Journal of Theoretical and Applied Information Technology
15th June 2020. Vol.98. No 11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# FILE CRYPTOCOMPRESSION BY USING H-RABIN AND GOLOMB RICE ALGORITHM

[1]DIAN RACHMAWATI, [2]AMALIA AMALIA, [3]TRISKA HANIS ILLAHI
[1,2,3]Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi,
Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia
E-mail: [1]dian.rachmawati@usu.ac.id

**ABSTRACT**

Security in delivering messages is essential, because it is useful to keep the confidentiality of the information distributed. Therefore, the message encoding or H-Rabin cryptography is necessary so that the negligent party does not easily detect or know the contents of the data. H-Rabin's excess of cryptography is to rely more on its security at the power of factoring one large integer into three large prime factor factors, making it safer than some attacks. The H-Rabin encryption process generates a larger file size. This leads to slower file delivery. Therefore, Golomb Rice's compression method is necessary to deliver fast and accurate file delivery. Golomb Rice has a very suitable code for situations where the occurrence of small value in the input flow is significantly more likely than the large value. In this study, the authors performed *cryptocompression*, which was a merger of the H-Rabin cryptographic algorithms and the Golomb Rice compression algorithm. The research will be conducted two phases of testing, namely the stage of encryption and compression process, and then continued with the process of decompression and decryption. The system implementation uses the C# programming language. The results showed that the H-Rabin cryptographic algorithm with the Golomb Rice compression algorithm was able to perform the message security process and be able to restore the already secured messages. From the test results that have been done that the file type does not affect processing time. In the compression result, the *Ratio of Compression* (RC), *Compression of Ratio* (CR), *Space Savings* (SS), and *Bitrate* of cipher files using the same H-Rabin key has no distinct value even though the original file size is used differently.

**Keywords:** *Cryptography, Compression, Cryptocompression, H-Rabin, Golomb Rice*

## 1. INTRODUCTION

Security in delivering messages is essential. The security of the messages is useful to keep the information to be delivered. This is aimed at the unauthorized party, not easily detecting or knowing the content of the information. Information security can be done in a variety of ways by encoding the message. One of the sciences that learns the message encoding technique is cryptography.

Cryptography is a science or an art that learns how to make a message sent by the sender can be conveyed to the recipient safely [1]. Cryptography aims to maintain the confidentiality of the information contained in the data so that the information cannot be found by unauthorized parties [14].

One of the cryptographic algorithms is H-Rabin. H-Rabin is one of the asymmetric public-key cryptographic algorithms built into the Rabin algorithm [13]. Rabin's algorithm is its secure encryption scheme depending on the difficulty of factoring one large integer into two major prime factor factors. On the other hand, the H-Rabin algorithm relies more on its security at the power of the factorization of one large integer into three major prime factor factors, making it safer than some attacks [12].

Using the H-Rabin cryptographic algorithm causes the results of the encryption (ciphertext) process to result in larger file sizes after a cryptographic operation. This leads to slower sending of files or messages. While the data transmission should be faster and more accurate, so the solution to this problem is to use a compression method [7] [9].

Data compression is a way to compress data, so it only needs a smaller and more efficient space to store or shorten the data exchange time. Data compression can be divided into two, namely lossless compression and lossy compression [6] [11]. Lossless compression means the compression data can be restored to its original data. In the lossy compression, data that has been compressed can not

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

be restored to the original data due to loss of information during the compression process [10][16].

The algorithm used in the compression method is Golomb Rice. Based on Basha et. al The Golomb Rice algorithm is a lossless compression algorithm. The Golomb code algorithm separates every bit of the desired number using a single parameter that is K. This makes it easier to set the code dynamically, which is useful in the change of the encoded value. Golomb Rice's algorithm is a good algorithm to maintain the efficiency of high compression techniques caused by the presence of the PSNR (Peak Signal to Noise Ratio), which can produce images with high visual quality. So this algorithm provides a wide scope for further research on the compression field, especially image files [2].

*Cryptocompression* is a composite of cryptography and compression [3]. So cryptocompression serves to secure while compacted data so that in storage and transmission of data is more efficient. Based on the descriptions, the authors want to conduct research by combining H-Rabin cryptographic algorithms and Golomb Rice compression algorithms.

## 2. LITERATURE REVIEW

### *2.1. Image Files*

#### *2.1.1 Binary Image*

A binary image is a digital image that only has two possible pixel values that are black and white. Binary images are often also referred to as black and white imagery. This image only takes 1 bit to represent the value of each pixel and the binary image. Examples of binary images can be seen in Figure 1.

***Figure 1.*** *Binary Image*

#### *2.1.2 Greyscale Image*

Greyscale imagery is a digital image that has only one channel value on each pixel; in other words, the red = Green – Blue part value. The colors are black, grey, and white. The following grayscale image has 8 bits of color depth (256 color combinations). The sample greyscale image can be seen in Figure 2.

***Figure 2.*** *Greyscale Image*

#### *2.1.3 Color Image*

In the color image, each point has a specific color that is a combination of 3 primary colors, namely red, green, and blue. Each base color has its own intensity with a maximum value of 255 (8 bits). Therefore, each point on the colour image requires 3-byte data. Examples of color imagery can be seen in Figure 3.

***Figure 3.*** *Color Image*

### *2.2. Text Files*

A text file is a file that contains information in the form of text. Data comes from word processing documents and numbers used in calculations. The name and address in the database are examples of text data input consisting of characters, numbers, and punctuation. In this study, the text format used was *.txt *.docx. To compress a text file refers to the ASCII code. ASCII is used by computers and communication tools to show text; tables from ASCII can be viewed in Figure 4.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

**ASCII table**

| Char | Dec | Oct | Hex | Char | Dec | Oct | Hex | Char | Dec | Oct | Hex | Char | Dec | Oct | Hex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (nul) | 0 | 0000 | 0x00 | (sp) | 32 | 0040 | 0x20 | @ | 64 | 0100 | 0x40 | ` | 96 | 0140 | 0x60 |
| (soh) | 1 | 0001 | 0x01 | ! | 33 | 0041 | 0x21 | A | 65 | 0101 | 0x41 | a | 97 | 0141 | 0x61 |
| (stx) | 2 | 0002 | 0x02 | " | 34 | 0042 | 0x22 | B | 66 | 0102 | 0x42 | b | 98 | 0142 | 0x62 |
| (etx) | 3 | 0003 | 0x03 | # | 35 | 0043 | 0x23 | C | 67 | 0103 | 0x43 | c | 99 | 0143 | 0x63 |
| (eot) | 4 | 0004 | 0x04 | $ | 36 | 0044 | 0x24 | D | 68 | 0104 | 0x44 | d | 100 | 0144 | 0x64 |
| (enq) | 5 | 0005 | 0x05 | % | 37 | 0045 | 0x25 | E | 69 | 0105 | 0x45 | e | 101 | 0145 | 0x65 |
| (ack) | 6 | 0006 | 0x06 | & | 38 | 0046 | 0x26 | F | 70 | 0106 | 0x46 | f | 102 | 0146 | 0x66 |
| (bel) | 7 | 0007 | 0x07 | ' | 39 | 0047 | 0x27 | G | 71 | 0107 | 0x47 | g | 103 | 0147 | 0x67 |
| (bs) | 8 | 0010 | 0x08 | ( | 40 | 0050 | 0x28 | H | 72 | 0110 | 0x48 | h | 104 | 0150 | 0x68 |
| (ht) | 9 | 0011 | 0x09 | ) | 41 | 0051 | 0x29 | I | 73 | 0111 | 0x49 | i | 105 | 0151 | 0x69 |
| (nl) | 10 | 0012 | 0x0a | * | 42 | 0052 | 0x2a | J | 74 | 0112 | 0x4a | j | 106 | 0152 | 0x6a |
| (vt) | 11 | 0013 | 0x0b | + | 43 | 0053 | 0x2b | K | 75 | 0113 | 0x4b | k | 107 | 0153 | 0x6b |
| (np) | 12 | 0014 | 0x0c | , | 44 | 0054 | 0x2c | L | 76 | 0114 | 0x4c | l | 108 | 0154 | 0x6c |
| (cr) | 13 | 0015 | 0x0d | - | 45 | 0055 | 0x2d | M | 77 | 0115 | 0x4d | m | 109 | 0155 | 0x6d |
| (so) | 14 | 0016 | 0x0e | . | 46 | 0056 | 0x2e | N | 78 | 0116 | 0x4e | n | 110 | 0156 | 0x6e |
| (si) | 15 | 0017 | 0x0f | / | 47 | 0057 | 0x2f | O | 79 | 0117 | 0x4f | o | 111 | 0157 | 0x6f |
| (dle) | 16 | 0020 | 0x10 | 0 | 48 | 0060 | 0x30 | P | 80 | 0120 | 0x50 | p | 112 | 0160 | 0x70 |
| (dc1) | 17 | 0021 | 0x11 | 1 | 49 | 0061 | 0x31 | Q | 81 | 0121 | 0x51 | q | 113 | 0161 | 0x71 |
| (dc2) | 18 | 0022 | 0x12 | 2 | 50 | 0062 | 0x32 | R | 82 | 0122 | 0x52 | r | 114 | 0162 | 0x72 |
| (dc3) | 19 | 0023 | 0x13 | 3 | 51 | 0063 | 0x33 | S | 83 | 0123 | 0x53 | s | 115 | 0163 | 0x73 |
| (dc4) | 20 | 0024 | 0x14 | 4 | 52 | 0064 | 0x34 | T | 84 | 0124 | 0x54 | t | 116 | 0164 | 0x74 |
| (nak) | 21 | 0025 | 0x15 | 5 | 53 | 0065 | 0x35 | U | 85 | 0125 | 0x55 | u | 117 | 0165 | 0x75 |
| (syn) | 22 | 0026 | 0x16 | 6 | 54 | 0066 | 0x36 | V | 86 | 0126 | 0x56 | v | 118 | 0166 | 0x76 |
| (etb) | 23 | 0027 | 0x17 | 7 | 55 | 0067 | 0x37 | W | 87 | 0127 | 0x57 | w | 119 | 0167 | 0x77 |
| (can) | 24 | 0030 | 0x18 | 8 | 56 | 0070 | 0x38 | X | 88 | 0130 | 0x58 | x | 120 | 0170 | 0x78 |
| (em) | 25 | 0031 | 0x19 | 9 | 57 | 0071 | 0x39 | Y | 89 | 0131 | 0x59 | y | 121 | 0171 | 0x79 |
| (sub) | 26 | 0032 | 0x1a | : | 58 | 0072 | 0x3a | Z | 90 | 0132 | 0x5a | z | 122 | 0172 | 0x7a |
| (esc) | 27 | 0033 | 0x1b | ; | 59 | 0073 | 0x3b | [ | 91 | 0133 | 0x5b | { | 123 | 0173 | 0x7b |
| (fs) | 28 | 0034 | 0x1c | < | 60 | 0074 | 0x3c | \ | 92 | 0134 | 0x5c | \| | 124 | 0174 | 0x7c |
| (gs) | 29 | 0035 | 0x1d | = | 61 | 0075 | 0x3d | ] | 93 | 0135 | 0x5d | } | 125 | 0175 | 0x7d |
| (rs) | 30 | 0036 | 0x1e | > | 62 | 0076 | 0x3e | ^ | 94 | 0136 | 0x5e | ~ | 126 | 0176 | 0x7e |
| (us) | 31 | 0037 | 0x1f | ? | 63 | 0077 | 0x3f | _ | 95 | 0137 | 0x5f | (del) | 127 | 0177 | 0x7f |

***Figure 4.*** *ASCII Table*

### *2.3. Audio Files*

Audio files are a form of data that generates sound or sound. The Format of the audio file used in this study is *.wav. WAV (Waveform Audio) files are uncompressed audio files so that all audio samples are stored in a digital form of storage.

### *2.4. Video Files*

Video is an electronic signal processing technology that represents a moving image. One of the video file types is Audio Video Interleaved with *.avi file extension. AVI files store audio and video data on an interleaved structure. This File is only a container, and audio-video data can be compressed using a variety of codecs.

### *2.5. Cryptography*

Cryptography is the science to secure data or documents using encryption techniques against data so that it cannot be publicly known. Cryptography aims to allow messages or materials to be sent not to be known to their contents by unauthorized persons, and falsification of messages and documents can also be proven so that the person sending the message cannot avoid the responsibilities that have been sent by the sender of the file.

Several cryptographic objectives are confidentiality that can guarantee a message sent kept confidential so as not to be known by the other party, integrity of a message that can not be manipulated with the addition, alteration or reduction of the information contained in the message, authenticity to the process to ensure that the message sent is the original message, and the last non-repudiation that could prevent a party from denying the message is not derived from it.

There are some terms in cryptography namely plaintext (m) is the message to be sent (containing the original message), ciphertext (C) is a message that has been encrypted, encryption is a process that can change the plaintext to ciphertext, Decryption is a process that can convert ciphertext to plaintext so that the resulting message can return to the original message or the actual message and the key that is a number that is kept secret to be used in the encryption process and decryption.

The cryptographic algorithm is divided into two, symmetric algorithms and asymmetric algorithms. Symmetrical algorithms are often referred to as classic algorithms as these algorithms use the same keys for encryption and decryption activities. Examples of symmetrical cryptographic algorithms are Blowfish, MARS, IDEA, DES (Data Encryption Standard). Asymmetric algorithms are often referred to as public-key algorithms as these algorithms have keywords used to perform different encryption and decryption. Examples of asymmetric cryptographic algorithms are RSA (Riverst Shamir Adleman), ECC (Elliptic Curve Cryptography), and H-Rabin. The encryption and decryption process is shown in Figure 5.

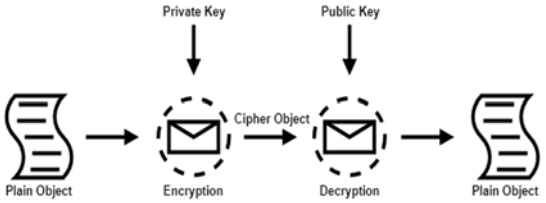


***Figure 5.*** *The Illustration of Cryptography*

### *2.6. Compression*

Compression is compacted with everything that is large to be smaller. So file compression means the process of compressing files for smaller sizes [4]. Compression differentiated into two types, namely lossy and lossless. Lossy data compression is a compression done by eliminating parts of the information that is considered unimportant, resulting in a very high compression ratio. Lossy data compression examples are CS&Q (Coarser Sampling and/or Quantization), JPEG, and MPEG. Lossless data compression is compressed data compression, then recompressed, then the data

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

is the same as the original data or compressions are done without removing any part of the information. Lossless data compression examples are Run-Length-Encoding, Lempel-Ziv-Welch Coding, Huffman Coding, and Golomb Rice. The compression process is shown in Figure 6 [8].

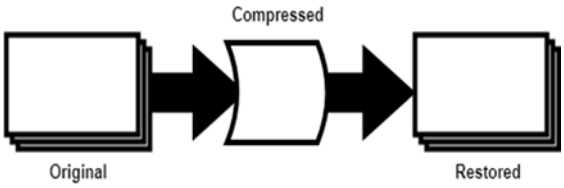


**Figure 6**. *The Illustration of Compression*

## 3. METHOD

The research uses the H-Rabin algorithm to process encryption and decryption of messages and Golomb Rice's algorithm to compress messages that are registered in encryption and decompression back to text files with extensions *.txt, *.docx, file image with extension *.bmp, an audio file with extension *.wav and video files with extension *.avi.

### *3.1. H-Rabin Algorithm*

The H-Rabin algorithm includes asymmetric cryptography. The H-Rabin algorithm is a change from the Rabin algorithm, which has more complicated durability than the Rabin algorithm. H-Rabin cryptography uses three random prime numbers with public key $n = p \times q \times r$ where $p$, $q$, and $r$ are random prime numbers whereas in Rabin algorithm, use only two random prime key numbers $n = p \times q$.

#### *3.1.1 Key Generation*

The first step of the recipient generates the private key, and the public key is as follows:

- Select 3 different prime numbers as the private keys are $p$, $q$, and $r$.
  With the condition $p \equiv q \equiv r \equiv 3$ (modulo 4).
  Example: the value $p = 31$, the value $q = 59$, and the value $r = 83$.
  These three values have been proven if in modulo with 4, then the result is 3.

- Count $n$, which is the public key where $\boldsymbol{n=p \times q \times r}$.
  Example: The value $n = p \times q \times r$
  $n = 31 \times 59 \times 83$
  $n = 151807$

#### *3.1.2 Encryption Method*

If the recipient has raised the public key ($n$), then the recipient sends the key to the sender for the message encryption process. The steps are as follows:

- Get the public key ($n$) from the receiver.

- Specify the message to be sent (plaintext) as m.
  Example: it is plaintext is T
  $m$ = "T" with ASCII value = 84
  has binary value = 1010100 (2)

- Extend $m$ with herself.
  $m_e = m\,|m$ or $m_e = m \times 257$
  Example: $m_e = m \times 257$
  $m_e = 84 \times 257$
  $m_e = 21588_{(10)}$
  $= 01010100_{(2)}|01010100_{(2)}$

- Calculate C value with the formula:
  $C \equiv m_e^2 \ (mod\ n)$
  Example: $m_e = 21588_{(10)}$
  $C = 21588^2 \ (mod\ 151807)$
  $C = 146061$

- Send C to the recipient.

#### 3.1.3 *Decryption method*

After the sender sends a ciphertext message, then the recipient is responsible for converting the password message to the original form so that it can be read with the private key ($p$, $q$, and $r$). The decrypted process is as follows:

- Accept the C value from the sender.

- Calculate $m_p$, $m_q$, and $m_r$ values with formula:
  $$m_p = C^{\left(\frac{p+1}{4}\right)} \ (mod\ p)$$
  $$m_q = C^{\left(\frac{q+1}{4}\right)} \ (mod\ q)$$
  $$m_r = C^{\left(\frac{r+1}{4}\right)} \ (mod\ r)$$
  From the formula then obtained the value of $m_p$, $m_q$, and $m_r$ as follows:
  $$m_p = 146061^{\left(\frac{31+1}{4}\right)} \ (mod\ 31) = 19$$
  $$m_q = 146061^{\left(\frac{59+1}{4}\right)} \ (mod\ 59) = 53$$
  $$m_r = 146061^{\left(\frac{83+1}{4}\right)} \ (mod\ 83) = 75$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Count each of the following values:

  $pm_p = m_p \pmod p = 19 \pmod{31} = 19$
  $mm_p = -m_p \pmod p = (-19) \pmod{31} = 12$
  $pm_q = m_q \pmod q = 53 \bmod 59 = 53$
  $mm_q = -m_q \pmod q = (-53) \bmod 59 = 6$
  $pm_r = m_r \pmod r = 75 \bmod 83 = 75$
  $mm_r = -m_r \pmod r = (-75) \bmod 83 = 8$

- Calculate $b_1$, $b_2$, $b_3$ with the CRT as follows:

$$b_1 = \left(\frac{n}{p}\right)^{-1} \pmod p$$
$$b_2 = \left(\frac{n}{q}\right)^{-1} \pmod q$$
$$b_3 = \left(\frac{n}{r}\right)^{-1} \pmod r$$

$$b_1 = \left(\frac{n}{p}\right)^{-1} \pmod p = (4897)^{-1} \pmod{31}$$
$$b_2 = \left(\frac{n}{q}\right)^{-1} \pmod q = (2573)^{-1} \pmod{59}$$
$$b_3 = \left(\frac{n}{r}\right)^{-1} \pmod r = (1829)^{-1} \pmod{83}$$

| $b_1$ | b1.4897 (mod 31) | $b_2$ | b2.2573 (mod 59) | $b_3$ | b3.1829 (mod 83) |
|---|---|---|---|---|---|
| 1 | 30 | 1 | 36 | 1 | 3 |
| 2 | 29 | 2 | 13 | 2 | 6 |
| 3 | 28 | 3 | 49 | 3 | 9 |
| … | … | … | …. | …. | … |
| 30 | 1 | 41 | 1 | 28 | 1 |

- Look for the $x_1$ to $x_8$ value in the following way:

$$x_1 = \left(\mathrm{pm}_p.b_1.\frac{n}{p} + \mathrm{pm}_q.b_2.\frac{n}{q} + \mathrm{pm}_r.b_3.\frac{n}{r}\right) \pmod n$$
$$x_2 = \left(\mathrm{mm}_p.b_1.\frac{n}{p} + \mathrm{pm}_q.b_2.\frac{n}{q} + \mathrm{pm}_r.b_3.\frac{n}{r}\right) \pmod n$$
$$x_3 = \left(\mathrm{pm}_p.b_1.\frac{n}{p} + \mathrm{mm}_q.b_2.\frac{n}{q} + \mathrm{pm}_r.b_3.\frac{n}{r}\right) \pmod n$$
$$x_4 = \left(\mathrm{pm}_p.b_1.\frac{n}{p} + \mathrm{pm}_q.b_2.\frac{n}{q} + \mathrm{mm}_r.b_3.\frac{n}{r}\right) \pmod n$$
$$x_5 = (n - x_1)$$
$$x_6 = (n - x_2)$$
$$x_7 = (n - x_3)$$
$$x_8 = (n - x_4)$$

Here is the $X_1$ to $x_8$ value we've been looking for:

$x_1 = 78759$
$x_2 = 113038$
$x_3 = 130219$
$x_4 = 139116$
$x_5 = 73048$
$x_6 = 38769$
$x_7 = 21588$
$x_8 = 12691$

- At modulo $x_1$ value up to $x_8$ with 257.

  $x_1 = 78759 \bmod 257 = 117$
  $x_2 = 113038 \bmod 257 = 215$
  $x_3 = 130219 \bmod 257 = 177$
  $x_4 = 139116 \bmod 257 = 79$
  $x_5 = 73048 \bmod 257 = 60$
  $x_6 = 38769 \bmod 257 = 219$
  **$x_7 = 21588 \bmod 257 = 0$**
  $x_8 = 12691 \bmod 257 = 98$

- If the modulo result value equals zero, share the $x_1$ value until $x_8$ with 257, which generates the ASCII code of the plaintext. Then convert those values to plaintext.

  **$x_7 = 21588 / 257 = 84 = T$**

### 3.2. *Prime Number Testing using Fermat Algorithm*

Fermat Little Theorem is a theory discovered by a French mathematician named Pierre De Fermat. In that theory says that the value of $p$ is expressed as the prime number if it meets the following requirements:

$$a^{p-1} \equiv 1 \pmod p$$

Where variable $a$ is a random integer that meets the requirements, $1 < a < p - 1$.
Examples:
Be sure to use the fermate algorithm below prime or not.

$p = 11$

$a^{p-1} \equiv 1 \pmod p$
$2^{11-1} \equiv 1024 \equiv 1 \pmod{11}$
$3^{11-1} \equiv 59049 \equiv 1 \pmod{11}$
$4^{11-1} \equiv 1048576 \equiv 1 \pmod{11}$
$5^{11-1} \equiv 9765625 \equiv 1 \pmod{11}$
$6^{11-1} \equiv 60466176 \equiv 1 \pmod{11}$
$7^{11-1} \equiv 282475249 \equiv 1 \pmod{11}$
$8^{11-1} \equiv 1073741824 \equiv 1 \pmod{11}$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$9^{11-1} \equiv 3486784401 \equiv 1\ (mod\ 11)$
$10^{11-1} \equiv\ 10000000000 \equiv 1\ (mod\ 11)$

Evident from the equation of the formula above that 11 is a prime number.

### 3.3. *Golomb Rice Algorithm*

Golomb Rice is a lossless data compression method that uses the family of data compression codes invented by Solomon W. Golomb in the 1960s. For the Golomb Rice code, find the quotient and remainder of the division by the divisor. Write the quotient in unary notation, then the remainder in truncated binary notation. In practice, we need a stop bit after the quotient. If the result is written as a sequence of zero numbers, the stop bit is one (or vice versa). The length of the remainder can be determined from the divisor. The following Table 1 is a table of code Golomb Rice with four dividers for numbers 0 to 21 as follows:

**Table 1.** *Golomb Rice Code*

| Value | Quotient | Remainder | Code |
|---|---|---|---|
| 0 | 0 | 0 | 1 00 |
| 1 | 0 | 1 | 1 01 |
| 2 | 0 | 2 | 1 10 |
| 3 | 0 | 3 | 1 11 |
| 4 | 1 | 0 | 0 1 00 |
| 5 | 1 | 1 | 0 1 01 |
| 6 | 1 | 2 | 0 1 10 |
| 7 | 1 | 3 | 0 1 11 |
| 8 | 2 | 0 | 00 1 00 |
| 9 | 2 | 1 | 00 1 01 |
| 10 | 2 | 2 | 00 1 10 |
| 11 | 2 | 3 | 00 1 11 |
| 12 | 3 | 0 | 000 1 00 |
| 13 | 3 | 1 | 000 1 01 |
| 14 | 3 | 2 | 000 1 10 |
| 15 | 3 | 3 | 000 1 11 |
| 16 | 4 | 0 | 0000 1 00 |
| 17 | 4 | 1 | 0000 1 01 |
| 18 | 4 | 2 | 0000 1 10 |
| 19 | 4 | 3 | 0000 1 11 |
| 20 | 5 | 0 | 00000 1 00 |
| 21 | 5 | 1 | 00000 1 01 |

### 3.4. *General Architecture*

General system architecture, as shown in Figure 7. The image shows the interaction made by system builders to the creation of the system [15]. The first process is that sender sends a message then in encryption using the public key and then generates a Cipher object. After creating a Cipher object indicating the message has been kept confidential, the compression is then useful to compress the message. The result of the cipher object becomes uncompressed, and then the message is reached to the receiver. Furthermore, the decompression process to restore the message size as the original and cipher object that still keep the message confidential. Then cipher objects in the decryption using the secret key so that the messages sent by the sender can be read.

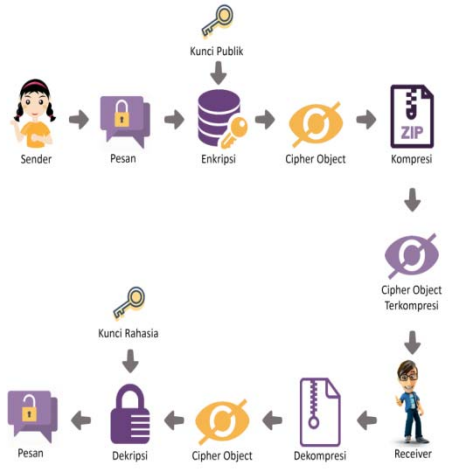


**Figure 7**. *General Architecture System*

## 4. RESULTS AND DISCUSSIONS

The result of this research is a system that is able to secure while compresses data, so as to help users to protect data. This research is also able to save computer memory because the data that is secured has a size smaller than the original size. This application is different from previous research because it uses a combination of cryptographic algorithms i.e. H-Rabin and the compression algorithm is Golomb Rice, in addition this research is able to process all kinds of file types. In the implementation of this system is done on Windows 10 with Intel Inside Core i3, 64-bit operating system, and 4.00GB RAM. The programming language used in the implementation of this system is C#, and the Integrated Development Environment

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(IDE) used for encoding is Microsoft Visual Studio 2017.

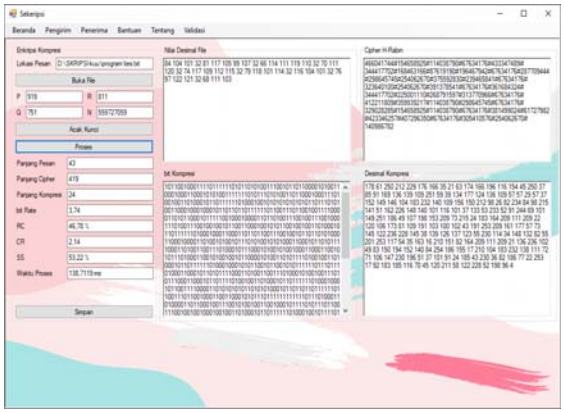

**Figure 8**. *Application of Encryption and Compression Processes*

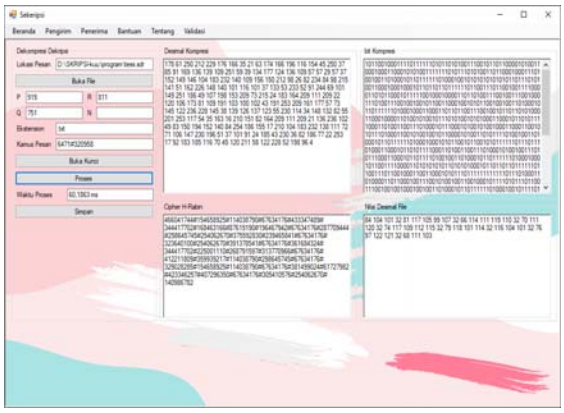

**Figure 9**. *Application of Decompression and Decryption Processes*

The key encryption process used is as follows:
$p = 947$
$q = 751$
$r = 307$
$n = p \times q \times r$
$= 947 \times 751 \times 307$
$= 218337479$

**Table 2.** *Calculation of the Encryption Process*

| Index | *Plain* ($m$) | $m_e = m \times 257$ | $C \equiv m_e^2$ | $C \equiv m_e^2 \pmod n$ |
|---|---|---|---|---|
| 1 | 84 | 21588 | 466041744 | 29366786 |
| 2 | 82 | 21074 | 444113476 | 7438518 |
| 3 | 73 | 18761 | 351975121 | 133637642 |
| 4 | 83 | 21331 | 455011561 | 18336603 |
| 5 | 75 | 19275 | 371525625 | 153188146 |

The result of ciphertext is as follows:

**29366786#7438518#133637642#18336603#153188146#60719546**

**Table 3.** *Calculation of Compression Process*

| Character | Codeword | Bit Length | Frequency | Frequency * Bit Length |
|---|---|---|---|---|
| 6 | 100 | 3 | 10 | 30 |
| 3 | 101 | 3 | 9 | 27 |
| 1 | 110 | 3 | 7 | 21 |
| 8 | 111 | 3 | 6 | 18 |
| # | 100 | 4 | 5 | 20 |
| 4 | 101 | 4 | 4 | 16 |
| 7 | 110 | 4 | 4 | 16 |
| 5 | 111 | 4 | 3 | 12 |
| 0 | 100 | 5 | 2 | 10 |
| 2 | 101 | 5 | 2 | 10 |
| 9 | 110 | 5 | 2 | 10 |
| Total | | | | 190 |

The result of the bits string in the compression process is as follows:

00101001101011001000110111100010001100101
10111101111101110100110101101100101011010
00101001010100110111101101100100001001010
10011001111011101111111100101100010010000
10001101100011001110101100 **00 00000010**

The results of the decompression process are as follows:

**29366786#7438518#133637642#18336603#153188146#60719546**

**Table 4.** *Calculation of Decryption Process*

Journal of Theoretical and Applied Information Technology
15th June 2020. Vol.98. No 11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

| Index | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| *Cipher* | 29366786 | 7438518 | 133637642 | 18336603 | 153188146 |
| *mp* | 193 | 707 | 768 | 497 | 335 |
| *mq* | 191 | 46 | 737 | 303 | 500 |
| *mr* | 209 | 109 | 273 | 148 | 66 |
| *pmp* | 193 | 707 | 768 | 497 | 335 |
| *mmq* | 754 | 240 | 179 | 450 | 612 |
| *pmq* | 191 | 46 | 737 | 303 | 500 |
| *mmq* | 560 | 705 | 14 | 448 | 251 |
| *pmr* | 209 | 109 | 273 | 148 | 66 |
| *mmr* | 98 | 198 | 34 | 159 | 241 |
| $x_0$ | 218315891 | 269655 | 157904495 | 21331 | 117366780 |
| $x / 257$ | 849478 | 1049 | 614414 | 83 | 456680 |
| $x \% 257$ | 45 | 62 | 97 | 0 | 20 |
| $x_1$ | 144998765 | 84653517 | 18878624 | 212825442 | 136041897 |
| $x_1 / 257$ | 564197 | 329391 | 73457 | 828114 | 529345 |
| $x_1 \% 257$ | 136 | 30 | 175 | 144 | 232 |
| $x_2$ | 117432928 | 218316405 | 139007110 | 50317448 | 199643087 |
| $x_2 / 257$ | 456937 | 849480 | 540883 | 195787 | 776821 |
| $x_2 \% 257$ | 119 | 45 | 179 | 189 | 90 |
| $x_3$ | 174221677 | 133974691 | 18761 | 173553399 | 19275 |
| $x_3 / 257$ | 677905 | 521302 | 73 | 675305 | 75 |
| $x_3 \% 257$ | 92 | 77 | 0 | 14 | 0 |
| $x_4$ | 21588 | 218067824 | 60432984 | 218316148 | 100970699 |
| $x_4 / 257$ | 84 | 848512 | 235147 | 849479 | 392882 |
| $x_4 \% 257$ | 0 | 240 | 205 | 45 | 25 |
| $x_5$ | 73338714 | 133683962 | 199458855 | 5512037 | 82295582 |
| $x_5 / 257$ | 285364 | 520171 | 776104 | 21447 | 320216 |
| $x_5 \% 257$ | 166 | 15 | 127 | 158 | 70 |
| $x_6$ | 100904551 | 21074 | 79330369 | 168020031 | 18694392 |
| $x_6 / 257$ | 392624 | 82 | 308678 | 653774 | 72740 |
| $x_6 \% 257$ | 183 | 0 | 123 | 113 | 212 |
| $x_7$ | 44115802 | 84362788 | 218318718 | 44784080 | 218318204 |
| $x_7 / 257$ | 171656 | 328259 | 849489 | 174257 | 849487 |
| $x_7 \% 257$ | 210 | 225 | 45 | 31 | 45 |

The following is the result of the decryption process:

84 82 73 83 75 65

In this test use 5 different file types (each 6 different sizes) and use the value $p = 947$, $q = 751$ and $r = 307$. The following are the results of testing on the part of the sender and recipient process by using, as shown in Table 5.

**Table 5.** *Test Results*

| | Initial Size (Byte) | Sending Process Time (ms) | Size Cipher (ms) | Compression Size (Byte) | RC (%) | CR | BR | SS (%) | Receiver Processing Time (ms) |
|---|---|---|---|---|---|---|---|---|---|
| *.txt | 51200 | 260.35.00 | 477480 | 226790 | 47% | 2.105 | 0,18056 | 53% | 13714.45 |
| | 102400 | 491.96 | 954463 | 452705 | 47% | 2.108 | 0,17986 | 53% | 23118.72 |
| | 153600 | 731.70 | 1431991 | 679425 | 47% | 2.108 | 0,18056 | 53% | 35490.26 |
| | 204800 | 993.35.00 | 1910314 | 906914 | 47% | 2.106 | 0,18056 | 53% | 50724.70 |
| | 512000 | 2678.82 | 4775661 | 2267152 | 47% | 2.106 | 0,18056 | 53% | 135149.92 |
| | 1024000 | 5399.79 | 9552829 | 4537329 | 47% | 2.105 | 0,18056 | 53% | 283343.64 |
| *.docx | 51200 | 232.28.00 | 477613 | 226589 | 47% | 2.108 | 0,18056 | 53% | 11194.77 |
| | 102400 | 483.21.00 | 954376 | 452582 | 47% | 2.109 | 0,17986 | 53% | 22346.05 |
| | 153600 | 740.93 | 1431674 | 679362 | 47% | 2.107 | 0,18056 | 53% | 36322.27 |
| | 204800 | 881.80 | 1909410 | 905449 | 47% | 2.109 | 0,17986 | 53% | 40818.35 |
| | 512000 | 2639.69 | 4772657 | 2265367 | 47% | 2.107 | 0,18056 | 53% | 131633.29 |
| | 1024000 | 4979.61 | 9544936 | 4529671 | 47% | 2.107 | 0,18056 | 53% | 245760.48 |
| *.wav | 51200 | 233.63 | 477285 | 226452 | 47% | 2.108 | 0,18056 | 53% | 11364.65 |
| | 102400 | 546.01.00 | 954471 | 453212 | 47% | 2.106 | 0,18056 | 53% | 27928.21 |
| | 153600 | 722.77 | 1432147 | 679414 | 47% | 2.108 | 0,18056 | 53% | 34691.18 |
| | 204800 | 925.80 | 1910446 | 906346 | 47% | 2.108 | 0,18056 | 53% | 44708.72 |
| | 512000 | 2635.28.00 | 4775424 | 2266634 | 47% | 2.107 | 0,18056 | 53% | 131268.55 |
| | 1024000 | 4891.31.00 | 9546431 | 4529556 | 47% | 2.108 | 0,18056 | 53% | 237929.33 |
| *.bmp | 51200 | 228.25.00 | 477584 | 226538 | 47% | 2.108 | 0,17986 | 53% | 10840.99 |
| | 102400 | 512.37.00 | 955140 | 453210 | 47% | 2.107 | 0,18056 | 53% | 24892.45 |
| | 153600 | 698.69 | 1431837 | 679045 | 47% | 2.109 | 0,17986 | 53% | 32561.21 |
| | 204800 | 1006.51.00 | 1910180 | 906974 | 47% | 2.106 | 0,18056 | 53% | 51899.98 |
| | 512000 | 2507.76 | 4773698 | 2264629 | 47% | 2.108 | 0,18056 | 53% | 119901.10 |
| | 1024000 | 5080.24.00 | 9544011 | 4530171 | 47% | 2.107 | 0,18056 | 53% | 254699.31 |
| *.avi | 51200 | 245.30.00 | 477203 | 226523 | 47% | 2.107 | 0,18056 | 53% | 12414.53 |
| | 102400 | 509.99 | 954456 | 452869 | 47% | 2.108 | 0,18056 | 53% | 24722.48 |
| | 153600 | 790.43.00 | 1432914 | 680405 | 47% | 2.106 | 0,18056 | 53% | 40691.70 |
| | 204800 | 955.95 | 1909195 | 906040 | 47% | 2.107 | 0,18056 | 53% | 47415.80 |
| | 512000 | 2449.85 | 4776148 | 2265246 | 47% | 2.108 | 0,17986 | 53% | 114767.71 |
| | 1024000 | 4852.00.00 | 9548754 | 4530289 | 47% | 2.108 | 0,18056 | 53% | 234469.07 |

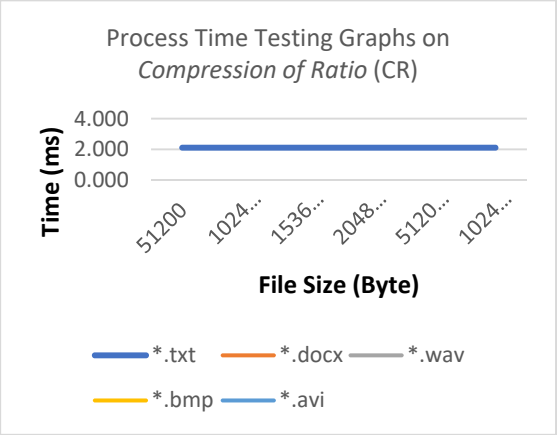


**Figure 10**. *Compression of Ratio Processing Time*

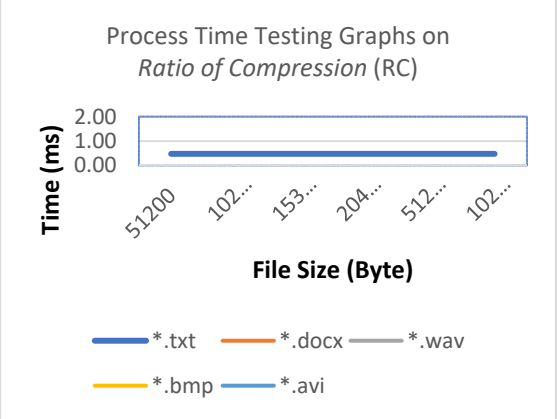


**Figure 11**. *Ratio of Compression Processing Time*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

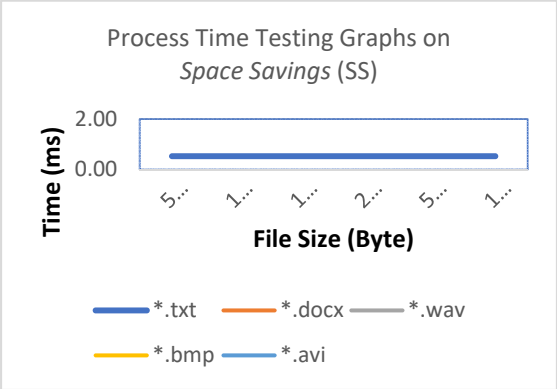


**Figure 12.** *Space Savings Processing Time*

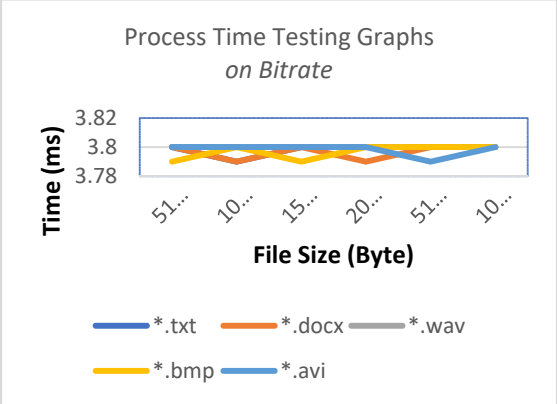


**Figure 13.** *Bitrate Processing Time*

## 5. CONCLUSION

Based on the results of studies that have been conducted in the analysis and testing phase of the system using H-Rabin cryptographic algorithms and Golomb Rice compression algorithm, the conclusions that can be taken are Implementation of the H-Rabin cryptographic algorithm with Golomb Rice compression algorithm is capable of securing the message and can restore the already secured messages. The implementation of the Golomb Rice compression algorithm is able to reduce the file size and can decompress it again. The factors affecting the sender and receiver process time are the message size; the larger the message size, the more time to process. The system designed successfully processes the file as a text file with extension *.txt *.docx, an image file with extension *.bmp, an audio file with extension *.wav and video files with extension *.avi. From the test results that the file type has done does not affect the processing time. The ratio of Compression (RC), Compression of Ratio (CR), Space Savings (SS), and Bitrate on compression results from cipher files using the same H-Rabin key do not have a different value even though the original file size used differently.

## REFERENCES

[1] Ariyus, Dony. 2008. *Pengantar Ilmu Kriptografi: Teori Analisis & Implementasi*. Penerbit Andi.

[2] Basha, Shaik Mahaboob, and BC Jinaga. 2013. "An Optimum Novel Technique Based on Golomb-Rice Coding for Lossless Image Compression of Digital Images." *International Journal of Signal Processing, Image Processing, and Pattern Recognition* 6 (5): 291–304.

[3] Borie, Jean-Claude, William Puech, and Michel Dumas. 2004. "Crypto-Compression System for Secure Transfer of Medical Images." In *MEDSIP: Medical Image and Signal Processing*, 327–331.

[4] M A Budiman and D Rachmawati 2017 *IOP Conf*. Ser.: Mater. Sci. Eng. 180 012062

[5] Ginting, Ikhsan Okto Kurnia Lidar. 2017. "Implementasi Algoritma Golomb-Rice Coding Untuk Kompresi File Citra Berbasis Android."

[6] Harahap, Winda Aprianti. 2017. "Studi Komparasi Kinerja Algoritma Goldbach G1 Code Dan Algoritma Unary Codes Dalam Kompresi Teks."

[7] Hashim, Hayder Raheem. 2014. "H-Rabin Cryptosystem." *Journal of Mathematics and Statistics* 10 (3): 258–262.

[8] Kiely, Aaron. 2004. "Selecting the Golomb Parameter in Rice Coding." *IPN Progress Report* 42: 159.

[9] Kurniadi, Karina Pramudita. 2018. "Implementasi Hybrid Cryptosystem Algoritma Stream Chipher Variable Permutation Composition (VMPC) Dan Algoritma Asimetris H-Rabin Untuk Pengamanan File."

[10] Merdiyan, Meckah, and Wawan Indarto. 2005. "Implementasi Algoritma Run Length, Half Byte Dan Huffman Untuk Kompresi File." In *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*.

[11] Moffat, Alistair, and Andrew Turpin. 2002. *Compression and Coding Algorithms*. Springer Science & Business Media.

[12] Nasution, Nadiya Khairani. 2018. "Implementasi Algoritma Kunci Publik H-Rabin Dan Algoritma Spritz Dalam Hybrid Cryptosystem Untuk Pengamanan File."

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[13] D. Rachmawati and M. A. Budiman, "An implementation of the H-rabin algorithm in the shamir three-pass protocol," 2017 2nd International Conference on *Automation, Cognitive Science, Optics, Micro Electro--Mechanical System, and Information Technology (ICACOMIT), Jakarta, 2017*, pp. 28-33. doi: 10.1109/ICACOMIT.2017.8253381

[14] Setyaningsih, Emy. 2015. "Kriptografi & Implementasinya Menggunakan Matlab." *Yogyakarta: Andi*.

[15] Whitten, Jeffery L, Lonnie D Bentley, and Kevin C Dittman. 2004. "Metode Desain Dan Analisis Sistem." *Yogyakarta: Andi & McGraw-Hill Education*.

[16] Yng, TL Bao, Byung-Gook Lee, and Hoon Yoo. 2008. "Low Complexity, Lossless Frame Memory Compression Using Modified Hadamard Transform and Adaptive Golomb-Rice Coding." In *IADIS International Conference Computer Graphics and Visualization*, 89–96.