Pertanika J. Sci. & Technol. 26 (4): 1913 - 1925 (2018)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

WBMFC: Efficient and Secure Storage of Genomic Data

Sanjeev Kumar*, Suneeta Agarwal and Ranvijay

Department of Computer Science and Engineering, MNNIT Allahabad, India

ABSTRACT

With the development of next-generation sequencing technology, a massive amount of genomic data are being generated day by day. To efficiently handle these data for storage, processing and transmission, some specialized genomic data compression techniques are need of today. In the near future, personalized genomics may come into existence where doctors may give the treatment on the basis of patient genome. It creates a huge challenge to securely store and transmit the genomic data over the cloud servers or remote servers. This problem can be solved by applying a combination of encryption and compression techniques. Most of the state of the art algorithms for secure and efficient storage of genomic data adopt the policy of encryption after compression. The computational costs of these algorithms are very high, so there is a need to develop a unified encryption-compression algorithm (encryption during compression) to provide the confidentiality/secrecy also to genomic data. In this paper an approach applying encryption during compression is proposed to efficiently and securely store the genomic data in fasta/multi-fasta file format. Here MWBTC (Modified Word Based Tag Code) and Delta Encoding are used for compression and AES-256 is used for encryption. Experiments show that the proposed algorithm (WBMFC) outperforms the state of the art algorithms in terms of processing time and compression ratio both.

Keywords: Compression, decompression, encryption, fasta, multi-fasta

INTRODUCTION

Genome data is a collection of genetic information (in the form of DNA sequence) of a living organism. Size of such data is very large. For example, one human genome contains 3.2 billion DNA base pairs, which takes 3 GB in memory (Danek, & Deorowicz, 2018). With the advancement of sequencing machine technology huge

ARTICLE INFO

Article history:
Received: 16 January 2018
Accepted: 28 May 2018
Published: 24 October 2018

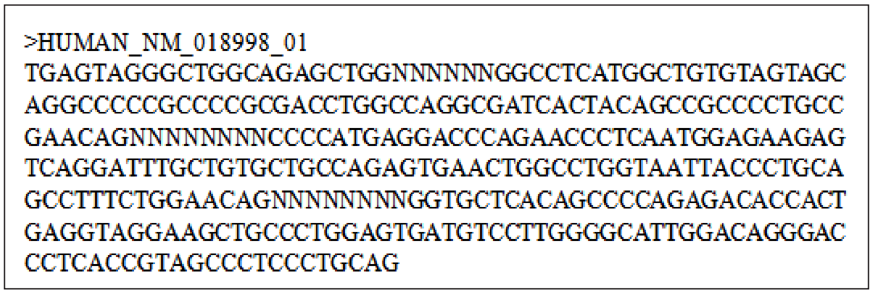
E-mail addresses:
skgcp86@gmail.com (Sanjeev kumar)
suneeta@mnnit.ac.in (Suneeta Agarwal)
ranvijay@mnnit.ac.in (Ranvijay)
* Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

volumes of genomic data are being deposited in public repositories (NCBI, DDBJ, EBI) and cloud servers for the purpose of research, forensic and diagnosis (Hosseini et al., 2016). The exponential growth of these data creates a severe challenge for secure storage, fast processing, and transmission. Compression with encryption is a crucial tool to address these challenges (Wiese et. al., 2018). It reduces storage space and processing cost along with security and also speeds up data transmission. To store and transmit genomic data efficiently and securely, a unified compression-encryption algorithm is the need of today.

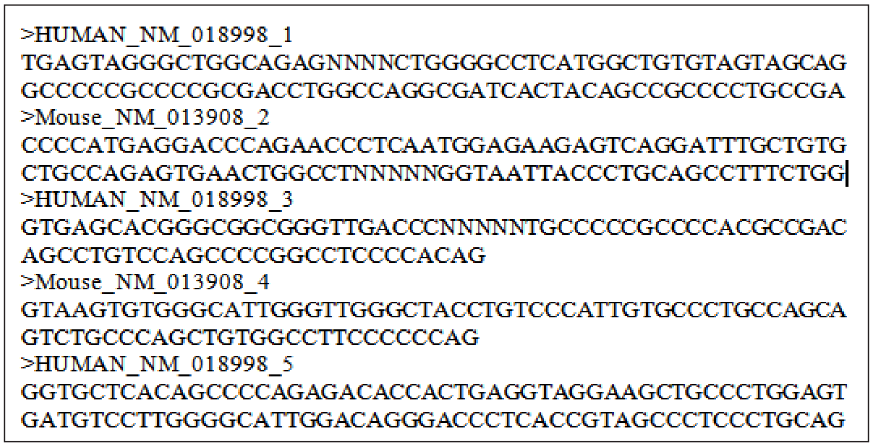
Most of the public repositories store the genomic data in fasta/multi-fasta file format. Fasta is a text based file format to represent a genomic sequence (Danek & Deorowicz, 2018). Fasta file always starts with ">" followed by the header which contains some descriptions about the genome and sequencing machine used, next is the sequence data which contains the characters A, C, G, T, and N. A Multi-fasta file contains several fasta files. Formats of Fasta and Multi-fasta files are shown in Figure 1 and Figure 2 respectively (Sardaraz et al., 2014).

Genomic data has some special characteristics such as large no of repeats (tandem & palindrome) and less no of nucleotides (A/ C/ G/ T) (Dix et al., 2006). To compress the genomic data, general purpose compression algorithms (Gzip, Bzip2, LZMA) do not take



```
>HUMAN_NM_018998_01
TGAGTAGGGCTGGCAGAGCTGGNNNNNNGGCCTCATGGCTGTGTAGTAGC
AGGCCCCCGCCCCGCGACCTGGCCAGGCGATCACTACAGCCGCCCCTGCC
GAACAGNNNNNNNNCCCCATGAGGACCCAGAACCCTCAATGGAGAAGAG
TCAGGATTTGCTGTGCTGCCAGAGTGAACTGGCCTGGTAATTACCCTGCA
GCCTTTCTGGAACAGNNNNNNNNGGTGCTCACAGCCCCAGAGACACCACT
GAGGTAGGAAGCTGCCCTGGAGTGATGTCCTTGGGGCATTGGACAGGGAC
CCTCACCGTAGCCCTCCCTGCAG
```

Figure 1. Fasta file



```
>HUMAN_NM_018998_1
TGAGTAGGGCTGGCAGAGNNNNCTGGGGCCTCATGGCTGTGTAGTAGCAG
GCCCCCGCCCCGCGACCTGGCCAGGCGATCACTACAGCCGCCCCTGCCGA
>Mouse_NM_013908_2
CCCCATGAGGACCCAGAACCCTCAATGGAGAAGAGTCAGGATTTGCTGTG
CTGCCAGAGTGAACTGGCCTNNNNNGGTAATTACCCTGCAGCCTTTCTGG
>HUMAN_NM_018998_3
GTGAGCACGGGCGGCGGGTTGACCCNNNNNTGCCCCCGCCCCACGCCGAC
AGCCTGTCCAGCCCCGGCCTCCCCACAG
>Mouse_NM_013908_4
GTAAGTGTGGGCATTGGGTTGGGCTACCTGTCCCATTGTGCCCTGCCAGCA
GTCTGCCCAGCTGTGGCCTTCCCCCCAG
>HUMAN_NM_018998_5
GGTGCTCACAGCCCCAGAGACACCACTGAGGTAGGAAGCTGCCCTGGAGT
GATGTCCTTGGGGCATTGGACAGGGACCCTCACCGTAGCCCTCCCTGCAG
```

Figure 2. Multi-fasta file

into account the characteristics of these data, while domain specific algorithms (BIND, Deliminate, MFCompress, SeqCompress and Cryfa) utilize the characteristics of these data (palindrome repeats, tandem repeats and less no of nucleotides) (Kumar et al., 2015). For security if required these algorithms use some external tools for encryption. Most of the existing DPCA adopt two level techniques to compress and encrypt the genomic data, i.e., encryption after compression (Figure 4). These techniques take large amount of time to compress & encrypt the genomic data. Thus to reduce the processing time there is a need to develop a unified encryption-compression technique (encryption during compression) (Figure 5).

There is a number of state of the art algorithms to compress the fasta/multi-fasta files. BIND (Sardaraz et al., 2014) uses two binary streams for compression. In the first stream, both A & T are assigned bit 0, and C & G are assigned bit1. In the second stream, both C and T are assigned bit 0, and A & G are assigned bit1. These two streams are independently compressed with LZMA general purpose compression algorithm. The average compression ratio of BIND is 4.3.

DELIMINATE (Mohammed et al., 2012) uses delta encoding for two most frequent characters, remaining characters are encoded with 0 and 1. Average compression ratio is 4.65. Compression time and decompression time are same as of BIND.

MFCompress (Pinho & Pratas, 2013) uses finite text models for encoding the fasta and multi-fasta files. It is based on a probabilistic model that determines probability distribution by calculating the probability of next nucleotide in the genome sequence based on k-previous nucleotides (order-k context). MFCompress uses single finite context model for encoding of header data and order-k context model for encoding sequence data. Compression ratio is same as of Deliminate, but compression and decompression speed are less in comparison to Deliminate.

SeqCompress (Sardaraz et al., 2014) uses arithmetic coding and statistical model. The statistical model is based on the frequency of fragments in the input sequence, decides whether to use fragment-based compression or binary compression. It is a two-pass algorithm. An average compression ratio of SeqCompress is 4.92 but compression time and decompression time are higher than that of BIND and Deliminate.

Cryfa (Pratas et al., 2017) first uses three-bit packing technique to reduce the size of genomic data, and thereafter encryption is applied to this packed data. The compression ratio of Cryfa is same as of DELIMINATE. It also provides encryption.

Existing state of the arts techniques for compression of fasta/multi-fasta files are either dictionary based or statistical-based. Dictionary-based techniques work in two phases: first, a dictionary is created, and then a substitution based method is used to encode the sequence. Such technique requires the large size of dictionary during decompression, it creates the problem of storage and transmission (Darok et. al., 2017). Statistical models used probabilistic method to predict the next character from the past occurrences of the characters. Such techniques require huge memory during compression and decompression. For security existing tools use two-level techniques, i.e., encryption after compression. The computational cost of such a technique is very high (Jahaan et al, 2017). So there is need to develop a unified technique (encryption during compression) for compressing the fasta/multi-fasta files. In this paper, a new approach to compress & encrypt the fasta/multi-fasta file is presented. The proposed approach uses unified encryption approach (encryption during compression) to encrypt and compress the genomic data. Proposed approach first divides the fasta/multi-fasta file into two streams: header stream and sequence stream. These two streams are compressed individually with the appropriate compression algorithms. Delta difference encoding is used to compress the header stream while MWBTC (Modified Word Based Tag Code) (Gupta & Agarwal, 2008) is used to compress the sequence stream. Encryption is applied only on the dictionary created by MWBTC at the time of encoding instead of applying it on the whole genome data. Since the size of a dictionary in MWBTC is very small, therefore whole genomic data is encrypted very fast.

METHODS

Proposed method first separates the genomic data (fasta / multifasta) in two streams: sequence stream (W1) and header stream (W2) (files). Stream W2 contains header parts of the input genome sequence while stream W1 contains remaining part of the input genome sequence. If input file is a Multi–fasta file, then along with header data lengths of the corresponding sequences are also stored in W2. Files W2 and W1 are compressed with delta difference encoding and MWBTC respectively as shown in Figure 3. Encryption is applied on the dictionary created by MWBTC to secure the whole genome data.

Details of compression and encryption method used are as follows:

Header Stream Compression

To compress the header stream Delta difference encoding method is used. Finally, it is archived by 7ZIP general purpose compression as shown in Figure 3.

Sequence Stream Compression

To compress the sequence stream all occurrences of non ACGT character N (if present) are first extracted. All positions of N are recorded in a separate file W3. Generally in genomic sequence, N's are present in clusters. Positions of first occurrence in each cluster along with the size of that cluster are stored in W3 as shown in step 1.1 of Example 1. Thereafter remaining characters A, C, G & T of the sequence stream W1 are encoded with the modified word based tag code (MWBTC). MWBTC reads the file W1 and segments it into words of size 4, maximum 256 words are possible. Thereafter frequencies of all words of

W1 are calculated and stored in column "count" of table1. A new table "tempvocab" is formed with three columns: index, word, and count. Index column is for indices of each word. Word column contains all the possible words (A, C, G, and T) of size four stored in lexicographical order. Count columns contain corresponding frequencies of the words obtained from Table 1. A new Table 3 is formed by sorting full rows of Table 2 with respect to the contents of the column count. First column (index) of Table 3 is renamed as shuffled index here. A new column "index" is also added for indices (0-255) as the first column of Table 3.

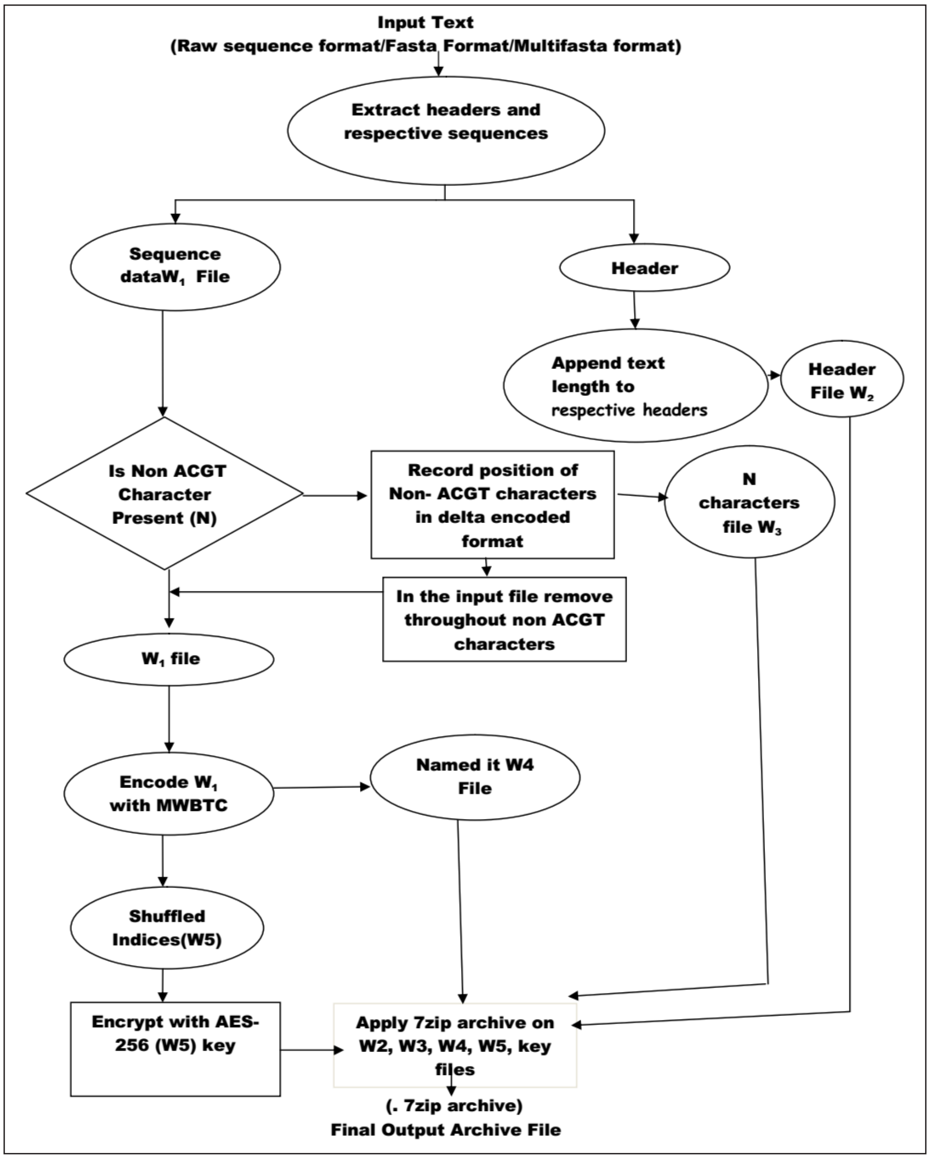

Input Text
(Raw sequence format/Fasta Format/Multifasta format)
Extract headers and respective sequences
Sequence dataW₁ File
Header
Append text length to respective headers
Header File W₂
Is Non ACGT Character Present (N)
Record position of Non- ACGT characters in delta encoded format
N characters file W₃
In the input file remove throughout non ACGT characters
W₁ file
Encode W₁ with MWBTC
Named it W4 File
Shuffled Indices(W5)
Encrypt with AES-256 (W5) key
Apply 7zip archive on W2, W3, W4, W5, key files
(. 7zip archive)
Final Output Archive File

Figure 3. Flow diagram of the proposed approach

Codes are assigned to each word of Table 3 using MEncode method (Algorithm1) and stored them in a new column (fifth column) codes. MWBTC allocate less number of bits for frequents words and more bits for lesser frequent words. Contents of file W1 are encoded using the generated codes in Table 3 and named it as W4 file. The complete encoding process of MWBTC is explained in step 1.2 of Example 1.

Example 1: Let the given genome sequence for secure and efficient storage be:
T = A A A A C G G A A T T N N N N N N G A T A A A A A N N
N N C G A C A A T T G A T A A A A A A A A A A C A C
The size of T is 54 characters.
Steps of compression and decompression by proposed approach are as follows:

Step 1. Compression Process
\\ **Input:** Genome Sequence T
\\ **Output:** Output.7zip - Compressed genome sequence along with secret key and a dictionary.

Step 1.1 (Extraction of non ACGT character N): Input sequence "T" has two clusters of N's at positions 13-18 & 27-30. All N's (if present) are extracted from input genome sequence T. They are stored as 13, 6, 27 and 4 in a new file W3. Here numbers at odd positions represent starting positions of clusters and numbers at even positions represent the size of the respective clusters.

Step 1.2 (Encoding): After extraction of non ACGT character N from input sequence T, resulting input sequence T'=A A A A C C G G A A T T G A T A A A A A C G A C A A T T G A T A A A A A A A A A A C A C
The size of T' is now 44 characters. The content of "vocab" table is presented in Table 1**.**
The compressed form of sequence T'' (W4 file):
01110110000101111010000101010010

Contents of the second column of Table 3 are stored in a new file W5. W5 alone is sufficient to decode the above compressed sequence T''. To enhance the security of whole genome sequence it is proposed to encrypt the contents of W5 only using AES-256 (Mahajan & Sachdeva, 2013).

Table 1
Contents of table "vocab"

Index	Word	Count
0	AAAA	4
1	CCGG	1
2	AATT	2
3	GATA	2
4	CGAC	1
5	ACAC	1

Table 2
Contents of table "tempvocab"

Index	Word	Count
0	AAAA	4
1	AATT	2
2	ACAC	1
3	CCGG	1
4	CGAC	1
5	GATA	2

Table 3
Contents of table "finalvocab"

Index	Shuffle indices (positions of words in table2)	Word	Count	Codes
0	0	AAAA	4	01
1	1	AATT	2	10
2	5	GATA	2	0001
3	2	ACAC	1	0010
4	3	CCGG	1	1101
5	4	CGAC	1	1110

At the end there are total five files for secure and efficient storage of fasta/multi-fasta file (T) W2, W3, W4, W5 & secret key used for encryption of W5. All these files are finally archived by "7zip" (an open source archival) to generate a single compressed output file (output.7zip) for input genome sequence T.

Step 2. Decompression Process
\\ **Input:** Output.7zip
\\ **Output:** Genome Sequence T

Step 2.1: Files W2, W3, W4, W5 & secret key are extracted from output.7zip archived file. W5 file is decrypted using the secret key to get the shuffled indices. A table "dvocab" with two columns: index and word is created. Index column is for indices of words and word column contains all the possible words (A, C, G and T) of size four stored in lexicographical order (here as an example only few entries are shown but it has 256 entries).

Step 2.2: Rearrange the contents of the "dvocab" table with the help of shuffled indices obtained by W5 file (0, 1, 5, 2, 3, and 4). A new column "new index" is also added at beginning of the Table 5.

Table 4
Contents of table "dvocab"

Index	**Word**
0	AAAA
1	AATT
2	ACAC
3	CCGG
4	CGAC
5	GATA
-	-

Table 5
Contents of table "rvocab"

New Index	Old Index	Word
0	0	AAAA
1	1	AATT
2	5	GATA
3	2	ACAC
4	3	CCGG
5	4	CGAC

Table 6
Contents of table "finaldvocab"

New Index	Old Index	Word	Codes
0	0	AAAA	01
1	1	AATT	10
2	5	GATA	0001
3	2	ACAC	0010
4	3	CCGG	1101
5	4	CGAC	1110

Step 2.3: Codes are assigned to each word of table 5 using MEncode method (algorithm1) and stored in a new column "codes". Once the "finaldvocab" table is ready decoding of binary words can be started. At any step i the decoder reads a binary word (two bits at a time) from W4 file till it ends with 01 or 10, which is decoded with the help of "finaldvocab" table (Table 6).

Compressed form of sequence T'': 01110110000101111010000101010010

Genome Sequence T': AAAACCGG AATT GATA AAAA CGACAATTGATAAAAAAAAAACAC

With the help of W3 file, positions of all 'N's (non A/C/G/T) are added in T'. W3 file contains 13, 6,27,4. To get back the original genome sequence T, non acgt character 'N' is added at positions 13 and 27 of cluster size 6 & 4 respectively. Thus the obtained sequence is:

Original genome sequence,

T= AAAACCGGAATTNNNNNNGATAAAAAN
NNNCGACAATTGATAAAAAAAAAACAC

This coding technique is a prefix free, no codeword is a proper prefix for any other code, and hence it is directly decodable. In MWBTC every code is ended with either 01 or 10, this property enables searching substrings directly over compressed data.

Encryption and Archival

Contents of W5 file are sufficient to decode (to get back the original genomic sequence) from compressed data (W4). To secure whole genome sequence it is proposed to encrypt the contents of W5 file only (Figure 5). AES 256 (Mahajan & Sachdeva, 2013) is used here to encrypt the content of W5. As the size of W5 file is very small so encryption will not take much time.

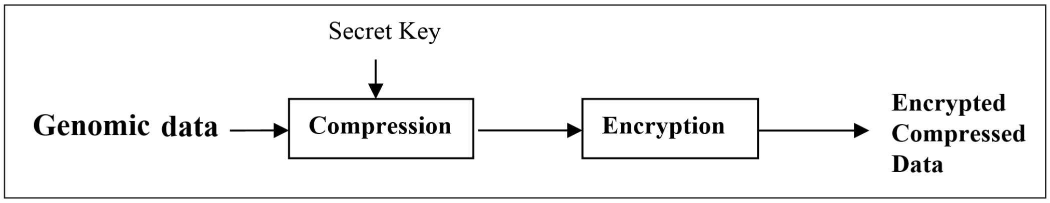


Figure 4. Existing two level approach for secure storage of genomic data: Encryption after compression

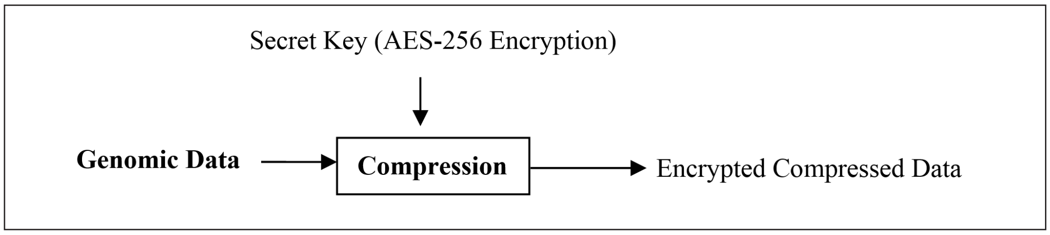


Figure 5. Proposed one level approach (unified) for secure storage of genomic data: Encryption during compression itself

In the proposed method encryption is applied only on the contents of W5 file (having indices from 0-255) only while other approaches apply the encryption on the whole compressed data (about 705Mb in case of Human genome). Thus proposed method requires very little additional computational effort to provide the security of genomic data.

After applying compression and encryption there are five files W2, W3, W4, W5 (encrypted shuffled indices) & secret key. These all are finally archived by "7zip" (an open source archival) to generate a single compressed output file (outut.7zip) for the input genome sequence.

Decompression

Files W2, W3, W4, W5 & secret key are extracted from output.7zip archived file. W5 file is decrypted using the secret key. A temporary "dvocab" table is created as in step 2.1 of example 1. Reshuffle the content of the "dvocab" table with the help of shuffled indices obtained by W5 file as shown in step 2.2 of example1. Once the "finaldvocab" table is ready decoding of binary words can begin immediately as shown in step 2.3 of example 1. From W4 file decoder reads binary words ending with 01 or 10 and decode it with the help of "finaldvocab" table.

Algorithm1

```
//MEncode method use to encode the binary word into binary code words
 1. T is the total no of segment present in vocab
 2. i is the level of n^th segment
 3.  T[0] ← 01
 4.  T[1] ← 10
 5.  x← 0,y←0,z← 2
 6.  for  i ←1 to 8
 7.  for  j←0 to power(2,i)
 8.  T[z].append(T[0])
10. T[z+1].append(T[1])
11.  z←z+1
12.  j←j+1
13.  i←i+1
14.  for j← 0 to power(2,i)
15.  T[k].append(T[1])
16.  T[k].append(T[y++])
17.  z←z+1
18.   j←j+1
19.  x←z-power(2,i+1)
20.  y←z-power(2,i+1)
21.  return T[i]
```

RESULT AND DISCUSSION

The performance of the proposed method has been compared with two state of the art fasta/multi-fasta file compressors: Seqcompress and Cryfa. Four different datasets (FAN (ftp://ftp.ncbi.nih.gov/genomes/ Bacteria/all.fna.tar.gz), FEN (ftp://ftp.ncbi.nih.gov/genomes/Bacteria/all.ffn.tar.gz), Camera (goldenPath/hg18/Chromosomes/) and Eukaryotic (https://portal.camera.calit2.net) are used for experiments. FAN, Camera and Eukaryotic datasets are in Fasta file formats while Fen dataset is in Multi-fasta file format. All experiments are performed on Ubuntu (64-bit) machine having a 2.33 GHz core i7- processor. The proposed method has been implemented in Java.

Table 7 confirms that compression ratio of proposed method is higher than other state of art algorithms. Figure 6 shows the times required for the SeqCompress, Cryfa and proposed method. SeqCompress and Cryfa are based on encryption after compression policy (Figure 4) while the proposed method uses unified compression-encryption policy (Figure 5). From Figure 6 it is confirmed that unified compression-encryption technique takes significantly less time in comparison to other techniques. Figure 7 shows memory consumption for the SeqCompress, Cryfa and proposed method. From Figure 8, it is confirmed that memory consumption of proposed method is least among existing techniques.

Table 7
Comparison of compression ratio of proposed method with SeqCom-press and Cryfa

Datasets	Uncompressed Size(in Megabits)	Compressed size using proposed method (in megabits)	Compression ratio**		
			Proposed	SeqCompress	Cryfa Method
FNA Datasets	40100.16	9155.92	**4.37**	4.21	3.89
FEN Datasets	38338.80	8168.48	**4.69**	4.34	*
Camera Data sets	19144.88	2102.24	**9.10**	8.65	7.64
Eurokaryotic	114703.68	22922.8	**5.03**	4.83	4.25

Note: *This method does not support multifasta file; ** CR=Uncompressed/Compressed

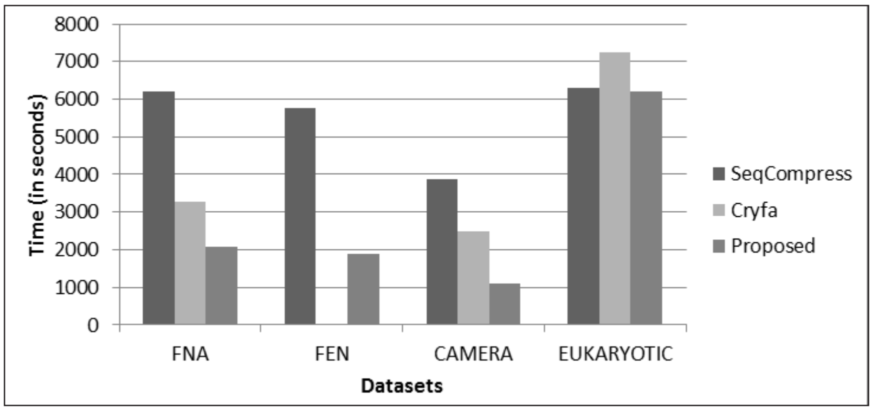


Figure 6. Time required for SeqCompress, Cryfa and Proposed method to securely and efficiently store various data sets (Time in seconds) (Cryfa does not support FEN dataset)

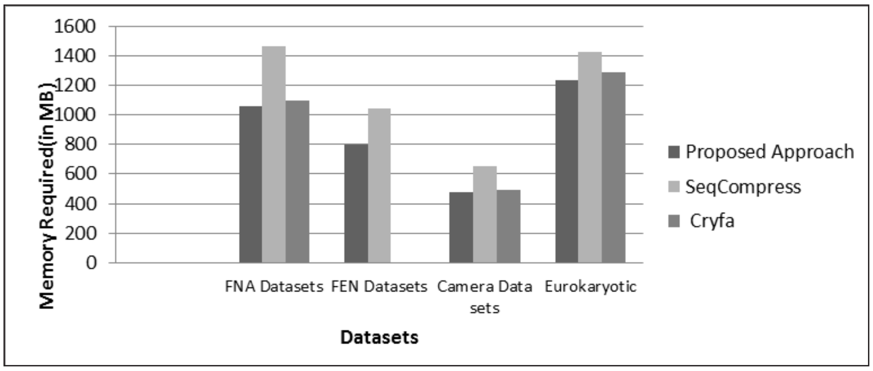


Figure 7. Memory required for SeqCompress, Cryfa and Propose method to securely and efficiently store various data sets (Memory in MB) (Cryfa does not support FEN dataset)

CONCLUSION AND FUTURE DIRECTION

In this paper, a new method to compress and encrypt the fasta/multifasta file is presented. The proposed method uses unified encryption approach (encryption during compression) to encrypt the genomic data. Here encryption is applied only on the permuted indices (0-255) created by MWBTC at the time of encoding instead of applying it on the whole genomic data. Experimental results show that proposed method has good compression ratio. Proposed algorithm also provides security of genomic data with very little computational effort. As a future work it can be extended for referential genome compression technique also.

REFERENCES

Bose, T., Mohammed, M. H., Dutta, A., & Mande, S. S. (2012). BIND–An algorithm for loss-less compression of nucleotide sequence data. *Journal of Biosciences, 37*(4), 785-789.

Chen, X., Li, M., Ma, B., & Tromp, J. (2002). DNACompress: fast and effective DNA sequence compression. *Bioinformatics, 18*(12), 1696-1698.

Danek, A., & Deorowicz, S. (2018). GTC: how to maintain huge genotype collections in a compressed form. *Bioinformatics, 34*(11), 1834-1840.

Dix, T. I., Powell, D. R., Allison, L., Yaeger, S., Bernal, J., & Stern, L. (2006). Exploring long DNA sequences by information content. In *Probabilistic Modeling and Machine Learning in Structural and Systems Biology* (pp. 97-102). Tuusula, Finland.

Dorok, S., Breß, S., Teubner, J., Läpple, H., Saake, G., & Markl, V. (2017). Efficient storage and analysis of genome data in databases. In, B. Mitschang, D. Nicklas, F. Leymann, H. Schöning, M. Herschel, J. Teubner, … & M. Wieland, (Eds.), *Datenbanksysteme für Business, Technologie und Web (BTW 2017)* (pp. 423-442). Gesellschaft für Informatik, Bonn.

Gupta, A., & Agarwal, S. (2008). A scheme that facilitates searching and partial decompression of textual documents. *International Journal of Advanced Computer Engineering, 1*(2), 99-109.

Hosseini, M., Pratas, D., & Pinho, A. J. (2016). A survey on data compression methods for biological sequences. *Information*, *7*(4), 56-78.

Jahaan, A., Ravi, D. R. T. N., & Arokiaraj, S. P. (2017). A Comparative Study and Survey on Existing DNA Compression Techniques. *International Journal of Advanced Research in Computer Science (IJARCS), 8*(3), 732-735.

Kumar, S., Agarwal, S., & Prasad, R. (2015, May). Efficient Read Alignment Using Burrows Wheeler Transform and Wavelet Tree. In *2015 Second International Conference on Advances in Computing and Communication Engineering (ICACCE)* (pp. 133-138). Dehradun, India.

Mahajan, P., & Sachdeva, A. (2013). A study of encryption algorithms AES, DES and RSA for security. *Global Journal of Computer Science and Technology, 13*(15), 14-22.

Mohammed, M. H., Dutta, A., Bose, T., Chadaram, S., & Mande, S. S. (2012). DELIMINATE—a fast and efficient method for loss-less compression of genomic sequences: sequence analysis. *Bioinformatics, 28*(19), 2527-2529.

Pinho, A. J., & Pratas, D. (2013). MFCompress: a compression tool for FASTA and multi-FASTA data. *Bioinformatics, 30*(1), 117-118.

Pratas, D., Hosseini, M., & Pinho, A. J. (2017, June). Cryfa: a tool to compact and encrypt FASTA files. In *International Conference on Practical Applications of Computational Biology & Bioinformatics* (pp. 305-312). Cham: Springer.

Sardaraz, M., Tahir, M., Ikram, A. A., & Bajwa, H. (2014). SeqCompress: An algorithm for biological sequence compression. *Genomics, 104*(4), 225-228.

Wandelt, S., Bux, M., & Leser, U. (2014). Trends in genome compression. *Current Bioinformatics, 9*(3), 315-326.

Wiese, L., Schmitt, A. O., & Gültas, M. (2018). Big Data Technologies for DNA Sequencing. In S. Sakr & A. Zomaya (Eds.), *Encyclopedia of Big Data Technology*. Cham: Springer.