Combination of the Blowfish and Lempel-Ziv-Welch algorithms for text compression

Robbi Rahim†, Muhammad Dahria‡, Muhammad Syahril‡ & Badrul Anwar‡

Universiti Malaysia Perlis, Perlis, Malaysia†
STMIK Triguna Dharma, Medan, Indonesia‡

ABSTRACT: Delivery of secure information in small sizes becomes a necessity in communication within networks, and to accommodate this, a cryptographic process is needed. In this research, the Blowfish algorithm was used for data security and the Lempel-Ziv-Welch (LZW) algorithm was used for data compression. In the test carried out, the authors found that the compression result of ciphertext result of compression was smaller and also safer, because the indirect ciphertext changed when the compression process was done. So, the combination of these two algorithms give a positive result.

INTRODUCTION

The transmission of information through an electronic network requires a process that ensures safety and integrity of the transmitted information [1-3]. The information must remain confidential during delivery and must continue to be original upon receipt at the destination [3-5]. To fulfil this condition, there must be a process of encryption and decryption of the information to be sent [6].

Encryption is made at the time of transmission by converting the original information into encrypted information, while the decryption process at the moment of receipt is effected by converting encrypted information into original information [5][7]. The information transmitted during the sending process is already encrypted, so the unauthorised parties cannot know the original information. The recipient can only know the original information by using a secret key [7][8].

In addition to information security, now users are also trying to reduce the size of data (compression of data). The purpose of this compression is to speed up the transmission of data or information [9]. Data compression also aims to reduce the size of data and can be stored on storage media that have a relatively small size due to security and speed. This is important in communication using computers and networks [9][10].

Many algorithms can be used to perform data security, such as Twofish, IDEA, MMB, GOST, AES and Blowfish. The Blowfish algorithm is the algorithm that was used in this research. The Blowfish algorithm is a modern symmetry algorithm that uses the S-Box table for the key as a decryption encryption process [11], the compression process also uses a variety of algorithms, one of which is the Lempel-Ziv-Welch (LZW) algorithm. The LZW algorithm is a lossless compression algorithm and uses the dictionary method [9].

The compression principle will occur when the bit value for the specified dictionary replaces the sequence of characters or strings formed, while in the decompression process, to obtain the same result with the file before compression, LZW recreates the dictionary during the decompression process [9][12-15]. The combination of Blowfish and LZW algorithms in this study is expected to be a good security solution with compressed ciphertext results, so that ciphertext will be much safer and with a smaller message length.

METHODOLOGY

Encryption is a crucial thing in cryptography, which is the security of data sent secretly [8]. The original message is called plaintext, which is transformed into codes that are not understood by those not privy to the transmission [1][6][8]. Encryption could be described asa cipher or code. Reverse to the case with decryption, it can be interpreted as a process to convert the ciphertext into plaintext [1-3], the process of encryption/decryption could be described as below:

Figure 1: Encryption and decryption process.

Blowfish Algorithm

Blowfish was created by cryptanalyst Bruce Schneier and published in 1994. Blowfish was made for use in computers with a large microprocessor (32-bit and above with a large data cache) [11]. Blowfish is a non-patented and license-free algorithm, and is available for free for a variety of uses [11][12]. When Blowfish was invented, it was expected to have the following design criteria:

1. Fast, Blowfish performs data encryption on 32-bit microprocessors with a rate of 26 clock cycles per byte.
2. Compact (lightweight), Blowfish can run on less than 5K memory.
3. Simple, Blowfish uses only simple operations: addition, XOR and lookup tables on 32-bit operands.
4. Having varying security levels, the key lengths used by Blowfish may vary and may be up to 448 bits long.

In the implementation, often this algorithm becomes non-optimal, due to improper implementation strategy. Blowfish algorithms will be more optimal if used for applications that do not often change keys, such as communication networks or automatic file encryption [12]. Also, since this algorithm requires a large amount of memory, this algorithm cannot be applied to applications that only have a small memory, such as smart cards. The key length used also affects the security of implementing this algorithm [11][12].

The Blowfish algorithm consists of two parts, the key expansion and data encryption [11][12]:

1. Key expansion: function lock switches (minimum 32-bit, a maximum of 448-bits) into several array subkeys with a total of 4,168 bytes (18 x 32-bit for the P-array and 4 x 256 x 32-bit S-Box for a total of 33,344 bits or 4,168 bytes) and keys are put in the K-array.
2. Encryption data: consisting of a simple iteration of functions (Feistel network) of 16 times round (iteration), the input is a 64-bit data element X. Each round consists of key-dependent permutations and key- and dependent data substitutions. All operations are additions and XOR on 32-bit variables. Other additional operations are just four indexed array table searches for each round.

LZW Algorithm

The Lempel-Ziv-Welch (LZW) algorithm uses adaptive and dictionary-based techniques. The LZW predecessor is LZ77 and LZ78 developed by Jacob Ziv and Abraham Lempel in 1977 and 1978. Terry Welch further developed the technique in 1984. LZW is widely used on UNIX, GIF for modems [15].

This algorithm performs compression using the dictionary, where text fragments are reconstructed with indexes obtained from the dictionary, where special codes are used to represent current words. This approach is adaptive and efficient because many characters could be encoded by reference to strings that have appeared before in the text [10][13][15]. The compression principle is reached, if the reference in the form of a pointer can be collected in a lower number of bits than the original string [15].

The combination process in this experiment was performed one by one, where the first process was done by encryption with the Blowfish and ciphertext algorithm, then, in compression by using the Lempel-Ziv-Welch (LZW) algorithm.

RESULTS AND DISCUSSION

The encryption process with the Blowfish algorithm is perform with manual calculations that occur in the encryption process, in this case the following parameters are used:

Plaintext= MATAHARI
Key = 2905

The calculation is as below:
1. Plaintext= MATAHARI

Table 1: Plaintext to binary.

Char	Hexadecimal	Binary
M	4D	01001101
A	41	01000001
T	54	01010101
A	41	01000001
H	48	01001000
A	41	01000001
R	52	01010010
I	49	01001001

Then, plaintext is divided into two parts XL and XR becomes:

XL = 01001101 01000001 01010101 01000001
XR = 01001000 01000001 01010010 01001001

2. Generating subkeys
Key: 2905

Table 2: Key to binary.

Char	Hexadecimal	Binary
2	32	00110010
9	39	00111001
0	30	00110000
5	35	00110101

Binary: 00110010 00111001 00110000 00110101

a. Key for first iteration:

$P_1 = P_1$ XOR Key
P_1 = 00100100 00111111 01101010 10001000 XOR
00110010 00111001 00110000 00110101
P_1 = 00010110 00000110 01011010 10111101

b. Key for second iteration:

$P_2 = P_2$ XOR P_1
P_2 = 10000101 10100011 00001000 11010011 XOR
00010110 00000110 01011010 10111101
P_2 = 10010011 10100101 01010010 01101110

3. The experiment can only do one iteration, because the total iteration of the encryption process is 16 rounds, the working principle of the same calculation process continues until the final result of iteration to 16 is obtained, for the first iteration i = 0 i.e.:

$XL = XL$ XOR P_1
XL = 01001101 01000001 01010101 01000001 XOR
00010110 00000110 01011010 10111101
XL = 01011011 01000111 00001111 11111100

F functions are derived from:

XL is divided into 4 (a, b, c, d) each 8 bits =
a = 01011011
b = 01000111
c = 00001111
d = 11111100

F Function:

$F(XL) = (((S_1.a + S_2.b \bmod 2^{32}) \text{ XOR } S_3, c) + S_4 d \bmod 2^{32}) = S_1.a + S_2.b \bmod 2^{32}$

= (11010001 00110001 00001011 10100110. 01011011) + (01001011 01111010 01110000 11101001. 01000111) mod 2^{32}
= 00011010 11110111 11111010 10111000 XOR S_3.c
= 00011010 11110111 11111010 10111000 XOR (11101001 00111101 01011010 01101000. 00001111)
= 11001111 10100000 01010011 01111110 00011000 + S4.d mod 2^{32}
= (11001111 10100000 01010011 01111110 00011000 + (00111010 00111001 11001110 00110111. 11111100) mod 2^{32}
= 11001010 00111010 10010010 01000001

F (XL) = 11001010 00111010 10010010 01000001
XR = *F (XL)* XOR *XR*
XR = 11001010 00111010 10010010 01000001 XOR 10010001 01111100 00111001 00111100
XR = 01011011 01000110 10101011 01111101

Changing the value of XL and XR:

XL = *XR*; XR = *XL*
XL = 01011011 01000110 10101011 01111101;
XR = 01000011 01010110 00010011 10011111

4. After doing 16 iterations, it will generate new values XL and XR each 32 bits. Switch back XL and XR. After that XOR the XL and XR: XR = XR XOR P_{16} and XL = XL XOR P_{17} XR XOR P_{16}

= 01000011 01010110 00010011 10011111 XOR 10010010 00010110 11010101 11011001
= 11010001 01000000 11000110 01000110 *XL* XOR P_{17}
= 01011011 01000110 10101011 01111101 XOR 10001001 01111001 11111011 00011011
= 11010010 00111111 01010000 01100110

5. Then, XL and XR are combined so that it becomes 64 bit. 11010010 00111111 01010000 01100110 11010001 01000000 11000110 01000110
6. The binary value is converted into ASCII code, so as to produce i.e. ciphertext: Ò?PfÑ@ÆF

The result of the Blowfish Ò?PfÑ@ÆF encryption would be compressed with the LZW algorithm, and the process is as follows (character used is ASCII standard character):

Table 3: Lempel-Ziv-Welch compression process.

Step	Input	Temp. C+W	Dictionary	New word	Output
1	Ò	Ò?	No	156 [Ò,?]	Ò
2	?	?P	No	157[?, P]	?
3	P	PF	No	158 [P,F]	P
4	F	FÑ	No	159 [F, Ñ]	f
5	Ñ	Ñ@	No	160 [Ñ,@]	Ñ
6	@	@Æ	No	161[@, Æ]	@
7	Æ	ÆF	No	162 [Æ, F]	Æ
8	F	F	No	163 [F]	F

Step 1 input value Ò, C + W is Ò,? Has it been there before? *No* new dictionary becomes 156 [Ò,?]; then, the output is Ò, so on until there is still the next character in the stream characters:

Stream characters: "Ò?PfÑ@ÆF"

Total initial bits are stored before compression =
Total Input * bit dictionary
= 8 * 8
= 64

After compressed = Total Output * bit dictionary
= 7 * 8
= 56

Ratio $= \left(\frac{\text{Compressed file size}}{\text{Original Size}}\right)$ X 100 %
= 100 - $\left(\frac{56}{64}\right)$ X 100 %
= 100 - 87.5 % = 12.5%

Based on the formula above, the compression ratio is 12.5%, it appears that the compression ratios are not too big. One of the weaknesses of the LZW algorithm is a smaller string that is recognised from the dictionary, then, the compression process is also low, so the LZW algorithm is either used for long strings or has a pattern looping, which is often the case, so that the compression process is also significant; here is a comparison of compression graphics to the Blowfish ciphertext with different lengths.

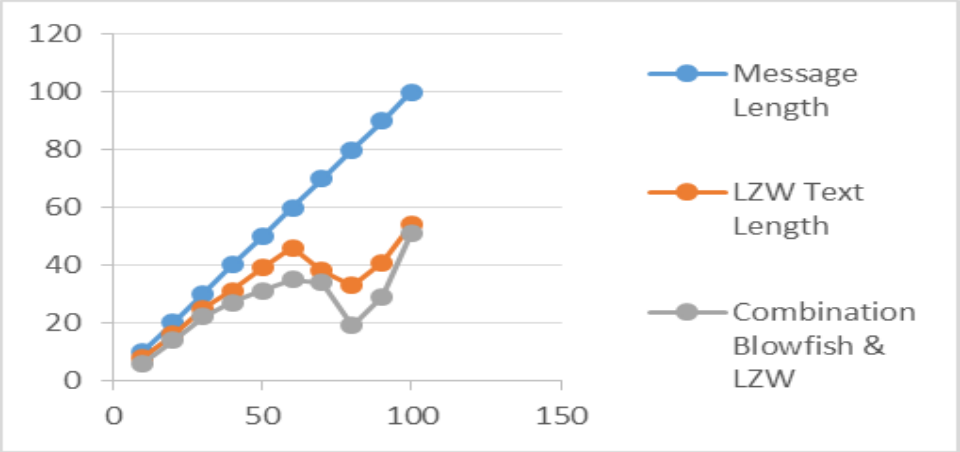

Figure 2: Comparison of compression ratios.

More specifically, the graph above shows that the larger the Blowfish ciphertext is generated and the smaller string produced by the LZW compression process, it proves that the length of a string affects the large compression ratio generated during the compression process.

The grey line is the result of compression text from the combination process of the Blowfish encryption and LZW compression, and the result of the test is that the compression result is smaller than using only LZW algorithm or encryption only, and security also has different ciphertext, because the first ciphertext (LZW) is already in compression with LZW.

CONLUSIONS

The combination of the Blowfish and LZW algorithms can give a good result in terms of compression security with the LZW algorithm to ciphertext generate a new form of strings with different patterns from the initial ciphertext obtained from the Blowfish encryption process, causing cryptanalyst to process the decryption over a longer time, due to having different patterns. Then, when viewed from the size and speed of data transmission, it is certainly faster due to its smaller size and does not require a tremendous amount of resources, so it can be concluded that the combination of these two algorithms gives a positive result.

REFERENCES

1. Rahim, R., 128 bit hash of variable length in short message service security. *Inter J. of Security and its Applications*, 11, **1**, 45-58 (2017).
2. Rahim, R. and Ikhwan, A., Cryptography technique with modular multiplication block cipher and playfair cipher. *IJSRST*, II, **6**, 71-78 (2016).
3. Rahim, R. and Ikhwan, A., Study of three-pass protocol on data security. *Inter. J. of Science and Research*, 5, **11**, 102-104 (2016).
4. Maulana, B. and Rahim, R., Go-Back-N ARQ approach for identification and repairing frame in transmission data. *Inter. J. of Research In Science and Engng.*, 2, **6**, 208-212 (2016).
5. Siahaan, P.U. and Rahim, R., Dynamic key matrix of hill cipher using genetic algorithm. *Inter. J. of Security and its Applications*, 10, **8**, 173-180 (2016).
6. Legito and Rahim, R., SMS encryption using word auto key encryption. *Inter. J. of Recent Trends in Engng. and Research*, 3, **1**, 251-256 (2017).
7. Saputra, Mesran, Hasibuan, N.A. and Rahim, R., Vigenere cipher algorithm with grayscale image key generator for secure text file. *Inter. J. of Engng. Research and Technol.*, 6, **1**, 266-269 (2017).
8. Hariyanto, E. and Rahim, R., Arnold's cat map algorithm in digital image encryption. *Inter. J. of Science and Research*, 5, **10**, 1363-1365 (2016).
9. Nasution, S.D., Ginting, G.L., Syahrizal, M. and Rahim, R., Data security using vigenere cipher and goldbach codes algorithm. *Inter. J. of Engng. Research and Technol.*, 6, **1**, 360-363 (2017).
10. Dheemanth, H.N., LZW data compression. *American J. of Engng. Research*, 3, **2**, 22-26 (2014).
11. Manku, S. and K. Vasanth, K., Blowfish encryption algorithm for information security. *ARPN J. of Engng. and Applied Sciences*, 10, **10**, 4717-4719 (2015).

12. Patil, P., Narayankar, P., Narayan, D.G and Meena, S.M., A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Computer Science*, 78, 617-624 (2016).
13. Taleb, S.A.A., Musafa, H.M., Khtoom, A.M. and Gharaybih, I.K., Improving LZW image compression. *European J. of Scientific Research*, 44, **3**, 502-509 (2010).
14. Shyni, K. and Manoj Kumar, K.V,, Lossless LZW data compression algorithm on CUDA. *IOSR J. of Computer*, 13, **1**, 122-127 (2013).
15. Cormen, T., Leiserson, C., Rivest, R. and Stein, C., Introduction to Algorithms. (3rd Edn), Massachusetts: MIT Press (2010).