Instal : Jurnal Komputer

E-ISSN: 2808-683X
Edisi : Volume 16 Nomor 05 | December 2024

Available online at https://journalinstal.cattleyadf.org/index.php/Instal/index

The Combination of the Spritz Algorithm and the Bit Plane Complexity Segmentation Technique in Text Security

Dessy Yusvika Yusdartono[1], Habib Muharry Yusdartono[2]
[1]Universitas Sumatera Utara, [2]Universitas Malikussaleh

ABSTRACT

Communication is the primary needs of the daily life. As human, communication is important. Through communication, some process of the message is exchanged between the sender and the recipient. The rapid advancement of information technology make security has become an important aspect of the communication, especially when it involves confidential messages. The presence of cryptography has proven to secure these secret messages by encrypting them into a form that cannot be easily understood. Additionally, the technique of steganography allows secret messages to be embedded into digital image, thereby enhancing the security of these messages. The Spritz Algorithm and Bit Plane Complexity Segmentation (BPCS) Technique are the methods of cryptography and steganography that can secure the secret messages, with the embedding message capacity is 20% up to 60% of the cover media's size. The results of these combination are represented in a table that includes the plaintext, key length, ciphertext, and the size of the generated stego image.

Keywords:

Cryptography,
Steganography,
Spritz Algorithm,
BPCS Technique

Corresponding Author:
Habib Muharry Yusdartono
Email: hmuharry1990@unimal.ac.id

INTRODUCTION

Communication is the primary needs of the daily life. As human, communication is important. At the moment, an individual acts as the sender of information, and at the same time, that individual serves as the recipient of information. This process continues endlessly throughout an individual's life. This situation is referred as the communication process. The slightly improvement of information technology as a medium also drive media communication to deliver information from one place to another making people easier to access the media communication. With everyone having easy access to communication media, information and communication security is becoming increasingly important and a new challenge in terms of security, accessibility, data management, and other information policy issues[1]. It affect to media information user's data security or messages using this media communication. If ignored, confidential messages or information can easily be exploited by irresponsible parties. Therefore, techniques for disguising and hiding messages, known as cryptography and steganography.

Cryptography is a branch of science whose the technique or art, disguise the data sent by the sender to the recipient for maintaining the data authenticity and integrity. Budiman and Poltak[2], explained that cryptography is a mathematics-based technique and is related to data security, including confidentiality, authentication, and data integrity. The process of cryptography shown in figure 1.

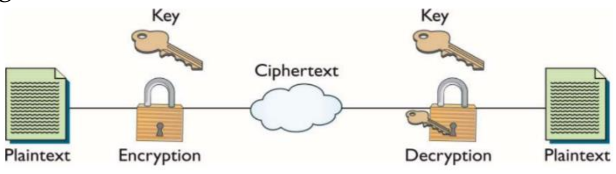


Figure 1. Process of Cryptography

In cryptography, the plaintext that has been encrypted becomes a message whose form and content are no longer understandable, potentially arousing suspicion from third parties who are not authorized to receive the information. These third parties may then attempt to uncover the information within the encrypted message and strive to decipher the actual content. To prevent it, steganography is needed.

Steganography is a technique for hiding secret message within digital media. The purpose of steganography *is* covert communication to hide a message from a third party[3]. The process of steganography shown in figure 2.

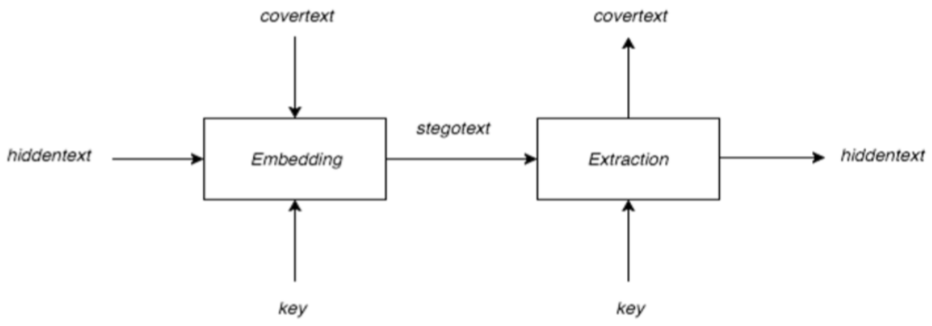


Figure 2. The Process Of Steganography

A. The Terminology of Steganography
   - Hiddentext or embedded message: The secret message to be inserted.
   - Covertext or cover-object or cover image: The original medium used to embed the hiddentext.
   - Stegotext or stego-object or stego image: The message that contains the embedded message (hiddentext and cover image).

B. The Criteria of Steganography

   The criteria for steganography that must be considered are as follows:
   - Fidelity: The quality of the carrier image should not change significantly. After the secret message is added, the resulting steganographic image should still appear normal. Observers should not be able to tell that there is a secret message within the image.
   - Imperceptibility: The existence of the secret message in the carrier medium should not be perceivable or detectable by sensory means.
   - Recovery: The hidden data must be recoverable. Since the goal of steganography is data hiding, the secret data in the carrier image should be retrievable for further use when needed.

The Spritz algorithm, is a symmetric cryptography algorithm improved from the RC4 algorithm, conducted by Ron Rivest and Jacob Schultz in 2014s. The element A is added to the value of N being the main procedure, makes the Spritz algorithm contrast to RC4. The encryption process of the Spritz algorithm is a variant of the RC4 encryption algorithm, where the encrypted data or message will be processed with the concept of a stream cipher, namely encryption one by one[4]. The Spritz algorithm has two main parts: the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA).

Bit Plane Complexity Segmentation (BPCS) is a steganography technique introduced by Eiji Kawaguchi and R. O. Eason in 1997. In BPCS, the original image is divided into "informative regions" and "noise-like regions," and the secret message is hidden within the noise-like region of the original image without altering the image quality. This traditional technique has limited data hiding capacity and they can hide up to 10 – 15% of the vessel data amount[5]. The BPCS technique is used on BMP format image documents that are uncompressed. The image document is divided into several segments, each with a size of 8×8 pixels. In an 8-bit image document, each segment will have 8 bit planes representing the pixels of each bit. These eight bit planes are called the PBC system (Pure Binary Coding). The embedding process is performed on the bit planes using the CGC system (Canonical Gray Coding) because the bit slicing process in CGC tends to be better compared to PBC.

METHODS

In this section, described about the steps involved in executing the Spritz algorithm and the Bit Plane Complexity Segmentation (BPCS) technique. And spritz algorithm is started and has two main parts: the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA).

The Key Scheduling Algorithm (KSA)

The Key-Scheduling Algorithm (KSA) in the Spritz algorithm is similar to the RC4 algorithm, as it generates a permutation of the 256 possible bytes in the array S. Table 1 is the pseudocode of the Key Scheduling Algorithm (KSA).

Table 1. The pseudocode of the Key Scheduling Algorithm (KSA)

```
N ← 256
S[ ] ← 0
for (i  = 0 to N −1)
S[i] ← i
Endfor
j ← 0
for (i = 0 to N  −1)
j ← j + S[i] + K[i mod K.length] mod 256
Swap values(S[i],S[j])
Endfor
```

In the Spritz algorithm, the variables *i* and *j* are integer-type variables with an initial value of 0. *S* is a 256-byte permutation initialized by the KSA. *N* represents the length of the formed array, and *K* is the ASCII value of the secret key that is inputted.

The Pseudo-Random Generation Algorithm (PRGA)

The Pseudo-Random Generation Algorithm (PRGA) is different from the RC4 algorithm. The PRGA generates random bytes for encryption. Table 2 shown the pseudocode for the Pseudo-Random Generation Algorithm (PRGA) process.

Table 2. The pseudocode of the Pseudo-Random Generation Algorithm (PRGA)

```
i, j, k, w, z ← 0
w ← random(2, 256)
plain ← plaintext.length
For (a = 0 to plain)
  i ← i + w
      j ← k + S[j + S[i]] mod 256
      k← i + k + S[j] mod 256
      Swap values (S[i],S[j])
      z ← S[j + S[i + S[z + k]]] mod 256
      output z
EndFor
```

The variable *k* is an integer-type variable with an initial value of 0, and *z* is an integer-type variable that functions as the output with an initial value of 0. Meanwhile, *w* is a variable that is relatively prime to the length of the array *S*. In the Key Scheduling Algorithm (KSA) phase, the values of the array are determined based on the length of the key entered. For example, if the key entered is "ab", the length of the key is 2 characters, with the key array being *K[0]* = "*a*" and *K[1]* = "*b*". Then, an initial array of 256 bytes will be formed, where the initial values of each array range from 0 to 255. The next step is to compute the first iteration of the Key Scheduling Algorithm (KSA).

- Compute the first iteration of the Key Scheduling Algorithm (KSA)
  i = 0; j = 0 → j = j + S[i] + K[i mod K.length] mod 256
  j = 0 + S[0] + K[0 mod 2] mod 256
  j = 0 + 0 + K[0] mod 256
  j = 0 + 0 + 65 (ASCII code "a") mod 256
  j = 65

Thus, the value of the 256-byte array at index 0 will be replaced with 65

The following steps for calculating KSA from the 3rd iteration to the 256th iteration are carried out by repeating the pseudocode from the Key Scheduling Algorithm (KSA) phase, where the value of *i* is incremented by one, and the value of *j* will change based on the last computed value of *j* from the previous step. Each time, *i* is incremented, and the new value of *j* is used to update the permutation array *S*, continuing this process until all 256 iterations (KSA 1 to KSA 256) are completed.

The plaintext to be encrypted is "Sah2018", with the ASCII code of each character in decimal being 83, 97, 104, 50, 48, 49, and 56. To generate the ciphertext, initialize the variables *i*, *j*, *k*, and *z* to 0, while the variable *w* is given a random value that is relatively prime to 256. Each iteration will be modulo 256. The calculation results of the Pseudo-Random Generation Algorithm (PRGA) for the first character "S" are as follows:
For the first character "S":

- ***i* = *i* + *w*** → *i* = (0 + 7) mod 256 (***assume that w* = 7**)
  i = 7 mod 256
  i = 7
- ***j* = *k* + *S*[*j* + *S*[*i*]]** → *j* = 0 + S[0 + S[7]] mod 256
  j = 0 + S[0 + 122] mod 256
  j = S[122] mod 256 = 22 mod 256
  j = 22
- ***k* = *i* + *k* + *S*[*j*]** → *k* = 7 + 0 + S[22] mod 256

k = (7 + 55) mod 256

k = 62

- ***S*[*i*] , *S*[*j*]** → **S[*i*]** = S[7] = 122, **S[*j*]** = S[22] = 55
- ***Swap S*[*i*] *with S*[*j*]** → **S[*i*]** = S[7] = 55, **S[*j*]** = S[22] = 122
- ***z* = *S*[*j* + *S*[*i* + *S*[*z* + *k*]]]** → z = S[22 + S[7 + S[0 + 62]]] mod 256

z = S[22 + S[7 + S[62]]] mod 256

z = S[22 + S[7 + 125]] mod 256

z = S[22 + S[132]] mod 256

z = S[22 + 197] mod 256

z = S[219] mod 256

z = 141 mod 256

z = 141

- XOR operation for the first character:

Now, we perform the XOR operation for the first character in the plaintext, which is "S" with ASCII code 83 and the output z = 141.

$$\begin{array}{l} 83 \;\; = 01010011 \\ \underline{141 = 10001101\ } \oplus \\ \quad = 11011110 = 222_{10} \end{array}$$

In the ASCII table, the value 222 corresponds to the character "Þ".
For the next characters: to process the next characters in the plaintext, repeat the calculations using the Pseudo-Random Generation Algorithm (PRGA). The same algorithm is also applied during the decryption process.

Bit Plane Complexity Segmentation (BPCS) Technique

The steps involved in the BPCS technique for embedding data are as follows:

- **Segmentation**: In this step, the original image (cover image) of size 80x72 pixels is divided into 8x8 pixel segments. After segmentation, 90 segments (10x9 segments) are formed. To simplify the segmentation process, the segments are adjusted to the size of n×nn \times nn×n, where nnn represents the smallest pixel size from the original image (cover image).
- **Convert intensity values**: The intensity values of each segment from the cover image are converted into binary values. These binary values are then transformed from the Pure Binary Coding (PBC) system to the Canonical Gray Coding (CGC) system.
- **Bit slicing**: Perform bit slicing or bit plane formation on each segment to create 8 bit planes.
- **Calculate complexity (α)**: The complexity (α) of each bit plane of the cover image is calculated. If $\alpha \geq \alpha_0$ then the segment is classified as a complex or noise-like region, and if $\alpha \leq \alpha_0$, it is classified as a simple or informative region. Here, α_0 is the threshold value, set at 0.3.
- **Read the secret message**: The secret message is read as a string, with the ASCII value of each character converted into binary. Then, the secret message is divided into 8x8 blocks, with each block representing 8 characters. The complexity (α) of each block is also calculated.
- **Create conjugation map**: A conjugation map of the secret message blocks is formed by adding padding (adding '0') to the last bits if the block size is not yet 8x8.
- **Embed secret message**: Each block of the secret message, combined with the stego key, is inserted into the bit planes of the cover image that are classified as noise-like regions.

- **Re-segment the stego image**: After embedding the secret message, segment the stego image back into 8x8 pixel blocks.
- **Convert intensity values back**: Convert the intensity values of each pixel in the stego image from the CGC system back to the PBC system.

RESULTS AND DISCUSSION

In this stage, the researcher conducts testing by comparing the results of the encryption and decryption processes performed by the system. The results of the system testing for the encryption-insertion process and the extraction-decryption process are displayed based on the text message file size, image file size, and processing time. The system testing results can be seen in Table 3.

Table 3. The Results of The Encryption and Embedding

Plaintext	*Ciphertext*	Waktu Proses (s)	*Cover Image*	*Stego Image*	Waktu Proses (s)
huhu.txt (12 *bytes*) Jumlah Karakter (12)	huhu_enkrip.txt (19 *bytes*) Jumlah Karakter (12)	0,562564	Ukuran *File* : 18,8 KB Ukuran Piksel (80x80)	Ukuran *File* : 7,72 KB Ukuran Piksel (80x80)	2,758587
minato_plain.txt (242 *bytes*) Jumlah Karakter (242)	minato_enkrip.txt (372 *bytes*) Jumlah Karakter (71)	2,341304	Ukuran *File* : 117 KB Ukuran Piksel (266x150)	Ukuran *File* : 89,2 KB Ukuran Piksel (266x150)	8,823945
ketiga_plain.txt (671 *bytes*) Jumlah Karakter (669)	ketiga_enkrip.txt (165 *bytes*) Jumlah Karakter (107)	3,354497	Ukuran *File* : 15,0 KB Ukuran Piksel (71x71)	Ukuran *File* : 11,7 KB Ukuran Piksel (71x71)	3,128515

Meanwhile, the results of the message extraction and decryption testing on the system are shown in the table 4.

Table 4. The Results of the Message Extraction and Decryption

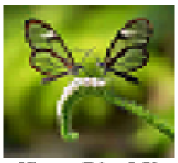
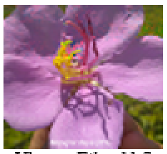
Stego Image	Waktu Proses (s)	*Ciphertext*	*Plaintext*	Waktu Proses (s)
 Ukuran *File* : 7,72 KB Ukuran Piksel (80x80)	4,467114	huhu_enkrip.txt (19 *bytes*) Jumlah Karakter (12)	huhu.txt (12 *bytes*) Jumlah Karakter (12)	0,183375

Table 5. The Results of the Message Extraction and Decryption

Ukuran *File* : 89,2 KB Ukuran Piksel (266x150)	8,684071	minato_enkrip.txt (372 *bytes*) Jumlah Karakter (71)	minato_plain.txt (242 *bytes*) Jumlah Karakter (242)	1,289066
 Ukuran *File* : 11,7 KB Ukuran Piksel (71x71)	2,786352	ketiga_enkrip.txt (165 *bytes*) Jumlah Karakter (107)	ketiga_plain.txt (671 *bytes*) Jumlah Karakter (669)	2,133458

The calculation of the storage capacity percentage for secret data using the BPCS technique is shown in table 5.

Table 6. The calculation of the storage capacity percentage

File Ciphertext & Cover Image	*Max. Embeddable Blocks*	*Total Embedded Blocks*	*% of Max. Hiding Capacity*
huhu_enkrip(txt) kupu_enkrip(bmp)	800	196	24,5%
minato_enkrip(txt) mimin(bmp)	4752	2194	46,17%
ketiga_enkrip(txt) bunga_stego(bmp)	512	305	59,57%

It can be seen that the data embedding capacity using the BPCS technique is between 20% up to 60% of the size of the provided cover image.

CONCLUSION

Based on the analysis of the design and implementation testing of the combination from spritz algorithm and the bit plane complexity segmentation technique in text security on bitmap image, the following conclusions are:

- The data security process using the combination of the Spritz algorithm and the Bit Plane Complexity Segmentation (BPCS) technique has not been fully successful. This issue arises because the number of plaintext characters before encryption exceeds 500, and the number of plaintext characters after decryption does not match, causing some characters to be lost or incomplete.
- The cryptography and steganography process using the combination of the Spritz algorithm and the Bit Plane Complexity Segmentation (BPCS) technique was successfully applied to text files as secret messages in txt, doc, and docx formats, as well as color images in bitmap format as the cover media.
- The shuffled Spritz key can be called repeatedly with the condition that the plaintext must contain at least one character, as the content of the Spritz key depends on the content of the entered plaintext.

- The file size difference between plaintext and ciphertext increases as the file size grows. On the other hand, the file size difference between the cover image and the stego image becomes smaller. This occurs because the BPCS technique has the ability to embed a significant amount of data, ranging from 20% to 60% of the size of the cover image.

REFERENCES

[1] J. C. Bertot, P. T. Jaeger, dan D. Hansen, “The impact of polices on government social media usage: Issues, challenges, and recommendations,” *Government Information Quarterly*, vol. 29, no. 1, hlm. 30–40, Jan 2012, doi: 10.1016/j.giq.2011.04.004.

[2] “(PDF) A cryptocompression system with Multi-Factor RSA algorithm and Levenstein code algorithm,” *ResearchGate*, Okt 2024, Diakses: 23 November 2024. [Daring]. Tersedia pada: https://www.researchgate.net/publication/352683599_A_cryptocompression_system_with_Multi-Factor_RSA_algorithm_and_Levenstein_code_algorithm

[3] G. C. Kessler dan C. Hosmer, “Chapter 2 - An Overview of Steganography,” dalam *Advances in Computers*, vol. 83, M. V. Zelkowitz, Ed., Elsevier, 2011, hlm. 51–107. doi: 10.1016/B978-0-12-385510-7.00002-3.

[4] M. Furqan dan R. Kurniawan, “Digital Image Security System Using Spritz Algorithm,” vol. 10, 2021.

[5] “(PDF) Review: Steganography – Bit Plane Complexity Segmentation (BPCS) Technique.” Diakses: 23 November 2024. [Daring]. Tersedia pada: https://www.researchgate.net/publication/50346770_Review_Steganography_-_Bit_Plane_Complexity_Segmentation_BPCS_Technique