Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 79 (2015) 30528-30531

ISSN: 2229-712X

Implementation of new secured data compression technique using huffman code and symmetric key algorithm

Prakash Kuppuswamy[1] and Saeed Q Y Al-Khalidi[2]

[1]Department of Computer Engineering & Networks, JAZAN University, KSA.

[2]Library Affairs, King Khalid University, KSA.

ARTICLE INFO

Article history:

Received: 10 September 2013;

Received in revised form:

29 January 2015;

Accepted: 19 February 2015;

Keywords

Data compression,
Encryption,
Decryption,
Symmetric,
Modulation.

ABSTRACT

Data compression is a common requirement for most of the computerized applications. There are many number of unsecured data compression algorithms, which are dedicated to compress different unsecured data formats. Even for a single data type there are number of different compression algorithms, which use different approaches. In this research, we propose a simple and efficient data compression algorithm particularly suited to be used on available commercial basis using secured manner. Our intention is transmitting text data in secured as well as compressed in the open environment. It is using double compression technique based on Huffman coding algorithm and simple symmetric key algorithm. Experiment itself evaluates the performance of new secured data compression algorithm with other data compression algorithm.

Introduction

Compression technique used for the purpose of utilization of storage space is important even with today's huge storage volumes [8]. Data compression has been adopted in hardware designs to improve performance and power. Cache compression increases the cache capacity by compressing block data and accommodating more blocks in a fixed space [5], [6]. It is the art of representing the information in a compact form rather than its original or uncompressed form [9]. In other words, using the data compression, the size of a particular file can be reduced. This is very useful when processing, storing or transferring a huge file, which needs lots of resources. If the algorithms used to encrypt works properly, there should be a significant difference between the original file and the compressed file. When data compression is used in a data transmission application, speed is the primary goal. Speed of transmission depends upon the number of bits sent, the time required for the encoder to generate the coded message and the time required for the decoder to recover the original collection. In a data storage application, the degree of compression is the primary concern [1]. Various lossless data compression algorithms have been proposed and used. Some of the main techniques in use are the Huffman Coding, Run Length Encoding, Arithmetic Encoding and Dictionary Based Encoding [3].

Symmetric or secret key cryptography, a single key is used for both encryption and decryption. Sender uses the key using some set of rules to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key or rule set to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric key algorithm. The biggest difficulty with this approach, of course, is the distribution of the key [2].

This algorithm is shown to be the best solution currently available in all situations, including archivers, distribution, and on-line compression such as disk compression or network datagram compression [7].

Related Works

David A. Huffman in the year 1952 proposed an Encoding Algorithms use the probability distribution of the alphabet of the source to develop the code words for symbols. The frequency distribution of all the characters of the source is calculated in order to calculate the probability distribution. According to the probabilities, the code words are assigned. Shorter code words for higher probabilities and longer code words for smaller probabilities are assigned. For this task a binary tree is created using the symbols as leaves according to their probabilities and paths of those are taken as the code words. Two families of Huffman Encoding have been proposed: Static Huffman Algorithms and Adaptive Huffman Algorithms. Static Huffman Algorithms calculate the frequencies first and then generate a common tree for both the compression and decompression processes [2]. Details of this tree should be saved or transferred with the compressed file. The Adaptive Huffman algorithms develop the tree while calculating the frequencies and there will be two trees in both the processes. In this approach, a tree is generated with the flag symbol in the beginning and is updated as the next symbol is read.

Glen G. Langdon, Jr. in 1984 discussed about An Introduction to Arithmetic Coding. In this paper presents the key notions of arithmetic compression coding by means of simple examples Arithmetic coding is a data compression technique that encodes data (the data string) by creating a code string which represents a fractional value on the number line between 0 and 1. The coding algorithm is symbol wise recursive; i.e., it operates upon and encodes (decodes) one data symbol per iteration or recursion. On each recursion, the algorithm successively partitions an interval of the number line between 0 and I , and retains one of the partitions as the new interval. Thus, the algorithm successively deals with smaller intervals, and the code string, viewed as a magnitude, lies in each of the nested

Tele:
E-mail addresses: varshiniprakash@rediffmail.com

intervals. The data string is recovered by using magnitude comparisons on the code string to recreate how the encoder must have successively partitioned and retained each nested subinterval. Arithmetic coding differs considerably from the more familiar compression coding techniques, such as prefix (Huffman) codes. Also, it should not be confused with error control coding, whose object is to detect and correct errors in computer operations. [2]

S.R. Kodituwakku, U. S.Amarasinghe, An experimental comparison of a number of different lossless compression algorithms for text data is carried out. Several existing lossless compression methods are compared for their effectiveness. Although they are tested on different type of files, the main interest is on different test patterns. By considering the compression times, decompression times and saving percentages of all the algorithms, the Shannon Fano algorithm can be considered as the most efficient algorithm among the selected ones. Those values of this algorithm are in an acceptable range and it shows better results for the large files [1].

Prakash Kuppuswamy, Dr. Saeed Q Y Al-Khalidi proposed Implementation of security through simple symmetric key algorithm based on modulo 37 in October 2012 proposed new symmetric key algorithm. Encryption and key generation became a vital tool for preventing the threats to data sharing and tool to preserve the data integrity so we are focusing on security enhancing by enhancing the level of encryption in network. This study's main goal is to reflect the importance of security in network and provide the better encryption technique for currently implemented encryption techniques in simple and powerful method. In our research we have proposed a modular 37 and select any number and calculate inverse of the selected integer using modular 37. The symmetric key distribution should be done in the secured manner. Also, we examine the performance of our new SSK algorithm with other existing symmetric key algorithm.[10]

Proposed Technique

One of the effective tools for ensuring the safety of compressed data transactions is the secured encryption techniques. It combines the Huffman encoding technique and simple symmetric algorithm. The proposed method of data compression technique focuses on the data confidentiality issue. Although security mechanisms, this method is very easy to adopt the coding of bulk and more compressed secured data. Also it is very safe enough on the other side. The tools for designing methods were as follows

a. Huffman Code

Huffman Code assigns shorter encodings to elements with a high frequency, F:e. It differs from block encoding in that it is able to assign codes of different bit lengths to different elements. Elements with the highest frequency, F:e, get assigned the shortest bit length code. The key to decompressing huffman code is a huffman tree.

b.Huffman tree

A huffman tree is a special binary tree called a trie. A binary trie is a binary tree in which a 0 represents a left branch and a 1 represents a right branch. The numbers on the nodes of the binary trie represent the total frequency, F, of the tree below. The leaves of the trie represent the elements, e, to be encoded. The elements are assigned the encoding which corresponds to their place in the binary trie.

c. Inverse function

An inverse of a matrix, usually written as $f^{-1}(x)$, is a reflection of the original function, f(x), around the line y = x. Basically, every x value is changed to a y value and every y value is change to an x value.

d. Modular Arithmetic

Modular arithmetic over a number 'n' involves arithmetic operations on integers between 0 and n – 1, where n is called the modulus. If the number happens to be out of this range in any of the operation the result, r, is wrapped around in to the range 0 and n – 1 by repeated subtraction of the modulus n from the result r. This is equivalent in taking the remainder of division operation r/n.

e. Selecting random positive and negative integer

The reason for selecting the random positive and negative integer to send the data compressed and secured. The random integer should satisfy ($1 \leq x \leq 37$) because we need inverse of the selected random integer for the purpose of decryption technique.

Encoding sequence

Step 1: Find out the element frequency from the given message
Step 2: Assign Huffman code
Step 3: Assign decimal value for the Huffman code
Step 4: Assign n=37 (prime number)
Step 5: Take random positive integer which satisfy mod 37;$(x*x^{-1})=1$
Step 6: Again take random negative integer for more securing
Step 7: multiply with the decimal value and selected positive, negative numbers
Step 8: Use mod 37
Step 9: Use again Huffman frequency code
Step 10: Now derived code is secured encoded message

Table 1. Uncompressed data

Data	D	A	D	B	A	D	C	A	B	C	A	F	E
Integer Value	4	1	4	2	1	4	3	1	2	3	1	6	5
Binary Value	100	001	100	010	001	100	011	001	010	011	001	110	101

Observed from the above table binary equivalent value is 39 bits. Now we are applying Huffman compression technic.

Table 2. Huffman compressed data

Text	No. of frequency	Huffman code	Total bits
A	4	10	8 Bits
B	2	000	6 Bits
C	2	001	6 Bits
D	3	01	6 Bits
E	1	111	3 Bits
F	1	110	3 Bits
			32 Bits

Table 3. Secured compressed data technique

Text	Huffman code Binary	Equivalent Integer value	Multiplying with Random positive integer Assume '3'and negative integer -8 using mod 37	Again using Huffman code
D	01	1	13	10
A	10	2	26	01
D	01	1	13	10
B	010	0	0	111
A	10	2	26	01
D	01	1	13	10
C	011	1	13	10
A	10	2	26	01
B	010	0	0	111
C	011	1	13	10
A	10	1	26	01
F	110	6	13	10
E	101	7	26	01
			Total secured data bits	30 Bits

Implementation

In order to provide quick and simple data compression/decompression, the bits size of the secret key has to be chosen effectively. For compression small amount of data, there should not be any overhead to the encrypting system as well as there should not be any compromise on the security level. Thus an optimized size of data "DAD BAD CAB CAFE" is chosen for experiment.

Table 4. Comparison of Un compressed, Huffman code, Secured compressing technique

Data	Equivalent Integer value	Binary digit	No of Frequency Occurs	Total Huffman	Decimal value	Multiply with decimal value (3*-8)%37	Use Huff
D	4	100	3 Times-01	6bits	1	13	10
A	1	001	4 Times-10	8bits	2	26	01
D	4	100			1	13	10
B	2	010	2Times-000	6bits	0	0	111
A	1	001			2	26	01
D	4	100			1	13	10
C	3	011	2Times-001	6bits	1	13	10
A	1	001			2	26	01
B	2	010			0	0	111
C	3	011			1	13	10
A	1	001			1	26	01
F	6	110	1Time-110	3bits	6	4	001
E	5	101	1Time-111	3bits	7	17	110
		39 Bits		32 Bits			30 Bits

Result Analysis

The proposed method of Data compression technique is the combination of the Huffman coding and symmetric key algorithm. More number of data transferring daily across the world. All the data transaction is not secured. Some of the data transferring method searching for secured transaction using various cryptography and data security algorithm. The other methods, looking for the new compression technique for bulk data transaction. This proposed new method of secured data compression technique, which will satisfy both the type of user.

The algorithm executes on PC computer of CPU Intel Pentium 4, 2.2 MHz Dual Core. The programs implemented using Microsoft Visual Studio 2008 (C#). It is tested with three messages and with different in length (1000, 2000, 3000 characters).

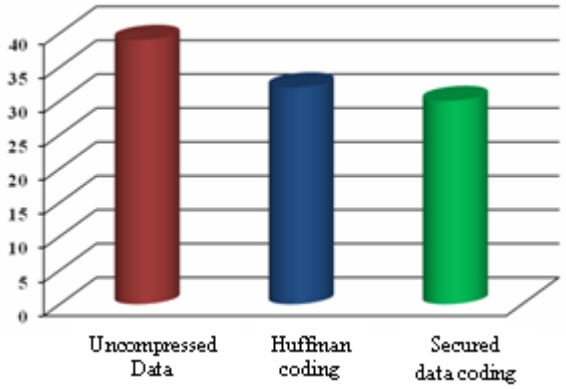

Figure 1. Comparison chart

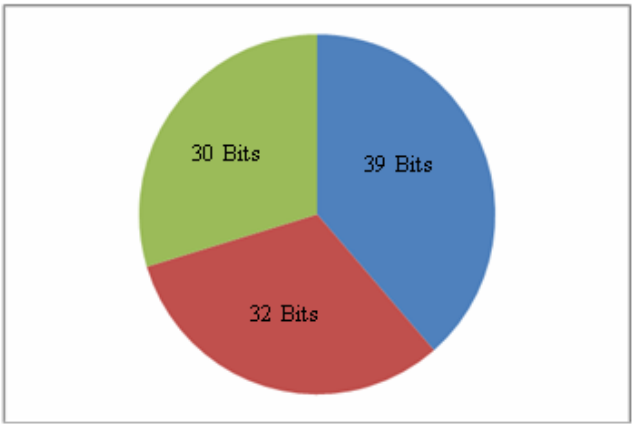

Figure 2. Average performance of data compression

Table 5.Comparison of execution timing

No. of Bits	Uncompressed Data (Un secured)	Huffman Coding (Un secured)	Secured data compressing
	Total execution timings		
1000	1mts 11 Sec	1 mts 0 Sec	1mts 20 Sec
2000	2mts 20 Sec	2 mts 0 Sec	2 mts 30Sec
3000	3mts 30 Sec	3 mts 0 Sec	3 mts 50 Sec

Conclusion

It has been clear that the result of our "new proposed technique" is better result producing as compared normal and Huffman coding. It is new technic of compressing data with secured manner. It is essential to achieve few goals like confidentiality and integration across the data transaction between the medium. The proposed compression technique is very simple in nature and there are two compressing methods present in this compression algorithm. So, It would make it more secured. For large amount of data transaction and commercial communication purpose this algorithm will work very smoothly. The proposed compression technique wouldn't be cost effective since those are not designed for large amount of data in minimal cost.

References

[1] S.R. Kodituwakku, U. S.Amarasinghe, "Indian Journal of Computer Science and Engineering", Vol.1 No 4 416-425, ISSN : 0976-5166.

[2] Glen G. Langdon, Jr. "An Introduction to Arithmetic Coding", IBM J. RES. DEVELOP, VOL. 28, NO. 2 MARCH 1984.

[3] Kesheng, W.J. Otoo and S. Arie, "Optimizing bitmap indices with efficient compression", ACM Trans. Database Systems, 31: 1-38.2006.

[4] Blelloch, E., Introduction to Data Compression, Computer Science Department, Carnegie Mellon University. 2002.

[5] A. R. Alameldeen and D. A. Wood, "Adaptive Cache Compression for High- Performance Processors" in Proceedings of ISCA, pp. 212–223, 2004.

[6] E. G. Hallnor and S. K. Reinhardt, "A Unified Compressed Memory Hierarchy", in Proceedings of HPCA, pp. 201–212, 2005.

[7] Charles Bloom, "a new data compression algorithm", cbloom@mail.utexas.edu.

[8] Peter A. James, "Data Compression for process historians", Chevron Research and Technology Company, Richmond, CA 94802-0627, 1995.

[9] Paul, I.M, "Fundamental Data Compression", 2006 Elsevier, Britain.

[10] Prakash Kuppuswamy, Dr. Saeed Q Y Al-Khalidi, "Implementation of security through simple symmetric key

algorithm based on modulo 37", IJOCT, ISSN: 2277-3061, Volume 3 No. 2, OCT, 2012.
[11]Herbert Edelsbrunner, LZW Data Compression, last modified: Feb 2004 http://www.cs.duke.edu
[12] Huffman, D.A., A method for the construction of minimum redundancy codes. Proc. IRE, Vol. 40, pp. 1098-1101, Sept. 1952.
[13] Jaradat, A. R. and Irshid, M. I., A Simple Binary Run - Length Compression Technique For Non-Binary Source Based on Source Mapping. Active and Passive Elec. Comp., 2001, Vol. 24, pp. 211 – 221.
[14] Kumar B., Point4: Working with data and Graphical Algorithms in C, c Reference Point Suite, skillsoft 2002.
[15] Lenat, Doug, Lempel-Ziv compression, 1999 http://foldoc.doc.ic.ac.uk
[16] Mark Nelson, LZW Data Compression, Dr. Dobb's Journal, October 1989 www.dogma.net
[17]Matt Powell, University of Canterbury, last updated November 20, 2001 http://corpus.canterbury.ac.nz

Prakash Kuppuswamy Lecturer, Computer Engineering & Networks Department in Jazan University, KSA He is research Scholar-Doctorate Degree yet to be awarded by 'Dravidian University'. He has published 12 International Research journals/Technical papers and participated in many international conferences in Rep. of Maldives, Libya and Ethiopia. His research area includes Cryptography, Bio-informatics and Network algorithms.

Dr. Saeed Q. Y. Al-Khalidi, Dean of Libraries Affairs at King Khalid University, Abha. KSA. He published many National & International papers, Journals. Also, he participated as a Reviewer in many international conferences worldwide. He completed Master Degree and Doctor of Philosophy in University of East Anglia. His research interests include: Information System development, approaches to systems analysis and the early stages of systems development process, IT/IS evaluation practices, E-readiness assessment.