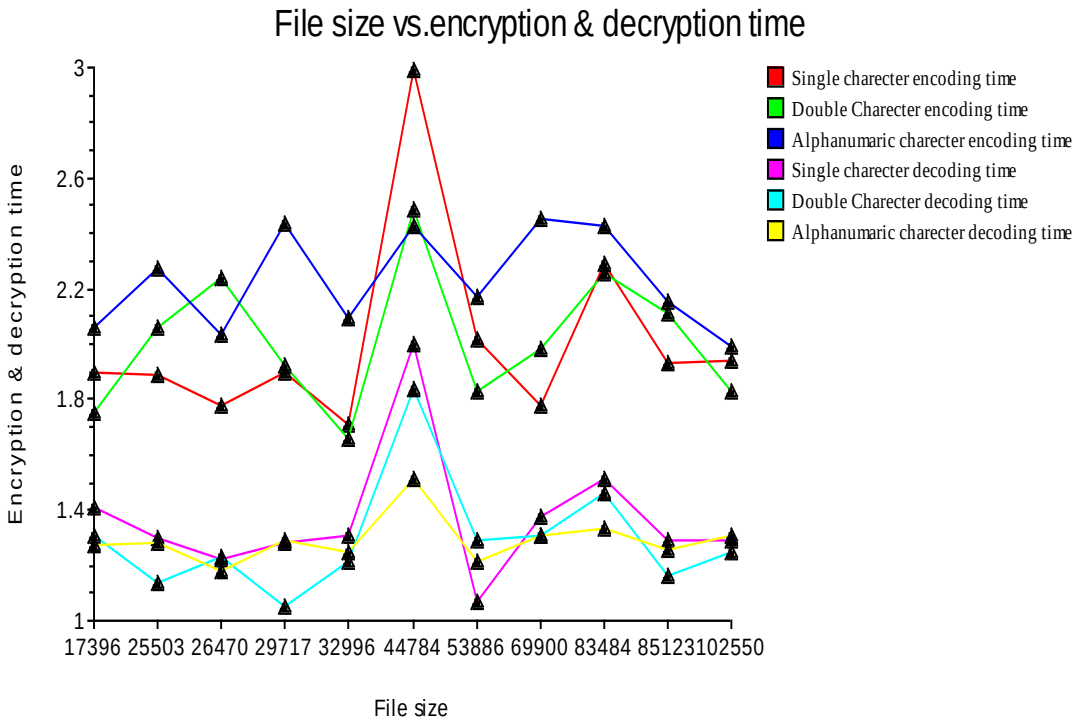

Fig.5.25 time in encryption & decryption vs. File size of data set-1

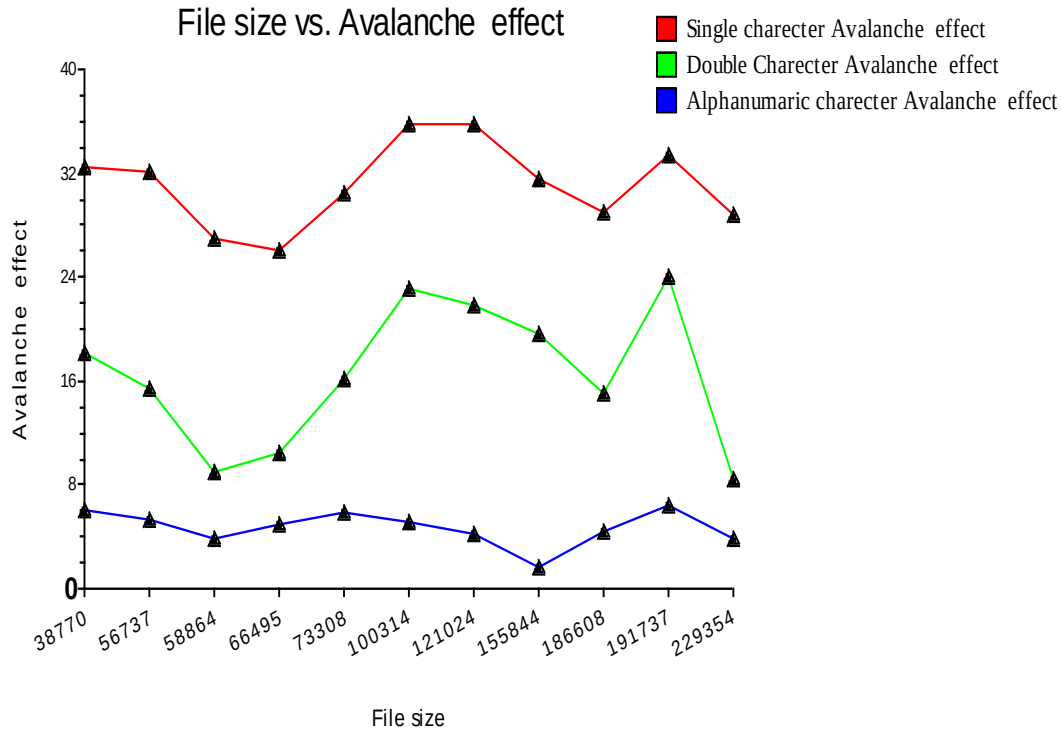

Fig.5.26 file size versus avalanche effect of data set-1

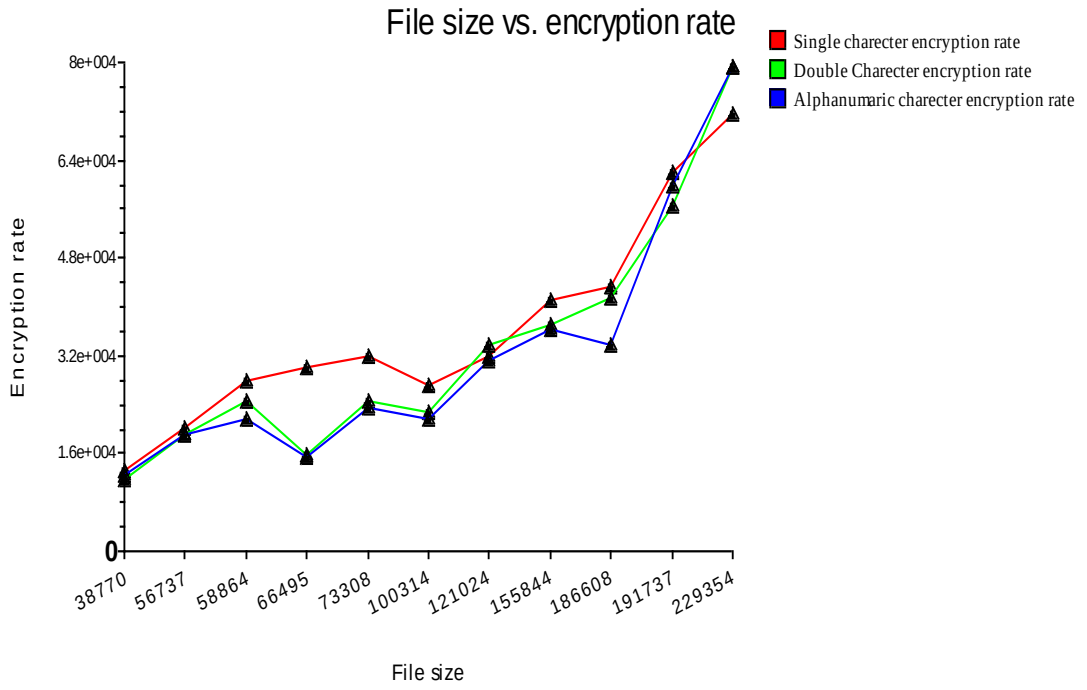

Fig.5.27 file size versus encryption rate of data set-1

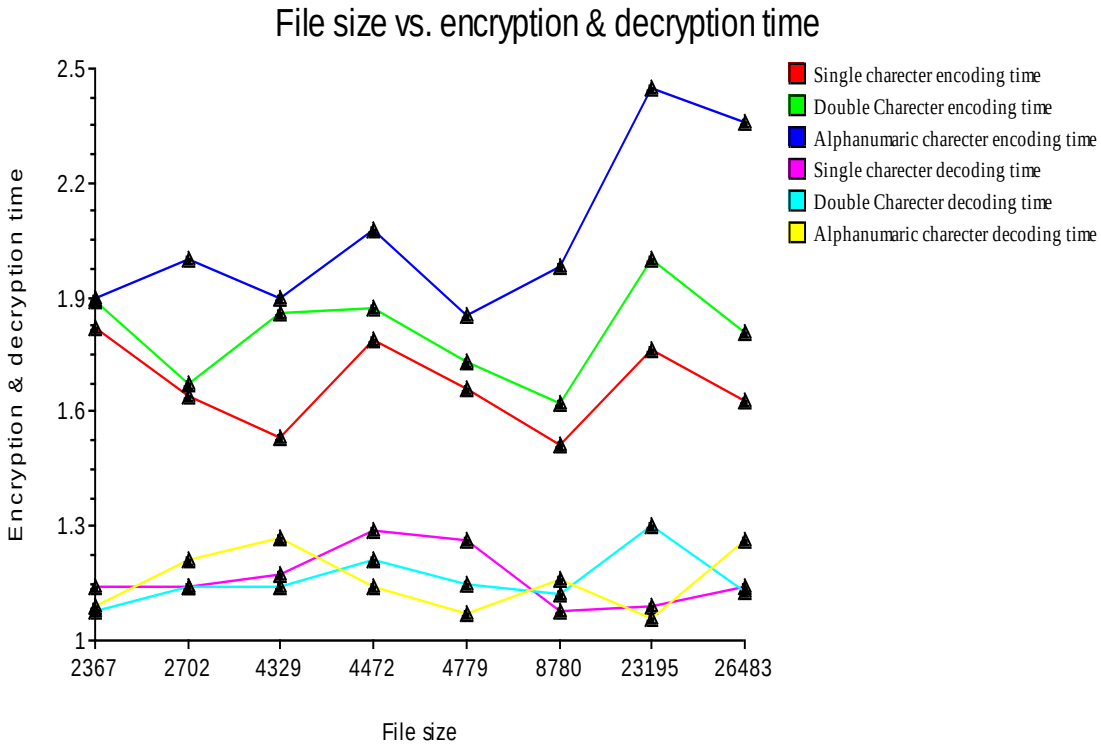

Fig.5.28 time in encryption & decryption vs. file size of data set-2

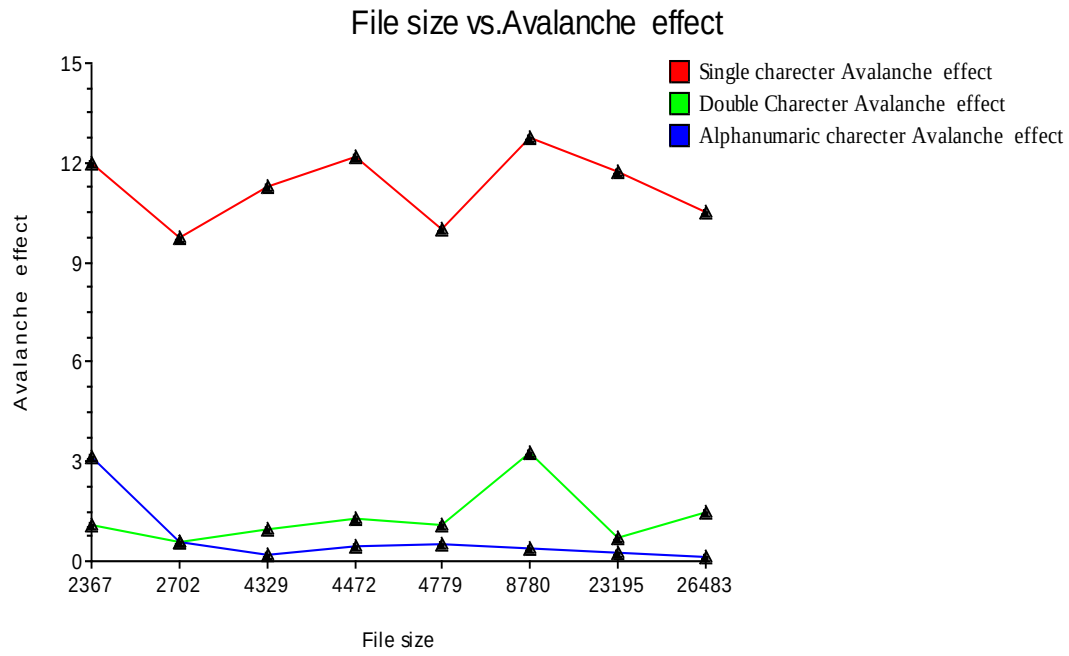

Fig.5.29 file size versus avalanche effect of data set-2

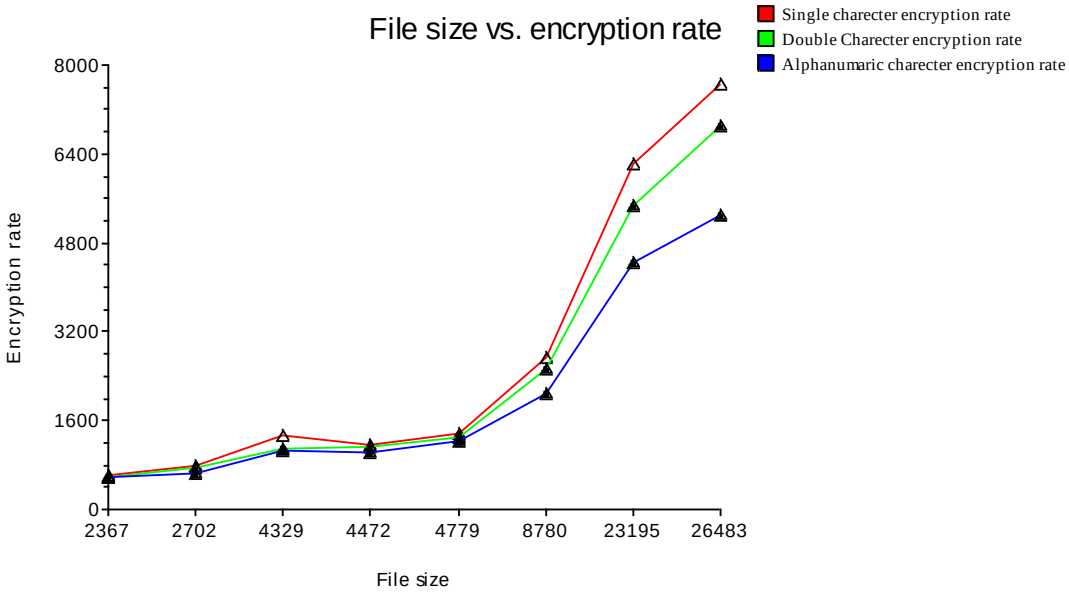

Fig.5.30 file size versus encryption rate of data set-2

Table 5.6 apply encryption algorithm on library file and calculate encryption & decryption time

Sequence Orientation	Sequences name	Library file size	RSA		DES		AES		Selection encryption					
			Encode Time	Decode Time	Encode Time	Decode Time	Encode Time	Decode Time	Choice selection	Encode Time	Decode Time	Hamming distance	Avalanche effect	Encryption rate (byte/sec)
	CHNTXX	192	1.12	1.75	0.14	0.13	0.04	0.05	S	0.64	0.4	64	33.33	300
									D	0.66	0.4	12	6.25	290.91
									A	1.01	0.4	3	1.563	190.1
	atef1a23	186	0.84	1.75	0.13	0.11	0.03	0.05	S	0.49	0.3	62	33.33	281.82
									D	0.48	0.2	4	2.151	387.5
									A	0.54	0.3	3	1.613	344.44
	atrdnai	180	0.84	2.45	0.12	0.08	0.03	0.07	S	0.42	0.3	60	33.33	428.57
									D	0.41	0.3	8	4.444	439.02
									A	0.	0.3	3	1.667	428.57

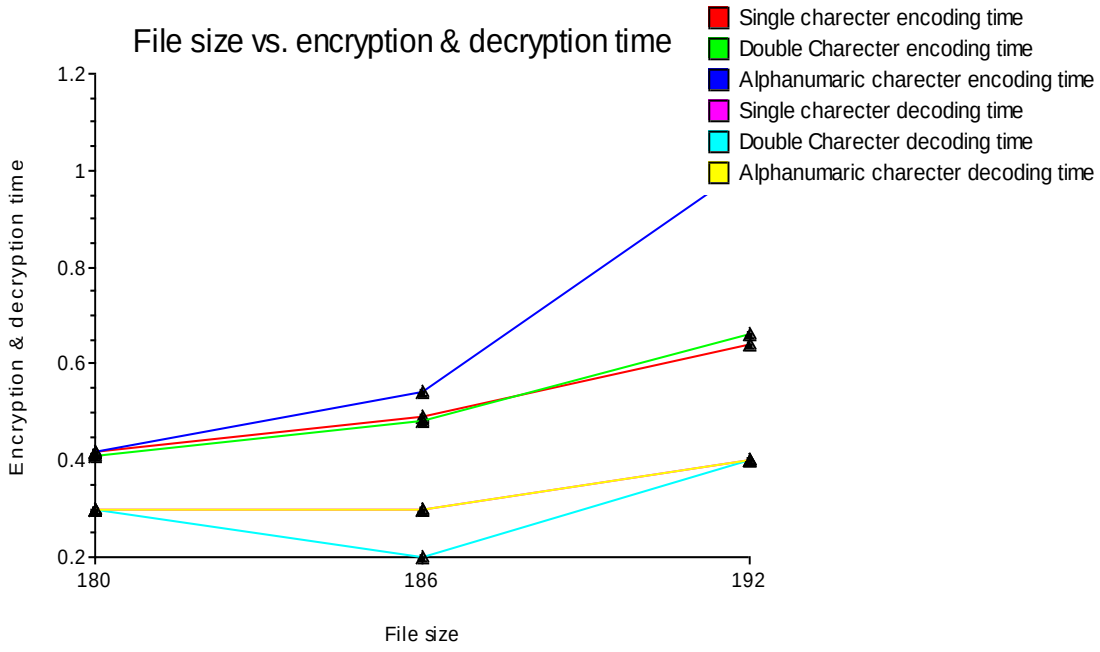

Fig.5.31 time in encryption & decryption vs. file size of library file

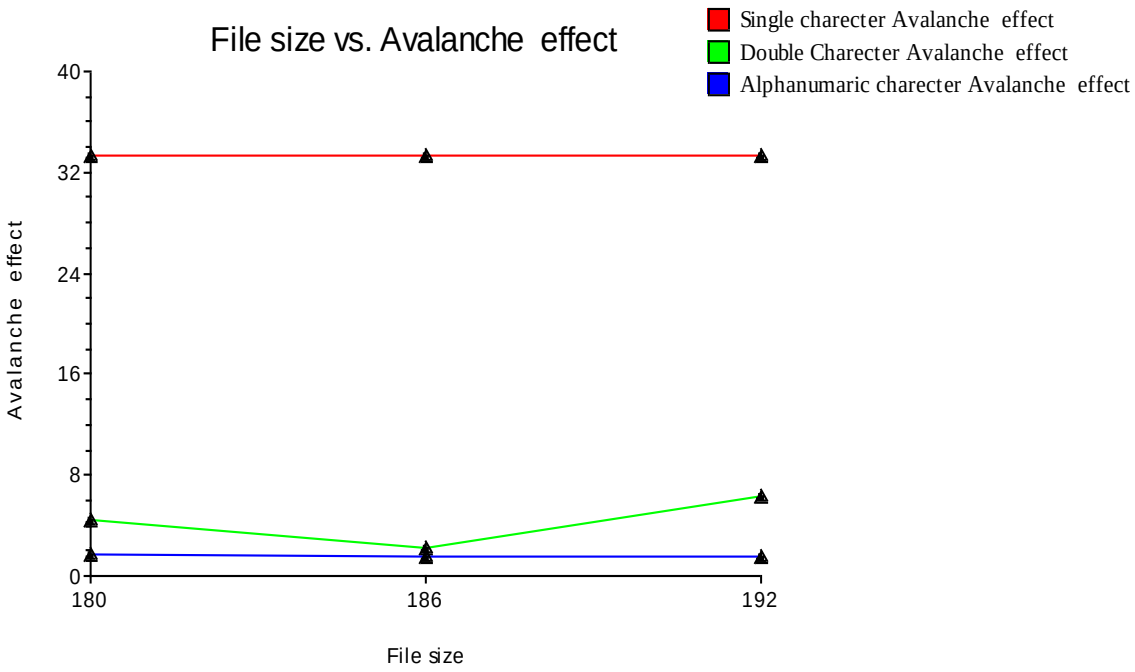

Fig.5.32 file size versus avalanche effect of library file

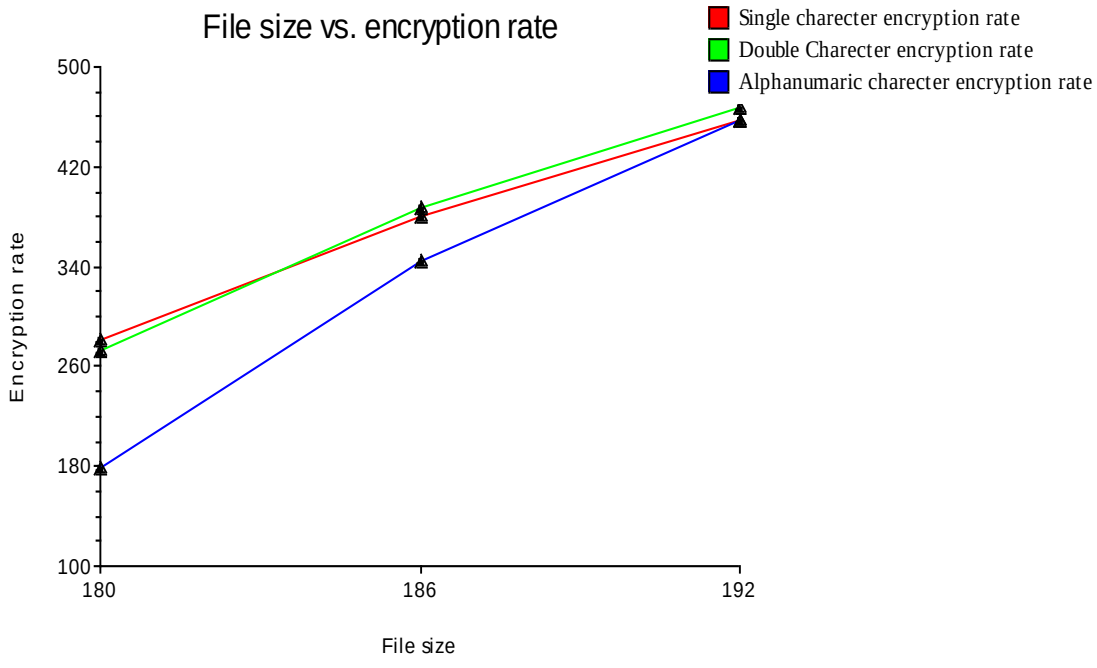

Fig.5.33 file size versus encryption rate of library file

Table 5.7 throughput of the process

Techniques	Data set-I		Data set-II	
	Encryption throughput (byte/sec)	Decryption throughput (byte/sec)	Encryption throughput (byte/sec)	Decryption throughput (byte/sec)
GP^{2}R	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	Independed of data set	

Table 5.8 compression rate using 2nd pass compression technique

Sequence size	Sequence Name	Base pair/ File size	Reduce file size	Lib. File	Total Compress	Compression ratio	Compression rate(bits /base)		
2nd Pass	MTPACGA	100314	42248	212	42460	-0.693084	3.386167	2nd pass Output	Data loss
	MPOMTCG	186608	80592	212	80804	-0.732059	3.464117	98138	2176
	CHNTXX	155844	66650	212	66862	-0.716126	3.432253	184543	2065
	CHMPXX	121024	50700	212	50912	-0.682708	3.365415	152987	2857
	HUMGHCSA	66495	28274	212	28486	-0.713572	3.427145	118131	2893
	HUMHBB	73308	32166	212	32378	-0.766683	3.533366	65491	1004
	HUMHDABCD	58864	25138	212	25350	-0.722615	3.44523	72617	691
	HUMDYSTROP	38770	16679	212	16891	-0.742688	3.485375	57867	997
	HUMHPRTB	56737	24326	212	24538	-0.729947	3.459894	38181	589
	VACCG	191737	81271	212	81483	-0.699891	3.399782	55909	828
	HEHCMVCG	229354	98622	212	98834	-0.723694	3.447387	188566	3171
	Average							227095	2259
								2nd pass Output	Data loss
2nd Pass	atatsgs	9647	4026	212	4238	-0.75723	3.51446	9471	176
	atef1a23	6022	2476	212	2688	-0.78545	3.57091	5931	91
	atrdnaf	10014	4222	212	4434	-0.77112	3.54224	9877	137
	atrdnai	5287	2129	212	2341	-0.77114	3.54227	5042	245
	celk07e12	58949	25549	212	25761	-0.74802	3.49604	58355	594
	hsg6pdgen	52173	22027	212	22239	-0.70502	3.41004	51372	801
	mmzp3g	10833	4416	212	4628	-0.70885	3.41771	10672	161
	xlxfg512	19338	8357	212	8569	-0.77247	3.54494	19087	251
	Average								

For **Cellular sequences**, the compression rate is presented in table 5.1 for data set-1 and 5.2 for the data set-2. The results are presented graphically in fig. 5.3 for the data set-1 and 5.8 for data set-2. The fig. 5.3 & 5.8 drawn on the basis of table 1 and 2 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 graph is heterogeneous in nature because sequences come under from different species as shown in fig. 5.3 & 5.8 for both the data sets. The result is presented in fig. 3 for the data set-1 and 8 for data set-2, shows that the increase in file size decreases the compression rate. The disk utilization and encryption rate are are shown in fig. 6 for data set-I and 11 for data set-II where it is shown that both changes are parallel with file size.

For **Artificial data**, the compression rate is presented in table 5.1 for data set-1 and 5.2 for the data set-2. The results are presented graphically in fig. 5.4 for the data set-1 and 5.9 for data set-2. The fig. 5.4 & 5.9 are drawn on the basis of table 5.1 & 5.2 considers only the compressed file, shows that the compression rate is dependent of 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 complement. The nature of graph is homogeneous in nature because sequences are randomly generated as shown in fig. 5.4 & 5.9. Now draw a fig. 5.5 for the data set-1 and 5.10 for data set-2 on the basis of cellular sequences versus artificial data, getting distinct fig. with naked eye shows two different graph characteristic. Where observed that cellular sequences have structure and non random data, whereas random data is unstructured. Also observed that library file is constant in size in case of an artificial sequence whereas library file is variable in size in case of cellular sequence.

If sequences encrypt by three/four character secret keys (Genetic palindrome, palindrome & reverse technique, sequences compressed by the sub-sequence/word of different size), calculate percentage of encryption and percentage of modification of actual text, this result is presented in table 5.3 for both the data set. From this it is observed that 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 table 5.3 shows that the entropy is increased two to three times before and after compression. As a results both the compressed file and library file increased the randomness, so, the attacker can not attack the sequence easily. The fig.5.13,5.14 & 5.15 for data set-1 and 5.16,5.17 & 5.18 shows the encryption is in increasing order. The fig. 5.19 for data set-I & 5.20 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.

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

In selection encryption, the file is encrypted on the basis of the basic principal of selection i,e selection of a single character (define by S in table 5.4 ,5.5 & 5.6),double character (define by D in table 5.4 ,5.5 & 5.6) and alphanumeric character (define by A in table 5.4 ,5.5 & 5.6). The result shown in table 5.4 for both the data set in the original file, 5.5 for both the data set in compressed file and table 5.6 for the library file. The table 5.4,5.5 & 5.6 shows that RSA exhibits highest avalanche effect. The Avalanche effect gives us the extent of diffusion of the message. One bit of change in the plaintext brings about the significant change in bits of the cipher text. The data is presented in fig. 5.21 to 5.24 for both the data set for original file, fig. 5.25 to 5.30 for compressed file and fig. 5.31 to 5.33 for 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 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 appropriate selection of character decode the encrypted text and get back the original text as get 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 5.7 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 presented in the table 5.4 & 5.5 both the data set and table 5.6 for the library file. This tabular result for both the data set is presented in graphically in fig. 5.21 to 5.24 for source file, 5.25 to 5.30 for compressed file and 5.31 to 5.33 for library file. It is observed that encryption time is always greater than decryption time. The encryption and decryption time is independent of file size. This is the basic principal for all encryption techniques (AES / DES / RSA / DNA). So these techniques are very much effective in security purpose.

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

13.Conclusion:

The results show that 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 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 pattern are similar in all kinds of sources. A major part is played by lookup table in finding the regularities and similarities of DNA order.

The output text file has ASCII symbol and non matched c,t,g & a, providing information safety. It is very useful for the data transmission and provides data protection. This process protects the particular source of DNA sequence. Here we can get better security than 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 DNA order. The output of our experiments also shows that our process is better than other standard processes. 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 limitation of

this research work , if non match 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 orientation.
iv) It is observed that the cellular DNA sequence compression rate & ratio are distinquisably different because this data set are collected from different species. The artificial data generated by the random string generation process, compression rate & ratio are similar both the data set
v) The first pass algorithm is lossless where as the 2^{nd} pass algorithm is lossy.
vi) Our algorithm works more efficiently on a short pattern than long pattern.
vii) The compression-encryption output file efficiency increases and is user friendly.

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 decrypt of the file by the unauthorised persons is very hard. Also 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 logical organization, structure, systematic and non random whereas artificial data are random and unstructured. Also observed that cellular DNA sequence compression ratio follows the equation as {1-Output/2Xinput} where as the artificial data, the compression ratio is followed by the equation as {1-Output/input}; where the output size is 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.