Chapter 4

DNA SEQUENCES COMPRESSION USING REPEAT TECHNIQUE AND SELECTIVE ENCRYPTION USING MODIFIED HUFFMAN'S TECHNIQUE

Abstract

The DNA (Deoxyribonucleic Acid) database size increases tremendously transmuting from millions to billions in a year. Ergo for storing, probing the DNA database requires efficient lossless compression and encryption algorithm for secure communication.

The DNA short pattern repetitions are of paramount characteristics in biological sequences. This algorithm is predicated on probing exact reiterate, substring substitute by corresponding ASCII code and engender a Library file, as a result get cumulating of the data stream. In this technique the data is secured utilizing ASCII value and engendering Library file which acts as a signature. The security of information is the most challenging question with veneration to the communication perspective. The selective encryption method is used for security purpose, this technique is applied on compressed data or in the library file or in both files. The fractional part of a message is encrypted in the selective encryption method keeping the remaining part unchanged, this is very paramount with reference to selective encryption system. The Huffman's algorithm is applied in the output of the first phase reiterate technique, including transmuting the Huffman's tree level position and node position for encryption. The mass demand is the minimum storage requirement and computation cost.

The artificial data of equipollent length is additionally tested by this algorithm. This modified Huffman technique reduces the compression rate & ratio. The experimental result shows that only 58% to 100% encryption on actual file is done when above 99% modification is in actual file can be observed and compression rate is 1.97bits/base.

Key word : Sequence, Compression, decompression, rate, ratio, Huffman, Node, speed, encryption & decryption .

1. Introduction

The whole DNA chains of many structures are already recognized and the entire human genome challenge is making regular progress. The statistics of DNA, RNA and amino-acid chains of proteins are stored in molecular biology databases. Now a day’s data reliability is a tough mission, a way to shield the DNA facts from the hackers [1].The size of DNA database for both prokaryotes and eukaryotes are extremely increasing [19], complex, and have a few logical structures [34], so this large database required a systematic compression approach for storing[67,68,8,73]. However, if one implements standard software such as “compack”,

"pkzip" and "arj", these software enlarges the file size with increasingly above 8 bits per base, albeit these standard compression software are used in text files and structure & function in DNA chains are more precise.

It means that traditional compression algorithm is inapplicable on DNA chains. To compress DNA content a better model is essential, higher statistics compression outcomes can be completed. The Huffman's code additionally is inapplicable also on genomic data both for the adaptive and static version, as the occurrence of probabilities of the 4 representatives are not extremely unlike [8]. In dictionary base compression, the genomic identity is not fully maintained, compromise genome identity. The use of reliability straight in the cellular DNA chain, obtain extremely small tag reliability because the DNA chain holds only 4 characters, by trial and error methods anyone can hack the data. Traditional symmetric key block ciphers like Data Encryption Excellence (DES), Advanced Encryption Excellence (AES), Rivest-Shamir-Adleman (RSA) and Escrowed Encryption Excellence are not systematic when the data size is large[74]. Also the problem is how to differentiate by naked eye the randomly generated artificial chains and Cellular DNA chains over communication point of view. Most of the available compression techniques are reducing the compression rate without considering the reliability concern. In selection- encryption, small part of the sequence is encrypted, other part is unencrypted, shown in fig. 4.1.

This string matching is scanned left-to-right by the use of individual shift rule[74], for the encryption purpose exchange inside the Huffman's tree branches at a specific node & level on the same key required for decoding the encoded representatives using the particular modified Huffman's tree. This approach is applicable on compressed genomic data string and has to be compressed due to constraint of the network bandwidth before communicating. As per Shannon theory [16,11], source entropy is equal to the average bit rate when lossless compression occurs. If joined both the compression & selection encryption process, getting another property as shown in fig. 4.2

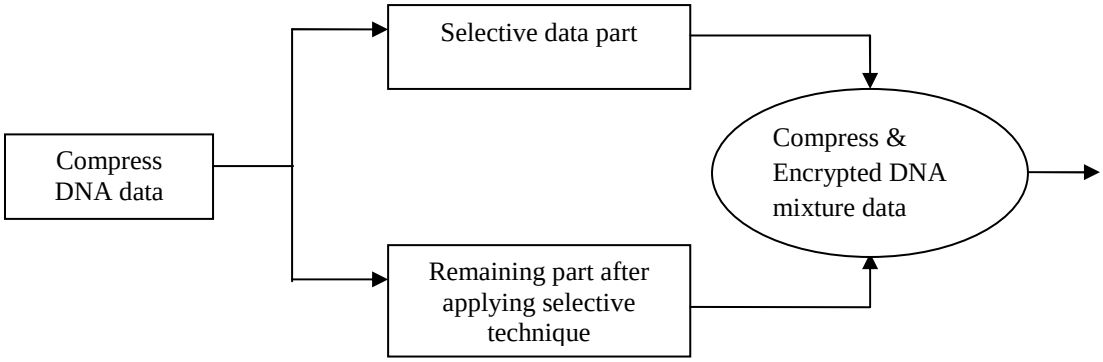

Fig.4.1 process of selection encryption

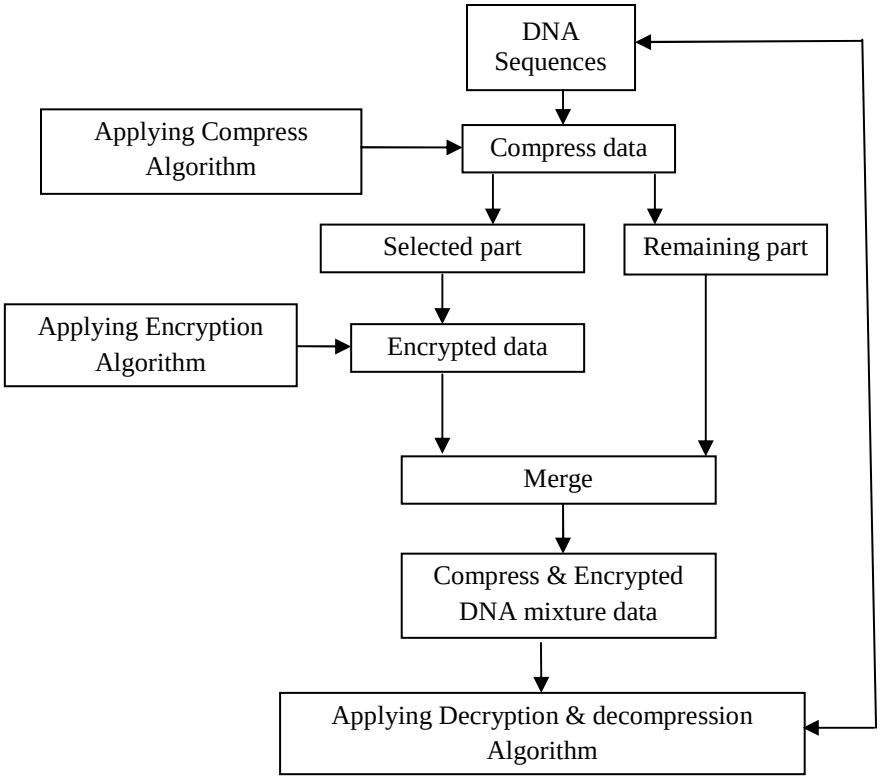

Fig. 4.2 compression- encryption process

The selected encryption is done only in the match region of plain text and group of nucleotide is selected for encryption. The match region identification is the basic principal of this technique. If the subsequence match is high in a sequence, in that situation the security level is increased, also increase the time for encryption. In a plain text to find the match region is the benefit of the compression. The redundancy of the plain text is reduced by this technique and cipher text size is not larger than the plain text. So, the technique is user friendly. The plain text is reformed by selective encryption where match region is higher, as a result producing great effect on encryption through decompression. The remaining part where no

match occurs produces no effect on encryption after decompression. The compression is done by the grouping of three/ four nucleotide sequence and replace by a single ASCII code, at the same time getting very high security by selection encryption. The group of nucleotide which is replaced by single ASCII code act as a key. This key is private is known only who encrypt and sending the sequence. If anyone tries to decrypt the sequence without proper grouping of sequence/key or library file, the sequence is changed. The decryption process is known to everyone and the private key is unknown to everyone so this process of cryptography is reliable.

To find out exact repeat is not an easy task in a very long sequence. The exact repeat searching algorithm requires more time for execution. Every compression algorithm has an aim to minimize the compression rate with operational time. It is based on dynamic repeat searching and repeat substring is placed in Library file and maximum repeat places is replaced by ASCII character.

The operating time is very less and it depends on the input file size. The evaluation of any encryption system depends on its speed and levels of security it provides. The operating time is minimum, required minute recollection and facilely utilizes this algorithm.

The Huffman [49] coding is used for lossless compression technique. It is a particular type of optimal prefix code and output can be viewed as a code of variable length. The Huffman's code is one sort of measuring code [75] and entropy coding [33] it allocates codes to images as to coordinate code lengths with the probabilities of the images.

Four phases of the proposed algorithm i) All exact repeat finding ii) finding un-match and match regions and encode match region iii) apply modified Huffman's technique on tree node position and label position for security purpose and iv) Selective encryption applied in a compressed, library file or in both.

Developed algorithms, discussed experimental results and compare this result with standard available results [76,37,77,78,29,79]. To complete this work, also developed other related algorithms as file size measurement, calculating numbers of bases in a file, i,e file size, file mapping algorithm; is developed because no mathematical formula for proving the two files are same or not, this check input-output file character one by one, change the DNA sequences orientation for reverse, reverse complement the sequence and the same work on random string also. We observed that compression rate, ratio and run time on benchmark [48] DNA

sequences is better than any standard technique, simultaneously. Also find out the run time performance of this algorithm.

2 : Motivation and contribution

In such applications, it is desirable that databases are stored compactly, that orders can be obtained not dependently of the order in which they were stored and that facts can be rapidly get back from auxiliary storage, since disk costs are often the bottleneck in looking for. These processes have need of much time for I/O compression-encryption and decryption decompression method. But in some examples, compression enlarges the over head like size of place for keeping records and processing time and so on. To prevent data loss during sending, many compression algorithms are used to get changed to other forms, the size of the facts during transmission. However, they are also deeply related to order and data mining and observations genomic facts compression, and the connected techniques coming from information theory, are often sensed as being of interest of facts exchange and space for storing. The repeat identification is an important characteristic of compression technique. A chief trouble in repeat identification comes about from the facts that the repeat unit can be certain, errorless and given details of measure end to end.

This algorithm is an effective tool for compression-encryption of DNA sequences by using exact repeat and modified Huffman's technique. The important feature of repetition in biological sequence has not been observed by all available algorithm. Our algorithm overcome this drawback & considered this issue.

3. Methods

3.1 Process diagram

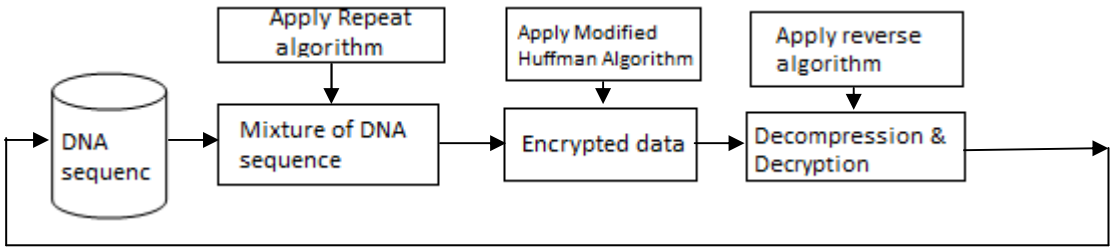

Fig.4.3 Process diagram

3.2 File format : The algorithm testing purpose use text file and end of file is indicated by the blank space. The result is also stored in a text file, the output file is the mixture of unmatched four base pair and ASCII notation.

3.3 Formation of substring / word of different size

Consider a DNA sequence is atggtagtaatgtacatg …… ...n. n is the size of file, required n byte to store this file. The sub-string formation method is in paper[80].

3.4 Merge Process :The Merge process is used for reducing the compression rate and other parameter of this work. The merge compression process is two pass, 1st pass followed by repeat and in 2nd pass uses excellent lossless compression techniques available in the market such as Huffman's technique. The first step consists of repeat coding (let each individual repeat process is called A, the output is O_1), and second step use excellence Huffman coding (let each individual process called B, the output is O_f) process. O_f is the final output, shown in fig.4.4.

The procedure for Multi step DNA Chains compression

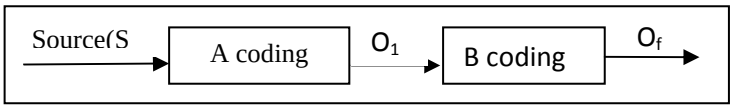

Fig. 4.4 Merge Process

Define as - Let, S be source file to be coded.

Step 1 : $O_1 \leftarrow 1^{st}$ pass of Repeat coding(s)

Step 2 : $O_f \leftarrow$ Huffman's' coding (O_1)

For reliability purpose, introduced a new reliable method two tier selection encryption method as shown in fig.4.5.

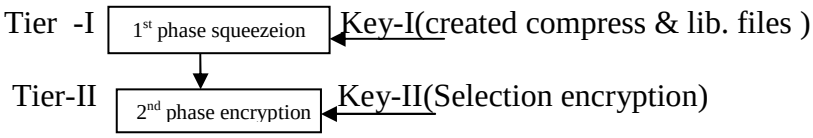

Fig. 4.5 Two tier compression-encryption method

Tier one- the input order has within only 4 symbols (c, t. g & a), after compression is changed to the other form, get four characters to 256 notation with un-match c, t. g & a and one sub-string has in it three characters, replaced by single ASCII symbol. As a result the output file is safer than the input file.

In tier two- In this way of encryption process the file is encrypted either in compressed file or in the library file or in both. The process of encryption is done by exchanging of the branches of Huffman's tree.

This technique bulwarks the DNA sequence information from hackers. The decoding time required the authentic encoding value, this value can provide the security, this security is applied in tier one. This technique uses only available ASCII code for encryption purpose and different pattern is utilized for cull encryption purport. The DNA sequences probing purpose the utilizer can send the encrypt compressed data to the receiver and the receiver decrypt the encrypted data by utilizing correct coded value. The transmission time is reduced over the Internet while the compressed file is decrypted followed by decompression at the client end. The utilizations of DNA sequence is incremented by applying compression and security techniques. This technique increases the efficiency of DNA uses.

3.5 The Complexity of this technique

3.5.1 Time complexity of Repeat algorithm

Let us assume "n" number of pattern strings and N is the total numbers of database string.

n=0,1,2,3 ……n; N=0,1,2,3 ……N ;M=0,1,2,3 ……n

To search n numbers of patterns we need a sequential search. Therefore, it needs N numbers of comparisons. (if, n<=N).If "m" numbers of patterns we need m X N numbers of comparisons.

We need N X N numbers of comparisons. (if, m<=N).Therefore, $O(N^2)$ is the time complexity.

3.5.2 Space complexity of Repeat algorithm

Let "n" numbers of pattern to be searched from the database. Then n numbers of space is required .

n=0,1,2,3 ……n;N=0,1,2,3 ……N

It need N numbers of space (if n<=N).Therefore space complexity is O(N)

3.5.3 Time complexity of Huffman algorithm*:* The weight of each tree, requires O(nlog n) iteration time to determine the cheapest weight and insert the new weight. Each item required O(n) iterations. i,e

O(n) + O(n log n) + O(1) = O(n log n).

3.5.4 Space complexity of Huffman algorithm

If n nodes in Huffman's tree, the space complexity of swapping nodes at the specified level is O(n log n).

3.6 Introduction of Repeat technique

In repeat technique, the highly repeated sub- sequence is replaced by a single ASCII code in source file and subsequence is placed into the library file dynamically. This dynamic library file work as a lookup table and act as security key, known only who encrypt the sequence. This substring length and ASCII code starting position depends on the user.

In two ways proposed algorithm work as first find out all the perfect match repeated substring. Second perfect match region is encoded by ASCII code and non match bases placed into the output file.

3.7 Methodology of Repeat Technique

Consider a DNA sequence as s=atgtggtagtaatgtacatgcatgtgg……..n

In repeat technique, the principal idea is as the substring s_1=atg is repeated in how many places, is shown by red color. The s_2=tgg sub-string repeated in how many places is shown by the green color and so on.

First replace maximum repeated substring by the corresponding ASCII code in appropriate places.

The input string ***S***, assume that a part w_r (variable word/sub-sequence size) has been compressed , it is defined as $S=w_{ri}$ (where i is denoted different word from 1 to n nos.) . The algorithm finds an optimal match position, stored in ascending order that can be encoded economically. This left to right scanning process is search character by character, if no optimal match is found, left the character, moved the process forward at the end of file, shown in fig. 4.6.

Fig. 4.6 Process of Repeat technique

3.8 Searching procedure

Searching for exact repetitions, encoding procedures of Repeat technique , the encoding analysis of Repeat technique and decoding procedure of Repeat technique are discussed in detail in paper[81]

3.9 Compression & decompression algorithm of Repeat technique

The DNA sequence compression algorithm based of Repeat technique

INPUTS:
i. DNA sequence & Artificial sequence in text format
ii. Word size of length l
iii. $S=w_{ri}$

ESTIMATED OUTPUT :
i. Compressed output file and Library file

START
i. Define ASCII code start position
ii. Word size 1 to <10 and count
iii. Product of different word
iv. Match word with the DNA sequence
v. Request to store output in two separate files

ITERATION

while $w_r \neq$ end of file **do**

Search for an optimal postfix of different word with the DNA sequence

if an optimal postfix is found , store in ascending order **then**

Encode the maximum repeat substring by ASCII code, where i is

Starting word position and l is the length of word. Output the code.

else Set w_{ri} in next step, encode and output it.

Remove the temporary compressed file and Library file

End

The DNA sequence decompression algorithm based on Repeat technique

INITIALIZATION OF INPUTS:

i. Enter the compressed text file and library file

ESTIMATED OUTPUT :
i. Exact original sequence

START
iv. Replace ASCII code by DNA sub-sequence

ITERATION
1. for(check library file size)
   flib[i]=fcom[i]
   for(match library file size with subsequence size)
   fname[i]=fcom[i]

2. Read the compressed file character by character.

3. if(ASCII code is matched with sub-sequence);
   produce original sequence
   Else repeat the process

4. Do step 2 to 3 until end of file is reached.

5. Generate original file.

7. End

3.10 Methodology of experiments performed in modified Hoffman's technique

In the first phase repeat experiment is done on different size of DNA sequences and Huffman's tree is generated using the output of statistical property of 1st phase compressed data. The main aim is to select the r part in the output of the 1st phase compressed data and on the basis of key swapping the Huffman's tree branches, this is called encoding, decoding required actual encoding key. This modified Huffman's technique is classified in the process I, II & III.

Process-I: Swapping the Huffman tree nodes at a particular level.

Process-II: Swap the Huffman tree particular nodes at different level. This process is done on character as well word.

Process-III : considering words instead of character

Process-I : First select the r part in the compressed text. On the key basis the Huffman's tree nodes is swapped at a particular level and decode using the encoding key of the modified Huffman's tree. The left and right nodes are interchanged at a particular level. Due to interchanging of node corresponding code is affected and remaining codes are also altered. Only interchanged nodes are affected other node is as usual and nodes related bit is altered.

The above process-I is explained below figure 4.7,4.8 & 4.9 where Huffman's codes are M=00, N=01, R=10, S=11.

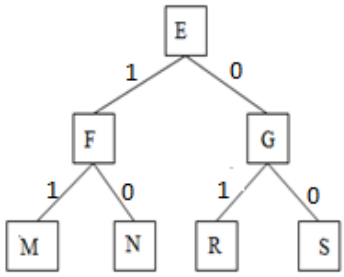

Fig. 4.7 example of process –I

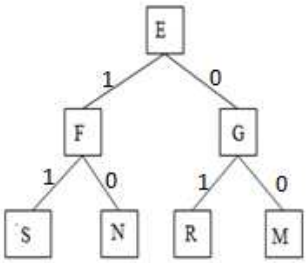

Fig. 4.8 level 1 swapping

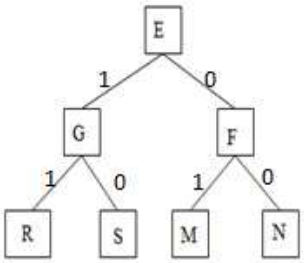

Fig. 4.9 level 2 swapping

Swapping is done at level 1 where E is single node is called root node and the level 0. Now change the child node position at level 1 with their sub tree is explained in fig. 4.9. Simultaneously sub tree position and value are changed as M=01, N=00, R=11, S=10. The actual text "MMNRNS" is encrypted as "010100110010". Decoding without actual encoding key the text is "RRSMSN". Here the Lavenstein distance is 6.

Next swapping at level 2, the interchange left node S with right node M explain in fig. 4.8, codes are also changed and remaining code is same. The actual text is "MMNRNS" and corresponding encrypted value is "000010011011". Decoding without actual encoding key the text is "SSNRNM". Here the Lavenstein distance is 3. The corresponding binary code is shown in table 4.1.

Table 4.1 Huffman code before and after encryption

Character	Before Encrypt	After Encrypt	
		Swapping at Level 1	Swapping at Level 2
M	11	01	00
N	10	00	10
R	01	11	01
S	00	10	11

To find Lavenstein distance on Modified Huffman techniques faced some problem on interchanging file.

Interchanging of nodes are not applicable in all the cases for example, if the frequency is E=2, F=1 and G=1 then tree is explained in fig. 10, interchanging binary node value is E=0, F=10, G=11 and assume string is 'EEFFG' it would be encoded as 00101011.

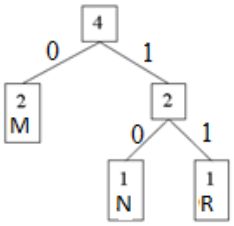

Fig. 4.10 Huffman tree after swapping at level 1

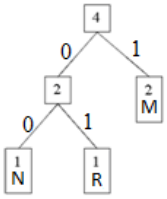

Fig.4.11 Huffman tree after swapping at level 0

The Huffman tree will look like as in the fig. 4.11, when the swapping is done at level 0 causing codeword as M=1, N=00 & R=01 and the string would be encoded as 11000001.
Finding the effect on the actual text and calculate the Lavenstein distance. Then decode the encoded string without swapping technique, the string will look like as table 4.2 with respect to fig.4.9.

Table 4.2 Huffman code after swapping

1	1	0	0	0	0	0	1
R	R	M	M	M	M	M	-

There is no corresponding character for 1 in the last column of the table-3. For accuracy purpose the actual text size is altered in the above cases.

Proceed-II :Two different node at specified level of swapping

The r part is selected by swapping two notes in specified level. This technique is useful for interchanging any two nodes of the Huffman's tree at its subtree level. This technique modifies the actual Huffman concept and enhance the security aspect. This technique required two level value with their corresponding binary codes. The corresponding binary codes are equivalent to their level value with respect to nodes. This process uses two key values for execution, it enhances the security aspect than process-I. This process is depicted as in fig. 12 with their code as M=11, N=10, R=01, S=00.

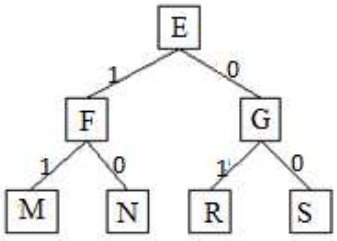

Fig. 4.12 Huffman Tree

Now, interchange in between F & S, as in fig. 4.13 of level 1, in this case the binary value of S is 1 and F is 00. Also the interchange in between N and G is depicted as in fig.4.14.

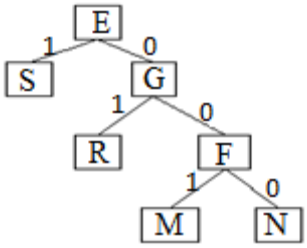

Fig. 4.13 swapping two nodes in level 1

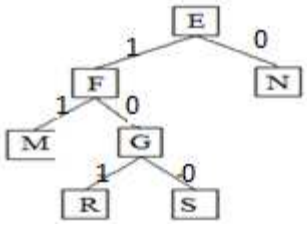

Fig. 4.14 swapping two nodes in level 1

The code value of M, N, R, S are changed as in table 3. If the actual text is as “MMNRNS”, the binary value is encrypted as “111110011000”. If decrypt is done without considering the changed value, the string is as “SNRSSNS” and in this case the D_{SID} is 10. The corresponding binary code shown in table 4.3.

Table 4.3 Huffman code before & after encryption

Character	Before Encrypt	After Encrypt	
		Swapping Between F and S	Swapping Between G and N
M	11	001	11
N	10	000	0
R	01	01	101
S	00	1	100

Process-III : considering words instead of character

In the previous section applied selective encryption considering each character as a symbol in a text document. The characters alone does not possess any meaning but words do have. Generally it is found that in any text document file, there are few vital words. It amends the protection of the whole document. So by swapping the Huffman tree at a lower level (i.e. encrypt a small % of the original file) can encrypt all the keywords. With this idea, new process-III is now illustrating below.

In this scheme take a small text file and using the statistical property of the words of the text, encode input text file. Then made swapping on the basis of process-I and compute the damage occurred due to the swapping with respect to the original text file and got the original text file decoding by modified Huffman tree. Here illustrate this with an example. Now take a simple text

“L.AA.ATG.ATGC.ATGCA.ATGATG.ATGCA.AA”

It contains words and also some punctuation marks. These punctuation marks are also considered as words. Frequencies of words are given in table 4.4.

Table 4.4 word frequency

Distinguishable words	Frequency
L	1
.	7
AA	2
ATG	1
ATGC	1
ATGCA	2
ATGATG.	1

The above string corresponding Huffman's tree is depicted in the fig.4.15.

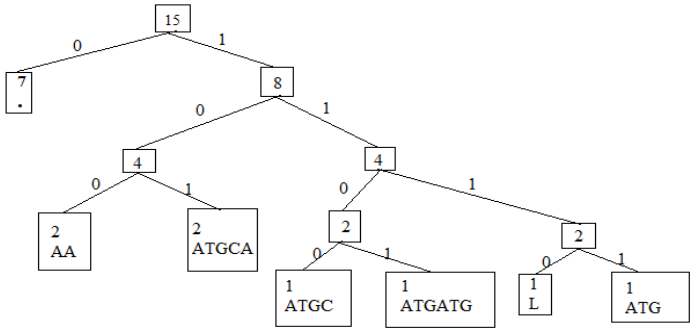

Fig.4.15 Huffman tree using word

Now generate Huffman's code from Huffman's tree which are given in table 4.5.

Table 4.5 word base Huffman code

Distinguishable words	Code
L	1110
.	0
AA	100
ATG	1111
ATGC	1100
ATGCA	101
ATGATG.	1101

So after encoding text message will be 11100100011110110001010110101010100. Next find the selected portion of text file in such a manner that security is not compromised. For this approach perform a swapping method at a specified level (same as performed in case of considering characters).

Now apply swapping method at level 2. Fig. 4.15 shows after swapping.

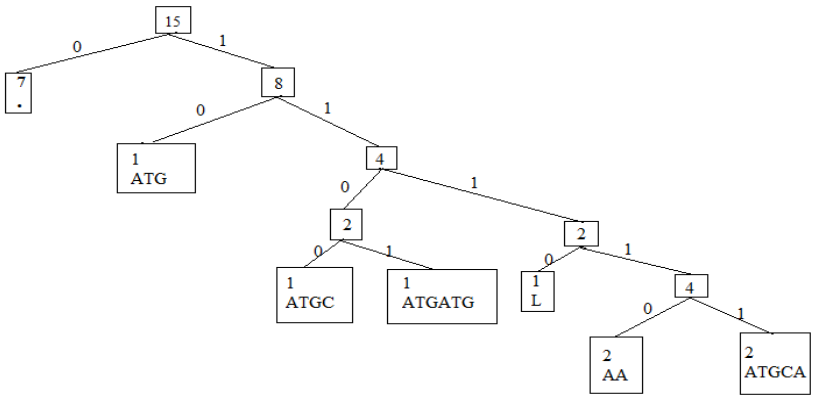

Fig. 4.16 Huffman tree after swapping

So corresponding codes of words are changed selectively, i.e. since in fig. 16, where 2 and 4 interchange their positions as "ATG", "AA.ATGCA" are only changed, other should be

unchanged. After modifying the tree codes corresponding to distinguishable words are changed which are shown in table 4.6

Table 4.6 Huffman code after swapping

Distinguishable words	Code
L	1110
.	0
AA	11110
ATG	10
ATGC	1100
ATGCA	11111
ATGATG.	1101

So after encoding text message will be encrypted like

11100111100100110001111101101011111011110

If not consider changed of level in decoding time the text will be look like "L.ATG.AA.ATGC. ATG. ATGCAATGCA. ATG. ATGCA.L". D_{SID} in this case is 19.

3.11 Encoding algorithm of modified Huffman's technique

Compressed DNA sequence encryption process using Swapping the Huffman tree nodes at a particular level (process-I)
INITIALIZATION OF INPUTS: i. The output of Reverse technique is input of this technique ii. Enter the level of 1st node iii. Enter the binary path of 1st node iv. Enter the level of 2nd node v. Enter the binary path of 2nd node ESTIMATED OUTPUT : i. Input file size ii. Output file iii. Output file size iv. Compression rate v. Encoding time START i. Generate Huffman tree ii. Swapping the Huffman tree nodes at a particular level iii. Request to store the output file ITERATION 1. The key basis of Huffman tree nodes are swapped at a specific level and modified Huffman tree is use for decoding. 2. The r part selection swaps two nodes at specified levels or same levels. To exchange right most node with left most node at specified level. The swapping nodes are only affected also other codes also altered including bits. Except swapping nodes, the remaining nodes are as usual. 3. End

Compressed DNA sequence decryption process using Swapping the Huffman tree nodes at a particular level (process-I)

INITIALIZATION OF INPUTS:
 	
i. Enter the compressed text file
ii. Enter the level of 1st node
iii. Enter the binary path of 1st node
iv. Enter the level of 2nd node
v. Enter the binary path of 2nd node
 	
ESTIMATED OUTPUT :
i. Exact original sequence
ii. Decoding time
 	
START
i. Generate Reverse Huffman tree
 	
ITERATION
1. for(sort the Huffman tree)
for(decoding the swapping tree)
if(frequency match)
reconstruct the code
else repeat the step
End
2. File decompressor for files compressed with HuffEnc.
3. End

Compressed DNA sequence encryption process using Swap the Huffman tree particular nodes at different level. This process also done on character as well word (Process-II).
INITIALIZATION OF INPUTS: i. The output of Reverse technique is input of this technique ii. Enter the lebel to change iii. Enter the binary value ESTIMATED OUTPUT : i. Input file size ii. Output file iii. Output file size iv. Compression rate v. Encoding time START i. Generate Huffman tree ii. Swap the Huffman tree particular nodes at different level iii. Request to store output file ITERATION 1: This process is done in any two particular nodes and including their subtree of Huffman tree. For interchanging purpose use more than one key of modified Huffman tree.

2: This process first identify two particular node of their corresponding level and their binary value, also kept in mind that the binary value is same as level value. This process uses two key value for two nodes, it improves the security than process-II.
3. End

Compressed DNA sequence decryption process using Swap the Huffman tree particular nodes at different level. This process is also done on character as well word (Process-II).

INITIALIZATION OF INPUTS:

i. Enter the compressed text file
ii. Enter the level to change
iii. Enter the binary value

ESTIMATED OUTPUT :
i. Exact original sequence
ii. Decoding time

START
i. Generate reverse Huffman tree

ITERATION
1. for(sort the Huffman tree)
for(decoding the swapping tree)
if(frequency match)
reconstruct the code
else repeat the step
End
2. File decompressor for files compressed with HuffEnc.
3. End

4. Results and discussion

The benchmark DNA data are used for testing this algorithm. This algorithm is tested on data (set-I & set-II) used in [48], this is standard benchmark data and artificial sequence generate by a random string creation method. All the results are described in different tables from 4.7 to 4.19, throughput result is described in table 4.20, improvement results described in table 4.21 and comparison results describe in table 4.22 & 4.23.

Table 4.7 cellular DNA sequences compression ratio, rate and gain where each column displays the result of a single algorithm and rate in bits per base for each sequence (data set-I).

Sequence Size	Sequence Name	Base pair/ File size	Cellular DNA Sequences							
			Original Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	atatsgs	9647	-0.83393	3.66787	-0.83974	3.67948	-0.75014	3.50036	-0.83974	3.67948
	atef1a23	6022	-0.82265	3.6453	-0.83062	3.66124	-0.82132	3.64264	-0.83062	3.66124
	atrdnaf	10014	-0.79029	3.58058	-0.79109	3.58218	-0.78949	3.57898	-0.79109	3.58218
	atrdnai	5287	-0.76811	3.53622	-0.75752	3.51503	-0.7666	3.53319	-0.75752	3.51503
	celk07e12	58949	-0.78004	3.56009	-0.7829	3.56579	-0.77991	3.55982	-0.7829	3.565794
	hsg6pdgen	52173	-0.80131	3.60262	-0.799015	3.598029	-0.801162	3.602323	-0.799015	3.598029
	mmzp3g	10833	-0.79414	3.58829	-0.805225	3.610449	-0.7934	3.58681	-0.80522	3.61044
	xlxfg512	19338	-0.78591	3.57182	-0.78757	3.57513	-0.7855	3.571	-0.78757	3.57513
	Average rate			3.59410		3.59842		3.57189		3.59842
	Average gain		55.18%		55.14%		55.3%		55.14%	
Sub string Size 4	atatsgs	9647	-0.65357	3.30714	-0.64527	3.29055	-0.65274	3.30548	-0.64527	3.30548
	atef1a23	6022	-0.65792	3.31584	-0.65659	3.31318	-0.65659	3.31318	-0.65659	3.31318
	atrdnaf	10014	-0.65688	3.31376	-0.65848	3.31695	-0.65608	3.31216	-0.65848	3.31216
	atrdnai	5287	-0.68413	3.36826	-0.6917	3.38339	-0.68262	3.36523	-0.6917	3.36523
	celk07e12	58949	-0.6005	3.201	-0.60641	3.21281	-0.60037	3.20073	-0.60627	3.20073
	hsg6pdgen	52173	-0.58626	3.17252	-0.594541	3.1890825	-0.586108	3.17221	-0.58978	3.17221
	mmzp3g	10833	-0.66565	3.3313	-0.66602	3.332041	-0.66491	3.32982	-0.66602	3.32982
	xlxfg512	19338	-0.57782	3.15565	-0.57803	3.15606	-0.57741	3.15482	-0.57803	3.15482
	Average rate			3.27068		3.27426		3.26920		3.27303
	Average gain		59.78%		59.67%		59.79%		59.71%	
Sub string Size 5	atatsgs	9647	-0.63242	3.26484	-0.62661	3.25323	-0.46739	2.93479	-0.62661	3.25323
	atef1a23	6022	-0.63932	3.27864	-0.61142	3.22285	-0.63799	3.27598	-0.61142	3.22285
	atrdnaf	10014	-0.59536	3.19073	-0.5682	3.1364	-0.99401	3.98801	-0.70082	3.40163
	atrdnai	5287	-0.57745	3.1549	-0.57897	3.15793	-0.57594	3.15188	-0.57897	3.15793
	celk07e12	58949	-0.77407	4.77117	-1.33008	4.66016	-0.5699	3.13979	-1.33008	4.660164
	hsg6pdgen	52173	-0.73101	3.46202	-0.757767	3.5155349	-0.76252	3.525042	-0.757767	3.5155349
	mmzp3g	10833	-0.70294	3.40588	-0.616173	3.2323456	-0.73138	3.46275	-0.61617	3.23234
	xlxfg512	19338	-0.66594	3.33188	-0.63409	3.26817	-0.57121	3.14241	-0.63409	3.26817
	Average rate			3.48250		3.43083		3.327587		3.46398
	Average gain		57.11%		52.45%		58.49%		52.26%	

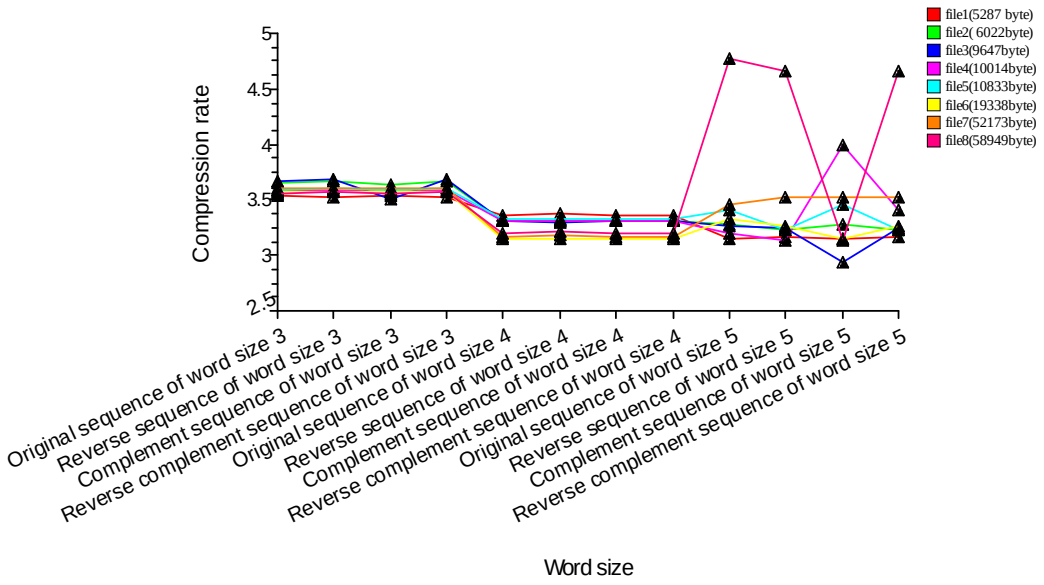

Fig. 4.17 compression rate versus word size among original, reverse, complement and reverse complement sequences using Repeat technique of data set-I

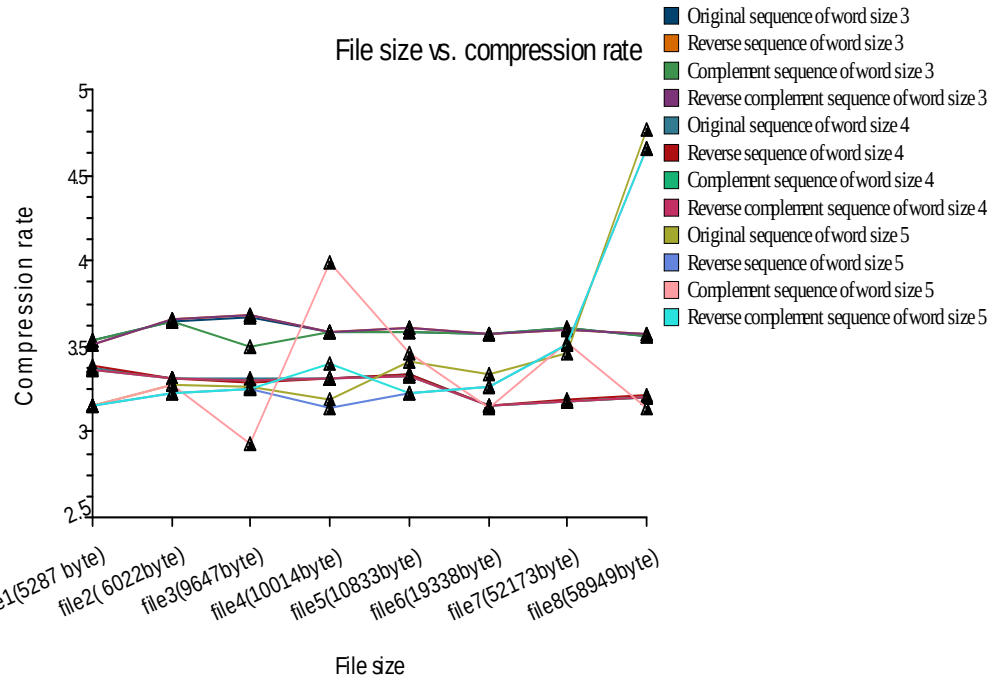

Fig. 4.18 compression rate versus file size among original, reverse, complement and reverse complement sequences using Repeat technique of data set-I

Table 4.8 cellular DNA sequences compression ratio, rate and gain where each column displays the result for a single algorithm and rate in bits per base for each sequence (data set-II).

Sequence Size	Sequence Name	Base pair/ File size	Cellular DNA Sequences							
			Original Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	MTPACGA	100314	-0.79748	3.594952	-0.79795	3.595909	-0.79772	3.59543	-0.79819	3.596387
	MPOMTCG	186608	-0.80258	3.605162	-0.80284	3.605676	-0.80271	3.605419	-0.80295	3.60589
	CHNTXX	155844	-0.80848	3.61695	-0.8094	3.618798	-0.80894	3.617874	-0.80984	3.619671
	CHMPXX	121024	-0.80146	3.602922	-0.80298	3.605962	-0.80222	3.604442	-0.80371	3.607417
	HUMGHCSA	66495	-0.81247	3.624934	-0.81355	3.6271	-0.81301	3.626017	-0.81409	3.628183
	HUMHBB	73308	-0.80832	3.616631	-0.80941	3.618814	-0.80886	3.617722	-0.8099	3.619796
	HUMHDABCD	58864	-0.80124	3.602473	-0.80219	3.604376	-0.80171	3.603425	-0.80246	3.60492
	HUMDYSTROP	38770	-0.80903	3.618055	-0.81068	3.621357	-0.80985	3.619706	-0.8115	3.623007
	HUMHPRTB	56737	-0.80292	3.60583	-0.80348	3.606958	-0.8032	3.606394	-0.80355	3.607099
	VACCG	191737	-0.80869	3.617372	-0.81123	3.622462	-0.80996	3.619917	-0.81244	3.624882
	HEHCMVCG	229354	-0.77948	3.55897	-0.78343	3.566853	-0.78146	3.562911	-0.78538	3.57076
	Average rate			3.605841		3.60857		3.607205		3.609819
	Average gain		54.98606		54.94463		54.96535		54.925	
Sub string Size 4	MTPACGA	100314	-0.60775	3.215503	-0.60815	3.216301	-0.60735	3.214706	-0.608748	3.217497
	MPOMTCG	186608	-0.59515	3.190303	-0.595387	3.190774	-0.59492	3.189831	-0.595773	3.191546
	CHNTXX	155844	-0.60091	3.201817	-0.60114	3.202279	-0.60068	3.201355	-0.601421	3.202844
	CHMPXX	121024	-0.58957	3.179138	-0.589833	3.179667	-0.5893	3.178609	-0.590130	3.180262
	HUMGHCSA	66495	-0.58099	3.161982	-0.581292	3.162584	-0.58069	3.161381	-0.581712	3.163426
	HUMHBB	73308	-0.59742	3.194849	-0.597752	3.195504	-0.5971	3.194194	-0.598133	3.196268
	HUMHDABCD	58864	-0.5793	3.158603	-0.579573	3.159147	-0.57903	3.158059	-0.579913	3.159826
	HUMDYSTROP	38770	-0.60794	3.215889	-0.608151	3.216301	-0.60774	3.215476	-0.608460	3.21692
	HUMHPRTB	56737	-0.59155	3.183108	-0.591766	3.183531	-0.59134	3.182685	-0.592047	3.184095
	VACCG	191737	-0.59287	3.185739	-0.59312	3.186239	-0.59262	3.185238	-0.593474	3.186949
	HEHCMVCG	229354	-0.60585	3.211699	-0.606111	3.212222	-0.60559	3.211176	-0.606407	3.212815
	Average rate			3.190785		3.1913226		3.190246		3.192040
	Average gain		60.08%		60.07%		60.08%		60.06%	

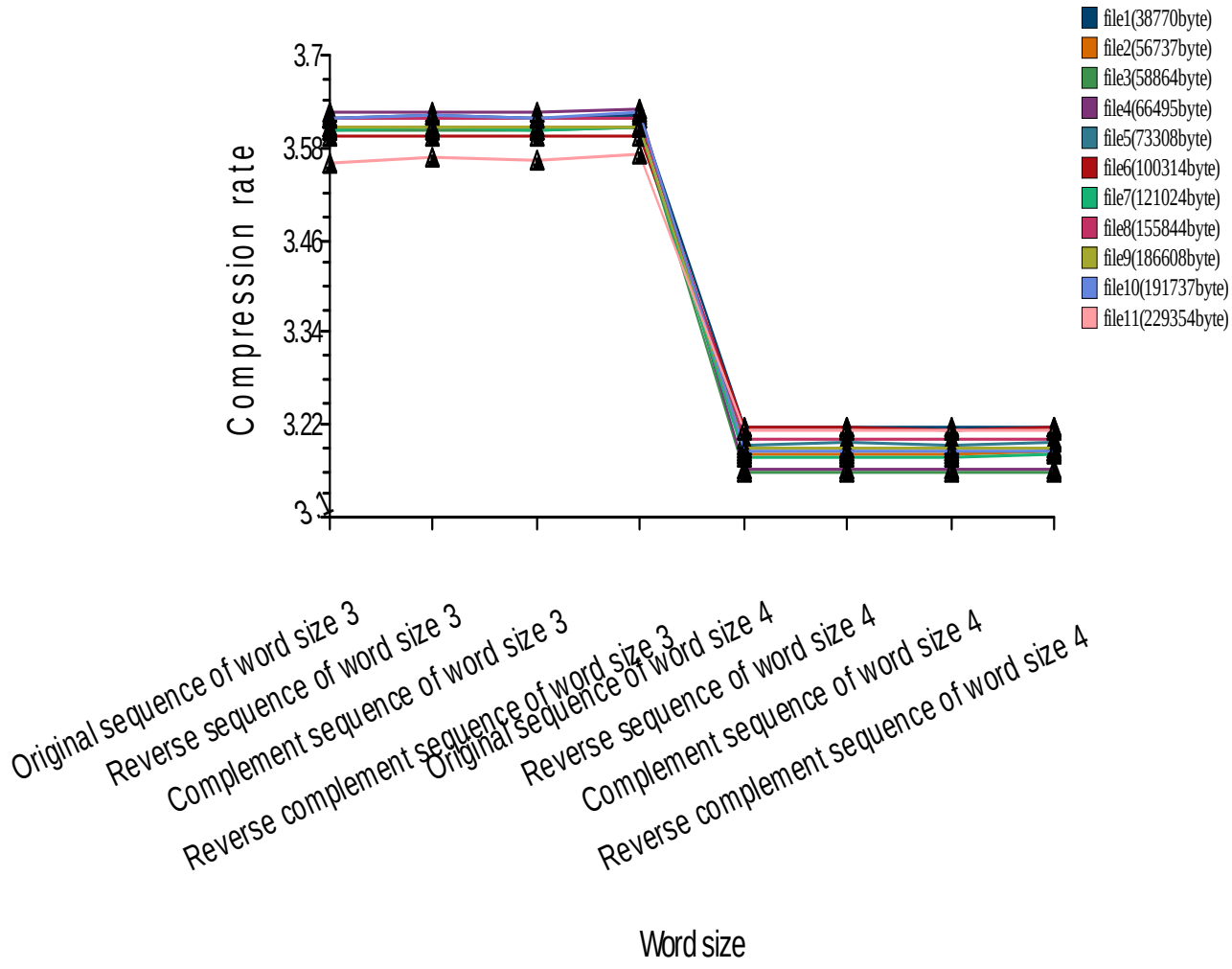

Fig. 4.19 compression rate versus word size among original, reverse, complement and reverse complement sequences using Repeat technique of data set-II

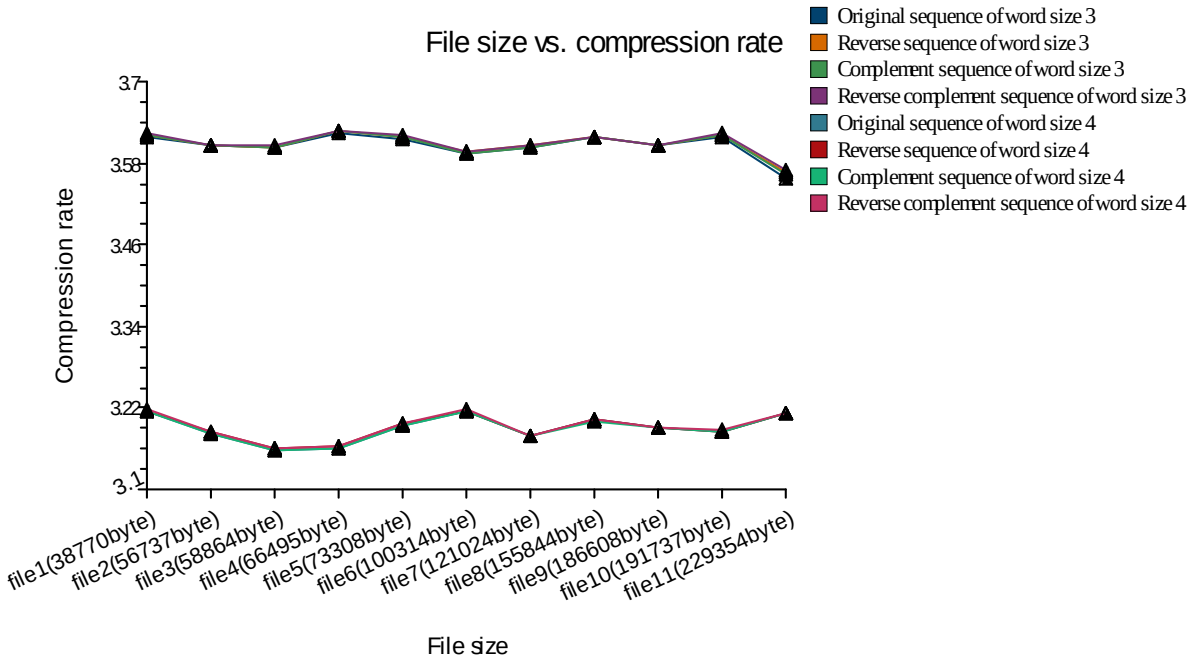

Fig.4.20 compression rate versus file size among original, reverse, complement and reverse complement sequences using Repeat technique of data set-II

Table 4.9 Artificial sequences compression ratio, rate and gain where each column displays the result for a single algorithm and rate in bits per base for each sequence of data set-I.

Sequence Size	Sequence Name	Base pair/ File size	Artificial Sequences							
			Original Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	XX1	9647	-0.82481	3.64963	-0.81818	3.63636	-0.82481	3.64963	-0.81818	3.63636
	XX2	6022	-0.81733	3.63467	-0.816	3.63201	-0.816	3.63201	-0.81601	3.63201
	XX3	10014	-0.81346	3.62692	-0.80786	3.61573	-0.81266	3.62532	-0.80787	3.61573
	XX4	5287	-0.82712	3.65424	-0.82409	3.648193	-0.82561	3.65122	-0.8241	3.64819
	XX5	58949	-0.81343	3.62686	-0.81275	3.6255	-0.81329	3.62659	-0.81275	3.6255
	XX6	52173	-0.81879	3.63759	0.818373	0.363253	-0.81814	0.36371	-0.81837	0.36325
	XX7	10833	-0.81925	3.63851	-0.81999	3.63998	-0.81925	3.63851	-0.81999	3.63998
	XX8	19338	-0.81197	3.62395	-0.81239	3.62478	-0.81156	3.62312	-0.81239	3.62478
	Average rate		3.63654		3.22323		3.22626		3.22323	
	Average gain		54.58%		67%		66.98%		67%	
Sub string Size 4	XX1	9647	-0.64113	3.28226	-0.64527	3.29055	-0.6403	3.2806	-0.64527	3.29055
	XX2	6022	-0.69976	3.39953	-0.68449	3.36898	-0.69844	3.39687	-0.68449	3.36898
	XX3	10014	-0.65927	3.31855	-0.65727	3.31455	-0.65848	3.31695	-0.65728	3.31456
	XX4	5287	-0.67505	3.3501	-0.68942	3.37885	-0.67354	3.34707	-0.68943	3.37885
	XX5	58949	-0.60721	3.21443	-0.60606	3.21213	-0.61652	3.23303	-0.61516	3.23031
	XX6	52173	-0.61018	3.22036	-0.61071	3.2214	-1.09533	4.19067	-0.61899	3.23799
	XX7	10833	-0.65457	3.30914	-0.63463	3.26926	-0.65457	3.30914	-0.63463	3.26926
	XX8	19338	-0.63801	3.27603	-0.63388	3.26776	-0.6376	3.2752	-0.63388	3.26776
	Average rate		3.29630		3.29044		3.41869		3.29478	
	Average gain		59.38%		59.42%		55.63%		59.28%	

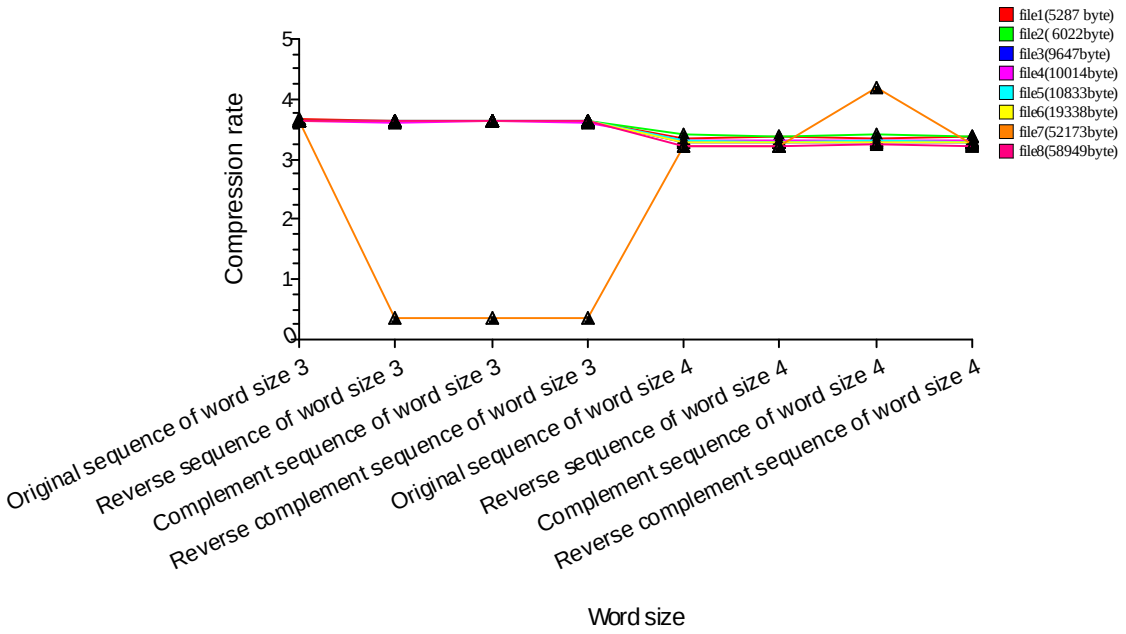

Fig.4.21 the word size versus compression rate of artificial data

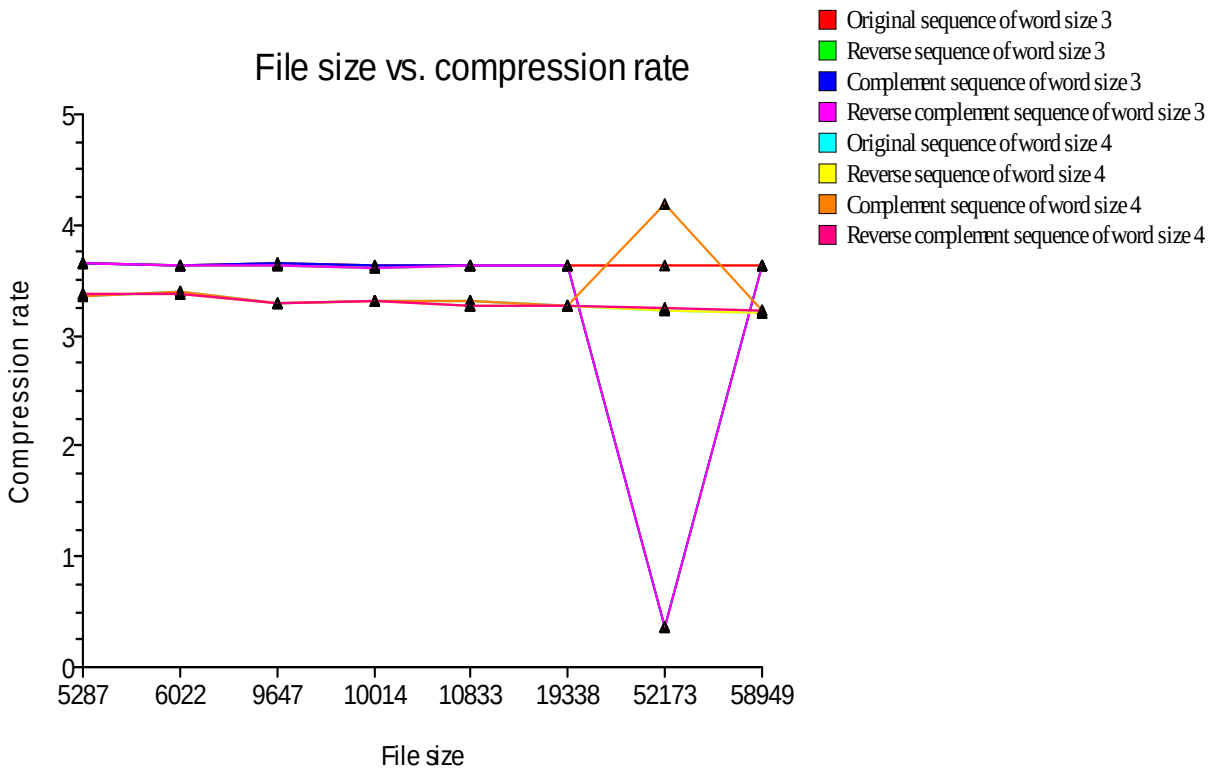

Fig.4.22 the file size versus compression rate of artificial data

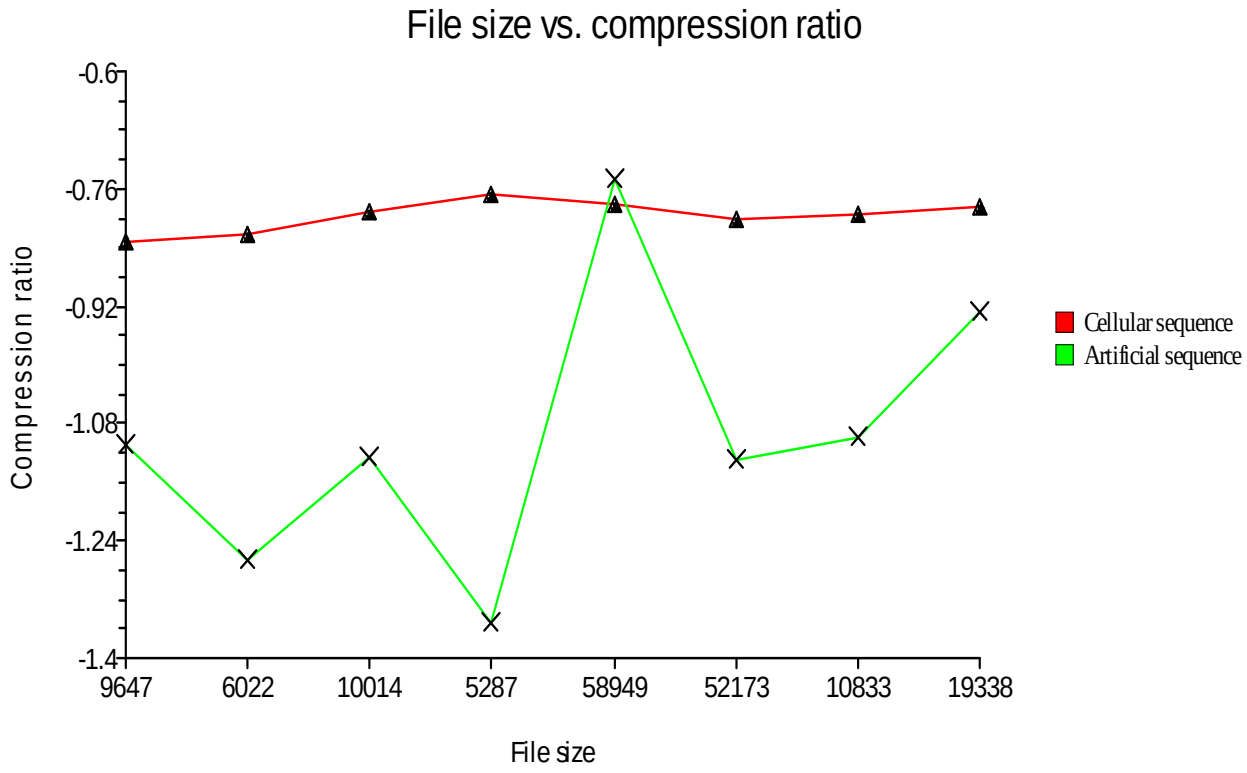

Fig. 4.23 file size versus compression rate of cellular DNA sequences and artificial data

Table 4.10 artificial sequences compression ratio, rate and gain where each column displays the result of a single algorithm and rate in bits per base for each sequence (data set-II).

Sequence Size	Sequence Name	Base pair/ File size	Cellular DNA Sequences							
			Original Sequences		Reverse Sequences		Complement Sequences		Reverse Complement Sequences	
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)
Sub string Size 3	X1	100314	-0.8151	3.6302	-0.81462	3.62924	-0.81063	3.62127	-0.81558	3.63116
	X2	186608	-0.81947	3.63894	-0.81926	3.63851	-0.81733	3.63466	-0.81973	3.63946
	X3	155844	-0.82359	3.64719	-0.82323	3.64647	-0.82105	3.6421	-0.82362	3.64724
	X4	121024	-0.82379	3.64759	-0.82333	3.64666	-0.8208	3.64159	-0.82661	3.65323
	X5	66495	-0.82156	3.64312	-0.82066	3.64132	-0.8165	3.63299	-0.82564	3.65128
	X6	73308	-0.82134	3.64268	-0.82047	3.64093	-0.8171	3.6342	-0.82408	3.64817
	X7	58864	-0.82558	3.65115	-0.82442	3.64884	-0.82074	3.64148	-0.82781	3.65561
	X8	38770	-0.82045	3.64089	-0.81859	3.63718	-0.81377	3.62755	-0.83234	3.66469
	X9	56737	-0.82408	3.64815	-0.82274	3.64548	-0.81998	3.63995	-0.83097	3.66195
	X10	191737	-0.81938	3.63876	-0.81896	3.63793	-0.8183	3.63661	-0.82106	3.64211
	X11	229354	-0.81821	3.63641	-0.81784	3.63568	-0.81742	3.63484	-0.8207	3.64139
	Average rate			3.64228 3.64228		3.64075 3.64075		3.63520		3.64875
	Average gain		54.96%		54.96%		54.95%		54.97%	
Sub string Size 4	X1	100314	-0.80624	3.61249	-0.80632	3.61265	-0.806049	3.61209801	-0.806607	3.61321451
	X2	186608	-0.81497	3.62994	-0.81501	3.63002	-0.814863	3.62972649	-0.815163	3.63032667
	X3	155844	-0.81822	3.63645	-0.81828	3.63655	-0.8181	3.6362003	-0.818459	3.63691897
	X4	121024	-0.81716	3.63432	-0.81722	3.63445	-0.816995	3.63398995	-0.817458	3.63491539
	X5	66495	-0.80988	3.61976	-0.81000	3.62000	-0.80958	3.61915934	-0.810422	3.62084367
	X6	73308	-0.81109	3.62219	-0.81120	3.62241	-0.810826	3.62165112	-0.811589	3.62317892
	X7	58864	-0.81326	3.62652	-0.8134	3.62680	-0.812925	3.62584942	-0.813876	3.62775211
	X8	38770	-0.80242	3.60484	-0.80263	3.60526	-0.801909	3.60381738	-0.803353	3.60670622
	X9	56737	-0.81222	3.62444	-0.81236	3.62472	-0.811869	3.62373762	-0.812856	3.62571162
	X10	191737	-0.81600	3.63201	-0.81605	3.63210	-0.815904	3.63180815	-0.816196	3.63239229
	X11	229354	-0.81549	3.63099	-0.81553	3.63106	-0.815412	3.63082397	-0.815656	3.63131229
	Average rate			3.62491		3.625097		3.62444		3.62575
	Average gain		54.64%		54.64%		54.65%		54.64%	

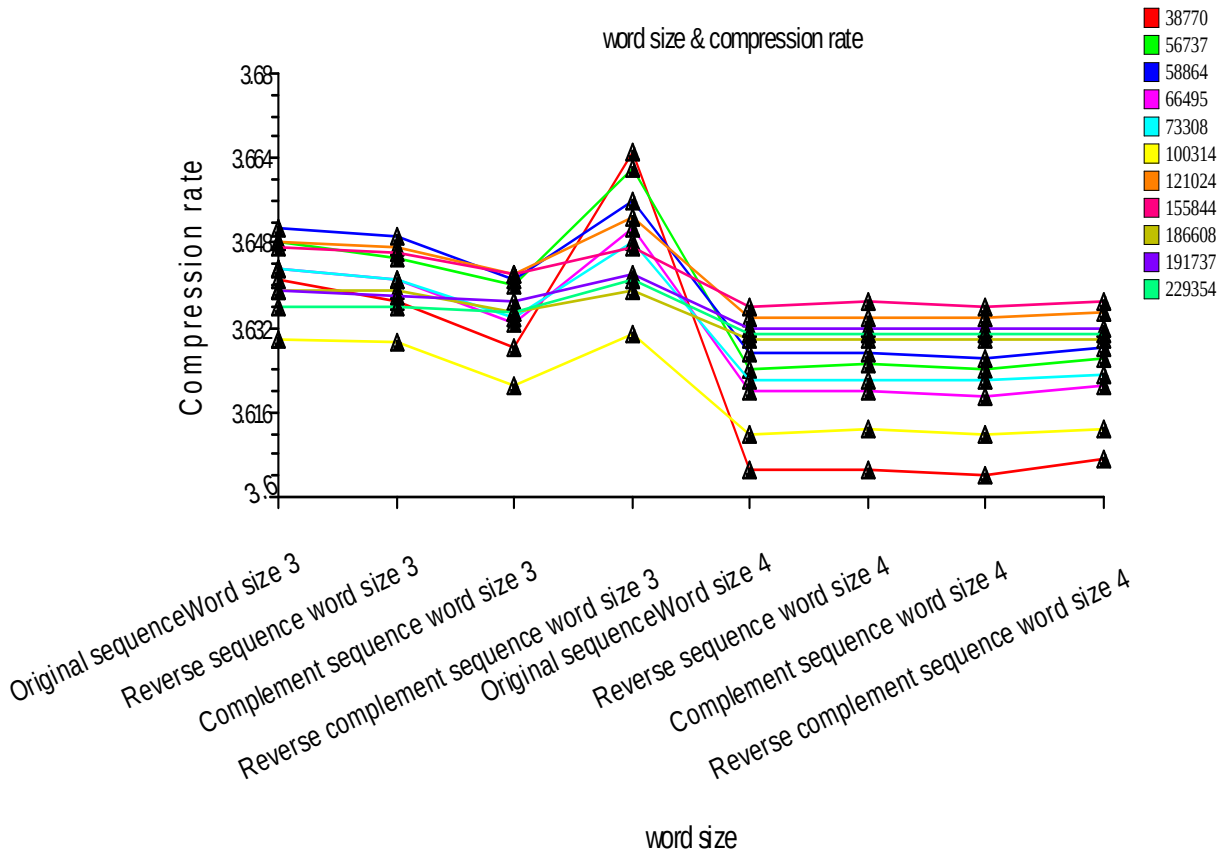

Fig.4.24 word size versus compression rate of artificial data

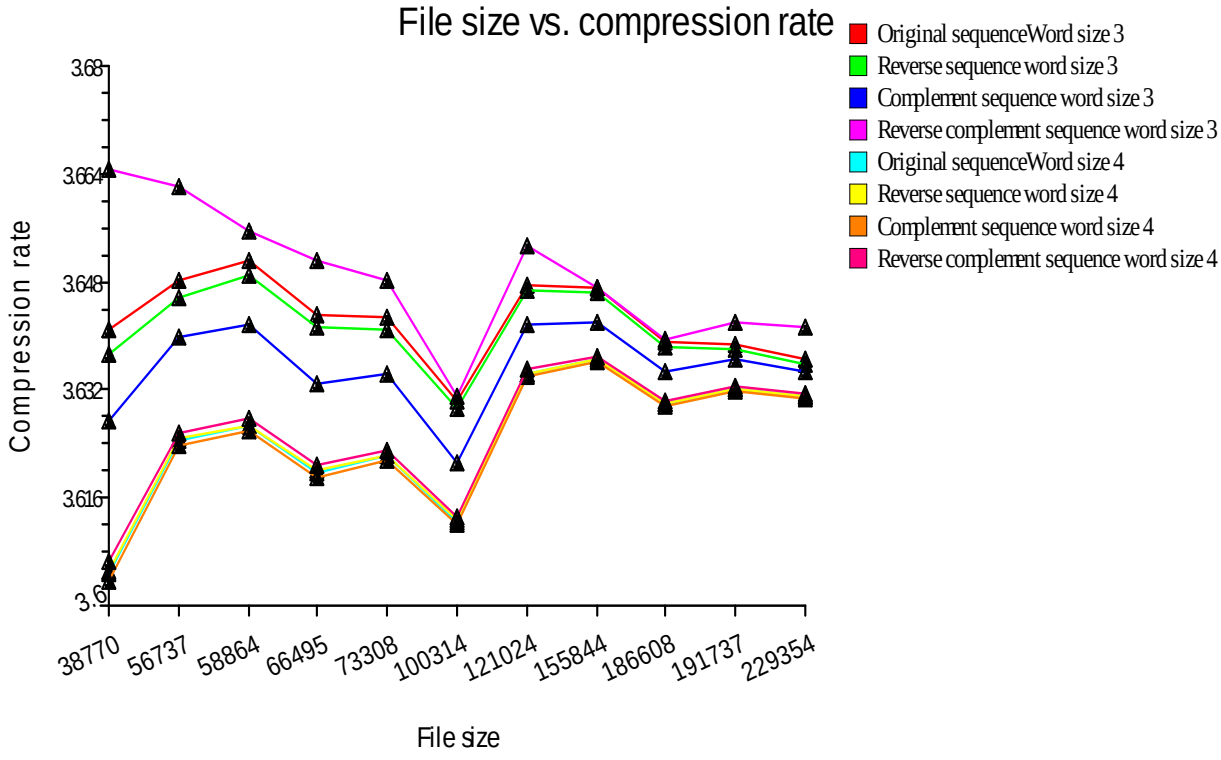

Fig. 4.25 file size versus compression rate of artificial data

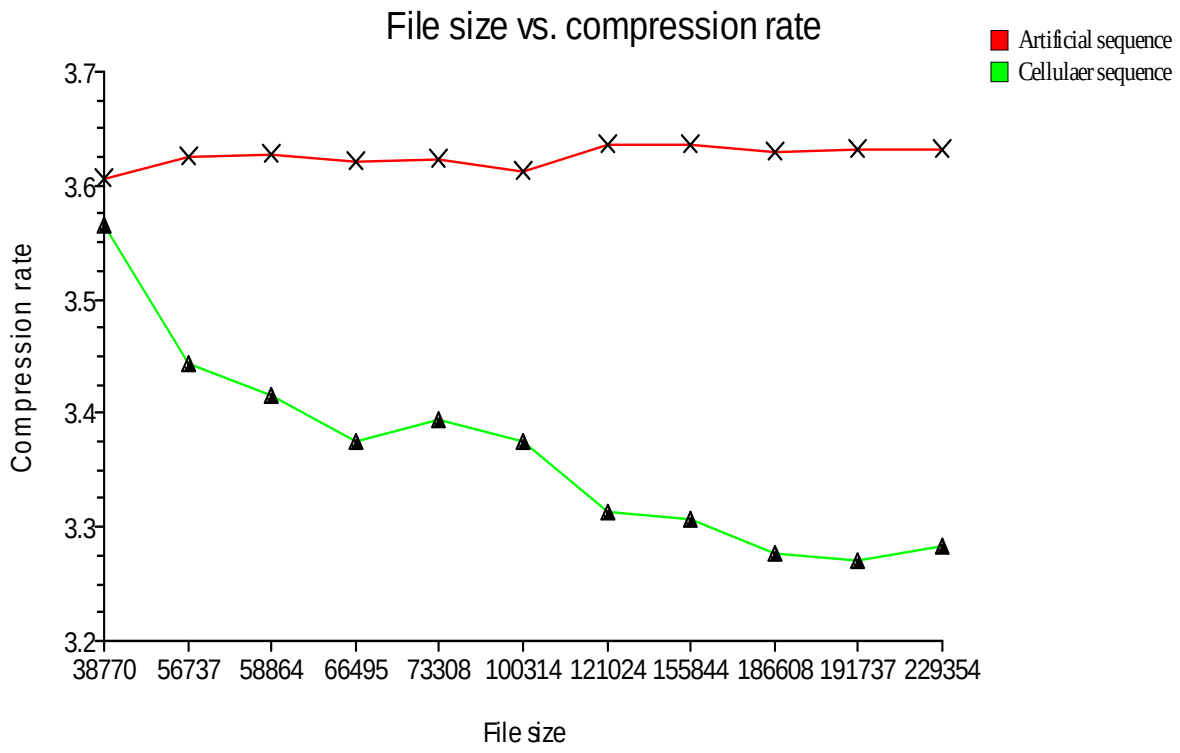

Fig. 4.26 file size versus compression rate of cellular DNA sequences and artificial data

Table 4.11 % encryption, % effect on original file and entropy change using Repeat technique

Data set	Sub string size	Sequence Name	Base pair/ File size	Compress file size	Library file size	Lavenstein Distance	% of encryption	%effect on actual text	Entropy before compression	Entropy after compression	
										Based on compressed file	Based on library file
Data set-I	Sub string size 3	atatsgs	9647	4421	413	9202	45.83	95.39	1.93269	4.59342	3.14512
		atef1a23	6022	2742	392	5735	45.53	95.23	1.97207	4.63627	3.13496
		atrdnaf	10014	4480	441	9597	44.74	95.84	1.99287	4.77932	3.15976
		atrdnai	5287	2335	343	5079	44.16	96.07	1.99159	4.66179	3.10756
		celk07e12	58949	26231	448	56392	44.5	95.66	1.96998	4.69271	3.16310
		hsg6pdgen	52173	23493	448	49801	45.03	95.45	1.99773	4.74455	3.16310
		mmzp3g	10833	4857	420	10375	44.84	95.77	1.99465	4.69741	3.14941
		xlxfg512	19338	8632	434	18474	44.64	95.53	1.99220	4.65393	3.15637
		Average							1.98047	4.68242	3.14742
Data set-II	Sub string size 3	MTPACGA	100314	45084	448	95756	44.94	95.46	1.87918	4.57314	3.16310
		MPOMTCG	186608	84100	448	178565	45.07	95.69	1.98324	4.81761	3.16310
		CHNTXX	155844	70478	448	147741	45.22	94.8	1.95680	4.70458	3.16310
		CHMPXX	121024	54528	448	115454	45.06	95.4	1.86621	4.55809	3.16310
		HUMGHCSA	66495	30139	448	63397	45.33	95.34	1.99957	4.70346	3.16310
		HUMHBB	73308	33154	448	70073	45.23	95.59	1.96758	4.41168	3.06042
		HUMHDABCD	58864	26514	448	56269	45.04	95.59	1.99739	4.71494	3.16310
		HUMDYSTROP	38770	17542	434	36932	45.25	95.26	1.94688	4.66074	3.15637
		HUMHPRTB	56737	25577	448	54261	45.08	95.64	1.97046	4.69900	3.16310
		VACCG	191737	86759	427	183250	45.25	95.57	1.91895	4.24001	3.11526
		HEHCMVCG	229354	102146	448	219856	44.54	95.86	1.98509	4.38365	3.10060
		Average							1.95194	4.5879	3.14312
Data set-I	Sub string size 4	atatsgs	9647	3986	952	9110	43.75412	94.4335	1.93269	4.86201	3.26543
		atef1a23	6022	2494	800	5678	43.92392	94.28761	1.97207	4.81496	3.23516
		atrdnaf	10014	4146	1056	9454	43.85445	94.40783	1.99287	4.98121	3.28601
		atrdnai	5287	2224	720	4996	44.51561	94.49593	1.99159	4.68617	3.21656

Data set / Sub string size	Sequence Name	Base pair/ File size	Compress file size	Library file size	Lavenstein Distance	% of encryption	% effect on actual text	Entropy before compression	Entropy after compression	
									Based on compressed file	Based on library file
	celk07e12	58949	23585	1848	55893	42.1967	94.81586	1.96998	4.32815	3.15037
	hsg6pdgen	52173	20616	1824	49670	41.50594	95.2025	1.99773	4.46300	3.20443
	mmzp3g	10833	4509	1000	10262	43.9388	94.72907	1.99465	4.87202	3.27572
	xlxfg512	19338	7626	1248	18399	41.4479	95.14428	1.99220	4.94624	3.22053
	Average							1.98047	4.74422	3.23177
Data set-II Sub string size 4	MTPACGA	100314	40320	2000	94866	42.50206	94.56905	1.87918	4.36590	3.14773
	MPOMTCG	186608	74417	2048	177498	41.92554	95.11811	1.98324	4.19289	3.08053
	CHNTXX	155844	62373	2048	147864	42.18268	94.87949	1.95680	3.94945	3.15131
	CHMPXX	121024	48094	2032	114705	41.92843	94.77872	1.86621	4.05246	3.17413
	HUMGHCSA	66495	26282	1784	63298	41.52106	95.19212	1.99957	4.22875	3.15894
	HUMHBB	73308	29276	1832	69522	42.11041	94.83549	1.96758	4.51482	3.19966
	HUMHDABCD	58864	23241	1896	55970	41.52403	95.08358	1.99739	4.14243	3.16624
	HUMDYSTROP	38770	15585	1696	36742	42.4174	94.76915	1.94688	4.37997	3.17874
	HUMHPRTB	56737	22575	1848	53851	41.92123	94.91337	1.97046	4.20130	3.22676
	VACCG	191737	76353	2048	181952	41.96327	94.89666	1.91895	4.45286	3.17549
	HEHCMVCG	229354	92077	2048	217803	42.27536	94.96368	1.98509	4.12430	3.13644
	Average							1.95194	4.23683	3.16327

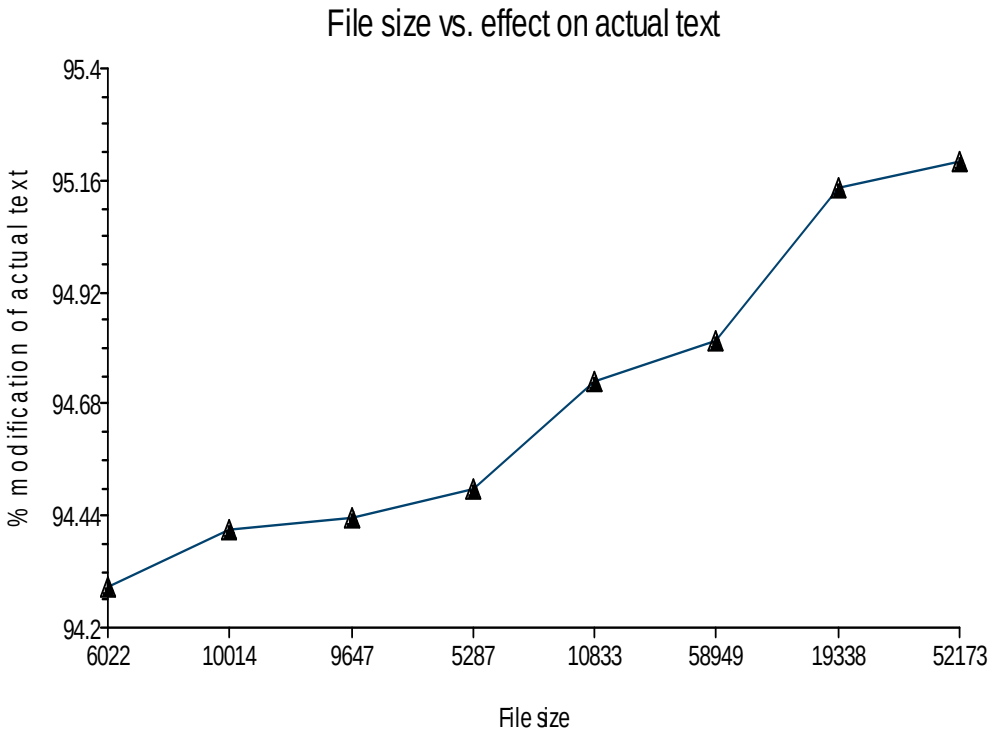

Fig. 4.27 file size vs. % modification for the actual text of data set-I

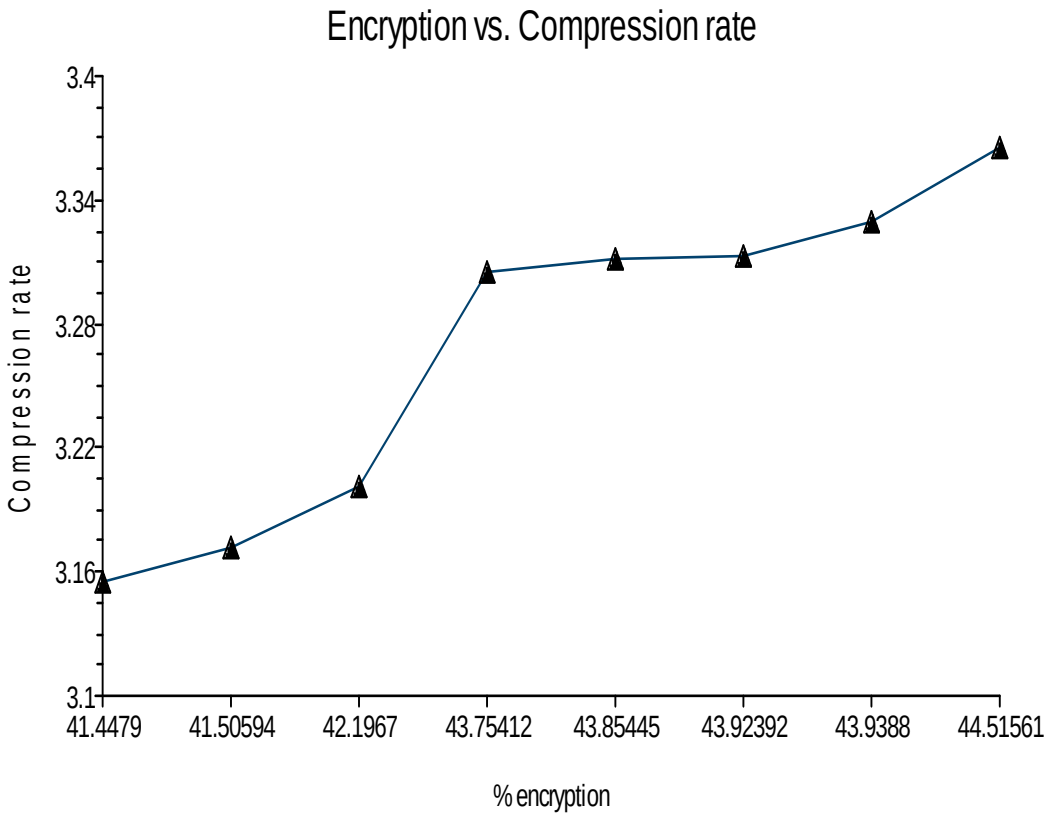

Fig. 4.28 % encryption vs. compression rate of the original file of data set-I

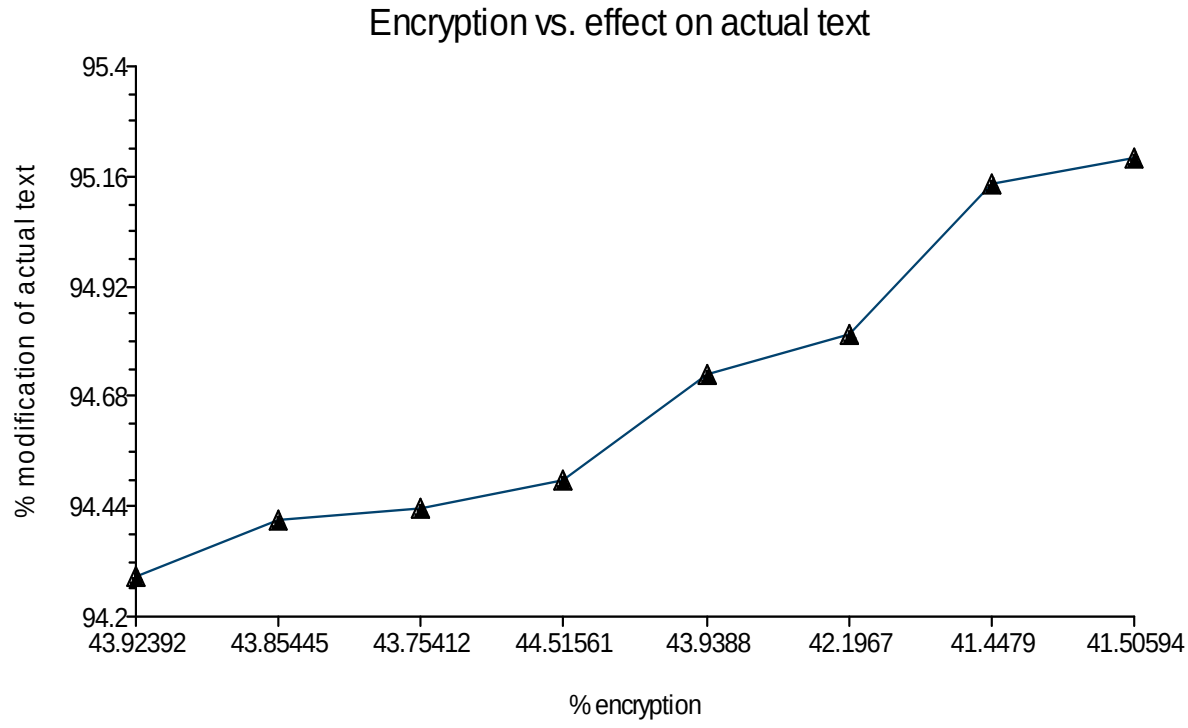

Fig. 4.29 % encryption vs. % modification of the original file of data set-I

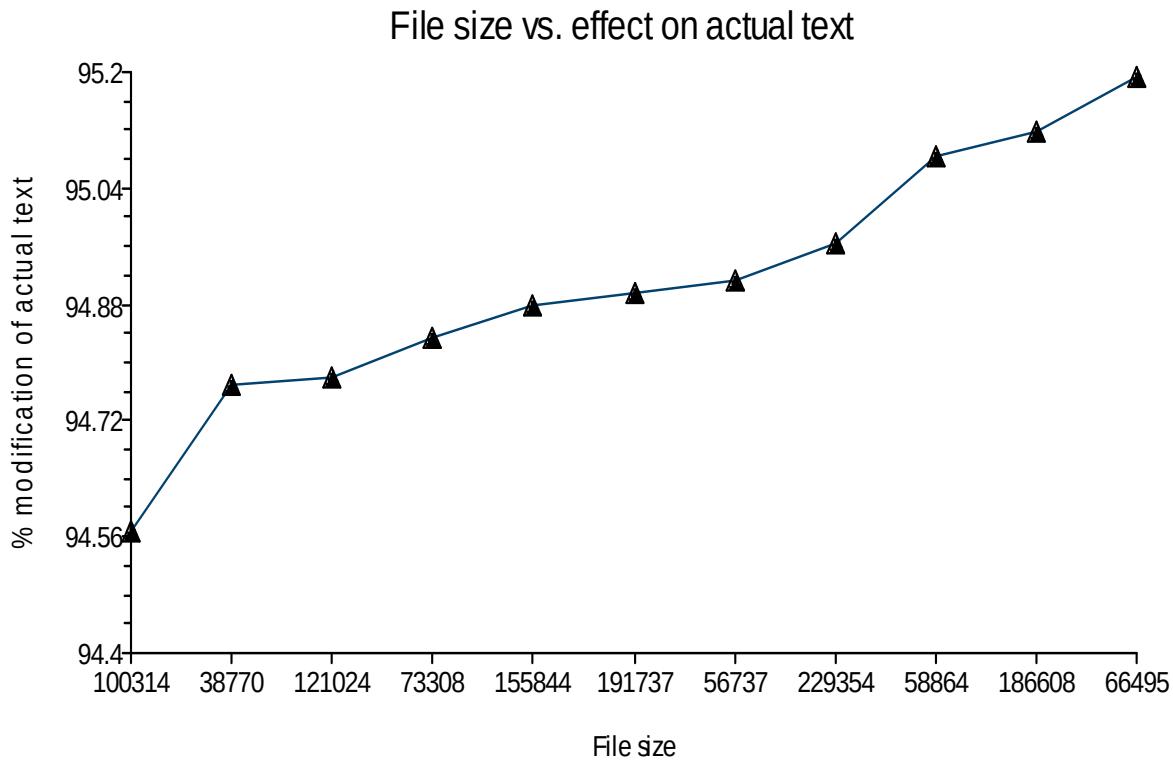

Fig. 4.30 file size vs. % modification for the actual text of data set-II

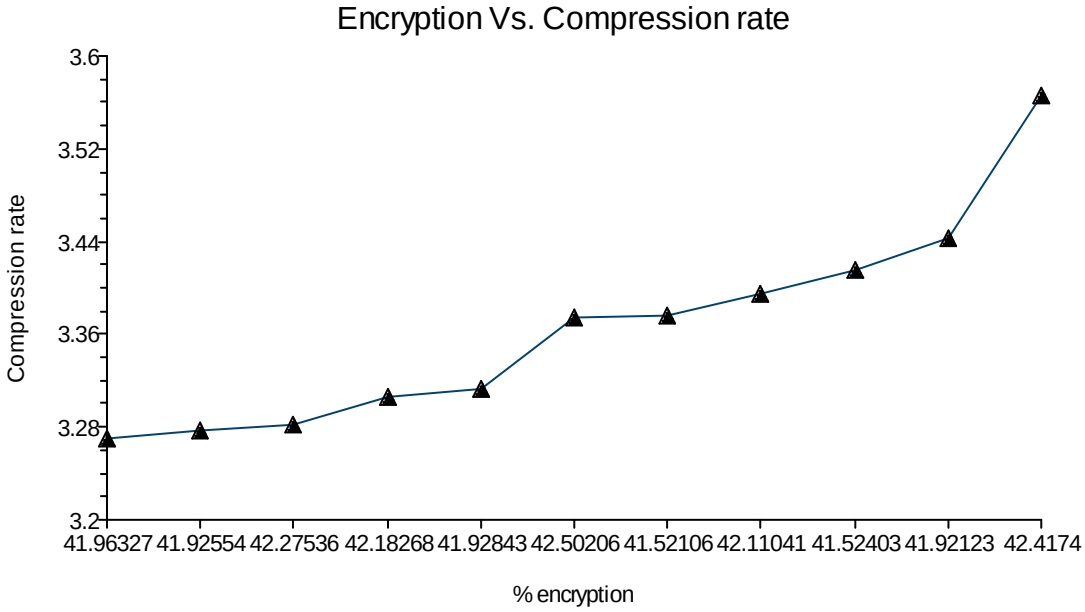

Fig. 4.31 % encryption vs. compression rate of the original file of data set-II

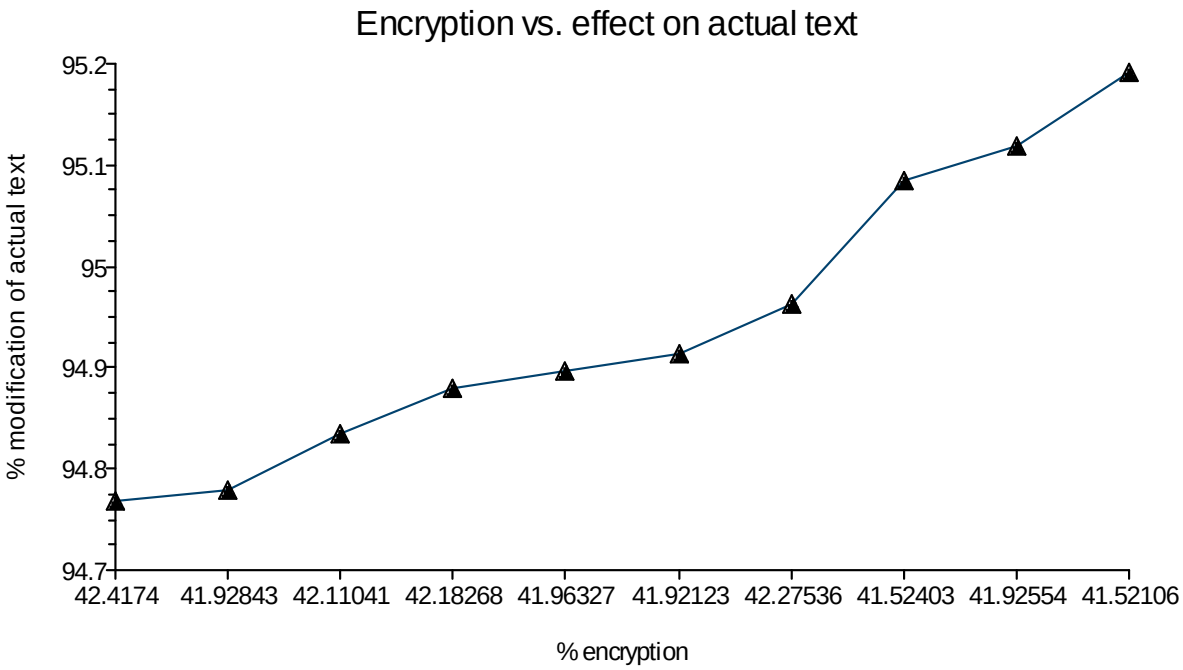

Fig. 4.32 % encryption vs. % modification of the original file of data set-II

Table 4.12 relative frequency

	Using Repeat technique			Using RHUFF			
Sequence Name	File size (byte)	Relative frequency for input	Relative frequency for	File size (byte)	Relative frequency for	level	Relative frequency for output
hehcmvcg	229354	57339	1244	92077	1244	1	233
						2	389
						3	242
						4	252
humdystrop	38770	9693	168	15585	166	1	40
						2	66
						3	41
						4	43
celk07e12	58949	14737	310	23585	310	1	61
						2	100
						3	66
						4	65
atrdnaf	10014	2504	30	4146	30	1	11
						2	17
						3	12
						4	11

Process-I

Table 4.13 encryption, effect on original file & change of entropy using process-I on compressed data of data set-I

Input file size (byte)	Reduce File size	Compress file size	Compression rate	Swapping at Level	Lavenstein Dist.	% of encryption	% effect on actual text	Encryption time(s)	Decryption time(s)	Entropy
atatsgs	3986	3467	2.875091	1	3974	86.98	99.69	15.12	8.56	**6.65007**
				2	3975	86.98	99.72	11.23	6.36	**6.64835**
				3	3974	86.98	99.69	12.36	7.16	**6.64570**
				4	3973	86.98	99.67	12.36	7.76	**6.64284 5567769**
atef1a23	2494	2541	3.375623	1	2527	100	100	13.56	10.76	**5.91886 6618924**
				2	2525	100	100	13.89	7.76	**5.90033 793533**
				3	2525	100	100	13.54	6.86	**5.89341 9038554**
				4	2528	100	100	9.56	7.26	**5.89759 1019287**
atrdnaf	4146	3626	2.896745	1	4133	87.46	99.68	10.25	6.66	**6.71692 1209919**
				2	4136	87.46	99.75	8.97	6.36	**6.73524 4411827**
				3	4136	87.46	99.75	10.12	7.46	**6.72780 1008784**
				4	4136	87.46	99.75	8.97	7.66	**6.72250 8888067**
atrdnai	2224	2343	3.5453	1	2332	100	100	8.45	7.06	**5.61727 4658538**
				2	2329	100	100	9.65	7.46	**5.61676 2665601**
				3	2328	100	100	8.58	6.26	**5.62987 0991289**
				4	2329	100	100	8.89	5.76	**5.62170 4036032**
celk07e12	23585	13859	1.880812	1	23531	58.76	99.77	8.56	5.76	**7.73680 9984371**
				2	23541	58.76	99.81	7.45	5.16	**7.74034 8963193**
				3	23543	58.76	99.82	9.48	5.76	**7.73840 2690865**
				4	23537	58.76	99.79	8.48	6.16	**7.73957 97091**
hsg6pdgen	20616	12586	1.929887	1	20583	61.05	99.83	8.98	6.66	**7.73184 6537191**
				2	20586	61.05	99.85	9.81	8.76	**7.73254 9649416**
				3	20584	61.05	99.84	9.91	8.06	**7.72577 0690086**
				4	20582	61.05	99.83	9.73	6.56	**7.71838 1022823**
mmzp3g	4509	3790	2.798855	1	4495	84.05	99.68	7.87	5.86	**6.77537 0719725**
				2	4491	84.05	99.60	8.84	6.36	**6.78737 486206**
				3	4496	84.05	99.71	7.98	6.26	**6.76832 4708497**
				4	4488	84.05	99.53	9.82	7.16	**6.76393 6617472**
xlxfg512	7626	5780	2.391147	1	7613	75.79	99.82	9.49	6.26	**7.22842**
				2	7607	75.79	99.75	9.37	6.56	**7.22435**
				3	7602	75.79	99.68	9.58	6.36	**7.04620**
				4	7602	75.79	99.68	9.83	5.96	**7.10392**
Average										**6.78584**

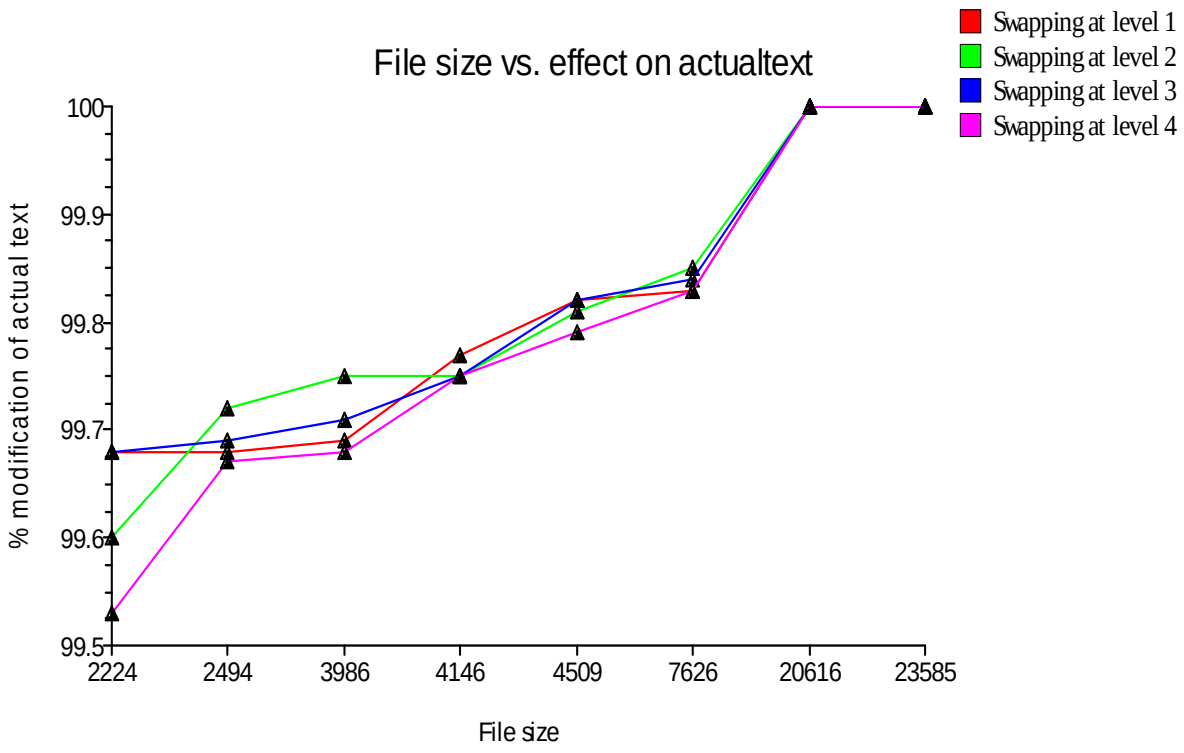

Fig.4.33 file size vs. % modification of the actual text of different level

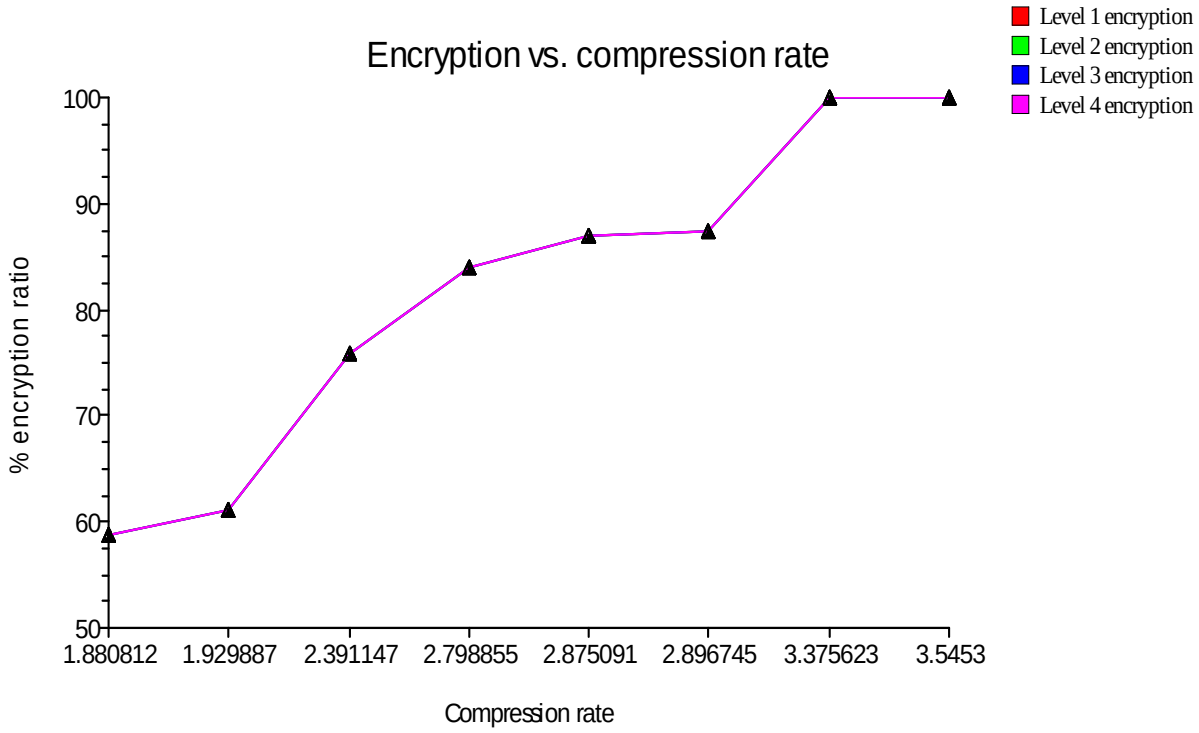

Fig. 4.34 the % encryption vs. compression rate of different level

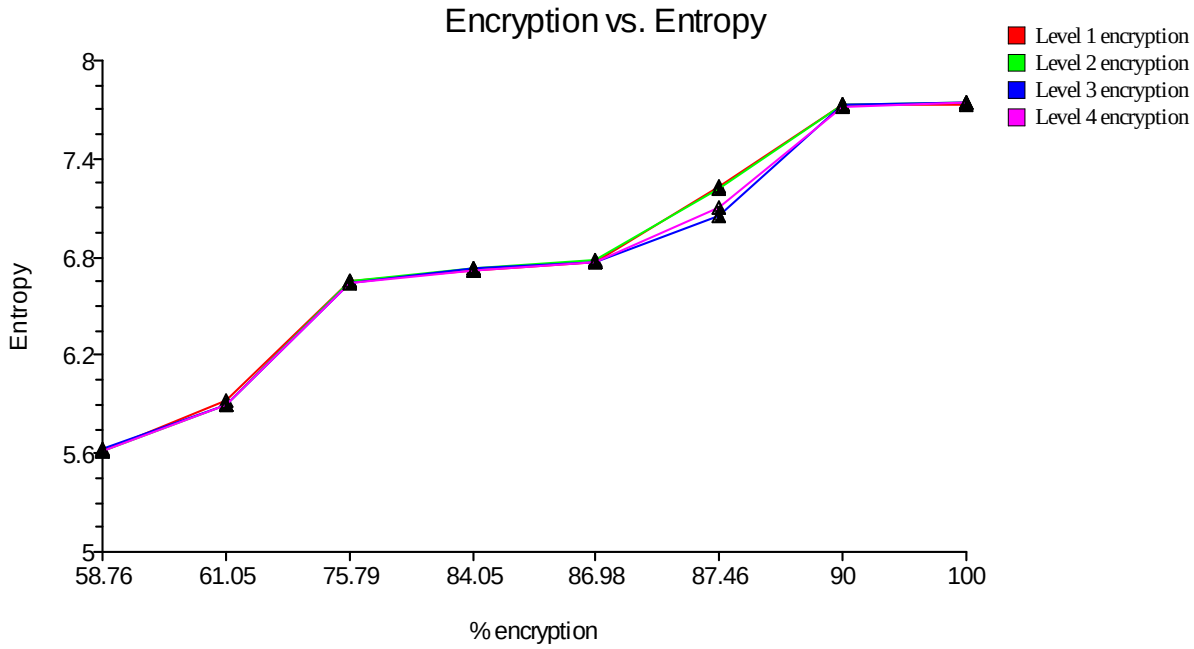

Fig. 4.35 file size vs. entropy of different level

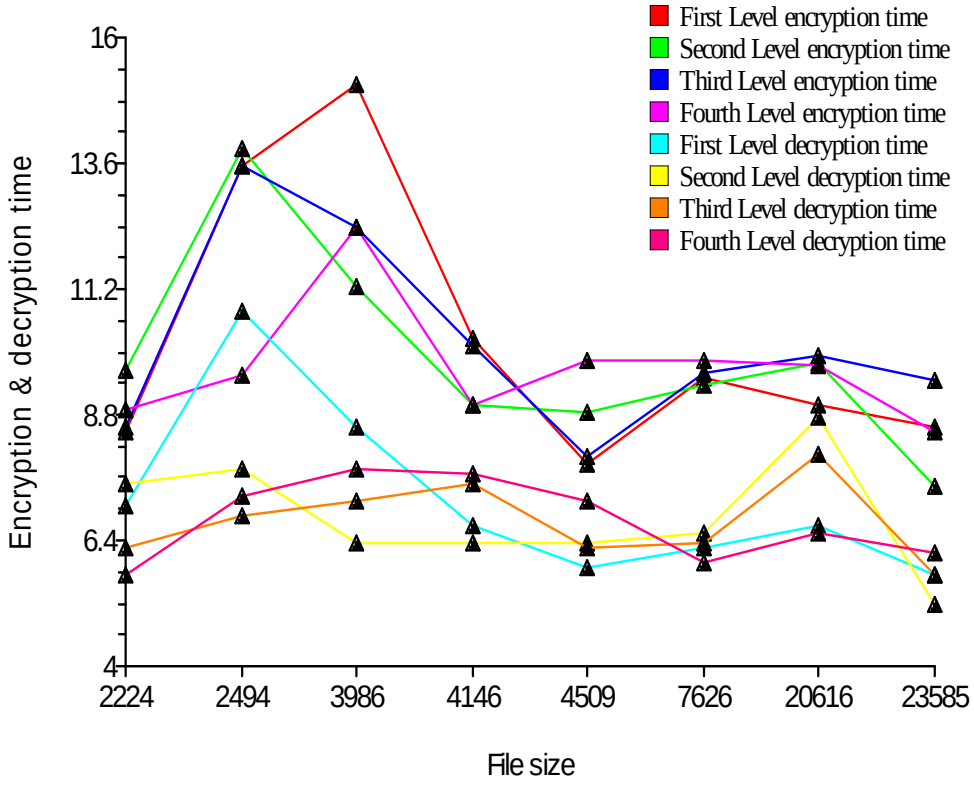

Fig. 4.36 different level time in encryption & decryption vs. file size

Table 4.14 encryption & effect on original file using process-I on compressed data (Data set-II)

Input file size (byte)	File size	Compress file size	Compression rate	Swapping at Level	Lavenstein Dist.	% of encryption	%effect on actual text	Encryption time(s)	Decryption time(s)	Entropy
MTPACGA	40310	23146	1.84588	1	40222	57.42	99.789	16.58	7.05	**7.86283**
				2	40213	57.42	99.75	16.95	6.65	**7.85758**
				3	40225	57.42	99.78	12.63	5.65	**7.86234**
				4	40234	57.42	99.81	15.45	7.45	**7.85525**
MPOMTCG	74406	40227	1.72455	1	74248	54.06	99.78	16.52	11.35	**7.91720**
				2	74271	54.06	99.81	12.32	4.95	**7.91655**
				3	74258	54.06	99.80	12.35	4.15	**7.91743**
				4	74259	54.06	99.80	13.25	7.55	**7.91194**
CHNTXX	62364	31930	1.63907	1	62230	51.2	99.78	15.62	7.25	**7.90273**
				2	62228	51.2	99.78	15.54	8.55	**7.90282**
				3	62241	51.2	99.80	11.56	6.15	**7.90350**
				4	62239	51.2	99.79	15.58	11.35	**7.78208**
CHMPXX	48086	25591	1.69163	1	47993	53.22	99.80	12.59	8.05	**7.87036**
				2	48010	53.22	99.84	11.56	6.75	**7.87361**
				3	48003	53.22	99.82	12.36	6.85	**7.81463**
				4	47998	53.22	99.81	12.58	6.25	**7.86506**
HUMGHCSA	26277	14956	1.79935	1	26226	56.92	99.80	11.48	5.35	**7.78244**
				2	26206	56.92	99.72	11.59	5.15	**7.77352**
				3	26210	56.92	99.74	11.69	4.95	**7.78146**
				4	26213	56.92	99.75	11.54	6.55	**7.77272**
HUMHBB	29270	17619	1.92273	1	29206	60.19	99.78	11.25	5.15	**7.82209**
				2	29210	60.19	99.79	11.20	8.05	**7.81923**
				3	29214	60.19	99.80	11.23	7.95	**7.81775**
				4	29213	60.19	99.80	11.56	5.95	**7.81096**
HUMHDABCD	23237	13114	1.78227	1	23186	56.44	99.78	11.54	6.55	**7.74001**
				2	23188	56.44	99.78	11.25	6.85	**7.74069**
				3	23188	56.44	99.78	11.45	6.05	**7.73122**
				4	23190	56.44	99.79	11.24	6.95	**7.72651**
HUMDYSTROP	15583	9603	1.98153	1	15555	61.62	99.82	11.23	5.85	**7.61497**
				2	15550	61.62	99.78	12.23	6.35	**7.61188**
				3	15549	61.62	99.78	15.56	7.75	**7.61555**
				4	15548	61.62	99.77	15.45	7.15	**7.60853**
HUMHPRTB	22572	12945	1.82526	1	22530	57.35	99.81	12.32	7.05	**7.73941**
				2	22528	57.35	99.80	10.22	6.65	**7.73190**
				3	22525	57.35	99.79	11.23	5.65	**7.73464**
				4	22516	57.35	99.75	12.32	7.45	**7.60177**
VACCG	76341	43773	1.82637	1	76175	57.34	99.78	10.23	11.35	**7.93338**
				2	76176	57.34	99.78	11.25	4.95	**7.92672**
				3	76171	57.34	99.77	11.25	4.15	**7.87645**
				4	76164	57.34	99.76	12.36	7.55	**7.92294**
HEHCMVCG	92062	48672	1.69770	1	91865	52.87	99.78	11.58	7.25	**7.94037**
				2	91870	52.87	99.79	12.57	8.55	**7.93478**
				3	91871	52.87	99.79	13.25	6.15	**7.93329**
				4	91879	52.87	99.80	13.51	11.35	**7.92203**
Average										**7.63749**

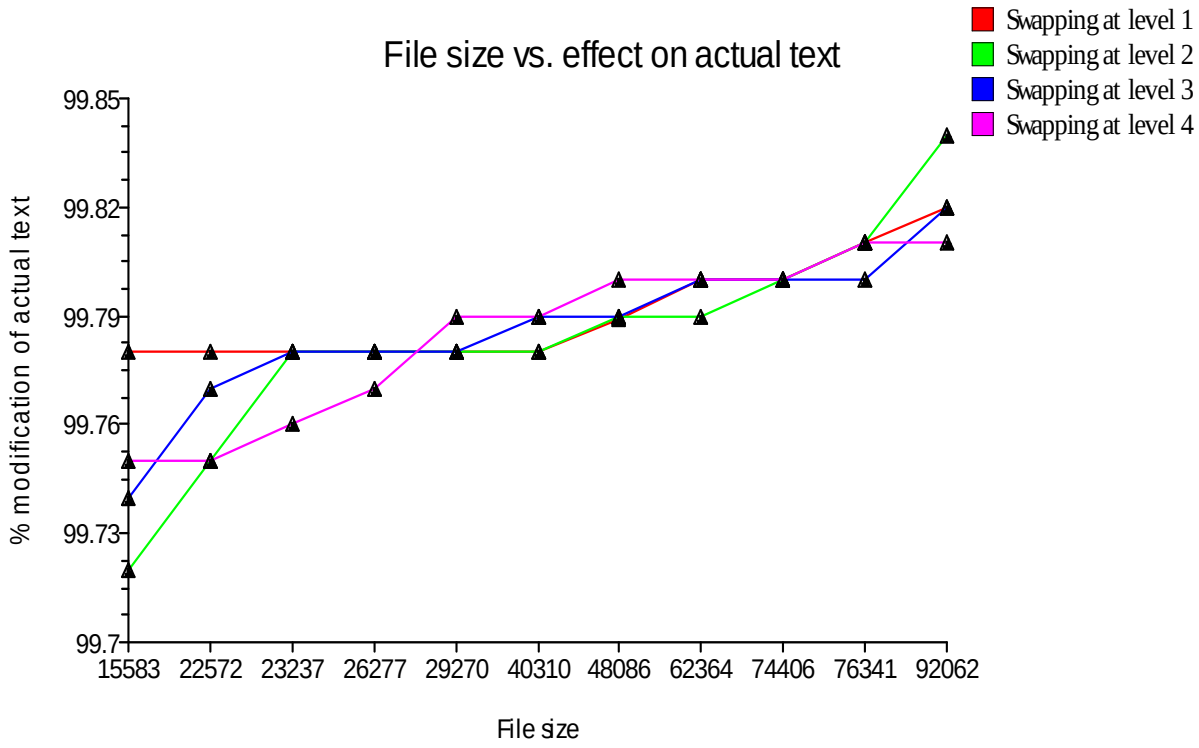

Fig. 4.37 file size vs. % modification of actual file of different level

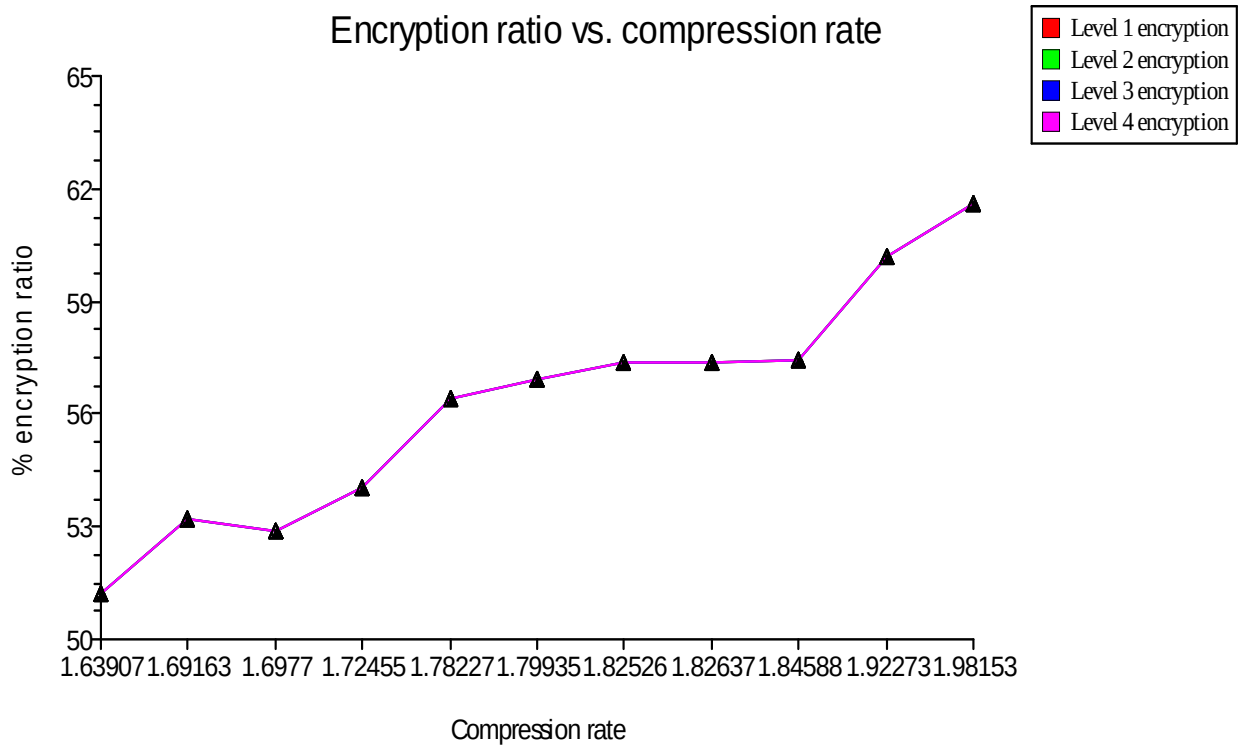

Fig. 4.38 % encryption vs. compression rate of different level

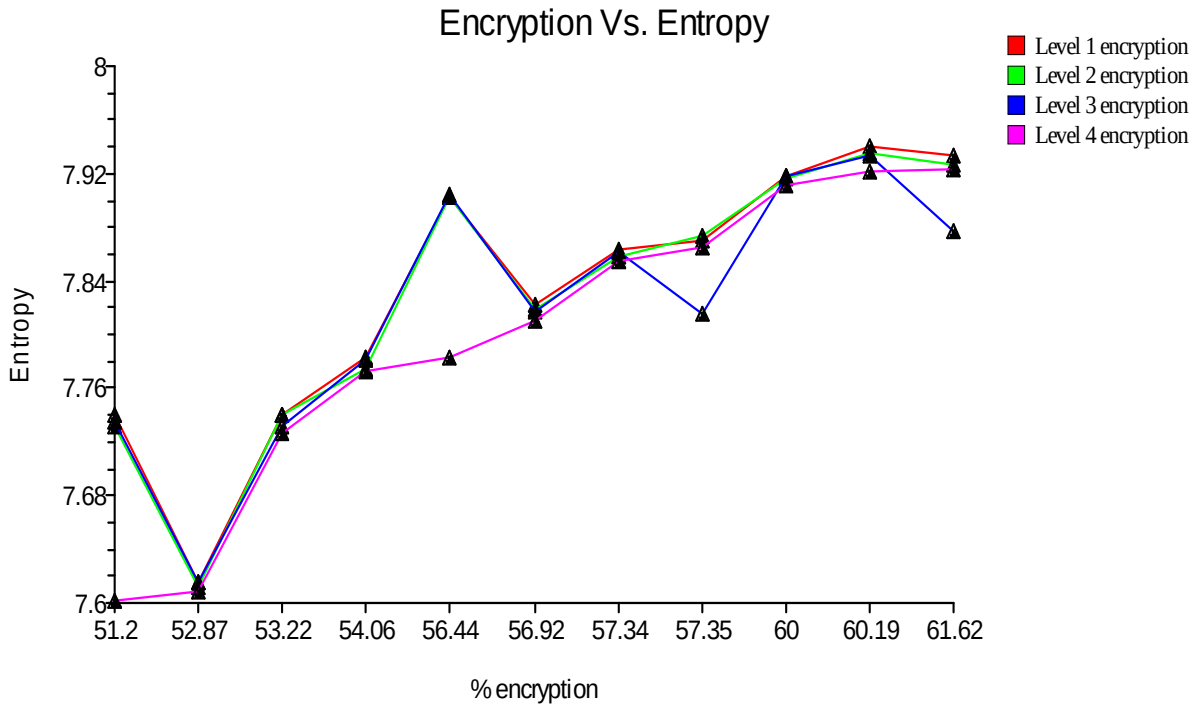

Fig. 4.39 % encryption vs. entropy of different level

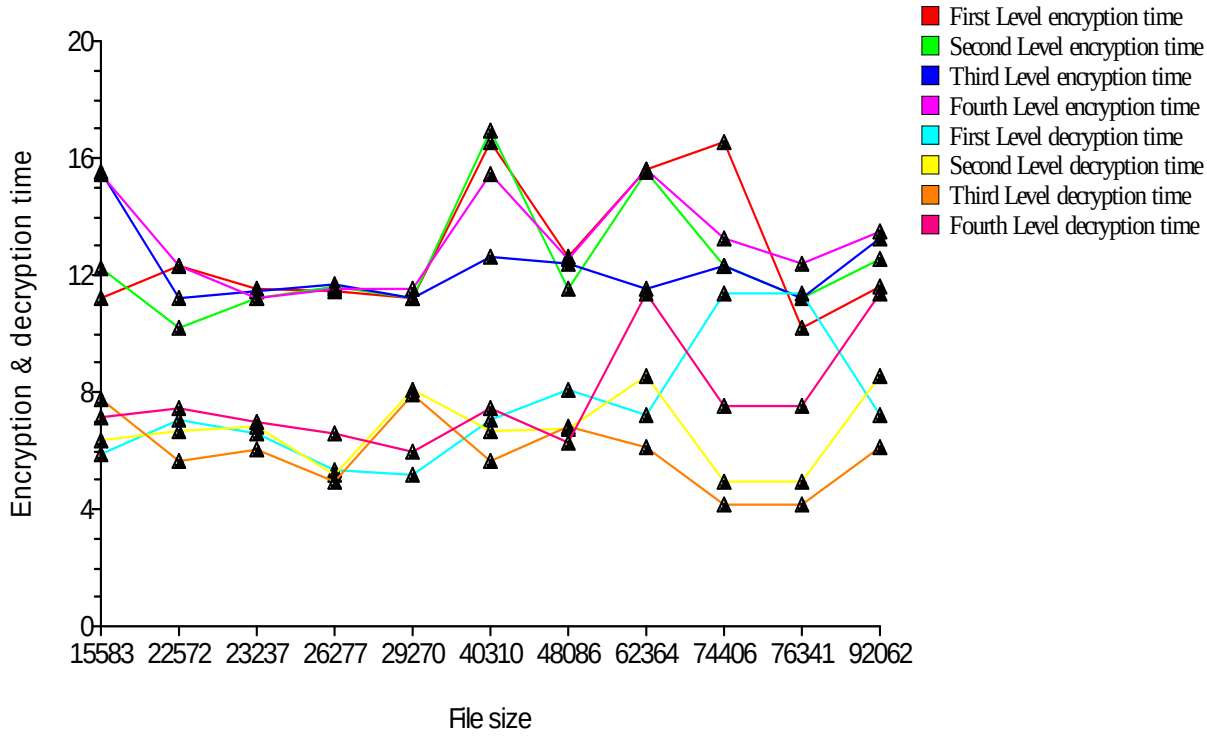

Fig. 4.40 different level time in encryption & decrypt vs. file size

Table 4.15 encryption, effect on original file & entropy using process-I on library file of Data set-II

Input file size (byte)	File size	Compress file size	Compression rate	Swapping at Level	Lavenstein Dist.	% of encryption	%effect on actual text	Encryption time(s)	Decryption time(s)	**Entropy**
MTPACGA	2000	1844	7.376	1	1651	92.2	82.55	11.11	3.72	**6.96988**
		1782	7.128	2	1650	89.1	82.5	13.92	4.66	**6.32727**
		1782	7.128	3	1650	89.1	82.5	15.29	5.11	**7.35867**
		1844	7.376	4	1642	92.2	82.1	16.77	5.61	**6.30857**
MPOMTCG	2048	1865	7.2851563	1	1693	91.06	82.67	12.98	4.34	**6.97240**
		1801	7.0351563	2	1693	87.94	82.67	11.17	3.74	**7.16278**
		1801	7.0351563	3	1690	87.94	82.52	15.94	5.33	**7.27689**
		1865	7.2851563	4	1685	91.06	82.28	11.55	3.86	**7.07412**
CHNTXX	2048	1865	7.2851563	1	1708	91.06	83.4	12.71	4.25	**6.99061**
		1801	7.0351563	2	1707	87.94	83.35	14.3	4.78	**7.16769**
		1801	7.0351563	3	1708	87.94	83.4	12.16	4.07	**7.27371**
		1865	7.2851563	4	1697	91.06	82.86	14.69	4.91	**7.14322**
CHMPXX	2032	1863	7.3346457	1	1693	91.68	83.32	14.91	4.99	**6.95007**
		1800	7.0866142	2	1690	88.58	83.17	15.24	5.1	**7.17534**
		1800	7.0866142	3	1689	88.58	83.12	16.28	5.44	**7.27940**
		1863	7.3346457	4	1681	91.68	82.73	15.68	5.24	**7.24072**
HUMGHCSA	1784	1758	7.8834081	1	1447	98.54	81.11	16.77	5.61	**6.93042**
		1703	7.6367713	2	1443	95.46	80.89	15.24	5.1	**7.08917**
		1702	7.632287	3	1446	95.4	81.05	13.48	4.51	**7.25482**
		1758	7.8834081	4	1434	98.54	80.38	14.36	4.8	**7.11266**
HUMHBB	1832	1785	7.7947598	1	1497	97.43	81.71	12.54	4.19	**7.00373**
		1731	7.558952	2	1499	94.49	81.82	14.19	4.75	**6.80638**
		1731	7.558952	3	1494	94.49	81.55	12.32	4.12	**7.33837**
		1785	7.7947598	4	1489	97.43	81.28	16.55	5.54	**7.13893**
HUMHDABCD	1896	1805	7.6160338	1	1565	95.2	82.54	15.18	5.08	**6.93388**
		1747	7.371308	2	1564	92.14	82.49	14.91	4.99	**7.01623**
		1747	7.371308	3	1564	92.14	82.49	12.93	4.32	**7.29324**
		1805	7.6160338	4	1551	95.2	81.8	14.8	4.95	**7.10867**
HUMDYSTROP	1696	1725	8.1367925	1	1379	100	81.31	11.44	3.83	**6.99463**
		1675	7.9009434	2	1359	98.76	80.13	9.41	3.15	**7.04693**
		1675	7.9009434	3	1358	98.76	80.07	12.21	4.08	**7.27954**
		1725	8.1367925	4	1371	100	80.84	10.29	3.44	**7.17083**
HUMHPRTB	1848	1798	7.7835498	1	1517	97.29	82.09	11.72	3.92	**7.00330**
		1743	7.5454545	2	1522	94.32	82.36	14.03	4.69	**7.22405**
		1743	7.5454545	3	1519	94.32	82.2	14.85	4.97	**7.29018**
		1798	7.7835498	4	1511	97.29	81.76	15.46	5.17	**7.15474**
VACCG	2048	1871	7.3085938	1	1713	91.36	83.64	12.21	4.08	**6.97878**
		1807	7.0585938	2	1704	88.23	83.2	13.31	4.45	**7.12356**
		1807	7.0585938	3	1700	88.23	83.01	17.6	5.89	**7.32420**
		1871	7.3085938	4	1707	91.36	83.35	16.11	5.39	**7.11197**
HEHCMVCG	2048	1860	7.265625	1	1698	90.82	82.91	11.99	4.01	**6.91295**
		1796	7.015625	2	1698	87.7	82.91	11.72	3.92	**7.09627**
		1796	7.015625	3	1699	87.7	82.96	13.2	4.41	**7.22691**
		1860	7.265625	4	1690	90.82	82.52	10.07	3.37	**7.14523**
Average			**7.4290933**							**7.03706**

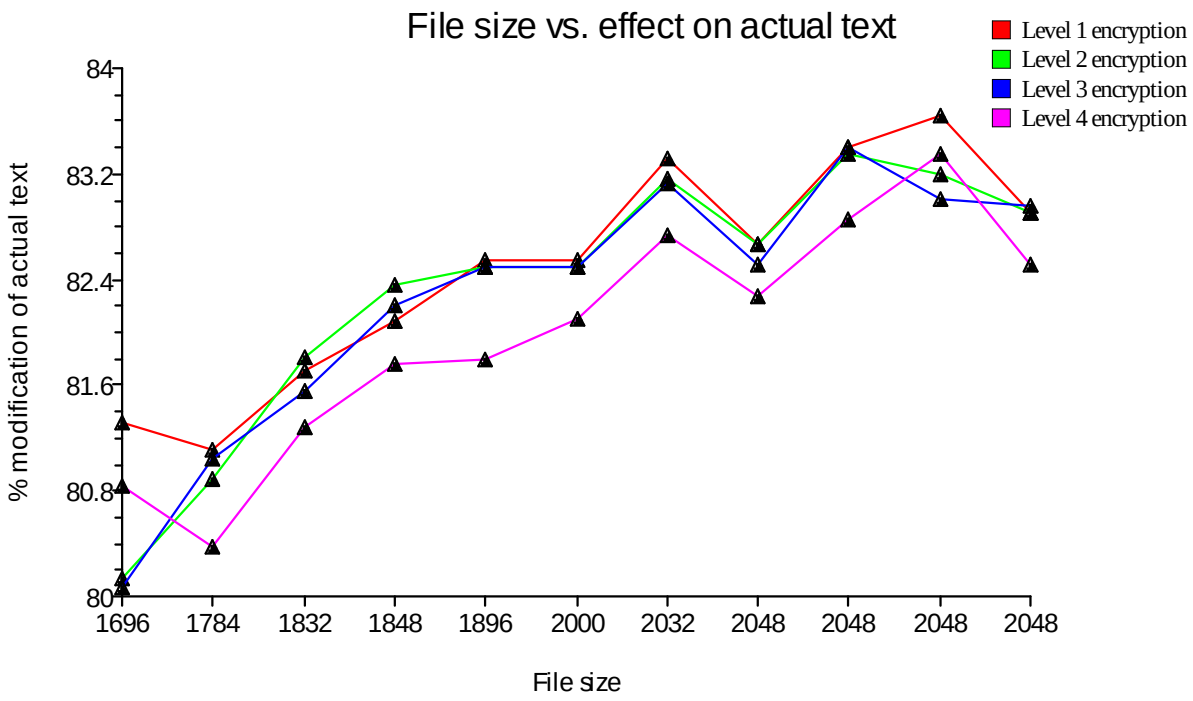

Fig. 4.41 file size vs. % modification of actual file

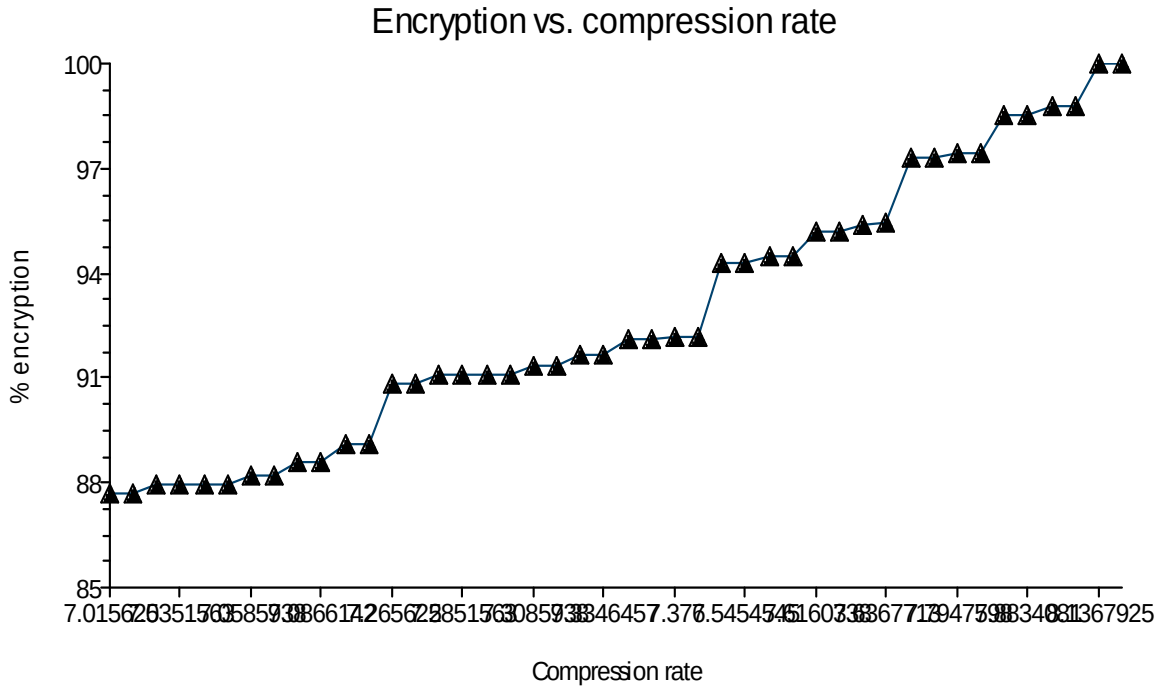

Fig.4.42 % encryption vs. compression rate

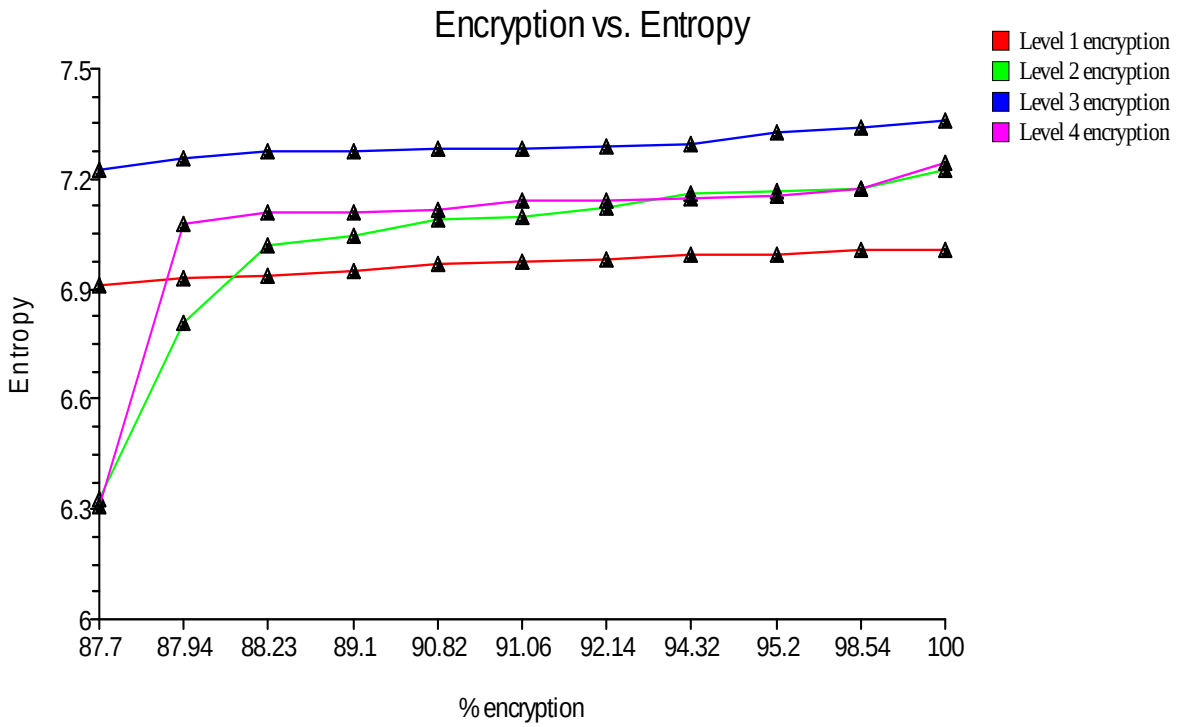

Fig. 4.43 % encryption vs. entropy of different level

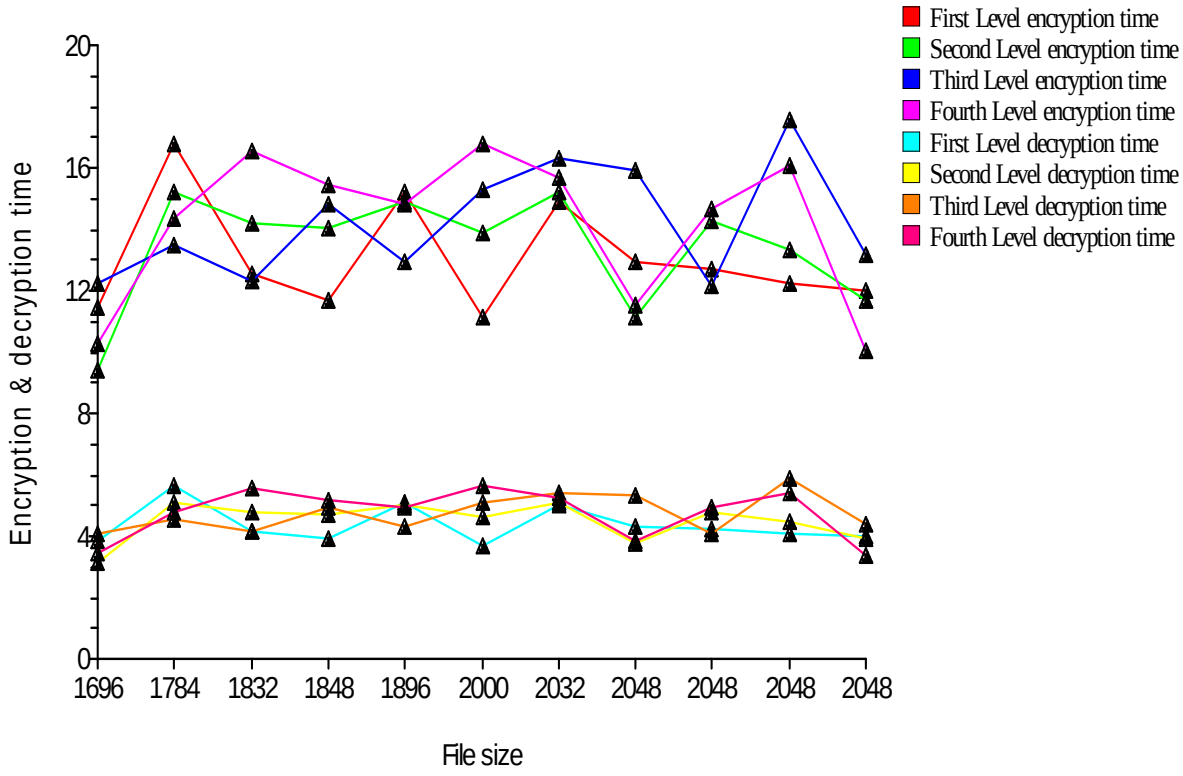

Fig.4.44 different level time in encryption & decryption vs. file size

Process-II

Table 4.16 encryption & effect on actual file using process-II on compressed data of data set-I

Input file size (byte)	Reduce File size	Compressed file size	Compression rate	Specification of two nodes				Lavenstein Dist.	% of encryption	% effect on actual text	Encryption time(s)	Decryption time(s)	Entropy
				1st node		2nd node							
				Level	Binary value	Level	Binary value						
atatsgs	3986	3467	2.87509	3	110	4	0100	3973	87.26	99.676	35.32	18.54	**6.608183421114**
			1	6	111111	4	0010	3973	87.26	99.67	46.81	23.14	**6.61469668338**
				4	0101	5	11111	3976	87.19	99.74	58.68	31.49	**6.612062460094**
				5	00001	6	111111	3973	87.26	99.67	92.91	51.73	**6.607472384914**
atef1a23	2494	2541	3.37562	5	10100	4	0001	2527	100	100	61.04	46.95	**5.868310996918**
			3	2	11	3	000	2529	100	100	35.65	18.83	**5.873629524142**
				3	001	4	1110	2524	100	100	55.76	25.21	**5.870888769054**
				2	00	2	10	2529	100	100	34.39	25.14	**5.871879419878**
atrdnaf	4146	3626	2.89674	4	1000	3	101	4138	87.62	99.80	47.58	27.38	**6.691133432667**
			5	5	00101	3	101	4133	87.73	99.68	82.03	51.78	**6.689809506392**
				2	01	2	10	4135	87.69	99.73	39.34	27.06	**6.684964060344**
				4	0000	4	1111	4137	87.64	99.78	72.80	60.85	**6.691133432667**
atrdnai	2224	2343	3.5453	4	110	4	0100	2329	100	100	38.46	30.32	**5.593507732783**
				5	111111	4	0010	2330	100	100	57.47	41.76	**5.598993063257**
				2	0101	5	11111	2330	100	100	53.13	36.19	**5.583700283306**
				4	00001	6	111111	2329	100	100	62.74	39.94	**5.593507732783**
celk07e12	23585	13859	1.88081	4	10100	4	0001	23531	58.89	99.77	56.09	33.76	**7.696915873256**
			2	5	11	3	000	23533	58.89	99.77	39.72	23.01	**7.689425459585**
				2	001	4	1110	23535	58.88	99.78	52.19	28.39	**7.706443312471**
				4	00	2	10	23531	58.89	99.77	34.61	23.91	**7.69720460198**
hsg6pdgen	20616	12586	1.92988	4	1000	3	101	20590	61.12	99.87	53.57	34.29	**7.683355536842**
			7	5	00101	3	101	20582	61.15	99.83	57.14	52.33	**7.675907854499**
				2	01	2	10	20569	61.18	99.77	34.12	29.1	**7.700569541568**
				4	0000	4	1111	20590	61.12	99.87	32.08	20.93	**7.683681551341**
mmzp3g	4509	3790	2.79885	4	110	4	0100	4498	84.25	99.75	45.82	31.38	**6.74476642548**
			5	5	111111	4	0010	4495	84.31	99.68	34.06	25.11	**6.737049392127**
				2	0101	5	11111	4498	84.25	99.75	34.45	22.41	**6.725983949165**
				4	00001	6	111111	4498	84.25	99.75	50.43	36.81	**6.744766425483**
xlxfg512	7626	5780	2.39114	4	10100	4	0001	7606	75.99	99.73	32.80	19.17	**7.164317210158**
			7	5	11	3	000	7607	75.98	99.75	25.43	16.77	**7.164317210158**
				2	001	4	1110	7606	75.99	99.73	24.83	15.52	**7.173765697193**
				4	00	2	10	7607	75.98	99.75	17.25	10.52	**7.164317210158**
Average													

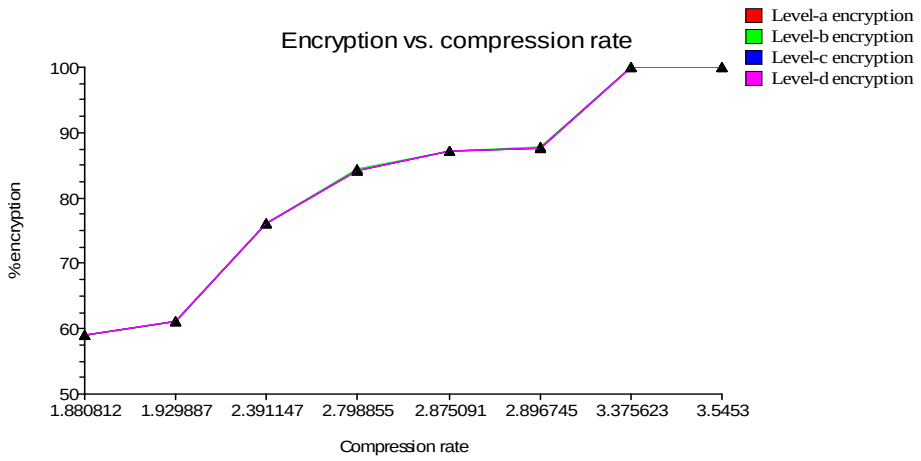

Fig. 4.45 % encryption vs. compression rate of different level

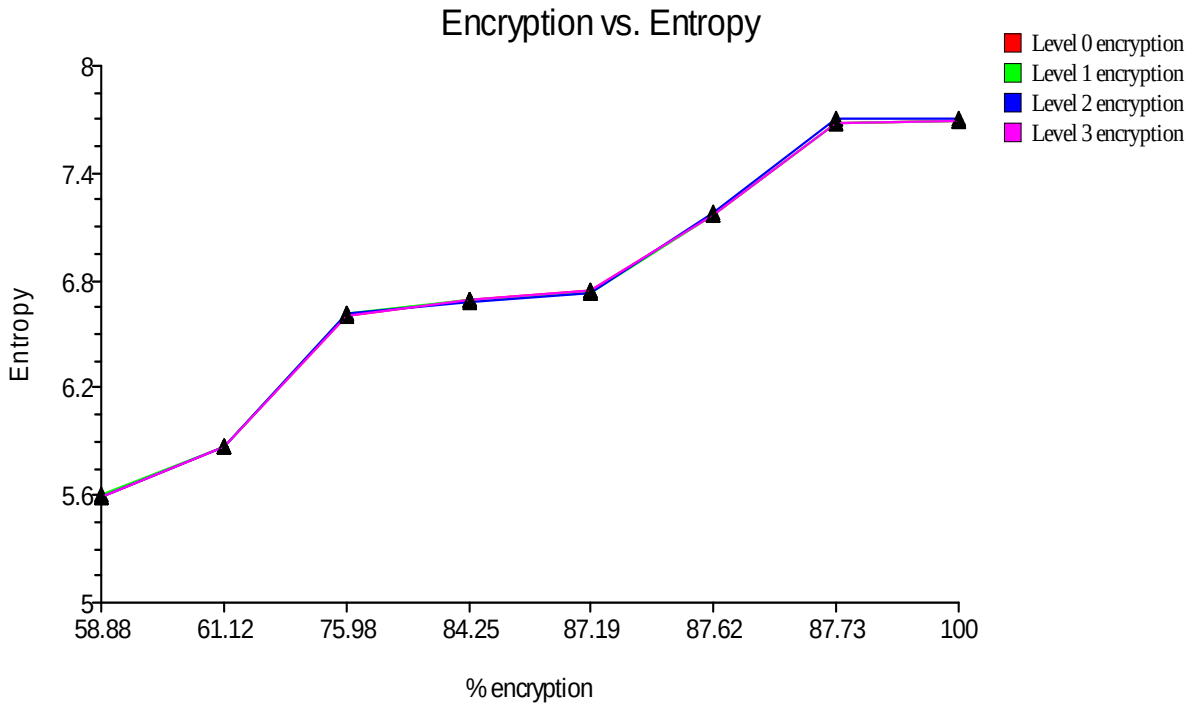

Fig.4.46 % encryption vs. Entropy of different level

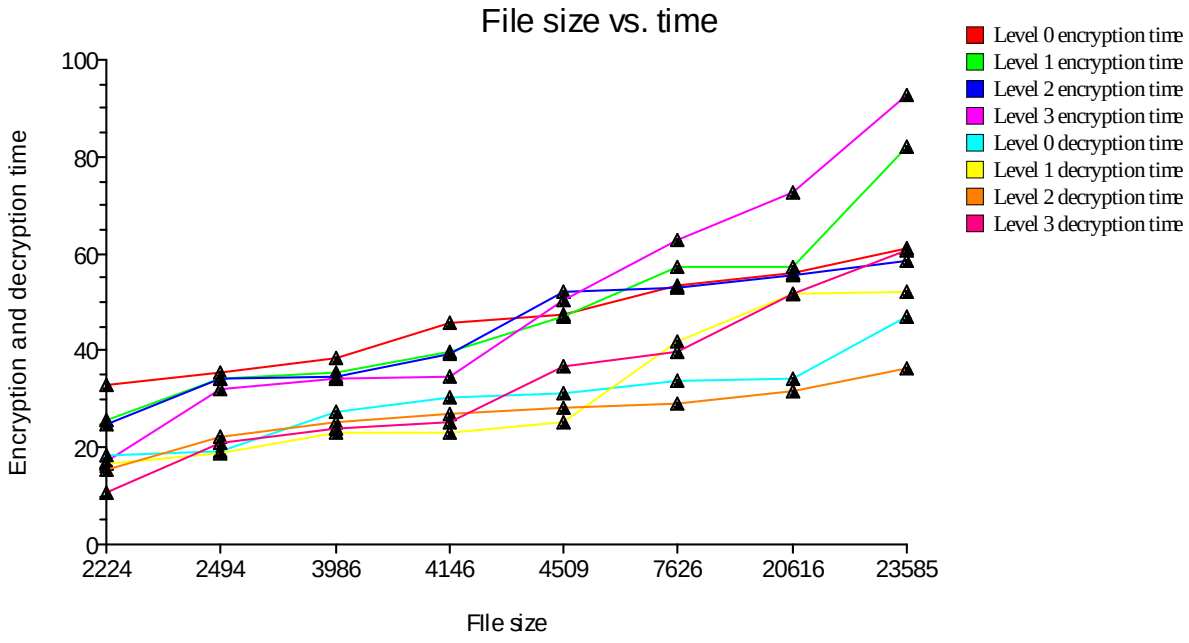

Fig.4.47 different level time in encryption & decryption vs. file size

Table 4.17 encryption & effect on original file using process-II on compressed data of data set-II

Input file size (byte)	File size	Compressed file size	Compression rate	Specification of two nodes				Lavenstein Dist.	% of encryption	% effect on actual	Encryption time(s)	Decryption time(s)	**Entropy**
				1st node		2nd node							
				Level	Binary value	Level	Binary value						
MTPACGA	40310	23146	1.8458	3	110	4	0100	40209	57.56	99.7	28.29	12.09	**7.82793817999**
				6	11111	4	0010	40219	57.54	99.7	29.28	11.43	**7.82849700268**
				4	0101	5	11111	40215	57.55	99.7	32.69	14.59	**7.82941603585**
				5	00001	6	11111	40218	57.55	99.7	34.56	15.97	**7.82743927986**
MPOMTCG	74406	40227	1.7245	5	10100	4	0001	74244	54.18	99.7	27.03	17.69	**7.88059666335**
				2	11	3	000	74258	54.17	99.8	29.89	12.39	**7.89353025059**
				3	001	4	1110	74243	54.18	99.7	21.86	7.511	**7.88214538177**
				2	00	2	10	74258	54.17	99.8	28.02	17.02	**7.89353025059**
CHNTXX	62364	31930	1.6390	4	1000	3	101	62239	51.30	99.7	29.39	13.65	**7.86678599809**
			7	5	00101	3	101	62239	51.30	99.7	23.84	12.08	**7.86668550244**
				2	01	2	10	62225	51.31	99.7	26.48	14.76	**7.87563786506**
				4	0000	4	1111	62239	51.30	99.7	23.57	17.22	**7.86668550244**
CHMPXX	48086	25591	1.6916	4	110	4	0100	47986	53.33	99.7	26.37	17.18	**7.83573258228**
			3	5	11111	4	0010	47995	53.32	99.8	24.72	14.66	**7.83750794570**
				2	0101	5	11111	47986	53.33	99.7	27.63	15.52	**7.84221015337**
				4	00001	6	11111	48000	53.31	99.8	28.57	13.81	**7.83586576502**
HUMGHCSA	26277	14956	1.7993	4	10100	4	0001	26215	57.05	99.7	28.68	13.54	**7.74437827786**
			5	5	11	3	000	26209	57.06	99.7	23.57	10.43	**7.74426244034**
				2	001	4	1110	26218	57.04	99.7	22.74	9.34	**7.75598531578**
				4	00	2	10	26215	57.05	99.7	20.98	11.73	**7.74437827786**
HUMHBB	29270	17619	1.9227	4	1000	3	101	29214	60.31	99.8	28.73	15.14	**7.77797870628**
			3	5	00101	3	101	29206	60.32	99.7	24.28	18.69	**7.77197111723**
				2	01	2	10	29213	60.31	99.8	19.94	15.03	**7.79473625365**
				4	0000	4	1111	29214	60.31	99.8	24.17	12.93	**7.77797870628**
HUMHDABCD	23237	13114	1.7822	4	110	4	0100	23186	56.55	99.7	20.65	11.54	**7.70049966164**
			7	5	11111	4	0010	23184	56.56	99.7	22.91	13.38	**7.69533528154**
				2	0101	5	11111	23197	56.53	99.8	22.08	10.99	**7.70579555472**
				4	00001	6	11111	23186	56.55	99.7	22.08	12.78	**7.70049966164**
HUMDYSTROP	15583	9603	1.9815	4	10100	4	0001	15551	61.75	99.7	22.74	10.51	**7.57660940702**
			3	5	11	3	000	15552	61.74	99.8	22.03	10.95	**7.5676188**
				2	001	4	1110	15557	61.72	99.8	20.00	10.21	**7.59089667731**
				4	00	2	10	15551	61.75	99.7	16.70	6.561	**7.57705902972**
HUMHPRTB	22572	12945	1.8252	3	110	4	0100	22532	57.45	99.8	20.38	12.09	**7.70511622952**
			6	6	111111	4	0010	22516	57.49	99.7	22.80	11.43	**7.6866464483**
				4	0101	5	11111	22516	57.49	99.7	19.17	14.59	**7.68633108306**
				5	00001	6	111111	22516	57.49	99.7	30.98	15.97	**7.68633108306**
VACCG	76341	43773	1.8263	4	110	4	0100	76176	57.46	99.7	21.81	17.69	**7.89009762673**
			7	5	111111	4	0010	76188	57.45	99.7	25.54	12.39	**7.88896161220**
				2	0101	5	11111	76188	57.45	99.7	18.73	7.511	**7.90507019722**
				4	00001	6	111111	76173	57.46	99.7	22.74	17.02	**7.89016464738**
HEHCMVCG	92062	48672	1.6977	4	10100	4	0001	91872	52.97	99.7	22.30	13.65	**7.90230983696**
			0	5	11	3	000	91878	52.97	99.8	18.84	12.08	**7.90172678239**
				2	001	4	1110	91866	52.98	99.7	19.89	14.76	**7.91774869399**
				4	00	2	10	91872	52.97	99.7	22.30	17.22	**7.90230983696**
Average													

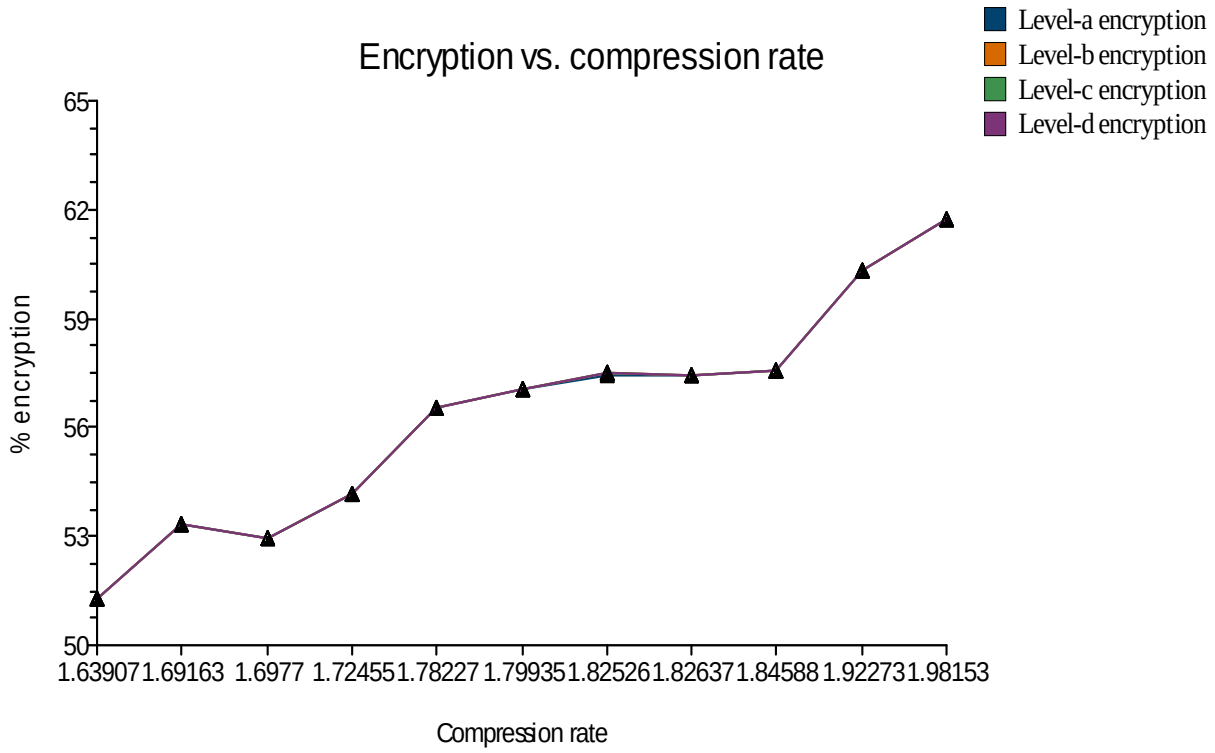

Fig.4.48 the % encryption vs. compression rate of different level

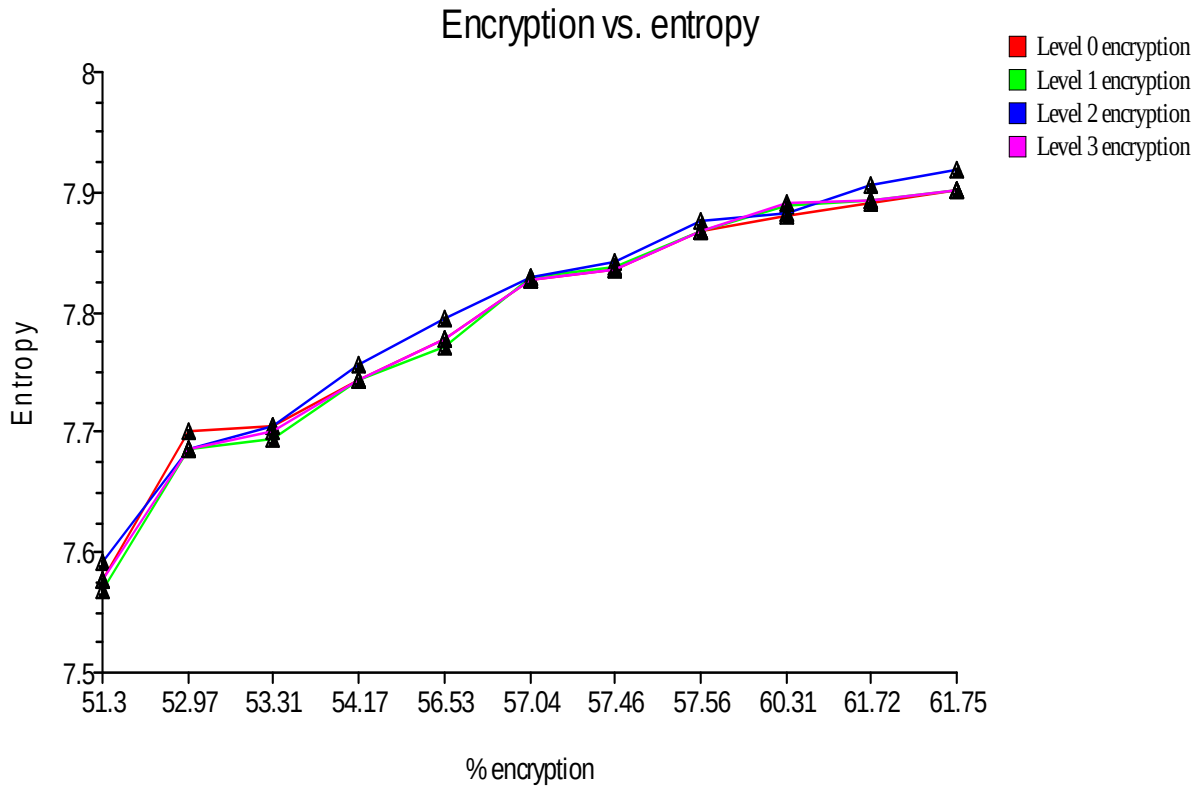

Fig.4.49 entropy vs. encryption of different level

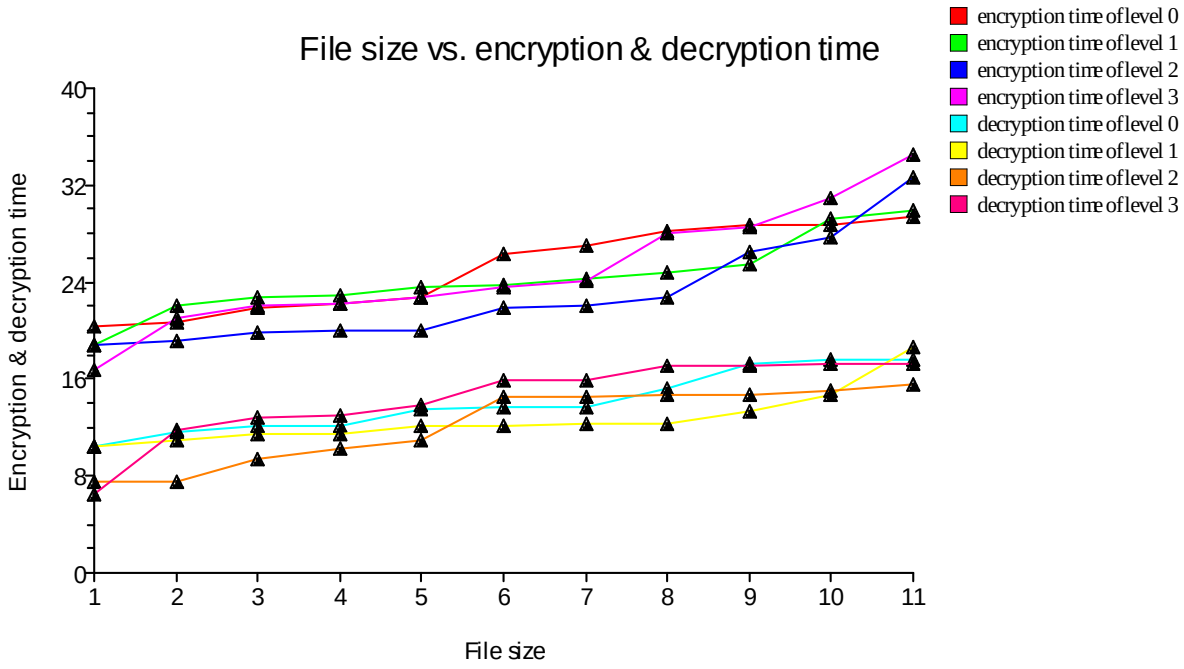

Fig. 4.50 different level time in encryption & decryption vs. file size

Table 4.18 encryption & effect on actual file using process-II on library file of data set-II

Input file size (byte)	File size	Compressed file size	Compression rate	Specification of two nodes				Lavenstein Dist.	% of encryption	% effect on actual text	Encryption time(s)	Decryption time(s)	Entropy
				1st node		2nd node							
				Level	Binary value	Level	Binary value						
MTPACGA	2000	1844	7.376	3	110	4	0100	1643	92.2	82.15	11.99	3.28	**7.097027579252**
				6	111111	4	0010	1643	92.2	82.15	14.8	4.05	**7.140484473138**
				4	0101	5	11111	1643	92.2	82.15	16.17	4.43	**7.140484473138**
				5	00001	6	111111	1643	92.2	82.15	17.65	4.84	**7.140484473138**
MPOMTCG	2048	1846	7.21094	5	10100	4	0001	1685	90.14	82.28	13.86	3.8	**7.077049166807**
				2	11	3	000	1686	90.14	82.32	12.05	3.3	**7.202799278834**
				3	001	4	1110	1680	90.14	82.03	16.82	4.61	**7.053848939736**
				2	00	2	10	1686	90.14	82.32	12.43	3.41	**7.202799278834**
CHNTXX	2048	1865	7.28516	4	1000	3	101	1697	91.06	82.86	13.59	3.72	**7.143221779508**
				5	00101	3	101	1697	91.06	82.86	15.18	4.16	**7.143221779508**
				2	01	2	10	1700	91.06	83.01	13.04	3.57	**7.30962291861**
				4	0000	4	1111	1697	91.06	82.86	15.57	4.27	**7.143221779508**
CHMPXX	2032	1863	7.33465	4	110	4	0100	1681	91.68	82.73	15.79	4.33	**7.240728456558**
				5	111111	4	0010	1681	91.68	82.73	16.12	4.42	**7.240728456558**
				2	0101	5	11111	1687	91.68	83.02	17.16	4.7	**7.323052703547**
				4	00001	6	111111	1681	91.68	82.73	16.56	4.54	**7.240728456558**
HUMGHCSA	1784	1758	7.88341	4	10100	4	0001	1434	98.54	80.38	17.65	4.84	**7.112662043453**
				5	11	3	000	1434	98.54	80.38	16.12	4.42	**7.112662043453**
				2	001	4	1110	1445	98.54	81	14.36	3.93	**7.240402524063**
				4	00	2	10	1434	98.54	80.38	15.24	4.18	**7.112662043453**
HUMHBB	1832	1785	7.79476	4	1000	3	101	1489	97.43	81.28	13.42	3.68	**7.141766785021**
				5	00101	3	101	1489	97.43	81.28	15.07	4.13	**7.141766785021**
				2	01	2	10	1492	97.43	81.44	13.2	3.62	**7.278142644736**
				4	0000	4	1111	1489	97.43	81.28	17.43	4.78	**7.141766785021**
HUMHDABCD	1896	1805	7.61603	4	110	4	0100	1553	95.2	81.91	16.06	4.4	**7.108676278135**
				5	111111	4	0010	1553	95.2	81.91	15.79	4.33	**7.108676278135**
				2	0101	5	11111	1560	95.2	82.28	13.81	3.78	**7.30774270527**
				4	00001	6	111111	1553	95.2	81.91	15.68	4.3	**7.108676278135**

Input file size (byte)	File size	Compressed file size	Compression rate	Specification of two nodes				Lavenstein Dist.	% of encryption	% effect on actual text	Encryption time(s)	Decryption time(s)	Entropy
				1st node		2nd node							
				Level	Binary value	Level	Binary value						
HUMDYSTROP	1696	1725	8.13679	4	10100	4	0001	1371	101.7	80.84	12.32	3.38	**7.165351185594**
				5	11	3	000	1371	101.7	80.84	10.29	2.82	**7.16535118559**
				2	001	4	1110	1374	101.7	81.01	13.09	3.59	**7.273265889495**
				4	00	2	10	1371	101.7	80.84	11.17	3.06	**7.170836221231**
HUMHPRTB	1848	1798	7.78355	3	110	4	0100	1571	97.29	85.01	12.6	3.45	**7.131234153214**
				6	111111	4	0010	1516	97.29	82.03	14.91	4.08	**7.153926544916**
				4	0101	5	11111	1516	97.29	82.03	15.73	4.31	**7.153926544916**
				5	00001	6	111111	1516	97.29	82.03	16.34	4.48	**7.153926544916**
VACCG	2048	1871	7.30859	4	110	4	0100	1699	91.36	82.96	13.09	3.59	**7.114507029907**
				5	111111	4	0010	1699	91.36	82.96	14.19	3.89	**7.114507029907**
				2	0101	5	11111	1701	91.36	83.06	18.48	5.06	**7.284656725915**
				4	00001	6	111111	1699	91.36	82.96	16.99	4.65	**7.114507029907**
HEHCMVCG	2048	1860	7.26563	4	10100	4	0001	1690	90.82	82.52	12.87	3.53	**7.145239955006**
				5	11	3	000	1690	90.82	82.52	12.6	3.45	**7.145239955006**
				2	001	4	1110	1691	90.82	82.57	14.08	3.86	**7.311276991984**
				4	**00**	**2**	**10**	**1690**	**90.82**	**82.52**	**10.95**	**3**	**7.145239955006**

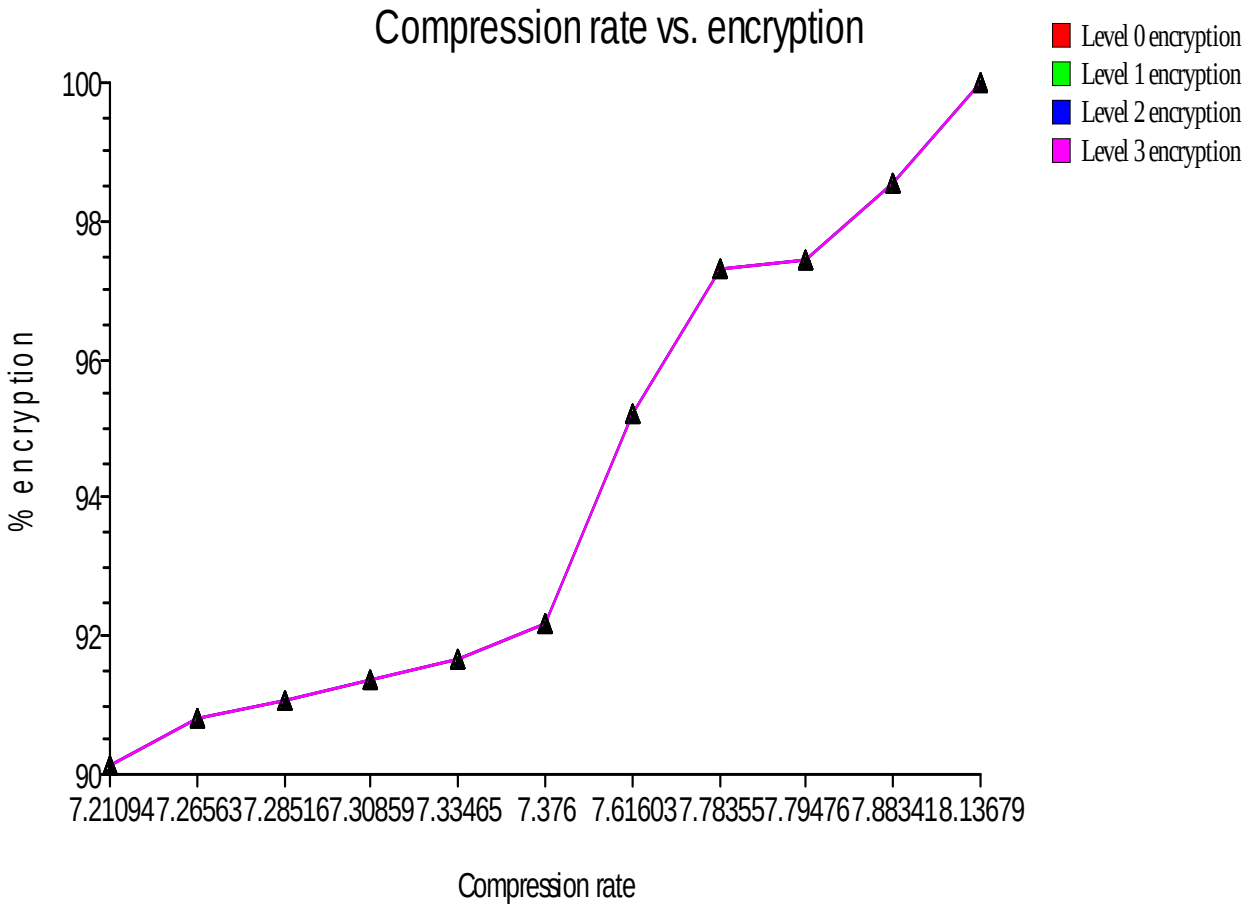

Fig. 4.51 % encryption vs. compression rate of different level

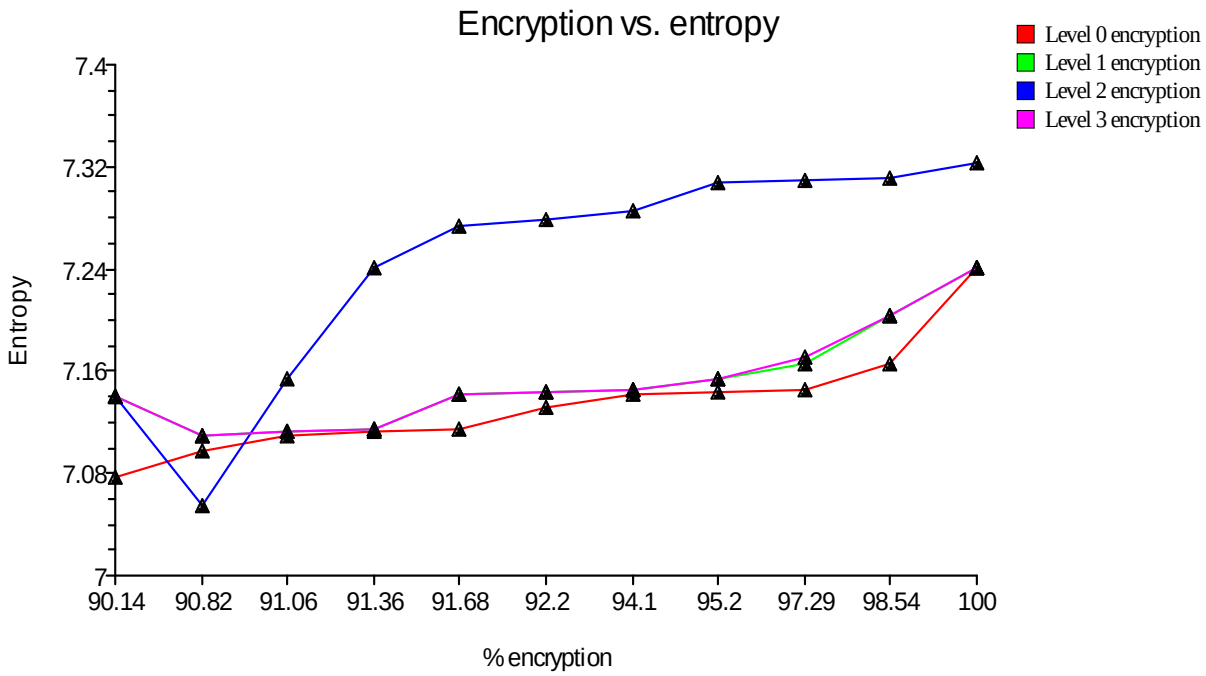

Fig.4.52 % encryption vs. entropy of different level

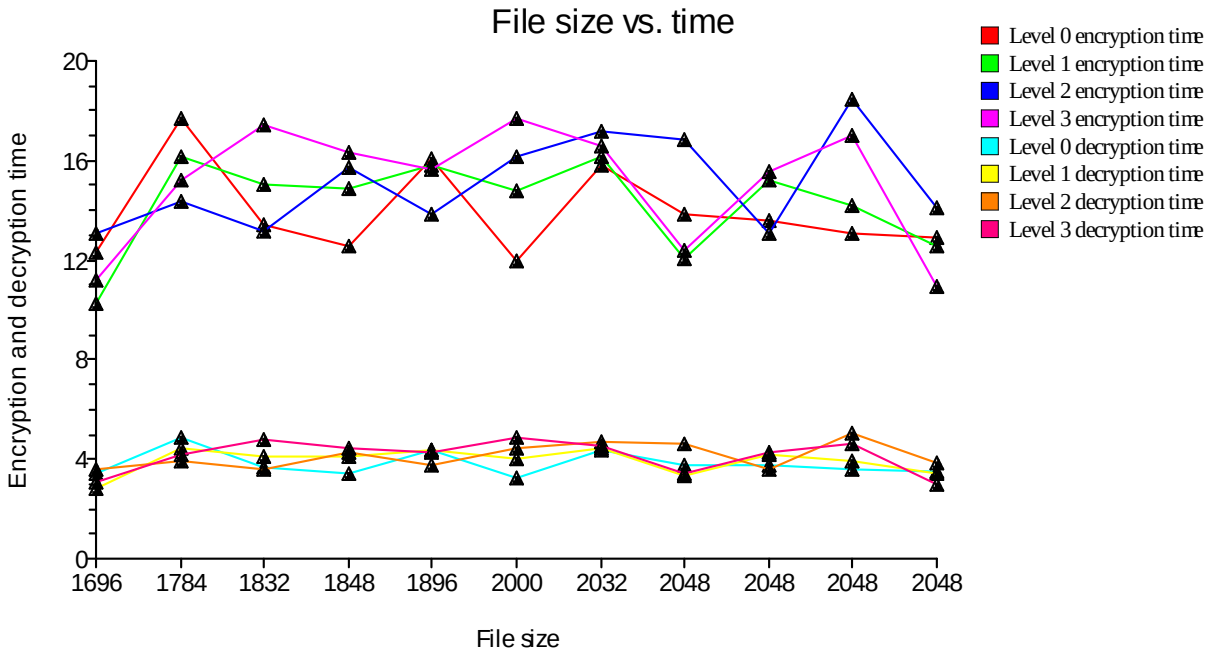

Fig. 4.53 different level time in encryption & decryption vs. file size

Process-III

Table 4.19 encryption & effect on original file using process-III

Level	Size	% of encryption	%of effect on actual text	L_{sid}
8	2681byte	52.11	86.34	**2315**
7		56.34	87.50	**2346**
6		59.77	91.23	**2446**
5		61.56	92.72	**2486**
8	4382byte	49.48	82.26	**3605**
7		53.68	86.17	**3776**
6		59.79	90.55	**3968**
5		**64.44**	**93.47**	**4095**

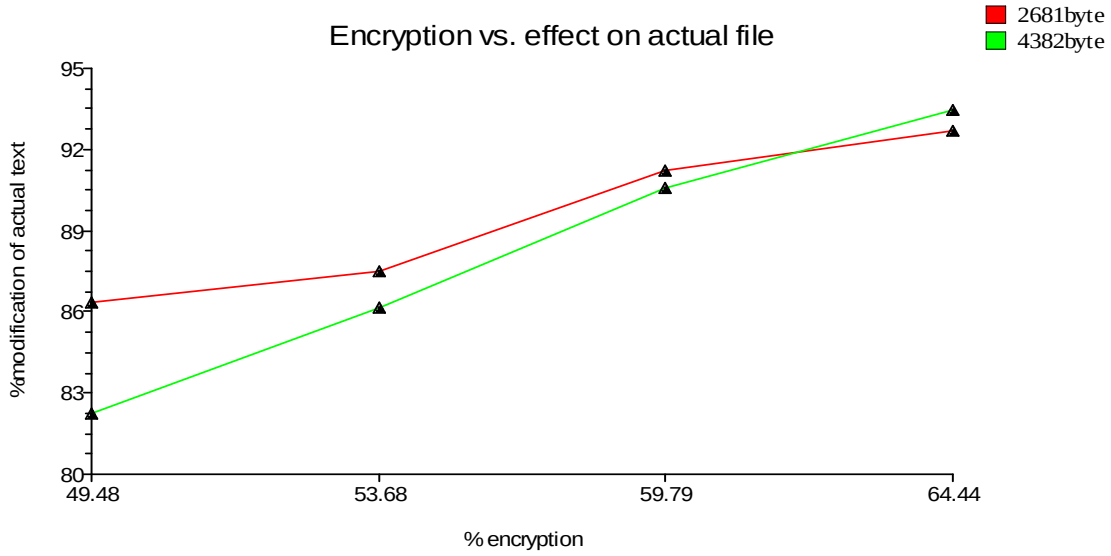

Fig. 4.54 % encryption Vs. % modification of original file

Table 4.20 data encryption & decryption throughput (Byte/See)

Data set	Process-I		Process-II	
	Encryption throughput(Byte/See)	Decryption throughput(Byte/See)	Encryption throughput(Byte/See)	Decryption throughput(Byte/See)
Data set-I	862.9911	869.3415	184.656	195.9417
Data set-II	3665.994	3672.331	1908.53	1916.386

Table 4.21 improvement of Repeat & Huffman Technique over Gzip

Data set	Sequence	File Size byte	Using Repeat algorithm.		Using RHUFF algorithm.		Improvement over gzip
			Compression ratio	Compression rate (bits /base)	Compression ratio	Compression rate (bits /base)	
Data set-I	atatsgs	9647	-0.65274	3.30548	0.016896	1.966207	12.01%
	atef1a23	6022	-0.65659	3.31318	0.014945	1.97011	
	atrdnaf	10014	-0.65608	3.31216	-0.016577	2.033154	
	atrdnai	5287	-0.68262	3.36523	-0.005485	2.01097	
	celk07e12	58949	-0.60037	3.20073	0.007413	1.985174	
	hsg6pdgen	52173	-0.586108	3.17221	-0.007724	2.015449	
	mmzp3g	10833	-0.66491	3.32982	-0.001754	2.003508	
	xlxfg512	19338	-0.57741	3.15482	-0.007343	2.014686	
	Average			3.26920		1.999907	
Data set-II	MTPACGA	100314	-0.68710	3.374205	0.07705804	1.84588	21.02%
	MPOMTCG	186608	-0.63881	3.27763	0.13772186	1.72455	
	CHNTXX	155844	-0.65324	3.306486	0.18046251	1.63907	
	CHMPXX	121024	-0.65646	3.31293	0.15418429	1.69163	
	HUMGHCSA	66495	-0.68800	3.376013	0.10032333	1.79935	
	HUMHBB	73308	-0.69705	3.394118	0.03863153	1.92273	
	HUMHDABCD	58864	-0.70786	3.415738	0.1088611	1.78227	
	HUMDYSTROP	38770	-0.78271	3.565437	0.00923394	1.98153	
	HUMHPRTB	56737	-0.72162	3.443256	0.08736803	1.82526	
	VACCG	191737	-0.63534	3.270688	0.08681162	1.82637	
	HEHCMVCG	229354	-0.64130	3.282611	0.15114626	1.6977	
	Average			**3.365374**		**1.794213**	

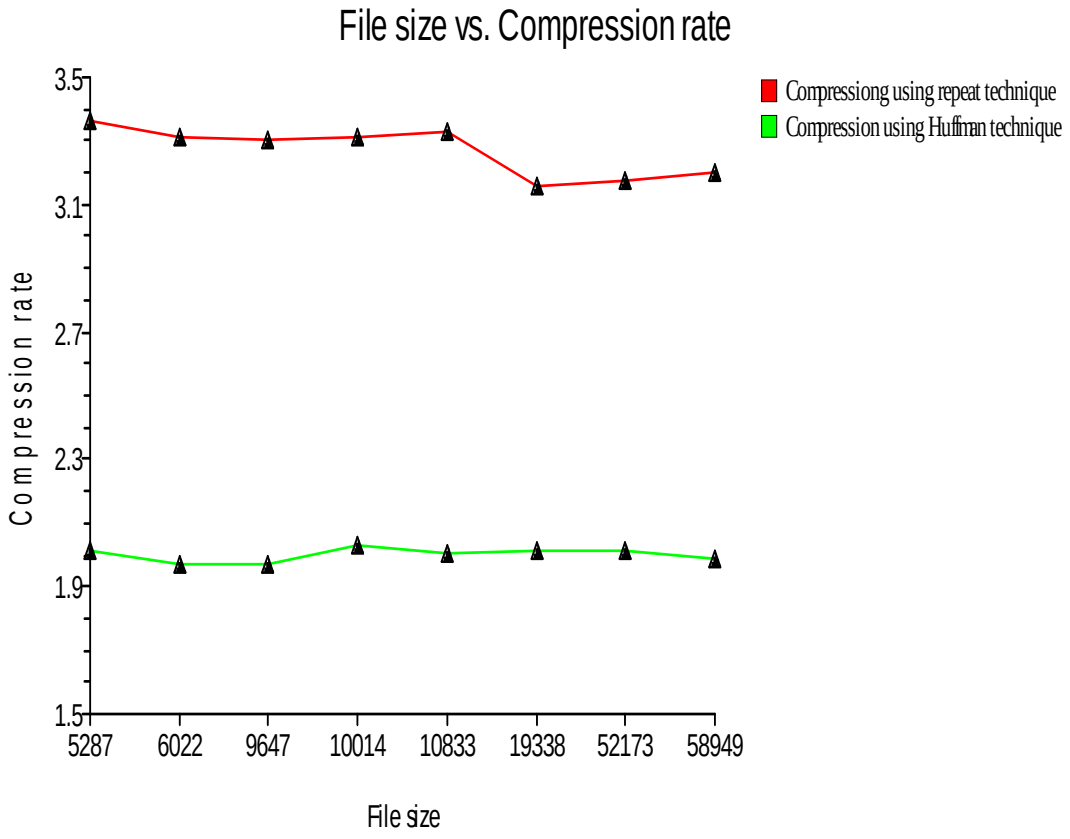

Fig. 4.55 compression rate vs. file size for data set-I using Repeat & Huffman's method

Fig.4.56 compression rate versus file size for data set-II using Repeat & Huffman's methods

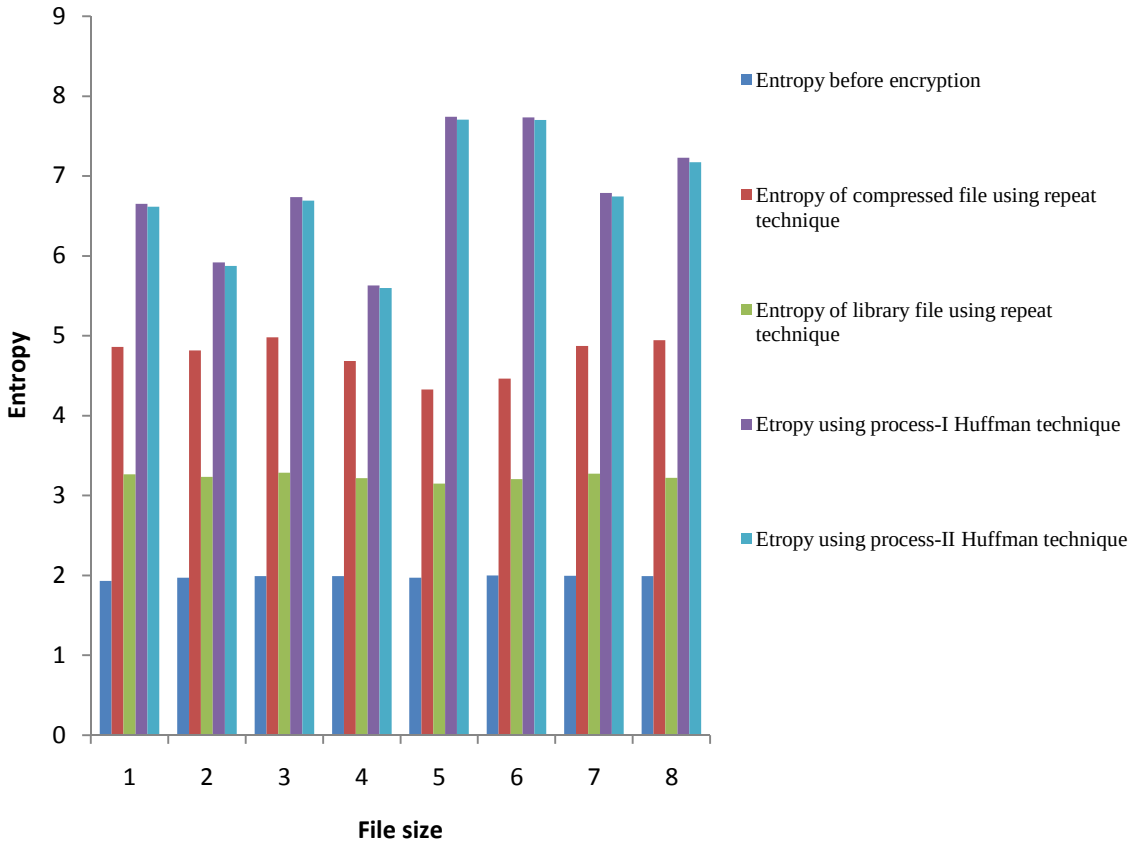

Fig. 4.57 entropy vs. file size for data set-I

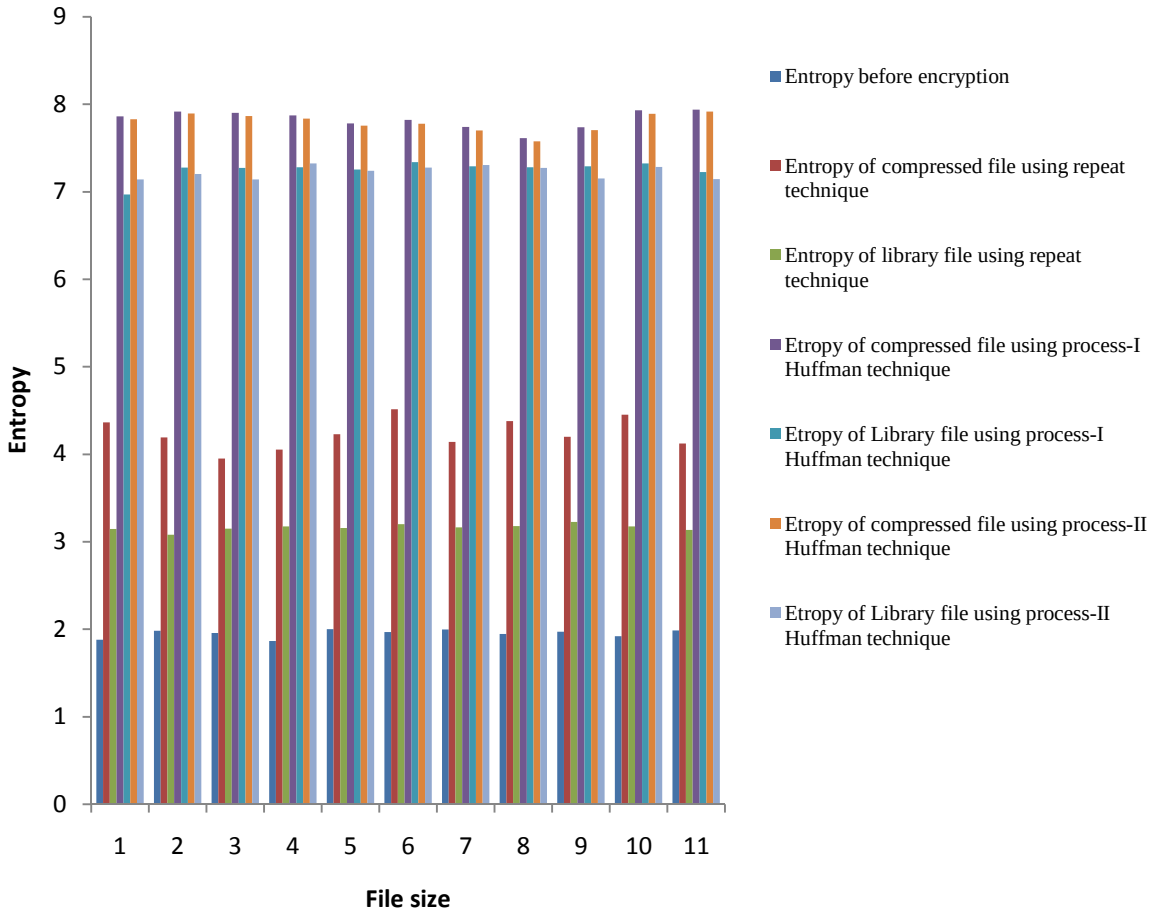

Fig.4.58 the entropy vs. file size for data set-II

Table 4.22 comparison our results with others standard results

Sequence	File Size byte	GZip	BZip2	Using RHUFF
atatsgs	9647	2.1702	2.15	**1.966207**
atef1a23	6022	2.0379	2.15	**1.97011**
atrdnaf	10014	2.2784	2.15	**2.033154**
atrdnai	5287	1.8846	1.96	**2.01097**
hsg6pdgen	52173	2.2444	2.07	**2.015449**
mmzp3g	10833	2.3225	2.13	**2.003508**
xlxfg512	19338	1.8310	1.80	**2.014686**
Average		**2.1098**	**2.058**	**1.999907**

Table 4.23 comparison our results with others standard results

DNA sequence name	MTPACGA	MPOMTCG	CHNTXX	CHMPXX	HUMGHCSA	HUMHBB	HUMHDABCD	HUMDYSTROP	HUMHPRTB	VACCG	HEHCMVCG	Average
Size →	100314	186609	155844	121024	66495	73308	58864	38770	56737	191737	229354	----
off-line	1.915	1.986	1.998	1.902	1.5993	1.969	1.9740	2.068	1.983	1.907	2.015	1.937
dna0	1.993	1.956	1.675	1.832	1.3860	1.939	1.9441	2.003	1.969	1.842	1.881	1.856
dna1	1.995	1.959	1.676	1.833	1.3946	1.945	1.9512	2.005	1.976	1.844	1.881	1.860
dna3	1.873	1.931	1.622	1.678	1.3750	1.880	1.9130	1.953	1.919	1.764	1.846	1.795
gzip	2.2919	2.3288	2.3345	2.2818	2.0648	2.245	2.2389	2.3618	2.2662	2.2518	2.3275	2.272
gzip-4	1.8827	1.9727	1.9519	1.8635	1.7372	1.8963	1.9141	1.9473	1.9207	1.874	1.9817	1.903
gzip -9	2.232	2.280	2.291	2.220	1.551	2.228	2.209	2.377	2.232	2.190	2.279	2.189
lz (32K)	2.249	2.289	2.300	2.234	1.580	2.255	2.241	2.427	2.269	2.194	2.286	2.211
WinRAR	2.23	2.30	2.24	2.25	1.38	2.22	2.19	2.37	2.23	2.23	2.32	2.19
lz (1M)	2.285	2.326	2.352	2.276	1.513	2.286	2.264	2.432	2.287	2.245	2.344	2.237
arith	1.880	1.984	1.957	1.867	2.001	1.969	1.999	1.949	1.972	1.919	1.985	1.952
arith+ (32k)	1.873	1.972	1.957	1.866	1.488	1.913	1.951	1.948	1.943	1.862	1.985	1.887
arith+ (1M)	1.873	1.961	1.956	1.866	1.438	1.911	1.950	1.948	1.942	1.862	1.985	1.881
normal PPMD+	2.018	2.075	2.062	1.977	2.077	2.116	2.130	2.237	2.130	2.002	2.053	2.079
adapted PPMD+	1.872(1)	1.966(2)	1.934(1)	1.840(1)	1.694(11)	1.921(2)	1.948(2)	1.921(1)	1.932(2)	1.910(2)	1.965(3)	1.900
PPMD+ escape	1.869(3)	1.964(3)	1.935(3)	1.839(3)	1.514(11)	1.923(3)	1.938(3)	1.931(3)	1.926(3)	1.908(3)	1.959(3)	1.882
normal CTW	1.902	1.989	1.974	1.879	1.376	1.917	1.909	1.960	1.922	1.897	1.997	1.883
CTW-4	1.866	1.962	1.933	1.838	1.363	1.892	1.897	1.920	1.913	1.857	1.958	1.854
Compress	2.12	2.20	2.19	2.09	2.19	2.20	2.21	2.23	2.23	2.14	2.20	2.18
bzip	2.1225	2.1701	2.1845	2.1218	1.7289	2.1481	2.0678	2.1802	2.0944	2.0949	2.1685	2.098
bzip-4	1.9847	2.0117	2.009	1.9667	1.8697	1.9957	1.9921	2.0678	2.0045	1.952	2.0091	1.987
bzip-2	2.12	2.17	2.18	2.12	1.31		2.07	2.18	2.09	2.09	2.17	2.05
ac-o2	1.8723	1.9654	1.9333	1.8364	1.9377	1.9176	1.9422	1.9235	1.9283	1.904	1.9647	1.920
ac-o3	1.8761	1.9689	1.9399	1.8425	1.9416	1.930	1.9466	1.9446	1.9352	1.906	1.9619	1.926
RHUFF	1.84588	1.72455	1.63907	1.69163	1.79935	1.92273	1.78227	1.98153	1.82526	1.826	1.6977	1.794
dna2	1.869	1.927	1.616	1.673	1.3668	1.867	1.9036	1.932	1.910	1.763	1.848	---
GenCompress	1.8624	1.9058	1.6146	1.673	1.0969	1.8204	1.8192	1.9231	1.8466	1.7614	1.847	---
CTW+LZ	1.8555	1.9000	1.6129	1.6690	1.0972	1.8082	1.8218	1.9175	1.8433	1.7616	1.8414	---
DNACompress	1.8556	1.8920	1.6127	1.6716	1.0272	1.7897	1.7951	1.9116	1.8165	1.7580	1.8492	---
Biocompress-2	1.8752	1.9378	1.6172	1.6848	1.3074	1.88	1.877	1.9262	1.9066	1.7614	1.848	---

This algorithm takes variable length sub-sequence size, starting from size 3 because less than 3 sub- sequence size is meaningless. Find out the results on normal & artificial sequences as well as reverse, genetic palindrome and invert complement of the sequences.

For c**ellular sequences**, the compression rate of data set-I is presented in table 4.7 and data set-II result is presented in table 4.8. The data set-I results are presented in graphically in fig. 4.17 & 4.18 and data set-II is presented in fig.4.19 & 4.20. The fig. 4.17 and 4.19 shows that the compression rate is independent of the file size but dependent on word size. The minimum average compression rate is 3.26920 bits/base for the data set-I and 3.190246 bit/base for the data set-II where word size is 4 and sequence orientation is complement. The compression rate is increased when the word size increases. To improve compression

rate/ratio the word size increase from 3 to 4, the compression rate decrease from 3.57189 bits/ byte to 3.327587 bits/base i,e 6% decreased, the library file increases about three to four times of word size 3 to 4 library size. 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. The nature of graph is heterogeneous in nature because sequences come from different species as shown in fig. 4.18 & 4.20 for both the data sets. In comparison library file size with compressed file size is too small. After applying Huffmans' technique the average compression rate of data set-I is 1.9999 bit/base and in case of data set-II the rate is 1.7942 bit/base, the result is presented in table 4.18/4.21/4.26 and 4.19/4.22/4.26. The result of data set-I is presented in fig.-4.74 and data set-II result is presented in fig. 4.75, shows that the increase in file size decreases the compression rate and Repeat plus modified Huffman technique is more acceptable.

For **artificial data**, the compression rate of data set-I is presented in table 4.9 and data set-II is presented in table 4.10. The results of data set-I are presented graphically in fig. 4.21,4.22 & 4.23 and data set-II is presented in fig. 4.24,4.25 & 4.26. The fig. 4.21 & 4.24 shows that the compression rate is dependent of file size as well as word size. The minimum average compression rate is 3.22323 bit/base for data set-I and 3.62444 bits/ base for data set-II, where word size is 3 and sequence orientation is reverse. The nature of graph is homogeneous in nature because sequences are randomly generated as shown in fig. 4.22 & 4.25. Now draw a fig. on the basis of data set-I is 23 and data set-II present in fig. 4.26 on the basis of cellular sequences versus artificial data, getting district fig. 4.23 & 4.26 where as 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 artificial sequence whereas library file is variable size in case of cellular sequence. In comparison library file size with compressed file size is too small.

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 mentioned the section in 5.1 where as the artificial data, the compression ratio is followed by the equation as {1-Output/input}; where the output size is number of bit. Also tables show the average compression gain of the sequences, observed that lower the compression rate, higher the compression gain.

If sequences encrypt by three/four characters secret key (Repeat technique, sequences compressed by the sub-sequence/word of different size), calculate percentage of encryption and percentage modification of actual text, this result of data set-I & II is presented in table 1. It is observed that in data set-I the average 41%-43% & data set-II the average 41%-42% of encryption on the actual text will be modified 95% for both the data set-I & II on the actual file.

The table 4.11,4.13,4.14,4.15,4.16,4.17 & 4.18 shows that the entropy is incensed two to three times before and after compression. As a result both the compressed file and library file increased the randomness, so, the attacker cannot attack the sequence easily. The data set-I is presented fig. 4.57 and data set-II is presented in fig. 4.58 shows the entropy of compressed file and library file before and after compression.

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

The table 4.12 shows the relative frequency for the different input file and encrypted output file, also shows the ratio is maintained in between input and output is 48:1 that mean possibility of frequency attack is minimized. The percentage of encryption did not vary significantly with compression rate, presented the data set-I result in fig. 4.34 and data set-II result is presented in fig. 4.38. The percentage of encryption varies due to sub-sequence/word consisting different characters with respect to file size & % modification of actual text also data set-I is presented in fig. 4.33 and data set-II is presented in fig. 4.37. The major effectiveness is found in the sequences. But decrypt the sequences without actual sub-sequence/word value or entered an incorrect sub-sequence the sequence will be different.

This is to observe that in case of repetition of input text having same frequency counting is not more. For key purpose use level number. The same frequency character ratio is approx 4:1 with respect to input and output text, in that situation frequency analysis attack is reduced. But if decrypt is done without applying key value or entered an incorrect key the text will be different.

In process –I : The result shown in table 4.13 (for data set-I) & 4.14 & 4.15 (for data set-II) that above 99% modification of actual file can be observed in data set-I & II by only when encrypting 58% to 100 for data set-I & 51% to 61% for data set-II of the actual file. If

consider the highest level of Lavenstein distance, the effect on original file is highest on the basis of top level interchanging. On the other at the lower level of Lavenstein Distance the effect is proportional. This result of data set-I is presented in fig.4.33 to 4.35 and data set-II result is presented in fig.4.37 to 4.39 on the basis of different file size, % effect on the original file, % encryption and compression rate. The encryption is increased with respect to the output text effectiveness. The percentage of effect on actual file in increased when input file size is increased and vice-versa.

In process-II : now taking the same text which was considered for previous experiment and do the same job i.e. first calculate frequency of each character and then measure the relative frequency to analyse probability of attack. Before encryption the text use two key values in binary form. These are actually specified two nodes at different levels. The results are shown in table 4.16 for data set-I & 4.17 & 4.18 for data set-II.

Now measure relative frequency in case of encrypted text in the same manner when calculating this for input text. For encrypt text the redundancy value is 389. But in case of input text this value is 1244. The attack is low when output and input sequences relative frequency ratio is nearly 4:1. The results in the table shown that different input text and encrypted output value relative frequency are different at different level. These are shown in the table 4.17.

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

The results of table 4.16, is presented in graphically of data set-I in fig. 4.45 & 4.46 and data set-II is presented in fig. 4.48 & 4.49; shown that 99% original text will affect and reduce percentage of encryption when swapping at lower level. If encrypt the sequence on the basis of two binary values as a key getting higher security. Also increase the encryption & efficiency of the sequences.

In process –III : Testing purpose use two text files of size 2681 & 4382 bytes, find out the actual effect on the text by interchanging the different nodes based on calculating % encryption and % modification on the actual text, result shown in table 4.24.

Based on data, drawn a graph of fig.4.54, observed that 88% actual text effected when 57% encryption is done on the actual text. Since number of distinguishable words are huge so it need more %of encryption, but in case of considering character if apply minimum 42% encryption of real text it will affect more than 90% of real text case.

Table 4.20 shows the encryption & decryption throughput. The result shows the decryption throughput is less than encryption throughput. Also observed that data set-II throughput is better than data set-I. Also observed that process-I encryption is better than process-II encryption.

Table 4.21 shows the improvement results on Gzip technique , the improvement is 12.01% on data set-I and 21.02% on data set-II, graphically shown in fig. 4.55 & 4.56.

The encryption & decryption time is presented in table 4.13 & 4.16 for data set-I and 4.14,4.15, 4.17 & 4.18 for data set-II. This tabular result is presented in graphically of fig. 4.36,4.40,4.44,4.47,4.50 & 4.53 of process-I & II. The decryption is always less than encryption time. The time in decryption and encryption is proportional to the file size. This is the basic principal for all encryption techniques (AES / DES / RSA / DNA). So this technique is very much effective for security purpose.

The Table 4.22 & 4.23 shows the comparison result on standard available techniques. This combine technique RHUFF (Repeat + Modified Huffman's) shown by red color is better than 23 nos. techniques (mention 1st coloum of table 4.23), also below RHUFF red color results are better than column 1 standard techniques.

This RHUFF technique is far better than gzip techniques with respect to compression rate and information security. The gzip technique is Lempel-Ziv "LZ"+Huffman [6] without any security concept. This method is the combination of method of Repeat with modified Huffman for compression as well as security.

Important observations are :

a) The cellular DNA sequences have logical organization, structure, systematic and non random where as artificial data are random and unstructured.
b) The substring measure end to end different from 2 to 5 and no match is discovered if the substring measure end to end becoming more than 6.

c) The cellular DNA sequence encode amino acid/protein that why sub-sequence of repeat/reverse/palindrome/genetic complement are found in the original sequence, more exact match are found in the repeat search method, other orientation the exact match are found in less number over repeat method.

d) The results are showing that cellular DNA sequence are reasonable compressible in any orientation (cellular DNA sequence, reverse sequence, complement sequence and reverse complement sequence) result is homogeneous in nature also where as artificially (random sting) generated sting of same length compression rate is heterogeneous in nature.

e) All compression rate are similar also suggests a highly similar sequences

f) This technique are also apply on corresponding other orientation of cellular DNA sequences like Reverse, Complement & reverse complement of cellular DNA sequence, the better result found on normal i.e cellular DNA sequence performance is better.

g) Best encryption result is found when modified Huffman's technique apply on library file

h) The output text having ASCII code and non match DNA bases, containing more than four character than the input text. So, after first pass Huffman's and two bit encoding technique is easily applicable and overcome the drawback of using Huffman's & two bit encoding technique.

5 Conclusion

The compression rate of Cellular DNA sequences lies between 3.25 bits/bases & 3.3 bits/base, where as in case of artificial data value lies in between 3.3 bits/bases & 3.5 bits/base. This combine technique, the compression rate lies between 1.69 bit/base to 1.92 bits/base. The nature of graph is homogeneous in case of cellular sequence where as artificial sequences the nature of graph is heterogeneous in nature.

The lowest compression rate is found when repeat technique run on the bench mark DNA sequences in complement orientation and the sub sequence size is 4, lowest compression rate is 3.26920 bit/base. Also the output is secured than input sequences in transmission point of view. The output contains 256 characters include a, t, g & c. So, the resultant test is highly secure than input text. The substring measure end to end different from 2 to 5 and no match discovered if the boat able to go under water line measure end to end becoming 6 or more.

The results show that the compressed pattern matching algorithm for DNA order is in competition among the best algorithm.

The results are showing that the compression rate & ratio are different to each other in case of reverse, complement and reverse complement, it make certain that the point of comparison facts are a part of same family.

If consider library with compress file, the overall compression rate is slightly increase. So better result find only when compression rate is calculated on compressed file size. In comparison to compressed file the library life is too minuscule.

In case of selective encryption of compressed text, as compression follows the encryption process, the effect on text based on statistical properties, plain text will not be possible because of the reduction of redundancy due to process of compression. This approach of selective encryption has got some advantage due to constraint of the bandwidth in network before communicating and also it needs to be encrypted so confidentiality is maintain or to protect the digital rights.

This is a static Huffman coding method applied on compressed text (1^{st} pass output use here as input) based on selection encryption and effectiveness compare by dissimilarity in original file. If anyone can decrypt without key, the cipher text resistance from the attacks based on statistical property of the plain text

For applying selection encryption, the result found that the encryption effectiveness is increases at where interchanging is done. So the attack is very low on the basis of probability of frequency analysis.

Another conclusion is that if consider word instead of characters then workspace is increased. But in this case, it is found the percentage of encryption is 57 then the actual effect on the text is 88%, so the different number of word is more and their frequency is less on character frequency. It is observed that relative frequency ratio of input text and encrypt text is nearly 4:1, so same frequency character ration in between input and output is approx 4:1.

In case of word consideration word's frequency are high but other word have very lower frequency. These lower frequency words are representing by higher bit. So percentage of compression decreases. In terms of security word encryption is more effective than character encryption. In case of character encryption, There is only 256 characters are available and

since workspace is short. So here is a possibility to break the security. But in case of word encryption, numbers of distinguishable words are huge, not known by all, so that workspace is also increased and breaking the security is not possible.

This techniques provide the better facts safety than other techniques. Also biological order compression is an useful apparatus for recovering useful knowledge from biological orders. This lossless compression technique achieve a moderate compression rate & ratio than that of existence DNA order compression algorithm and provides the better knowledge safety with encryption least decompression time, the execution time of this algorithms is fraction of second and optically points the different in between cellular DNA order and not natural DNA of equal measure end to end.

The process-II encryption techniques requires optically higher number of bits for encoding the data and repeat technique requires optically less number of bits, indicating process-II Huffman's requires transmission of higher bandwidth.

The entropy increased from 1.98 to 7.90 per byte of encryption. The information entropy is measure by a degree of randomness. Randomness is an important and desirable property of compression-encryption algorithm. Hence, both compressed file and library file yield high degree of randomness in output information, making the output less susceptible to attacks.