Research Journal of Applied Science, Engineering and Technology 3(7): 625-632, 2011
ISSN: 2040-7467

Received: May 07, 2011 Accepted: June 10, 2011 Published: July 25, 2011

Securing Messages in Wireless Channel by Using New Method of Compression

Ahmed Chalak Shakir, Gu Xuemai and Jia Min
Harbin Institute of Technology, School of Electronics and Information Technology,
Harbin, 150001, China

Abstract: This study attempts to secure the message by using the new method of compression in which the form of the message is completely changed and it is protected against the passive attack so that the sender and receiver of the information are only the hosts who can understand the encoding scheme. In unsecured channels like wireless, the messages are vulnerable to be read by the unauthorized third parity person, for that reason they must be protected. The method can be used for any language but the concentration of the paper is on the security of Chinese message, so that it using Unicode and needs so many bits to be represented in the computer. The Compression is not only useful because it helps for reducing the consumption of expensive resources, such as hard disk space or transmission bandwidth, especially the message that will be send is always saved as the Rich Text file format that has so many overheads and increases the size of original message, but it also used for encoding the message which is exploited for the security.

Key words: Email protocols, lossless compression, rich text file format, Unicode

INTRODUCTION

In the last decade we have been witnessing a transformation_some call it a revolution, in the way we communicate, and the process is still under the way. This transformation includes ever present, ever growing Internet, the explosive development of mobile communications; and ever increasing importance of video communication. Data compression is one of the enabling technologies for each of these aspects of the multimedia revolution, (Sayood, 2000).

The compression algorithms can be classified broadly in two categories. Lossless algorithms: which can reconstruct the original message exactly from the compressed message, and lossy algorithms which can only reconstruct an approximation of the original message.Lossless algorithms are typically used for text, and lossy for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable (Umesh and Trivedi, 2011).

Data compression is often referred to as coding, where coding is a very general term encompassing any special representation of data which satisfies a given need. Information theory is defined to be the study of efficient coding and its consequences, in the form of speed of transmission and probability of error. Data compression may be viewed as a branch of information theory in which the primary objective is to minimize the amount of data to be transmitted. The purpose of this paper is to present and analyze new data compression algorithm and applying this algorithm to the email message written in Chinese language.

A simple characterization of data compression is that it involves transforming a string of characters in some representation (such as Unicode) into a new string (i.e., of bits) which contains the same information but whose length is as small as possible (Debra *et al*., 1987).

Data compression is particularly useful in communications because it enables devices to transmit or store the same amount of data in fewer bits. There are a variety of data compression techniques, but only a few have been standardized. In many cases, only some parts of the exchanged information are really useful. Thus, in these cases, information can be compressed before being transmitted and/or stored. This compression enables to reduce the processing and transfer time as well as the storage capacity needed for data (Anne *et al*., 2008).

Typically, e-mail messages are moved across the Internet using the Simple Mail Transfer Protocol (SMTP) which utilizes the connection-oriented Transmission Control Protocol (TCP) to establish the connections between two mail servers. The POP3 (Post Office Protocol) is used to retrieve mails for individual users from a server (Nand and Tong, 2002). The designed and implemented e-mail system which contains embedded data compression mechanisms that can be used for any text written in any language is produced. The SMTP protocol is extended to allow for the mail client and server to negotiate compression which is transparent to the users. The Unicode is used for representing the Chinese stroke which requires 16-bit for each stroke representation.

Corresponding Author: Ahmed Chalak Shakir, Harbin Institute of Technology, School of Electronics and Information Technology, Harbin, 150001, China

MATERIALS AND METHODS

The system design is based on the Unicode manipulation and processing regardless of the language used. Due to the using of the Unicode, so that the email message can be written in any language, like Chinese or Arabic languages, but in our study the concentration is on the email written in Chinese language. Notice that any email written in any language will be saved and send as a Rich Text File format in which automatically represent each character using 8-bit. For that reason the size of the file totally is increased because it has so many overhead.

Generating the table of characters: Let firstly name all the characters, numbers, and symbols etc. before compression for simplicity as *M* which is an array contains *K* elements. And name the table of *M* as an encoding message table.The algorithm of compression and decompression is based on the generating the table of *M* which is using and reading the message without redundancy. The proposed system or program will give each element *K* in the *M* the index starting from zero including the space.

Note: the system is used for the Rich Text file contents as whole, but for clarification, the message without the overhead of the rich Text format is taken.

Determining the number of bits: After the construction of the encoding message table, the number of required bits that represents the index number is determined as follows:

Let *n* be the number of required bits, then $2^n \geq$ the number of *M* in the encoding message table. For example if the table contains 24 elements, then $2^5 \geq 24$, so in this example the number of required bits =5.

Conversion to the binary system: The number that represents each *K* in *M* as index is converted to binary number from the decimal. The number of bits of these binary numbers must be equal to the number of bits. So each *M* has the same number of bits to represent its index number produced in section of (Generating the table of characters). The number of bits that represents each *K* can be seen as small as it was in the original, prior of generating the encoding message table and giving the index.

Recompensing *M*: Each *K* member is recompensedby its corresponding binary number so that each *K* member in *M* has its own binary representation.

Bit stream balancing: The total length of bits for all members of *M* must be divisible by 8 without reminder for balancing the stream of binary. That means, the number of bit stream/8 = integer number. If it does not equal to integer number then zeros (pad) in the right of the final bit in the stream bits of *M* elements must be added. To determine the number of required zeros to be added for balancing, the following pseudo code must be used:

```
If BinaryStringLength mod 8! = 0
then
Pad = 8- BinaryStringLength mod 8
else
Pad = 0
```

In example, let the *Binary String Length* =132, 132 mod 8! = 0 then the Pad (no. of zeros to be added) = 8-(132 mod 8) = 4. So, four zeros must be add.

Determining the size of the message: Prior of computing the size of the message, the most important note must be mentioned is: We cannot determine the size of the message exactly because we don't know the size of the overhead so that the file as a whole (original message with its Rich Text File format overhead) is compressed. But in the following example the message without overhead is supposed to clarify the algorithm.

The form of the message is always a collection of the sequences of bits, their values are from 0 to 255 and divided into the bytes. These sequences of bits are for:

- Encoding message table
- 8 bits for the pad (balancing bits)
- The compressed message

Let the number of bits that represent the index in the encoding message table = X. The number of bits that are used to represent the stroke (Chinese character) =16 bits for each stroke (Unicode).Note, the meaning of the characters in our paper may be the stroke of Chinese language or it may be any other character like (“).
Let *Y* be the number of bits for the compressed message. Then the size of the message after the compression = encoding message table size + 8 (pre-saved for the pad)+ *Y*.

Compression ratio = (The size of the message after the compression - the size of the message before the compression) / the size of the message before the compression * 100%

RESULTS AND DISCUSSION

The results of the algorithm can be illustrated by the following example:

Consider the following text (email message) which is in plain text format and used for explaining the algorithm only written in Chinese language:

“哈尔滨工业大学（哈工大）成立于 1920 年。89-年后货柜码头已经发展成为著名的多学科与科学工程和研究型大学为核心的大型国籍。哈工大是全国九个中国重点大学之一。”

First of all the encoding message table as shown in Table 1 must be generated without redundancy.

The number of *M* in the Table 1 is from 0 to 57 which means there are58 elements. So, the number of bits needed = 6, $2^6 \geq 58$

The Unicode, which requires 16 bits to represent each *K* in the example, be represented by 6 bits.

The size of the text before compression = number of characters in the text*number of bits that represents each character.

83*16=1328 bits

Note:

- All the binary numbers must have the same number of bits, the zeros are added to the left of the last bit because the maximum number is 57 and it requires 6-bit to be represented as binary number (3 in instance is represented as 000011à 6 bits, not as 11à 2 bits).
- The email that will be sending is in Rich Text format which has text overhead containing many attributes. But we take the plain text email message in the paper as an example and for explaining the algorithm only.
- The greater the size of the message leads to increase the compression ratio.

Binary representation of the original text: Each character in the original text is compensate by the corresponding 6-bit gotten from Table 1.

After each character is represented by 6-bit binary number, and in order to make these numbers in a form that can be stored in the computer; the following must be done:

Let *N* be the name of the binary stream that is gotten from the Table 2 and must be re-arranged in the form of 8-bits as follows:

00000000||00010000||10000011||00010000||01010001||10000111||00100000||00010001||00000110||00100100||10100010||11001100||00110100||11100011||11010000||01000101||00100100||11010100||01010101||00010101||10101111||01100001||10010110||10011011||01110001||11010111||10011111||00101010||00001000||01100010||10001110||01000001||11100101||10011010||01010001||11010111||00010010||01111010||00101001||10101010||10110001||10000111||10000010||11001011||01100011||00011010||10111011||10101111||01001000||00010001||00000110||11000011||00011011||10110010||11001111||01001011||10110101||11011000||01100001||11110111||11100001||00101110||01000000||.

Note: the characters || are used only for clarification and splitting each 8-bit. And the pad =6.

Converting the numbers from binary to decimal: In this section, *N* must be converted to decimal system as follows:

0||16||131||16||81||135||32||17||6||36||162||204||52||227||208||69||36||212||85||21||175||97||150||155||113||215||159||42||8||98||142||65||229||154||81||215||18||122||41||170||177||135||130||203||99||26||187||175||72||17||6||195||27||178||207||75||181||216||97||247||225||46||64|| .

Getting the compressed text: The text file of format UTF-8 is generated by the program for saving the decimal numbers as follows:

- *M* element’s from the Table1 are written in the first line of the file, and as follows:

“哈尔滨工业大学（ ）成立于 1920 年。89-后 space 货柜码头已经发展为著名的多科与程和研究型核心国籍是全九个中重点之一”

- Enter the new line.
- Pad with the array of generated decimal numbers

So, the final compressed text that is to be send has the following parts:

Part 1:

“哈尔滨工业大学（ ）成立于1920 年。89-后研究型核心国籍是全九个中重点之一”

Part 2: New line: for splitting the table from the text code.

Table 1: Encoding message

Index (decimal)	M(K)	Index (Binary)	Index (decimal)	M(K)	Index(Binary)
0	“	000000	29	经	011101
1	哈	000001	30	发	011110
2	尔	000010	31	展	011111
3	滨	000011	32	为	100000
4	工	000100	33	著	100001
5	业	000101	34	名	100010
6	大	000110	35	的	100011
7	学	000111	36	多	100100
8	(	001000	37	科	100101
9	)	001001	38	与	100110
10	成	001010	39	程	100111
11	立	001011	40	和	101000
12	于	001100	41	研	101001
13	1	001101	42	究	101010
14	9	001110	43	型	101011
15	2	001111	44	核	101100
16	0	010000	45	心	101101
17	年	010001	46	国	101110
18	。	010010	47	籍	101111
19	8	010011	48	是	110000
20	9	010100	49	全	110001
21	-	010101	50	九	110010
22	后	010110	51	个	110011
23	space	010111	52	中	110100
24	货	011000	53	重	110101
25	柜	011001	54	点	110110
26	码	011010	55	之	110111
27	头	011011	56	一	111000
28	已	011100	57	”	111001

Table 2: Binary representation of Chinese characters according to the example

“	哈	尔	滨	工	业	大	学	(	哈
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	1	1	1	0	0
0	0	1	1	0	0	1	1	0	0
0	1	0	1	0	1	0	1	0	1

工	)	大	成	立	于	1	9	2	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	1	1	1	1	1	1	1	0
1	1	0	0	0	1	1	1	1	0
0	1	0	1	1	0	0	1	1	0
0	0	1	0	1	0	1	0	1	0

年	∘	8	9	-	年	后		货	柜
0	0	0	0	0	0	0	1	0	0
1	1	1	1	1	1	1	0	1	1
0	0	0	0	0	0	0	1	1	1
0	0	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0
1	0	1	0	1	1	0	1	0	1

码	头	已	经	发	展	成	为	著	名
0	0	0	0	0	0	0	1	1	1
1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0	0
1	1	0	0	1	1	1	0	0	1
0	1	0	1	0	1	0	0	1	0

的	多	学	科	与	科	学		工	程
1	1	0	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	1	0	1
1	0	1	1	0	1	1	1	0	1

和	研	究	型	大	学	为	核	心	的
1	1	1	1	0	0	1	1	1	1
0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	1	0
0	0	0	0	1	1	0	1	1	0
0	0	1	1	1	1	0	0	0	1
0	1	0	1	0	1	0	0	1	1

Table 2: (Continued)

大	型	国	籍	。	哈	工	大	是	全
0	1	1	1	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	1
0	1	1	1	0	0	0	0	0	0
1	0	1	1	0	0	1	1	0	0
1	1	1	1	1	0	0	1	0	0
0	1	0	1	0	1	0	0	0	1

国	九	个	中	国	重	点	大	学	之
1	1	1	1	1	1	1	0	0	1
0	1	1	1	0	1	1	0	0	1
1	0	0	0	1	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1
1	1	1	0	1	0	1	1	1	1
0	0	1	0	0	1	0	0	1	1

–	o	”
1	0	1
1	1	1
1	0	1
0	0	0
0	1	0
0	0	1

Part 3: â œå“ˆå°”æ»¨å·¥ä¸šå¤§å|ï¼ˆï¼‰æˆ ç«‹äºŽ1920å¹´ã ,8å Žè´§æŸœç å¤´å·²ç» å ‘å±•ä¸ºè‘å çš„å¤šç§‘ä¸Žç¨‹å’Œç ”ç©¶åž‹æ ¸å¿ƒå›½ç± æ˜¯å…¨ä¹ ä¸ªä¸é‡‡ ç‚¹ä¹‹ä¸ â

Notice that the word “space” in the Table 1 is written only for calcification, in reality it is empty.

Determining the size of compressed file:

Size of *M* in Table 1 =
no. of characters in the table *
no. of bits needed for each character, in the example:

Size of *M* in Table 1 =
58 * 16 bytes (Unicode) = 928 bits.

Number of bytes needed for new line (\n\r) which splits the table from the text code = 3 bytes = 24 bits

Number of bits used for storing the pad =
8 bits and its location is in the start of the text code to be easy for retrieving the original text and knowing the number of exceeding bits.

Size of the text before the compression =
number of character * number of bits used for representing each character.

Size of the text after compression =
928+(6 for pad)+24(for new line)+
8(used for saving 6-bit pad)=966.

In the example 6 zeros are added for pad and saved as 00000110

Applying the algorithm on a Rich Text File format message used in the email: The algorithm is applied now on the same Chinese text but in the format of rich text not plain text. The result will be shown in the Fig. 1 which is the interface of the program written in C# programming language (Weldon, 2007).

As shown in the Fig. 1, the Chinese text is translated to the Rich Text File format. The whole message is:

{\rtf1\fbidis\ansi\ansicpg1256\deff0\deflang2049{\fonttbl{\f0\fswiss\fprq2\fcharset0Calibri;}{\f1\fnil\fprq2\fcharset134SimSun;}{\f2\fnil\fcharset0SegoeUI;}}\viewkind4\uc1\pard\ltrpar\lang1033\b\f0\fs22\ldblquote\f1\'b9\'fe\'b6\'fb\'b1\'f5\'b9\'a4\'d2\'b5\'b4\'f3\'d1\'a7\'a3\'a8\'b9\'fe\'b9\'a4\'b4\'f3\'a3\'a9\'b3\'c9\'c1\'a2\'d3\'da\f01920\f1\'c4\'ea\'a1\'a3\f089\f2-f1\'c4\'ea\'ba\'f3\f0\f1\'bb\'f5\'b9\'f1\'c2\'eb\'cd\'b7\'d2\'d1\'be\'ad\'b7\'a2\'d5\'b9\'b3\'c9\'ce\'aa\'d6\'f8\'c3\'fb\'b5\'c4\'b6\'e0\'d1\'a7\'bf\'c6\'d3\'eb\'bf\'c6\'d1\'a7\f0\f1\'b9\'a4\'b3\'cc\'ba\'cd\'d1\'d0\'be\'bf\'d0\'cd\'b4\'f3\'d1\'a7\'ce\'aa\'ba\'cb\'d0\'c4\'b5\'c4\'b4\'f3\'d0\'cd\'b9\'fa\'bc\'ae\'a1\'a3\'b9\'fe\'b9\'a4\'b4\'f3\'ca\'c7\'c8\'ab\'b9\'fa\'be\'c5\'b8\'f6\'d6\'d0\'b9\'fa\'d6\'d8\'b5\'e3\'b4\'f3\'d1\'a7\'d6\'ae\'d2\'bb\'a1\'a3\f0\rdblquote\lang2049\b0\f2\fs24\par}

Then the encoding message table of the above rich text is generated as shown in Fig. 2.

Finally the message is send as a series of bits as shown in Fig. 3 and they are splitting as a byte which means each character is represented as 8-bit.

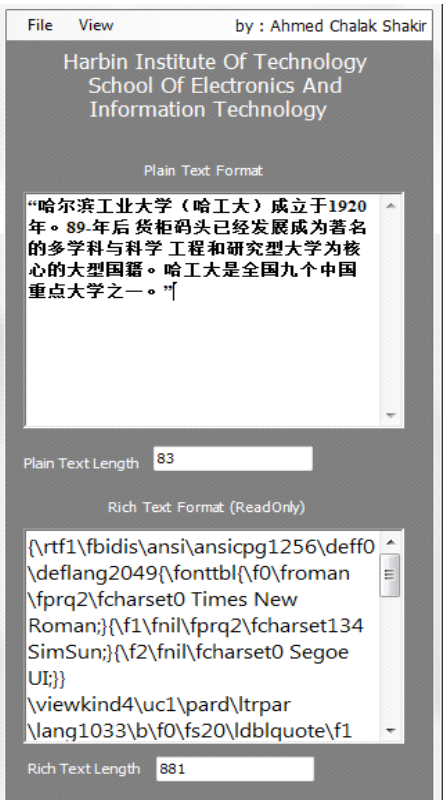

Fig. 1: Algorithm implementation interface

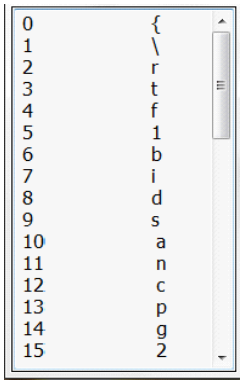

Fig. 2: The generation of the encoding message

Compression ratio = (710-877)/877 *100% = -19%

The negative sign means that the size of the message is decreased. But if the sign is positive that means there is increasing in the size and the compression method is not work properly, this may happen if the size of the message is small. So, the algorithm is working efficiently and properly for the big sizes of the messages. For that reason, the algorithm can be expanded in using; such that it can be used for the attached file which always has a big size.

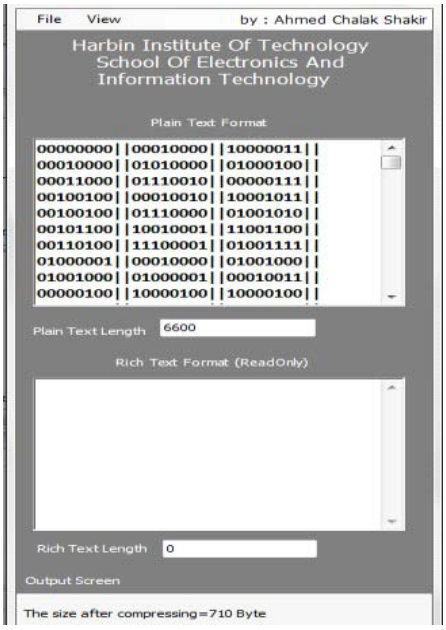

Fig. 3: The binary representation of the message

CONCLUSION

The compression method provides some features of changing the message, so that the message is encoded. By this, it empowers the next operation-encryption if needed because this method of compression provides the security and make the text file contents are changed completely which cannot be recognized by third parity person who wants to make a passive attack. The efficiency and the ratio of compression with the power of security, are increasing as the size of the message is increased. That mean our scheme is better for using not only in email message compression and security, but also in the file attached to such email which almost has a big size.

ACKNOWLEDGMENT

Authors would like to thank Harbin Institute of Technology, School of Electronics and Information Technology, China, for supporting this research.

REFERENCES

Anne, C., A. Ayman and H. Hamam, 2008. Optical video image compression: A multiplexing method based on the spectral fusion of information. Data Compression Conference. Snowbird, Utah, USA. 24-26 March 2010, IEEE Computer Society, pp: 1-6.

Debra, A.L. and S.H. Daniel, 1987. Data compression. ACM Comput. Surv., 19(3), DOI: 10.1145/45072.45074.

Nand, A. and L.Y. Tong, 2002. Mail servers with embedded data compression mechanisms. Data compression conference, 1998. DCC '98. Proceedings. Snowbird, UT, USA. (30 Mar 1998 - 01 Apr 1998). Paging Syst. Group, Motorola Inc., Fort Worth, TX, pp: 566. Retrieved from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=672308.

Sayood, K., 2000. Introduction to Data Compression. 2nd Edn., Harcourt Place, 32 Jamestown Road, London, NW17 By United Kingdom: Morgan Kaufmann Publishers. Academic Press, ISBN: 1-55860-558-4.

Umesh, S.B. and A.I. Trivedi, 2011. Lossless text compression using dictionaries. Int. J. Comput. Applic., 13(8): 0975-8887, DOI:10.5120/1799-1767

Weldon, W.N., 2007. Accelerated C# 2008. Printed and Bound in the United States of America 9 8 7 6 5 4 3 2 1. ISBN: 13 (pbk): 978-1-59059-873-3.