Asian Journal of Information Technology 15 (18): 3449-3454, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Enhancing Security for Gnome Data Using Referential Compression with Symmetric Cryptography Scheme

[1]E. Wiselin Kiruba and [2]K. Ramar
[1]Department of Computer Science and Engineering, University VOC College of Engineering, Tuticorin, 628008 Tamil Nadu, India
[2]Department Computer Science and Engineering, Einstein College of Engineering, Tirunelveli, 627012 Tamil Nadu, India,

Abstract: Recently, Genome research focuses on sequencing novel genomes in sequencing technology. The cost to scalable storage, analyze and transmit the data is provide crucial challenge for genome sequencing in bio-medical application. So, it is very much essential to design efficient data compression techniques for biological sequencing data. The genome sequencing is foremost a claim to deal with the huge generated data. In this study, we proposed a Referential Compression with Symmetric Cryptographic Scheme (RCSCS) for dealing with both the issues storage demands and security. RCSCS reduces the storage demands of genome sequences with security and exploits the similarity between the genome sequences of the evolutionary species. RCSCS stores the compressed file which has differences between compressed and an identified reference sequence. It gives high compression rates than existing schemes. The proposed scheme reducing running time that allows partial decompression of the target genome in local regions of interest. The results show that our algorithm reduces the 3000 (MB) genome down to 6.87 MB for genome data with hg18 and 19 as a reference.

Key words: Genome data, referential compression, symmetric cryptographic, scalable, security

INTRODUCTION

While technology keeps growing the world keeps shrinking. Necessity of genome compression (Schatz *et al*., 2010) is playing a predominant role in the real vogue. Genome data can be classified into DNA and RNA textures which encoded by four literals of A, C, G and T. Due to the excessive storage of living organism in the public data bases like GenBank and EMBL their size is exponentially growing for ease of processing the data in the network and storage in the data base genetic compression striving into the world as a major concern. Many classical algorithms are fails to explain genetic sequences and compress effectively. Lossless compression (Bahadili, 2008) is very much essential for gnome data processing.

The main challenge for reference-based encoding is to find long matches efficiently. This can be done by indexing sequences either in the form of suffix trees or by using hash-based approaches. The growth of Next Generation Sequencing technologies (Sardaraz *et al*., 2014) presents significant research challenges, specifically to design bioinformatics tools that handle massive amount of data efficiently. Biological sequence data storage cost has become a noticeable proportion of total cost in the generation and analysis. Compression of genome data is significantly outstripping the rate of increase in disk storage capacity which may go beyond the limit of storage capacity. It is essential to develop algorithms that handle large genome data sets through optimal memory management. Secure management of the cryptographic keys is critically important, since the security and reliability of cryptographic processes depend upon the strength of the keys. The effectiveness of the protocols associated with the keys and the protection given to the keys. The proposed scheme used a security mechanism and compression technique is applied for enhancing security and solves the issue of storage demands. The objective of the proposed referential compression scheme is to construct an encoding and decoding scheme that minimizes the length of the representation. The proposed scheme also used a NONCE mechanism to generated key that is transmitted along with the message. The objective of the NONCE mechanism is to prevent message loss without loss of compression in genomic data that prevent the chosen plain text attack.

Corresponding Author: Wiselin Kiruba, Department of Computer Science and Engineering, University VOC College of Engineering, Tuticorin, 628008 Tamil Nadu, India

Literature: Now a days, a huge volume of Genome data is stored on storage devices. In order to reduce storage demand, it is very much essential to implement novel Genome compression techniques. Reese *et al.* (2010), Saada and Zhang (2015) compared two algorithms using the binary representation of DNA sequences. They applied the RLE technique to enhance the compression of entire genomes. Muhammad presents a DNA sequence compression algorithm Seq Compress that copes with the space complexity of biological sequences. The algorithm is based on lossless data compression and uses statistical model as well as arithmetic coding to compress DNA sequences. Research in Hanus *et al.* (2010) introduced statistical evolutionary models and prediction techniques from lossless binary image compression to reduce computation for Genome data sets. Madanayake *et al.* (2013) proposed Genpack technique that uses public 32 bit key for encoding and decoding process. Gen pack will work on both tandem repeats and non tandem repeats and observed that compression is 1.002 bits/characters. It is a finest technique among all existed ones and by using this technique data easily managed by substantially reducing its infrastructure in networks. Wandelt and Leser (2012, 2013) proposed a RE ferential Sequence COmpression (FRESCO) to compress large amounts of biological sequence data. They also proposed two techniques to increase compression ratios including selecting a good reference sequence and rewriting a reference sequence to allow for better compression. Ravindu presented an algorithm to compress a target genome given a known reference genome. The proposed algorithm first generates a mapping from the reference to the target genome and then compresses this mapping with an entropy coder.

Pinho *et al.* (2012) presented GReEn compression tool based on arithmetic coding to parse the target genome into factors each of which occurs in the reference sequence. Wang and Zhang (2011) presented a compression tool for storing and analyzing Genome Re-Sequencing data (GRS). GRS process the genome sequence data without the use of the reference single nucleotide polymorphisms and other sequence variation information and automatically rebuild the individual genome sequence data using the reference genome sequence. Heath *et al.* (2010) focused on compression of source sequence such as target genome given reference genome. It also provides an efficient manipulation and studying the tradeoff between the two. Ma *et al.* (2012), proposed a novel algorithm to compress the source sequence. It encodes the edits contained in runs of different extents separately. Hossein *et al.* (2010), Indran *et al.* (2011) proposed a web page compression technique help to reduce the space for storing and transmitting data also introduce one new technique along with exiting Huffman Technique of compression routine. Sebastian and Ulf Leser proposed an adaptive referential compression schemes for genomes. The compression speed controlled by varying the amount of main memory for the compressor.

MATERIALS AND METHODS

Proposed work: Before Recently, Genome research was focusing on sequencing new genomes in sequencing technology. It is very much essential to have re-sequencing of genomes that was increasing demands for translational medicine (Chin *et al.*, 2011). In this study, we proposed a referential compression technique for genome data in which all re-sequenced genomes are from the same species, the resulting sequences exhibit extremely high levels of similarity. The key plan of this scheme is to encode sequences with respect to an external reference sequences. The objective of the proposed referential compression scheme is to construct an encoding and decoding scheme that minimizes the length of the representation. The proposed scheme achieved high compression rates by long matches in the reference. The proposed referential compression algorithm is shown in Algorithm 1. The value of X determines whether short matches are encoded as a reference or as a raw string. In many implementations any match longer than two characters are encoded as a referential match.

Algorithm1; Input; Characters, reference set:

```
While characters do
Find longest matching substring in reference for current input position
If length of match >X then
Encode match as match position with length
Else
Encode match with raw symbols
End if
End while
```

In the proposed scheme, the encoding algorithm divided into two phases such as mapping and entropy coder. Initially, mapping is generated from the reference genome to the target genome. In the second phase, this mapping performed in lossless compression using an entropy coder. The mapping generation is performed by the sliding window Lempel-Ziv algorithm (Kuruppu *et al.*, 2011). The decoder then takes the compressed representation of this mapping and decompresses it, obtaining the mapping. It can recover reference genome perfectly by inverting the original mapping.

Figure 1 shows the referential compression algorithm for two interval matches. In any instance, interval match (Kuruppu *et al*., 2011; Sardaraz *et al*., 2014) indicates that the input matches the reference for four symbols starting at position seven. Then, a tiny sequence is stored as raw bases when there is no match in the reference sequence for GAT.

Algorithm 2 for mapping:

```
Input: Genome data, reference sequence
Initialize Instruction Set n1 = 0 and i = 1
While (not sequence $x_1^n$ is exhausted)
    Compute $F_i = F (n_i, W_i, L_i, R_i, x^8, y^8)$
    Represent the i-th idiom by $X_i+L_i+1$, $X_{ni}+1$ where $l_i = \max( F_i)$
    $p_i = \arg\min (F_i)$
    $z_i = x_{ni+li+1}$
    Generate Instruction Set $F_i = (p_i, l_i, z_i)$
    $n_{i+1} = n_i+l_i+1$
I = i + 1
Update $(W_i, L_i, R_i)$
End while
```

Where W_i is the slide window, L_i is the length of a symbol, R_i is the reference sequence, x^8 and y^8 is the position of the sequence. The algorithm is performed from a position in the target sequence. The aim of the algorithm is to find the largest length and symbol for window and then encode the position for finding best match. It also finds a set of instructions F that suffices to reconstruct the target genome based on the reference.

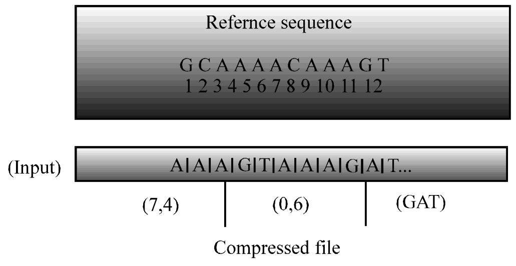

Fig. 1: Referential compression

security: The proposed scheme used a short nonce mechanism to generated key that is transmitted along with the message. The objective of the nonce mechanism is to prevent message loss without loss of compression in genomic data. In this, an encryption scheme is the counter mode of operation that uses a sequence number as a nonce. Counter mode allows a random access property during decryption. Counter mode is suitable to operate on a multi-processor machine where blocks can be encrypted in parallel. Furthermore, it does not suffer from the short-cycle problem. In order to reduce chosen plaintext attack, XOR the nonce and counter into a single value.

In this study, we design a modified integer arithmetic code (Huang and Liang, 2011) which owns the capabilities of compression and encryption simultaneously. The encoder algorithm is given below. It is mainly used to generate the key vector and encode the symbol based on the NONCE approach.

Algorithm 3 for encoder:

```
Initialize source interval to [0, W-1].
For i = 1 to N-1
Divide the source interval as NONCE integer
If KNV[i] = 1, Then
    Increase the size of the source interval based on symbol with the largest probability $(p_i)$
    If (size >1 and $p_i$<0)
Decrease the size
    End if
    End if
Encode the symbol $x_i$
Update the source interval according to the encoded symbol
End for
Key vector = KNV[i]
```

The encoding algorithm can be extended by the size of more source intervals that is increased by one. Figure 2 show that the proposed encoded algorithm used eight key bits at a time to establish the source intervals. Then, the symbol is encoded in which the size of the interval associated with the symbol is increased by one. The positions of all of the intervals are changed a little and so on.

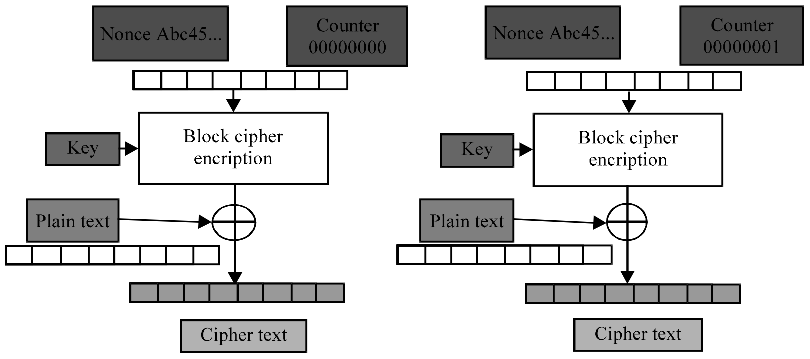

Fig. 2: Counter mode encryption

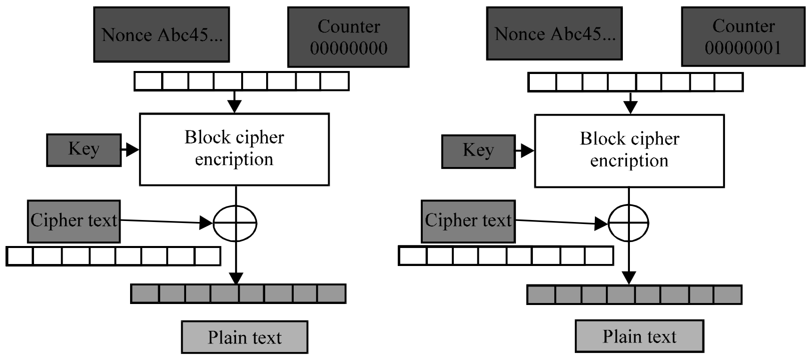

Fig.3: Counter mode decryption

Figure 3 show that it used nonce for generation key instead of random bit generator. In the decoder, once the secret key K is known, the key nonce vector KNV can be generated successfully. Then, the symbol sequence was decoded correctly by mimicking the encoder algorithm. In the proposed scheme, it is still difficult for the attacker to correctly find out those chosen symbols because XOR the nonce and counter into a single value and the source intervals are not static in every time. User was prevented to find the secret keys from the plaintext that the whole input symbol sequence is encrypted by repeatedly using NONCE key vector.

RESULTS AND DISCUSSION

The proposed RCSCS scheme was experimented with the real genomic data such as hg 18 and 19 (Human Genome version 18 and version 19). The hg 18 and 19 genome data is released from the UCSC Genome database. Table 1 shows some of sample size of the hg 18 and 19 genome data.

Table 2 show the best result of compression ratio of proposed RCSCS. The proposed scheme compressed genome data efficiently because it uses NONCE mechanism that also reduced the computation time.

Complexity analysis: The complexity analysis of encryption and decryption algorithm includes computation complexity and space complexity analysis. In proposed encoding, the NONCE LSB substitution replaces k1 bits of LSB of each pixel of gnome. Where k1 represents total number of eight bits replaced per pixel at a time constant (t1). The time complexity of encoding is (t1 × k1). Furthermore to access respected pixel of genome data in decoding approach there will be time complexity of n2 during encoding and another O (n2) during decoding. So total time complexity Tc is as given below,

$$Tc = O\{(t1 \times k1) + (t1 \times k1) + n2 + n2\} \quad (1)$$

Table 1: Sample size of the hg18 and 19 genome data

Character	Total size
Chr1	2.47E+08
Chr2	2.43E+08
Chr3	2E+08
Chr4	1.91E+08
Chr5	1.81E+08
Chr6	1.71E+08
Chr7	1.59E+08
Chr6_cox_hap1	4731698
Chr6_qbl_hap2	4565931
Chr17_random	2617613
Chr21_random	1679693
Chr9_random	1146434
Chr8_random	943810

Table 2: Compression results in MB of RCSCS and RBGC

Reference Genome	Target Genome	Size in MB	RBGC	RCSCS
Hg18	JW	2,991	6.99	5.09
hg18	YH	2,987	8.5	7.2
Hg19	JW	3,478	7.12	6.87
hg19	YH	3,085	9.67	8.37

Compressing ratio of genome data: The proposed RCSCS scheme obtained a compression ratio of about 790: 1 (3GB to 3.6MB) when compressing an entire genome data that used referential compression mechanism. The compression ratio drops to 750: 1 (3GB to 4.1MB) when compressing the entire genome as a single file. The proposed scheme takes less time for compressing the entire genome as a single file. RCSCS takes about 44 seconds to draw to a close the compression of an entire genome in a single file. The previous scheme requires more memory space to finish the execution. The proposed scheme performs well than 40% than existing scheme. The proposed approach manages to reduce the target genome down to 6.43 MB and performs well with the data like hg18 and 19. It also reduces the size of the compressed files by 50% more than existing scheme.

Security analysis: The results of the proposed RCSCS compared with DES and RBGC in terms of compression ratio encryption and decryption time.

Table 3: Compression ratio, encryption and decryption time

Parameters	DES	RBGC	RCSCS
Compression ratio	27.13	30.23	42.23
Encryption time (m sec)	567	489	457
Decryption time (m sec)	560	491	456

Table 3 shows the compression ratio, encryption and decryption time of proposed scheme with existing scheme. Compared with the effectiveness of other compression algorithms, the NONCE mechanism of the RCSCS gives better compression ratio in terms of encryption and decryption time. The proposed scheme provide better compression ratio about 42.23% than the related schemes. The proposed RCSCS takes less time for encryption and decryption because it uses NONCE mechanism to generate key vector. In SCMRP, data was protected by Encryption Key at different segments. It prevent the Byzantine attack. It also prevent the chose plain text attack because it is very difficult for the attacker to correctly find out those chosen symbols because XOR the nonce and counter into a single value and the source intervals are not static in every time.

CONCLUSION

In this study, we applied an efficient compression scheme named reference compression technique to genome data to solve the storage demands. It avoids reference sequence to increase the compression ratio. The proposed RCSCS scheme also enhanced the security by using symmetric cryptographic approach. In this, the counter mode of operation is used as an encryption scheme that uses a sequence number as a NONCE. Counter mode allows a random access property during decryption in parallel. This security mechanism reduces chosen-plaintext attack by XOR the nonce and counter into a single value. We exploit the redundancy and compress the genome from 3000 MB for genome data with hg18 and 19 release as the reference. The results show that the proposed RCSCS scheme provides better compression ratio as well as enhancing security.

REFERENCES

Bahadili, A.H., 2008. A novel lossless data compression scheme based on the error correcting Hamming codes. Comput. Math. Appl., 56: 143-150.

Chin, L., J.N. Andersen and P.A. Futreal, 2011. Cancer genomics: From discovery science to personalized medicine. Nat. Med., 17: 297-303.

Hanus, P., J. Dingel, G. Chalkidis and J. Hagenauer, 2010. Compression of whole genome alignments. IEEE. Trans. Inf. Theory, 56: 696-705.

Heath, L.S., A.P. Hou, H. Xia and L. Zhang, 2010. A genome compression algorithm supporting manipulation. Proc. LSS Comput. Syst. Bioinform Conf., 9: 38-49.

Hossein, M.S.M., A. Mukherjee and S. Ghosh, 2010. Notice of retraction webpage development for genome compression technique. Proceeding of the ICMLC 2010 2nd International Conference on Machine Learning and Computing, February 9-11, 2010,IEEE,Jhgargram,India,ISBN:978-1-4244-6007-6, pp: 282-287.

Huang, Y.M. and Y.C. Liang, 2011. A secure arithmetic coding algorithm based on integer implementation. Proceeding of the 2011 11th International Symposium on Communications and Information Technologies (ISCIT), October 12-14, 2011, IEEE, Nantou, Taiwan, ISBN:978-1-4577-1295-1, pp: 518-521.

Indran, I.R., G. Tufo, S. Pervaiz and C. Brenner, 2011. Recent advances in apoptosis, mitochondria and drug resistance in cancer cells. Biochimica Biophysica Acta, 1807: 735-745.

Kuruppu, S., S.J. Puglisi and J. Zobel, 2011. Optimized relative lempel-ziv compression of genomes. Proceedings of the 34th Conference on Australasian Computer Science, Vol. 113, January 17-20, 2011, Australian Computer Society Inc., Perth, Australia, ISBN: 978-1-920682-93-4, pp: 91-98.

Ma, N., K. Ramchandran and D. Tse, 2012. A compression algorithm using mis-aligned side-information. Proceedings of the 2012 IEEE International Symposium on Information Theory, July 1-6, 2012, IEEE, Berkeley, California, ISBN:978-1-4673-2579-0, pp: 16-20.

Madanayake, P.R.D., M.D.N.S. Peiris, G.H. Ranaweera, K.U.K.K. Jayathilake and A. Senarathne et al., 2013. Advanced encryption algorithm using fuzzy logic. Proceeding of the 2012 International Conference on Information and Computer Networks, July 1-11, 2013, Sliit, Kotte, Sri Lanka, pp: 32-36.

Pinho, A.J., D. Pratas and S.P. Garcia, 2012. GReEn: A tool for efficient compression of genome resequencing data. Nucleic Acids Res., 40: e27-e27.

Reese, M.G., B. Moore, C. Batchelor, F. Salas and F. Cunningham et al., 2010. A standard variation file format for human genome sequences. Genome Biol., 11: R1-R88.

Saada, B. and J. Zhang, 2015. DNA sequences compression algorithms based on the two bits codation method. Proceeding of the 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), November 9-12, 2015, IEEE, China, East Asia, ISBN: 978-1-4673-6799-8, pp: 1684-1686.

Sardaraz, M., M. Tahir, A.A. Ikram and H. Bajwa, 2014. Seq compress: An algorithm for biological sequence compression. Genomics, 104: 225-228.

Schatz, M.C., B. Langmead S.L. Salzberg, 2010. Cloud computing and the DNA data race. Nat. Biotechnology, 28: 691-693.

Wandelt, S. and U. Leser, 2012. Adaptive efficient compression of genomes. Algorithms Mol. Biol., 7: 1-1.

Wandelt, S. and U. Leser, 2013. FRESCO: Referential compression of highly similar sequences. IEEE. ACM. Trans. Comput. Biol. Bioinf., 10: 1275-1288.

Wang, C. and D. Zhang, 2011. A novel compression tool for efficient storage of genome resequencing data. Nucleic Acids Res., 39: e45-e45.