An Implementation Of Elias Delta Code And ElGamal Algorithm In Image Compression And Security

Dian Rachmawati*, Mohammad Andri Budiman*, and Cut Amalia Saffiera*

1Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia

2Email: dian.rachmawati@usu.ac.id, mandrib@usu.ac.id, amalia.ssaifiera@gmail.com

Abstract. In data transmission such as transferring an image, confidentiality, integrity, and efficiency of data storage aspects are highly needed. To maintain the confidentiality and integrity of data, one of the techniques used is ElGamal. The strength of this algorithm is found on the difficulty of calculating discrete logs in a large prime modulus. ElGamal belongs to the class of Asymmetric Key Algorithm and resulted in enlargement of the file size, therefore data compression is required. Elias Delta Code is one of the compression algorithms that use delta code table. The image was first compressed using Elias Delta Code Algorithm, then the result of the compression was encrypted by using ElGamal algorithm. Prime test was implemented using Agrawal Biswas Algorithm. The result showed that ElGamal method could maintain the confidentiality and integrity of data with MSE and PSNR values 0 and infinity. The Elias Delta Code method generated compression ratio and space-saving each with average values of 62.49%, and 37.51%.

1. Introduction

Cryptography is the science of using mathematics to transform the contents of information or message in secure mode and also immuned to attack [10]. The main objectives of cryptography are confidentiality, integrity, authentication, and non-repudiation [5]. The process of converting the original message (plaintext) into a secured message (cipher text) is called encryption, while on the other hand, the process of converting ciphertext to plaintext is called decryption [2]. The security process is performed on every bit of the message to allow data size enlargement [4]. ElGamal is one of the algorithms of cryptography that includes an asymmetric algorithm. ElGamal is based on the difficulty of calculating discrete logs in a large prime modulus [9].

Data compression is a technique that commonly used to reduce the size of data in order to store it much more compactly and also to decrease its transfer time [3]. Data compression has been widely applied to various types of data, text, images, and video [11].

The images may cover a huge amount of the memory both in the RAM and in the storage. Image compression is for reducing the redundancy and irrelevance of image to allow them to either store or transmit the data in a better way[8]. Image compression can be classified as lossy and lossless. Lossy compression is a method that bases on the assumption that the current data files save more information than human beings can perceive. Thus the irrelevant data can be removed, so there is information lost...
during the compression process [7]. Lossless compression is a data compression method that allows the original data to be completely reorganized, without losing any information [6]. Elias Delta Code belongs to the lossless method. In this code, each character is represented by mapping the source code to a number of bit variables [11].

In Singh and Kumar [9], the weakness of the ElGamal algorithm is caused by the magnification of size. Therefore, it appears that data compression technique can reduce or decrease the size of the data. Elias Delta code is one of universal code that encodes the positive integers. An advantage of universal code is not mandatory to find probabilities of the source string exactly[1]. So, this research will combine both Elias Data Code to compress and ElGamal Algorithm to encrypt the image for getting the better result.

2. Method
In this implementation there are 3 main menus namely: Key Generator to generate public and private key, Sender Menu to compress image and encryption, and Recipient Menu to decrypt and decompress files.

2.1 Steps for generating key:
1. Randomize a large prime number \( p \) with Agrawal Biswas test prime Algorithm. The test as follows:
   a. Choose the number \( p \) that will be tested, with \( p > 100 \).
   b. Generate random \( Z \) such that \( 3 < Z < p - 2 \).
   c. If \((1 + Z)^p \mod p = (1 + Z^p \mod p) \mod p\), then \( p \) is prime.
   d. If not, \( p \) is not a prime number.
2. Randomize \( \alpha \) as primitive root of \( p \).
3. Randomize an integer \( d \) such that \( 1 < d < p - 2 \).
4. Compute \( \beta = \alpha^d \mod p \).
5. Get the public key \((p, \alpha, \beta)\) and private key \((p, d)\).

Then on the Sender menu, image was first compressed using Elias Delta Code Algorithm, then the result of the compression is encrypted by using ElGamal algorithm.

2.2 Steps for compressing the image:
1. Get all the frequency of all color value on each pixel in the selected image.
2. Create a compression table based on frequency (descending order).
3. Create Elias Delta Code Table
   a. Write \( n \) in binary. The left most bit will be 1.
   b. Calculate the number of bits, remove the leftmost bit from.
   c. Add the calculation in the binary on the left of \( n \), after the leftmost bit of \( n \) is removed.
   d. Subtract 1 from the calculation in step 2 and add the number of zeros to the code.
4. Replace File Content According To Elias Delta Code Table. Add padding bits and flag.
5. Generate Elias Delta Code To ASCII Code.
6. Calculate compression ratio and space saving.

2.3 Steps for encrypting:
1. Get the public key \((p, \alpha, \beta)\).
2. Encryption the compression value \((P)\) on each block, with calculate \( c_1 = \alpha^r \mod p \) and \( c_2 = P \times \beta^r \mod p \)
3. Save the result value of encryption.
Then on the Recipient menu, get the result of compressed and encrypted file. Then, the file is decrypted using Elgamal and decompressed using Elias Delta Algorithm to get the original image.

2.4 Steps for decrypting the file:
1. Get the private key \((p, d)\).
2. Decryption on each block using \(P = c_2 \times c_1^{p-1-d} \mod p\).

2.5 Steps for decompressing:
1. Convert \(P\) from string to binary digit.
2. Read bit flag (8-bit from right) to know how many padding bits.
3. Check starts from the first bit (from left), \(n = 1\)
   a. If the bit is contained in the Elias Delta Code table, change it to the appropriate string.
   b. If bit is not in the table, read the string in front of it, bit \(n + 1\) until the bits are contained in the table.
4. Repeat steps a and b until finished bit before padding.
5. Return the string value on each pixel to get the original image.

Figure 1 shows that pixel coordinate \((300,1)\) of the image with intensity value \((210,210,190)\).

![Sample of RGB value for true color image](image)

Figure 1. Sample of RGB value for true color image

From Figure 1, the steps for compression and encryption of the pixel \((300,1)\) are as follows:

1. Compression Process
   a. Create a compression table based on frequency (descending order) as showed in table 1.

| Value | Frequency | Elias Delta Code |
|-------|-----------|------------------|
| 210   | 2         | 1                |
| 190   | 1         | 0100             |

b. Replace value of pixel according to elias delta code table.
The pixel values of 210,210,190 changes to string bit = “110100”.
c. Add padding bits and flag.
For the number of string bits is 6, it has to take addition two extra bit 0. Then the padding bits is “00”, and the flag is the binary of 2 such as “00000010”. Finally the result of compression string is “1101000000000010”.
d. Generate Elias Delta Code To ASCII Code equal to 208, 2.
2. Generate Key  
   a. Randomize a prime number \( p = 271 \). Test with Agrawal Biswas Algorithm as follows:  
      - Generate random \( Z \) such that \( 3 < Z < p - 2 \). Example \( Z = 200 \).  
      - Do the first calculation  
        \[
        (1 + Z)^p \mod p = (1 + 200)^{271} \mod 271 = 201
        \]
      - Do the second calculation  
        \[
        (1 + Z^p \mod p) \mod p = (1 + 200^{271} \mod 271) \mod 271 = (1 + 200) \mod 271 = 201
        \]
      
      For the result of both calculation is equal, then 271 is a prime number.  
   b. Randomize primitive root \( \alpha = 107 \). The test as follows.  
      - Calculate \( q = \frac{p-1}{2} = \frac{271 - 1}{2} = 135 \).  
      - Calculate \( \alpha^2 \mod p \) and \( \alpha^3 \mod p \).  
        \[
        \alpha^2 \mod p = 107^2 \mod 271 = 67
        \]
        \[
        \alpha^3 \mod p = 107^{135} \mod 271 = 270
        \]
      107^2 \mod 271 \neq 1 and 107^{135} \mod 271 \neq 1, then \( \alpha \) is primitive root.  
   c. Randomize \( d = 96 \).  
      \[
      \beta = \alpha^d \mod p = 107^{96} \mod 271 = 39
      \]
      d. Then get the public key \((p, \alpha, \beta) = (271, 107, 39)\) and private key \((p, d) = (271, 96)\).  

3. Encryption Process  
   a. Get the public key \((p, \alpha, \beta) = (271, 107, 39)\).  
   b. Randomize \( r < p - 1 \). \( r = 50 \).  
   c. Calculate \( c_1 \) and \( c_2 \) on each block  
      - \( c_1[1] = \alpha^r \mod p = 107^{50} \mod 271 = 238 \)  
      - \( c_2[1] = P_1 x \beta^r \mod p = 208 \times 39^{50} \mod 271 = 193 \)  
      - \( c_1[2] = \alpha^r \mod p = 107^{50} \mod 271 = 238 \)  
      - \( c_2[2] = P_1 x \beta^r \mod p \)
4. Decryption Process
   a. Get the private key \((p, d) = (271, 96)\).
   b. Decryption on each block with calculate
      \[ P = c_2 x c_1^{p^{-1}d} \mod p \]
      \[ P_1 = c_2[1] \times c_1[1]^{p^{-1}d} \mod p \]
      \[ = 193 \times 2381^{27 \times 1-96} \mod 271 \]
      \[ = 208 \]
      \[ P_2 = c_2[2] \times c_1[2]^{p^{-1}d} \mod p \]
      \[ = 67 \times 2381^{27 \times 1-96} \mod 271 \]
      \[ = 2 \]

5. Decompression Process
   a. Convert P from string to binary digit.
      String bit = “11010000000010”.
   b. Read bit flag (8-bit from right) = “00000010”, then padding bits = 2.
   c. Check starts from the first bit, the value is 1. Because bit “1” is contained in the Elias Delta Code table, change it to the appropriate string of value 210.
   d. Check the next bit, the value is 1, change to 210.
   e. Check the next bit, the value is 0. Because bit is not available in the table, read the next string or bit n + 1, until those bits are available in the table. That is “0100”, then change it to appropriate string of the value 190.
   f. Input those value to get the intensity value of pixel of (210,210,190).

3. Results and Discussions

The experiments were performed on Windows 10 Notebook with Intel Core i3 processor, 64-bit architecture, and 2048MB RAM. The development environment used for coding C# scripts is SharpDevelop. The results of the experiments of each set are presented in Tables 2 and 3 as follows.

| Resolution (pixel) | Original Size (kb) | Compressed Size (kb) | Compressed Ratio (%) | Space Saving (%) |
|--------------------|--------------------|----------------------|----------------------|------------------|
| 200 x 200          | 156.30             | 99.59                | 63.72                | 36.28            |
| 400 x 400          | 625.05             | 392.88               | 62.86                | 37.14            |
| 600 x 600          | 1406.30            | 875.85               | 62.28                | 37.72            |
| 800 x 800          | 2500.05            | 1546.66              | 61.87                | 38.13            |
| 1000 x 1000        | 3906.30            | 2410.73              | 62.49                | 38.29            |

It can be seen in tabel 1 that the average compressed ratio is 62.49% and the average space saving is 37.51 %.

| Resolution (pixel) | MSE | PSNR |
|--------------------|-----|------|
| 200 x 200          | 0   | Infinity |
| 400 x 400          | 0   | Infinity |
| 600 x 600          | 0   | Infinity |
| 800 x 800          | 0   | Infinity |
| 1000 x 1000        | 0   | Infinity |
It can be showed in table 3 that input and output image are exactly the same.

4. Conclusions
In conclusion, the integrity of the image is maintained, with the value of MSE and PSNR 0 and infinity. Also, the average of compression ratio and space saving is 62.49% and 37.51% respectively.

5. Acknowledgments
The authors gratefully acknowledge that the present research is supported by Fund Dissemination IPTEKS Research Results for Lecturers / Researchers Universitas Sumatera Utara.

References
[1] Aarthi, R.S, Muralidharan, D, and Swaminathan, P 2012 Double Compression Of Test Data Using Huffman Code Journal of Theoretical and Applied Information Technology 39 (2) 104-113
[2] Ayushi. 2010. A Symmetric Key Cryptographic Algorithm. International Journal of Computer Applications 1 (15) 1-2
[3] Budiman, M.A and Rachmawati, D. 2017. On Using Goldbach GO Codes and Even-Rodeh Codes for Text Compression. Material Science and Engineering 180.
[4] Cai, Jin-Yi and Cusick, Thomas W. 1999. A Lattice-Based Public-Key Cryptosystem. Information and Computation 151 17-31.
[5] Garg et al.2014. Comparison of Asymmetric Algorithms in Cryptography. International Journal of Computer Science and Mobile Computing 3 (4) 1190-1196.
[6] Kavitha, P. 2016. A Survey on Lossless and Lossy Data Compression Methods. International Journal of Computer Science & Engineering Technology 7 (03) 110-114.
[7] Mamta Sharma. 2012. Compression Using Huffman Coding. International Journal of Computer Science And Network Security 5 (5)
[8] Singh, Manjari. 2016. Various Image Compression Techniques: Lossy and Lossless. International Journal of Computer Applications 142 (6) 23-26
[9] Singh, Rashmi dan Shiv Kumar. 2012. Elgamal’s Algorithm in Cryptography. International Journal of Scientific & Engineering Research 3 (12) 1-4.
[10] Thakkar, Jaydip. 2015. An Encryption and Decryption More Secure Elgamal Cryptosystem. International Journal of Science Technology & Engineering 1 (12)
[11] Walder et al. 2011. Fast Decoding Algorithms For Variabl-Lengths Codes. 2012. Information Sciences 183 66–91.