Journal of Theoretical and Applied Information Technology
30th June 2024. Vol.102. No. 12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TEXT STEGANOGRAPHY BASED ON UNICODE CHARACTERS AS MARKER IN INDONESIAN EXCEL FILE

KUKUH ADI PRASETYO[1], ROJALI[2]

[1,2]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia

E-mail: [1]kukuh.prasetyo@binus.ac.id, [2]rojali@binus.edu

ABSTRACT

There are many applications to simplify office activities in Indonesia today. When the confidential information file from the office application has been spread, employees will definitely be confused about which one of the several users of the office application has downloaded the file. The message as a mark was given to the confidential information file using steganography based on Unicode characters utilizing the Latin letters that appeared most frequently in Indonesian (a, n, e, and i). The letters that appeared most frequently were made to have a different Unicode but with the same Latin letter appearance to represent certain binary (2-bit, 3-bit, or 4-bit binary). In using steganography, the results of Security Ratio, Size Increasing Ratio, and Capacity were measured and Invisibility result was also seen. The message as a mark in the confidential information file was successfully inserted and the best steganography algorithm in this research was steganography based on Unicode character which used 4 letters to represent 3-bit binary producing 100% for Security Ratio, no Size Increasing Ratio (0% for SIR), and 1954.76 bits for Capacity (increased compared to steganography based on Unicode character which used 3 letters to represent 3-bit binary, 3 letters to represent a 2-bit binary, and 4 letters to represent 2-bit binary)

Keywords: *Steganography*, *Unicode*, *Character*, *Letter*, *Excel*

1. INTRODUCTION

There are many applications to simplify office activities in Indonesia today. Every application in the office must have a lot of information. This information can be public or confidential information. Confidential information is protected as far as possible so that it can only be seen by people who have the right to receive or access it.

Based on a survey by Asosiasi Penyelenggara Jasa Internet Indonesia in 2023 to 8,510 respondents, it was found that 66.82% of internet users in Indonesia had never changed their passwords [1]. In addition, based on a survey conducted by YouGov as requested by Google to more than 13,000 respondents in 11 Asian markets in September 2021, it was found that 3 out of 5 respondents shared passwords with friends or family [2].

So, it is very possible that unauthorized employees can log into applications in the office and download files in formats such as Excel, Word, or PDF that contain confidential information texts and then spread them. If confidential information files have been spread, employees will definitely be confused about which user has downloaded the file. Therefore, it is necessary to provide a message hidden in the confidential information file as a mark. This hiding can be done using steganography. Encryption in Cryptography and information hiding are important methods in information security [3]. Steganography has been around since ancient times [4]. This is one of the data communications that has the highest level of effectiveness [5].

The media used in steganography is called a cover file, while the result file that has a message inserted is called a stego file. Commonly used cover files are images and audio but it is also possible to use text. The use of cover files in the form of text to hide messages is called text steganography [6]. Text steganography is considered the most difficult compared to steganography such as image or audio media [7].

There were several examples of inserting mark in a file. Martono and Irawan protected

ownership of digital data using marker by inserting information into the digital data using steganography with the End of File method [8]. This algorithm inserted the watermark marker at the end of the digital file. The results showed that the file quality was still the same, the file size increased according to the inserted size, and the insertion and extraction were successful.

Pratomo Djati Nugroho and Mahbubul Wathoni saw that the world of photography produced images that were widely spread on social media which could be acknowledged by unauthorized parties [9]. They inserted ownership marker in the form of text (which had been encrypted with the Advanced Encryption Standard algorithm) in the image using steganography with the LSB (Least Significant Bit) method so that the marker bits were inserted at the end of the image bits. Insertion and extraction of text messages in images could be done well but there was an increase with an average 616.85% for size.

Yulita Salim and Huzain Azis were worried that the research data of lecturers at the Muslim Indonesia University would be taken and acknowledged by unauthorized people, so they put a marker of ownership on the research data in pdf form [10]. The marker used was an image with the Digital Watermarking steganography algorithm. The system could be used by Muslim Indonesia University to provide mark of ownership.

Damian Victor Putra Rape Tupen, Wahyu Eko Sridaryanto, and Lukman Hakim inserted markers in images to make searching easier when an image was searched based on these markers [11]. This research was implemented on Android and used steganography with the LSB (Least Significant Bit) method. The average image search time in the Android gallery was 8.17 seconds, search accuracy was 100%, and there was an increase 0.2 to 0.6 KB for image size.

Ahmad Khuzaifi, Fauziah, and Iskandar Fitri inserted message marker into an image so that the image would not be acknowledged by unauthorized people [12]. This research used steganography with the LSB (Least Significant Bit) method. Marker in the form of messages could be extracted even though there was a change 675.1 KB (1186.5%) in file size.

There were several steganography methods that could be performed on text but what was interesting was the use of Unicode which developed from year to year. Aliea Salman Saber and Wid Akeel Awadh conducted research in Excel files, unlike generally in Word, PDF, PPT, and others [13]. The method used to hide messages was to use Unicode from Arabic and Persian numbers (0, 1, 2, 3, 7, 8, 9) which were similar when seen with the human eye but different Unicode. Invisibility was good because there was no visible difference from the outside with 1 bit per number in the Excel file (average 67.5%) for Capacity.

Abdul Monem S. Rahma, Wesam S. Bhaya, and Dhamyaa A. Al-Nasrawi carried out steganography on text, where this type of steganography was more difficult than images and audio [14]. Secret messages were hidden in Microsoft Word through the Unicode of certain Latin letters (A, B, E, G, H, I, M, O, P, S, T, j, o) which had similar characters to other languages but with different Unicode. 2-bit binary "00" was used when using Unicode from Latin letters, 2-bit binary "01" was used for Unicode from another language, 2-bit binary "10" was used for another language, and 2-bit binary "11" was used for the other language that had the same character appearance. The average size increase was 11.1%, capacity was 4177.6 bits (200% compared to the number of selected letters), and invisibility was good.

Salwa Shakir Baawi, Mohd Rosmadi Mokhtar, and Rossilawati Sulaiman used the Unicode Zero Width Non-Joiner (ZWNJ) and Zero Width Joiner (ZWJ) to perform steganography on text [7]. The words on the cover text were divided into words with 2 letters. The secret message was converted into binary bits which are further divided into 2-bit binary for later insertion. The inserted 2-bit binary depended on ZWNJ Unicode placement in a 2-letter word. Unicode ZWJ was inserted at the end when the secret message had been completely inserted. The average increase in size of the cover was 22.61%, good perceptual transparency, and average capacity was 5650.8 bits.

Salwa Shakir Baawi, Dhamyaa A. Nasrawi, Lina Talib Abdulameer tried to improve previous research related to Steganography in texts that used Unicode from other languages [14]. The choice of letter characters used was based on the four letters most frequently used in English [15]. If previous research used letters from other

languages, this time what was used were custom letters that had a different Unicode than the original letters. 2-bit binary "00" was used for the Unicode of the original letter, 2-bit binary "01" was used for the Unicode of the first custom letter, 2-bit binary "10" was used for the second custom letter, and 2-bit binary "11" was used for the third custom letter which had the same appearance. The cover file used was in TXT format from Chapter 1 of The Lost Girl by D. H. Lawrence. The results obtained were 0% for Size Increasing Ratio (SIR), Capacity increased to 26174 bits (200% compared to the number of selected letters), Invisibility was good.

The aim of this research is to insert mark in confidential information file and find out the way to insert it using text steganography. This research also has scope which is that the text steganography used is based on Unicode character. Then the confidential information file used is in Excel format and in Indonesian. The message as a mark is inserted using the Latin letters that appear most frequently in Indonesian. If in English the 4 letters that appear most frequently are e, t, a, o, then in Indonesian the 4 letters that appear most frequently are a, n, e, i [16]. The letters that appear most frequently are made to have different Unicode but with the same Latin letter appearance to represent certain binary. In this research there is also improvement in terms of the binary represented. If previously Salwa Shakir Baawi [15] only used 2-bit binary (00, 01, 10, and 11) then this research used 2-bit binary and then upgraded it to 3-bit binary (000, 001, and so on) and 4-bit binary (0000, 0001, and so on). Among the different binary represented, the best steganography algorithm is determined based on the performance of the steganography characteristics of each steganography algorithm. The Improvement in terms of the binary represented, the use of Latin letters that appear most often in Indonesian, and the use of Indonesian excel file in text steganography based on Unicode character are novelty.

2. LITERATURE REVIEW

2.1 Characteristics of Steganography

There are several characteristics of steganography that will be used in this research, namely Capacity, Size Increasing Ratio, and Security Ratio. Inserting a message which will make the cover file change to a stego file is not an easy task and there are characteristics that must be taken into account such as increasing of the file size and other characteristics [17].

Capacity is the main thing in text steganography [18]. Capacity is determined by how much data can be inserted and hidden in the cover file. The formula for calculating Capacity can be seen in Equation 1 [18].

$$\text{Capacity} = \text{represented bits x total number of selected cover characters} \quad (1)$$

Size Increasing Ratio (SIR) is the ratio of the increase in the size of the stego file compared to the size of the cover file. The formula used can be seen in Equation 2 [15].

$$\text{SIR} = \frac{(\text{Size of Stego File} - \text{Size of Cover File})}{\text{Size of Cover File}} \text{ x } 100\% \quad (2)$$

The homogeneity ratio between two texts, namely between the cover file and the stego file, can be measured by calculating Excess Visual Characters. If Excess Visual Characters in the stego file increases, it will reduce the homogeneity ratio which will also result in a lack of Security in the method [19]. Equation 3, Equation 4, and Equation 5 are used to calculate digitally the Security Ratio [19] [20].

$$\text{Discount Value} = \frac{\text{Original Character} * \text{Excess Character}}{100} \quad (3)$$

$$\text{Amount After Discount} = \text{Original Character} - \text{Discount Value} \quad (4)$$

$$\text{Security Ratio} = \frac{\text{Amount After Discount}}{\text{Original Character}} \text{ x } 100\% \quad (5)$$

Excess Character needs to be paid attention because it greatly influences the Security Ratio results. If the Excess Character (Total Number of Stego File Characters - Total Number of Cover File Characters) is equal to 0, then the Security Ratio result is 100%.

Apart from that, Invisibility will be seen. Invisibility is seen from whether humans or readers can or cannot differentiate between the cover file and the stego file to detect the inserted message [15].

2.2 ASCII

ASCII, which stands for American Standard Code for Information Interchange, is an international standard code for representing numbers, letters and other symbols on computers. ASCII can also be called numeric which represents commands or

characters on a computer [21]. ASCII is the same as Unicode which encodes various numbers, letters and symbols. ASCII uses 8 bits for coding. ASCII has 256 codes (0-255). Half are codes for text manipulation while the other half are codes for image or graphic manipulation.

2.3 Unicode

Unicode is a standard that is universally used to represent non-ASCII characters [22]. Unicode emerged because of ASCII's limitations in representing letter characters and symbols from various countries in the world. Unicode is capable of representing much more characters through code. The characters that can be represented by Unicode exceed 1 million characters [23]. Unicode has code point that is usually written in hexadecimal and begin with "U+" [24].

2.4 Glyph

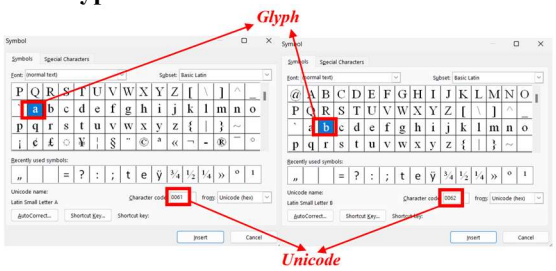


Figure 1: Glyph Examples in Microsoft Office

Glyphs can be interpreted as the physical appearance/representation of a character. If connected to Unicode then each Unicode has its own glyph. So, the glyph for Unicode 0061 is different from the glyph for Unicode 0062 which can be seen in Figure 1.

3. PROPOSED METHOD

3.1 Data Collection

In this research, the Excel file that would be inserted by message as a mark was an Excel file about the secret election of officials at the XYZ office. There were 34 files covering almost all provinces in Indonesia which can be seen in Table 1. The files contained the location of provinces, positions, current officials, proposed officials and elected officials.

Table 1: Excel Data of Confidential Information

Experiment	Cover File Name	Province
1	Aceh.xlsx	Aceh
2	Sumatera Utara.xlsx	North Sumatera
3	Sumatera Barat.xlsx	West Sumatera
4	Riau.xlsx	Riau
5	Kepulauan Riau.xlsx	Riau islands
6	Jambi.xlsx	Jambi
7	Sumatera Selatan.xlsx	South Sumatera
8	Bangka Belitung.xlsx	Bangka Belitung Islands
9	Bengkulu.xlsx	Bengkulu
10	Lampung.xlsx	Lampung
11	Jakarta Metropolitan.xlsx	Jakarta
12	Banten.xlsx	Banten
13	Jawa Barat.xlsx	West Jawa
14	Jawa Tengah.xlsx	Central Jawa
15	D. I. Yogyakarta.xlsx	Special Region of Yogyakarta
16	Jawa Timur.xlsx	East Jawa
17	Kalimantan Barat.xlsx	West Kalimantan
18	Kalimantan Tengah.xlsx	Central Kalimantan
19	Kalimantan Selatan.xlsx	South Kalimantan
20	Kalimantan Timur.xlsx	East Kalimantan
21	Kalimantan Utara.xlsx	North Kalimantan
22	Bali.xlsx	Bali
23	Nusa Tenggara Barat.xlsx	West Nusa Tenggara
24	Nusa Tenggara Timur.xlsx	East Nusa Tenggara
25	Sulawesi Utara.xlsx	North Sulawesi
26	Sulawesi Tengah.xlsx	Central Sulawesi
27	Sulawesi Tenggara.xlsx	Southeast Sulawesi
28	Sulawesi Barat.xlsx	West Sulawesi
29	Sulawesi Selatan.xlsx	South Sulawesi
30	Gorontalo.xlsx	Gorontalo
31	Maluku.xlsx	Maluku
32	Maluku Utara.xlsx	North Maluku
33	Papua.xlsx	Papua
34	Papua Barat.xlsx	West Papua

3.2 Creation of Special Font File

Journal of Theoretical and Applied Information Technology
30th June 2024. Vol.102. No. 12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

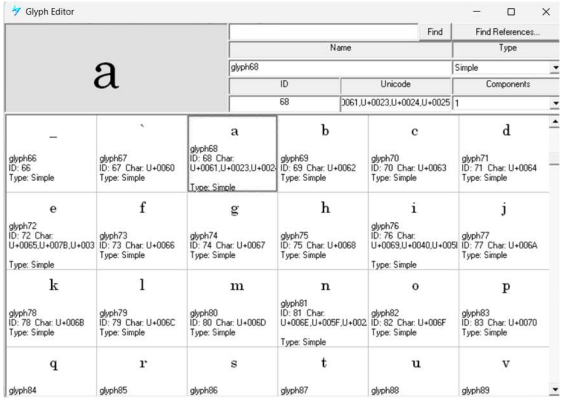


Figure 2: Use of Microsoft VOLT

Using the Microsoft VOLT program, a special letter file was created which the same glyph of a previously selected Latin letter had many different Unicode characters. As an illustration, the glyph for the letter 'a' will have the Unicode characters U+0061, U+0023, U+0024, and U+0025. Then did the same thing for the glyphs for the letter 'n', 'e', and 'i' with different Unicode characters as could be seen in Figure 2.

The Unicode characters used depended on the number of binaries represented. In Table 2 you could see the use of Unicode characters from 4 Latin letters to represent 2-bit binary. S was the binary represented. The Unicode of the characters used here all had 1 byte for the size.

Table 2: Use of Unicode characters on 4 Selected Letters Representing 2-Bit Binary

Letter	Unicode			
	S=00	S=01	S=10	S=11
a	U+0061	U+0023	U+0024	U+0025
n	U+006E	U+005f	U+002A	U+002B
e	U+0065	U+007b	U+003D	U+007d
i	U+0069	U+0040	U+005B	U+005C

Then the use of Unicode characters from 4 Latin letters to represent 3-bit binary could be seen in Table 3. The Unicode of the characters used here all had 1 byte for the size.

Table 3: Use of Unicode characters on 4 Selected Letters Representing 3-Bit Binary

Letter	Unicode			
	S=000	S=001	S=010	S=100
a	U+0061	U+0023	U+0024	U+0025
n	U+006E	U+005F	U+002A	U+002B
e	U+0065	U+007B	U+003D	U+007D
i	U+0069	U+0028	U+0029	U+0022

Letter	Unicode			
	S=011	S=101	S=110	S=111
a	U+0040	U+005B	U+005C	U+005D
n	U+005E	U+007C	U+007E	U+0026
e	U+0060	U+003F	U+0021	U+003B
i	U+003A	U+0027	U+003C	U+003E

Meanwhile, the use of Unicode characters of 4 Latin letters to represent 4-bit binary could be seen in Table 4. The Unicode of the characters used here were not all worth 1 byte. This happened because of the limited number of Unicode characters worth 1 byte. The first 32 Unicode characters (U+0061 to U+003E) were worth 1 byte. Meanwhile, the next 32 Unicode characters (U+0192 to U+00BB) were not worth 1 byte.

Table 4: Use of Unicode characters on 4 Selected Letters Representing 4-Bit Binary

Letter	Unicode			
	S=0000	S=0001	S=0010	S=0100
a	U+0061	U+0023	U+0024	U+0025
n	U+006E	U+005F	U+002A	U+002B
e	U+0065	U+007B	U+003D	U+007D
i	U+0069	U+0028	U+0029	U+0022
Letter	Unicode			
	S=1000	S=1001	S=1010	S=1100
a	U+0040	U+005B	U+005C	U+005D
n	U+005E	U+007C	U+007E	U+0026
e	U+0060	U+003F	U+0021	U+003B
i	U+003A	U+0027	U+003C	U+003E
Letter	Unicode			
	S=0110	S=0101	S=0011	S=1101
a	U+0192	U+02C6	U+0160	U+0152
n	U+017E	U+0178	U+00A1	U+00A2
e	U+00A7	U+00A8	U+00A9	U+00AA
i	U+00B0	U+00B1	U+00B4	U+00B5
Letter	Unicode			
	S=1011	S=1110	S=0111	S=1111
a	U+017D	U+02DC	U+0161	U+0153
n	U+00A3	U+00A4	U+00A5	U+00A6
e	U+00AB	U+00AC	U+00AE	U+00AF
i	U+00B6	U+00B7	U+00B8	U+00BB

The use Unicode characters from 3 Latin letters representing 2-bit, 3-bit, or 4-bit binary used the same table as before but the letters used were the top 3 letters, namely 'a', 'n', and 'e'. The custom font file was installed after special font file had been created where the same glyph of the selected Latin fonts had many different Unicode characters.

3.3 Message Insertion

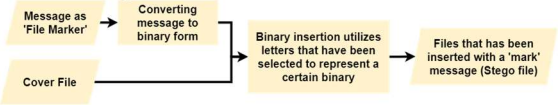


Figure 3: Flowchart of Message Insertion Overview

In Figure 3, flowchart of message insertion overview can be seen. The message as a mark is inserted using the Latin letters that appear most frequently in Indonesian (a, n, e, and i) to represent 2-bit, 3-bit, and 4-bit binary. The different binary representation gives different insertion algorithms.

Algorithm 1: Representing 2-Bit Binary
Input: Excel file E and message M
Output: Stego file S
Steps:
1. Open file E
2. Perform scan of selected letters in file E
3. Count the number of letters in Step 2 to check the maximum capacity (in bits) of insertion in file E
4. Calculate the length of characters in message M and write it in 2-digit format
5. Convert each digit into 8-bit binary so there are 16 bits
6. Convert M to binary (8-bit format) and add 16 bits (from Step 5) to the start of M
7. If the maximum insertion capacity is greater than or equal to the total bits from Step 6 then go to step 8, otherwise go directly to step 11
8. Divide M into blocks consisting of 2-bit binary
9. For each block of message M:
   - **If** block = 00, then use the original Unicode of the letter
   - **Otherwise,** use another Unicode according to the 2-bit binary represented by the other Unicode in table 2
10. Give file S
11. Finish

Algorithm 2: Representing 3-Bit Binary
Input: Excel file E and message M
Output: Stego file S
Steps:
1. Open file E
2. Perform scan of selected letters in file E
3. Count the number of letters in Step 2 to check the maximum capacity (in bits) of insertion in file E
4. Calculate the length of characters in message M and write it in 2-digit format
5. Convert each digit into 6-bit binary so there are 12 bits
6. Convert M to binary (8-bit format) and add 12 bits (from Step 5) to the start of M
7. For the total number of bits in Step 6:
   - **If** the total bits modulo 3 = 0 then continue to Step 8
   - **If** the total bits modulo 3 = 1 then add 2 bits with the value 0 at the end of M then continue Step 8
   - **If** the total bits modulo 3 = 2 then add 1 bit with the value 0 at the end of M then continue Step 8
8. If the maximum insertion capacity is greater than or equal to the total bits from Step 7 then go to step 9, otherwise go directly to step 12
9. Divide M into blocks consisting of 3-bit binary
10. For each block of message M:
   - **If** block = 000, then use the original Unicode of the letter
   - **Otherwise,** use another Unicode according to the 3-bit binary represented by the other Unicode in table 3
11. Give file S
12. Finish

Algorithm 3: Representing 4-Bit Binary
Input: Excel file E and message M
Output: Stego file S
Steps:
1. Open file E
2. Perform scan of selected letters in file E
3. Count the number of letters in Step 2 to check the maximum capacity (in bits) of insertion in file E
4. Calculate the length of characters in message M and write it in 2-digit format
5. Convert each digit into 8-bit binary so there are 16 bits
6. Convert M to binary (8-bit format) and add 16 bits (from Step 5) to the start of M
7. If the maximum insertion capacity is greater than or equal to the total bits from Step 6 then go to step 8, otherwise go directly to step 11
8. Divide M into blocks consisting of 4-bit binary
9. For each block of message M:
   - **If** block = 0000, then use the original Unicode of the letter
   - **Otherwise,** use another Unicode according to the 4-bit binary represented by the other Unicode in table 4
10. Give file S
11. Finish

Journal of Theoretical and Applied Information Technology
30th June 2024. Vol.102. No. 12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NO	Provinsi	Kode Satker	Pejabat Saat Ini	
			Jabatan	Nama	
1	JAWA TIMUR	xxxxxx	Kepala Satker Balai Prasarana Permukiman Wilayah Jawa Timur	Pratama Ade Kurniawan	xxxx
2	JAWA TIMUR	xxxxxx	Pejabat Pembuat Komitmen Perencanaan	Apri Ayu Angelica	xxxx
3	JAWA TIMUR	xxxxxx	Pejabat Pembuat Komitmen Tata Laksana	M. Sarengat	xxxx

Figure 4: Example of a Cover File (Jawa Timur.xlsx) to be Inserted

The letters colored red in Figure 4 depicted examples of letters that had been selected, namely those that appeared most frequently in Indonesian (a, n, e, and i) which would be inserted with certain binary messages.

For example, the cover file used here was an excel file from East Java Province (Jawa Timur.xlsx). The message inserted as a sign was "Oleh SESditCK tanggal 29-02-2024 pukul 14:15:16 PM" using steganography based on Unicode characters utilizing 4 letters to represent 3-bit binary

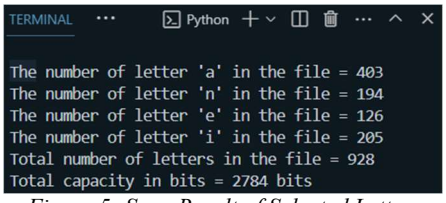


Figure 5: Scan Result of Selected Letters

First, Scan was carried out for the number of selected letters in the file with the results which could be seen in Figure 5. The total number was 928 so the insertion capacity was 928 x 3 = 2784 bits.

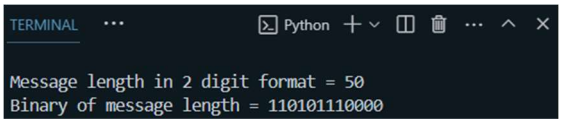


Figure 6: Result of Marker Length

Then calculated the length of the characters in the message and wrote it in 2-digit format and converted each digit to binary with 6-bit format so that there were 12 bits as in Figure 6.

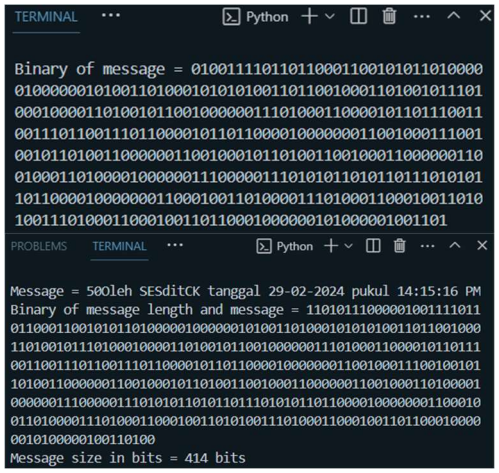


Figure 7: Binary of Inserted Message

After that, converted the message into binary form (8-bit format) and add the 12 bits at the beginning of the message so that the actual message inserted was "50Oleh SESditCK tanggal 29-02-2024 pukul 14:15:16 PM". Because the total bits were 414 as in Figure 7 so 414 modulo 3 produced 0 then went straight to the next step.

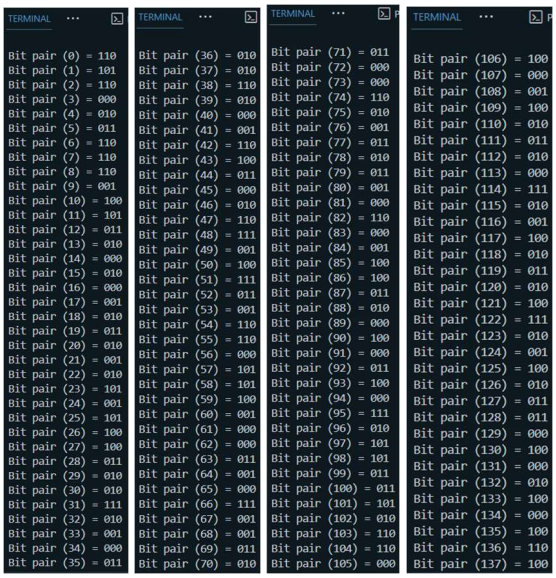


Figure 8: Bit Blocks of Message

Because the insertion capacity (2784 bits) was greater than or equal to the total bits of the entire message (414 bits), The binary message was divided into blocks which consisted of 3-bit binary as in Figure 8.

In Table 5 it could be seen that for each block of the message binary, if block = 000, then used the original Unicode of the letter and if not then used another Unicode according to the 3-bit

binary represented by the other Unicode in table 3. Finally, the stego file was saved with the name "Jawa Timur Stego.xlsx".

Table 5: Unicode Character Replacement Overview

Cover File	i	n	i	e	a …
Unicode before Hiding	U+ 0069	U+ 006E	U+ 0069	U+ 0065	U+ 0061 …
Binary Block	110	101	110	000	010 …
Unicode after Hiding	U+ 003C	U+ 007C	U+ 003C	U+ 0065	U+ 0024 …

3.4 Message Extraction

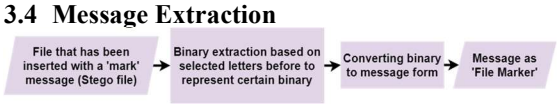


Figure 9: Flowchart of Message Extraction Overview

In Figure 9, flowchart of Message Extraction Overview can be seen. Marker in the form of messages is extracted from the stego file. The message comes from a binary collection that is extracted using the Latin letters that appear most frequently in Indonesian (a, n, e, and i) to represent 2-bit, 3-bit, and 4-bit binary. The different binary representation gives different extraction algorithms.

Algorithm 4: Representing 2-Bit Binary
Input: Stego file S
Output: Message M and message length L
Steps:
1. Open file S
2. Extract L which is a 2-digit number (8 bits each) in the first 8 selected letters (1 letter represents 2-bit binary)
3. For each selected letter:
   - **If** the Unicode of the selected letter = the Unicode of the original letter then L = 00
   - **Otherwise,** check the value of L in table 2 for the 2-bit binary represented by the other Unicode
4. Divide L into blocks consisting of 8-bit binary then convert it into ASCII values so that the digit values can be read
5. Apart from the first 8 letters, scan the selected letters again in the S file as much as ((8 x value of the digit) / 2) letters
6. For each selected letter:
   - **If** the Unicode of the selected letter = the Unicode of the original letter then M = 00
   - **Otherwise,** check the value of M in table 2 for the 2-bit binary represented by the other Unicode
7. Divide M into blocks consisting of 8-bit binary then convert to ASCII values so that the message can be read
8. Give message M
9. Finish

Algorithm 5: Representing 3-Bit Binary
Input: Stego file S
Output: Message M and message length L
Steps:
1. Open file S
2. Extract L which is a 2-digit number (6 bits each) in the first 4 selected letters (1 letter represents 3-bit binary)
3. For each selected letter:
   - **If** the Unicode of the selected letter = the Unicode of the original letter then L = 000
   - **Otherwise,** check the value of L in table 3 for the 3-bit binary represented by the other Unicode
4. Divide L into blocks consisting of 6-bit binary then convert it into ASCII values so that the digit values can be read
5. Apart from the first 4 letters, scan the selected letters again in the S file as much as ((8 x value of the digit + y) / 3) letters, where the value of y:
   - If 8 x value of the digit modulo 3 = 0 then y=0
   - If 8 x value of the digit modulo 3 = 1 then y=2
   - If 8 x value of the digit modulo 3 = 2 then y=1
6. For each selected letter:
   - **If** the Unicode of the selected letter = the Unicode of the original letter then M = 000
   - **Otherwise,** check the value of M in table 3 for the 3-bit binary represented by the other Unicode
7. From all the bits of the M value, remove y bit at the end
8. Divide M into blocks consisting of 8-bit binary then convert to ASCII values so that the message can be read
9. Give message M
10. Finish

Algorithm 6: Representing 4-Bit Binary
Input: Stego file S
Output: Message M and message length L
Steps:
1. Open file S

2. Extract L which is a 2-digit number (8 bits each) in the first 4 selected letters (1 letter represents 4-bit binary)
3. For each selected letter:
   - **If** the Unicode of the selected letter = the Unicode of the original letter then L = 0000
   - **Otherwise,** check the value of L in table 4 for the 4-bit binary represented by the other Unicode
4. Divide L into blocks consisting of 8-bit binary then convert it into ASCII values so that the digit values can be read
5. Apart from the first 4 letters, scan the selected letters again in the S file as much as ((8 x value of the digit) / 4) letters
6. For each selected letter:
   - **If** the Unicode of the selected letter = the Unicode of the original letter then M = 0000
   - **Otherwise,** check the value of M in table 4 for the 4-bit binary represented by the other Unicode
7. Divide M into blocks consisting of 8-bit binary then convert to ASCII values so that the message can be read
8. Give message M
9. Finish

For example, the stego file used here was “Jawa Timur Stego.xlsx” which previously had the message "50Oleh SESditCK tanggal 29-02-2024 pukul 14:15:16 PM" inserted. Message as mark was extracted using steganography based on Unicode characters utilizing 4 letters to represent 3-bit binary.

Table 6: Binary Extraction of Message Length

Stego File (SF)	i	n	i	e
Unicode hidden in SF	U+003C	U+007C	U+003C	U+0065
Binary Block	110	101	110	000

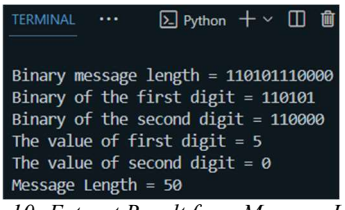


Figure 10: Extract Result from Message Length

First of all, the length of the message which was a 2-digit number (6 bits each) was extracted in the first 4 selected letters (1 letter represented 3-bit binary). In Table 6, it could be seen that for each selected letter, if the Unicode of the selected letter was the same as the Unicode of the original letter, then the message binary was 000. If not, then checked the message binary value in table 3 for the 3-bit binary represented by the other Unicode.

Then, divided the binary message into blocks consisting of 6-bit binary then converted it into ASCII values so that the digit values could be read by humans as in Figure 10.

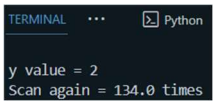


Figure 11: Additional Number of Letters to Scan

Apart from the first 4 letters, scanned the selected letters again in the Stego file for ((8 x 50 + 2) / 3 = 134) letters as in Figure 11.

Table 7: Binary Extraction of Message

Stego File (SF)	a	e	e	a	a …
Unicode hidden in SF	U+ 0024	U+ 0060	U+ 0021	U+ 005C	U+ 005C …
Binary Block	010	011	110	110	110 …

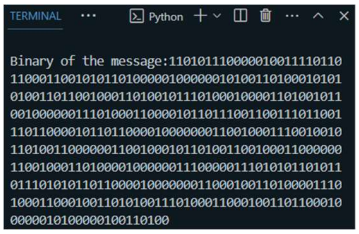


Figure 12: Bit Collection from the Length and Content of the Extraction Message

In Table 7, it could be seen that for each selected letter, if the Unicode of the selected letter was the same as the Unicode of the original letter, then the message binary was 000. If not, checked the message binary value in table 3 for the 3-bit binary represented by the other Unicode. After completing the scanning, bit collection from the Length and Content of the Extraction Message obtained which could be seen in Figure 12. Later the 2 bits at the end of the message are removed (because y=2).

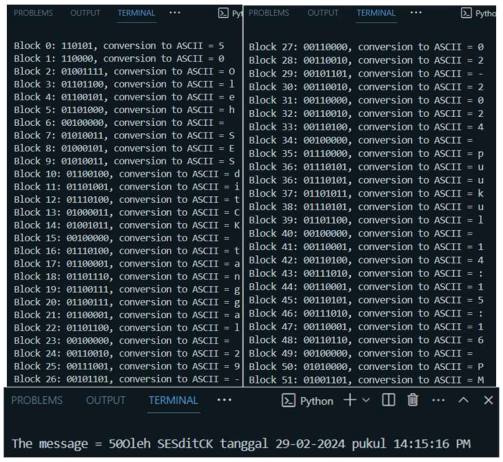


Figure 13: Result of Message Extraction

After that, divided the message binary into blocks which consist of 6-bit binary for the first 12 bits of the message (2-digit of message length) and 8-bit binary after that for the original message. After that, converted it to ASCII values so that the message could be read by humans as in Figure 13. Finally, combined the result of ASCII values from these blocks so that the message was obtained which was 100% the same as the inserted message, namely "50Oleh SESditCK tanggal 29-02-2024 pukul 14:15:16 PM".

3.5 Steganography Performance Testing

The message inserted as a mark contained 3 main contents, namely user name, date, and time. Steganography performance testing on 34 excel files seen from the parameters Size Increasing Ratio (SIR), Capacity, and Security Ratio. Each file was used in text steganography based on Unicode characters utilizing 3 or 4 letters to represent 2-bit, 3-bit, or 4-bit binary.

3.6 Steganography Performance Evaluation

The evaluation carried out was comparing the results of the average Size Increasing Ratio (SIR), Capacity, and Security Ratio parameters. It was seen which algorithm has the best results among all.

4. RESULTS AND DISCUSSION

4.1 Result from Steganography Performance Testing

In Table 8, it could be seen that there were two message sizes inserted in this research namely 52 bytes and 99 bytes to see the parameters Capacity, Size Increasing Ratio (SIR), and Security Ratio. Previously, each message was tested against 34 cover files which used text steganography based on Unicode characters utilizing 3 or 4 letters representing 2-bit, 3-bit, or 4-bit binary. The first message was a message with size of 52 bytes containing "50Oleh SESditCK tanggal 29-02-2024 pukul 14:15:16 PM". The number 50 in the message was useful during the message extraction process and indicated that 50 characters had been inserted. The second message was a message with size of 99 bytes containing "97This official election file was downloaded by Pras from the application on 02-29-2024 14:15:16 PM". The number 97 in the message was useful during the message extraction process and indicated that 97 characters had been inserted.

Table 8: Test Results for Capacity, Size Increasing Ratio (SIR), and Security Ratio

Experiment	Total Number of Selected Letters in Cover File	Capacity (bits)	Size of Cover File (bytes)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Total Number of Cover File Characters	Total Number of Stego File Characters	Security Ratio (%)
Steganography utilized 3 letters (a, n, and e) to represent 2-bit binary												
1	403	806	39490	52	39490	0	99	39490	0	2784	2784	100
2	845	1690	52372	52	52372	0	99	52372	0	5785	5785	100
3	449	898	39243	52	39243	0	99	39243	0	3018	3018	100
4	418	836	37426	52	37426	0	99	37426	0	2635	2635	100
5	425	850	38987	52	38987	0	99	38987	0	3182	3182	100
6	439	878	37451	52	37451	0	99	37451	0	2681	2681	100
7	464	928	39670	52	39670	0	99	39670	0	3109	3109	100
8	443	886	38302	52	38302	0	99	38302	0	2821	2821	100
9	413	826	37520	52	37520	0	99	37520	0	2729	2729	100
10	400	800	37503	52	37503	0	99	37503	0	2754	2754	100
11	502	1004	40321	52	40321	0	99	40321	0	3313	3313	100
12	439	878	37480	52	37480	0	99	37480	0	2773	2773	100
13	725	1450	49346	52	49346	0	99	49346	0	4862	4862	100
14	901	1802	54471	52	54471	0	99	54471	0	6087	6087	100

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

Experiment	Total Number of Selected Letters in Cover File	Capacity (bits)	Size of Cover File (bytes)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Total Number of Cover File Characters	Total Number of Stego File Characters	Security Ratio (%)
15	458	916	39901	52	39901	0	99	39901	0	3280	3280	100
16	723	1446	47987	52	47987	0	99	47987	0	4932	4932	100
17	720	1440	47277	52	47277	0	99	47277	0	4518	4518	100
18	478	956	38253	52	38253	0	99	38253	0	2874	2874	100
19	462	924	37959	52	37959	0	99	37959	0	3042	3042	100
20	717	1434	51968	52	51968	0	99	51968	0	5132	5132	100
21	485	970	40696	52	40696	0	99	40696	0	3310	3310	100
22	447	894	38566	52	38566	0	99	38566	0	2974	2974	100
23	466	932	39798	52	39798	0	99	39798	0	3237	3237	100
24	967	1934	54186	52	54186	0	99	54186	0	6111	6111	100
25	478	956	38940	52	38940	0	99	38940	0	3162	3162	100
26	478	956	39713	52	39713	0	99	39713	0	3146	3146	100
27	475	950	39861	52	39861	0	99	39861	0	2997	2997	100
28	445	890	38354	52	38354	0	99	38354	0	2975	2975	100
29	643	1286	45958	52	45958	0	99	45958	0	4405	4405	100
30	459	918	37656	52	37656	0	99	37656	0	2823	2823	100
31	462	924	37818	52	37818	0	99	37818	0	2880	2880	100
32	409	818	37692	52	37692	0	99	37692	0	2880	2880	100
33	594	1188	44621	52	44621	0	99	44621	0	3899	3899	100
34	409	818	37552	52	37552	0	99	37552	0	2782	2782	100
Average	**530.62**	**1061.24**	**41598.18**	**52**	**41598.18**	**0**	**99**	**41598.18**	**0**	**3526.24**	**3526.24**	**100**
Steganography utilized 4 letters (a, n, e, and i) to represent 2-bit binary												
1	494	988	39490	52	39490	0	99	39490	0	2784	2784	100
2	1052	2104	52372	52	52372	0	99	52372	0	5785	5785	100
3	533	1066	39243	52	39243	0	99	39243	0	3018	3018	100
4	525	1050	37426	52	37426	0	99	37426	0	2635	2635	100
5	545	1090	38987	52	38987	0	99	38987	0	3182	3182	100
6	533	1066	37451	52	37451	0	99	37451	0	2681	2681	100
7	553	1106	39670	52	39670	0	99	39670	0	3109	3109	100
8	539	1078	38302	52	38302	0	99	38302	0	2821	2821	100
9	512	1024	37520	52	37520	0	99	37520	0	2729	2729	100
10	504	1008	37503	52	37503	0	99	37503	0	2754	2754	100
11	624	1248	40321	52	40321	0	99	40321	0	3313	3313	100
12	531	1062	37480	52	37480	0	99	37480	0	2773	2773	100
13	913	1826	49346	52	49346	0	99	49346	0	4862	4862	100
14	1104	2208	54471	52	54471	0	99	54471	0	6087	6087	100
15	573	1146	39901	52	39901	0	99	39901	0	3280	3280	100
16	928	1856	47987	52	47987	0	99	47987	0	4932	4932	100
17	908	1816	47277	52	47277	0	99	47277	0	4518	4518	100
18	564	1128	38253	52	38253	0	99	38253	0	2874	2874	100
19	571	1142	37959	52	37959	0	99	37959	0	3042	3042	100
20	900	1800	51968	52	51968	0	99	51968	0	5132	5132	100
21	592	1184	40696	52	40696	0	99	40696	0	3310	3310	100
22	567	1134	38566	52	38566	0	99	38566	0	2974	2974	100
23	577	1154	39798	52	39798	0	99	39798	0	3237	3237	100
24	1179	2358	54186	52	54186	0	99	54186	0	6111	6111	100
25	572	1144	38940	52	38940	0	99	38940	0	3162	3162	100
26	558	1116	39713	52	39713	0	99	39713	0	3146	3146	100
27	557	1114	39861	52	39861	0	99	39861	0	2997	2997	100
28	542	1084	38354	52	38354	0	99	38354	0	2975	2975	100
29	771	1542	45958	52	45958	0	99	45958	0	4405	4405	100
30	540	1080	37656	52	37656	0	99	37656	0	2823	2823	100
31	559	1118	37818	52	37818	0	99	37818	0	2880	2880	100
32	502	1004	37692	52	37692	0	99	37692	0	2880	2880	100
33	735	1470	44621	52	44621	0	99	44621	0	3899	3899	100
34	497	994	37552	52	37552	0	99	37552	0	2782	2782	100
Average	**651.59**	**1303.18**	**41598.18**	**52**	**41598.18**	**0**	**99**	**41598.18**	**0**	**3526.24**	**3526.24**	**100**

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

Experiment	Total Number of Selected Letters in Cover File	Capacity (bits)	Size of Cover File (bytes)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Total Number of Cover File Characters	Total Number of Stego File Characters	Security Ratio (%)
Steganography utilized 3 letters (a, n, and e) to represent 3-bit binary												
1	403	1209	39490	52	39490	0	99	39490	0	2784	2784	100
2	845	2535	52372	52	52372	0	99	52372	0	5785	5785	100
3	449	1347	39243	52	39243	0	99	39243	0	3018	3018	100
4	418	1254	37426	52	37426	0	99	37426	0	2635	2635	100
5	425	1275	38987	52	38987	0	99	38987	0	3182	3182	100
6	439	1317	37451	52	37451	0	99	37451	0	2681	2681	100
7	464	1392	39670	52	39670	0	99	39670	0	3109	3109	100
8	443	1329	38302	52	38302	0	99	38302	0	2821	2821	100
9	413	1239	37520	52	37520	0	99	37520	0	2729	2729	100
10	400	1200	37503	52	37503	0	99	37503	0	2754	2754	100
11	502	1506	40321	52	40321	0	99	40321	0	3313	3313	100
12	439	1317	37480	52	37480	0	99	37480	0	2773	2773	100
13	725	2175	49346	52	49346	0	99	49346	0	4862	4862	100
14	901	2703	54471	52	54471	0	99	54471	0	6087	6087	100
15	458	1374	39901	52	39901	0	99	39901	0	3280	3280	100
16	723	2169	47987	52	47987	0	99	47987	0	4932	4932	100
17	720	2160	47277	52	47277	0	99	47277	0	4518	4518	100
18	478	1434	38253	52	38253	0	99	38253	0	2874	2874	100
19	462	1386	37959	52	37959	0	99	37959	0	3042	3042	100
20	717	2151	51968	52	51968	0	99	51968	0	5132	5132	100
21	485	1455	40696	52	40696	0	99	40696	0	3310	3310	100
22	447	1341	38566	52	38566	0	99	38566	0	2974	2974	100
23	466	1398	39798	52	39798	0	99	39798	0	3237	3237	100
24	967	2901	54186	52	54186	0	99	54186	0	6111	6111	100
25	478	1434	38940	52	38940	0	99	38940	0	3162	3162	100
26	478	1434	39713	52	39713	0	99	39713	0	3146	3146	100
27	475	1425	39861	52	39861	0	99	39861	0	2997	2997	100
28	445	1335	38354	52	38354	0	99	38354	0	2975	2975	100
29	643	1929	45958	52	45958	0	99	45958	0	4405	4405	100
30	459	1377	37656	52	37656	0	99	37656	0	2823	2823	100
31	462	1386	37818	52	37818	0	99	37818	0	2880	2880	100
32	409	1227	37692	52	37692	0	99	37692	0	2880	2880	100
33	594	1782	44621	52	44621	0	99	44621	0	3899	3899	100
34	409	1227	37552	52	37552	0	99	37552	0	2782	2782	100
Average	**530.62**	**1591.85**	**41598.18**	**52**	**41598.18**	**0**	**99**	**41598.18**	**0**	**3526.24**	**3526.24**	**100**
Steganography utilized 4 letters (a, n, e, and i) to represent 3-bit binary												
1	494	1482	39490	52	39490	0	99	39490	0	2784	2784	100
2	1052	3156	52372	52	52372	0	99	52372	0	5785	5785	100
3	533	1599	39243	52	39243	0	99	39243	0	3018	3018	100
4	525	1575	37426	52	37426	0	99	37426	0	2635	2635	100
5	545	1635	38987	52	38987	0	99	38987	0	3182	3182	100
6	533	1599	37451	52	37451	0	99	37451	0	2681	2681	100
7	553	1659	39670	52	39670	0	99	39670	0	3109	3109	100
8	539	1617	38302	52	38302	0	99	38302	0	2821	2821	100
9	512	1536	37520	52	37520	0	99	37520	0	2729	2729	100
10	504	1512	37503	52	37503	0	99	37503	0	2754	2754	100
11	624	1872	40321	52	40321	0	99	40321	0	3313	3313	100
12	531	1593	37480	52	37480	0	99	37480	0	2773	2773	100
13	913	2739	49346	52	49346	0	99	49346	0	4862	4862	100
14	1104	3312	54471	52	54471	0	99	54471	0	6087	6087	100
15	573	1719	39901	52	39901	0	99	39901	0	3280	3280	100
16	928	2784	47987	52	47987	0	99	47987	0	4932	4932	100
17	908	2724	47277	52	47277	0	99	47277	0	4518	4518	100
18	564	1692	38253	52	38253	0	99	38253	0	2874	2874	100
19	571	1713	37959	52	37959	0	99	37959	0	3042	3042	100
20	900	2700	51968	52	51968	0	99	51968	0	5132	5132	100
21	592	1776	40696	52	40696	0	99	40696	0	3310	3310	100

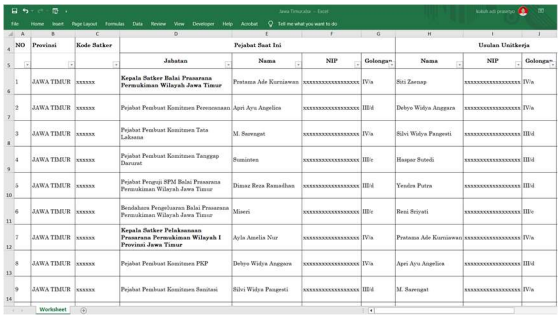
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Experiment	Total Number of Selected Letters in Cover File	Capacity (bits)	Size of Cover File (bytes)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Total Number of Cover File Characters	Total Number of Stego File Characters	Security Ratio (%)
22	567	1701	38566	52	38566	0	99	38566	0	2974	2974	100
23	577	1731	39798	52	39798	0	99	39798	0	3237	3237	100
24	1179	3537	54186	52	54186	0	99	54186	0	6111	6111	100
25	572	1716	38940	52	38940	0	99	38940	0	3162	3162	100
26	558	1674	39713	52	39713	0	99	39713	0	3146	3146	100
27	557	1671	39861	52	39861	0	99	39861	0	2997	2997	100
28	542	1626	38354	52	38354	0	99	38354	0	2975	2975	100
29	771	2313	45958	52	45958	0	99	45958	0	4405	4405	100
30	540	1620	37656	52	37656	0	99	37656	0	2823	2823	100
31	559	1677	37818	52	37818	0	99	37818	0	2880	2880	100
32	502	1506	37692	52	37692	0	99	37692	0	2880	2880	100
33	735	2205	44621	52	44621	0	99	44621	0	3899	3899	100
34	497	1491	37552	52	37552	0	99	37552	0	2782	2782	100
Average	**651.59**	**1954.76**	**41598.18**	**52**	**41598.18**	**0**	**99**	**41598.18**	**0**	**3526.24**	**3526.24**	**100**
Steganography utilized 3 letters (a, n, and e) to represent 4-bit binary												
1	403	1612	39490	52	39548	0.15	99	39607	0.30	2784	2784	100
2	845	3380	52372	52	52430	0.11	99	52489	0.22	5785	5785	100
3	449	1796	39243	52	39301	0.15	99	39360	0.30	3018	3018	100
4	418	1672	37426	52	37484	0.15	99	37543	0.31	2635	2635	100
5	425	1700	38987	52	39045	0.15	99	39104	0.30	3182	3182	100
6	439	1756	37451	52	37509	0.15	99	37568	0.31	2681	2681	100
7	464	1856	39670	52	39728	0.15	99	39787	0.29	3109	3109	100
8	443	1772	38302	52	38360	0.15	99	38419	0.31	2821	2821	100
9	413	1652	37520	52	37578	0.15	99	37637	0.31	2729	2729	100
10	400	1600	37503	52	37561	0.15	99	37620	0.31	2754	2754	100
11	502	2008	40321	52	40379	0.14	99	40438	0.29	3313	3313	100
12	439	1756	37480	52	37538	0.15	99	37597	0.31	2773	2773	100
13	725	2900	49346	52	49404	0.12	99	49463	0.24	4862	4862	100
14	901	3604	54471	52	54529	0.11	99	54588	0.21	6087	6087	100
15	458	1832	39901	52	39959	0.15	99	40018	0.29	3280	3280	100
16	723	2892	47987	52	48045	0.12	99	48104	0.24	4932	4932	100
17	720	2880	47277	52	47335	0.12	99	47394	0.25	4518	4518	100
18	478	1912	38253	52	38311	0.15	99	38370	0.31	2874	2874	100
19	462	1848	37959	52	38017	0.15	99	38076	0.31	3042	3042	100
20	717	2868	51968	52	52026	0.11	99	52085	0.23	5132	5132	100
21	485	1940	40696	52	40754	0.14	99	40813	0.29	3310	3310	100
22	447	1788	38566	52	38624	0.15	99	38683	0.30	2974	2974	100
23	466	1864	39798	52	39856	0.15	99	39915	0.29	3237	3237	100
24	967	3868	54186	52	54244	0.11	99	54303	0.22	6111	6111	100
25	478	1912	38940	52	38998	0.15	99	39057	0.30	3162	3162	100
26	478	1912	39713	52	39771	0.15	99	39830	0.29	3146	3146	100
27	475	1900	39861	52	39919	0.15	99	39978	0.29	2997	2997	100
28	445	1780	38354	52	38412	0.15	99	38471	0.31	2975	2975	100
29	643	2572	45958	52	46016	0.13	99	46075	0.25	4405	4405	100
30	459	1836	37656	52	37714	0.15	99	37773	0.31	2823	2823	100
31	462	1848	37818	52	37876	0.15	99	37935	0.31	2880	2880	100
32	409	1636	37692	52	37750	0.15	99	37809	0.31	2880	2880	100
33	594	2376	44621	52	44679	0.13	99	44738	0.26	3899	3899	100
34	409	1636	37552	52	37610	0.15	99	37669	0.31	2782	2782	100
Average	**530.62**	**2122.47**	**41598.18**	**52**	**41656.18**	**0.14**	**99**	**41715.18**	**0.29**	**3526.24**	**3526.24**	**100**
Steganography utilized 4 letters (a, n, e, and i) to represent 4-bit binary												
1	494	1976	39490	52	39548	0.15	99	39607	0.30	2784	2784	100
2	1052	4208	52372	52	52430	0.11	99	52489	0.22	5785	5785	100
3	533	2132	39243	52	39301	0.15	99	39360	0.30	3018	3018	100
4	525	2100	37426	52	37484	0.15	99	37543	0.31	2635	2635	100
5	545	2180	38987	52	39045	0.15	99	39104	0.30	3182	3182	100
6	533	2132	37451	52	37509	0.15	99	37568	0.31	2681	2681	100

Journal of Theoretical and Applied Information Technology
30th June 2024. Vol.102. No. 12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Experiment	Total Number of Selected Letters in Cover File	Capacity (bits)	Size of Cover File (bytes)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Message Size (bytes)	Size of Stego File (bytes)	Size Increasing Ratio (%)	Total Number of Cover File Characters	Total Number of Stego File Characters	Security Ratio (%)
7	553	2212	39670	52	39728	0.15	99	39787	0.29	3109	3109	100
8	539	2156	38302	52	38360	0.15	99	38419	0.31	2821	2821	100
9	512	2048	37520	52	37578	0.15	99	37637	0.31	2729	2729	100
10	504	2016	37503	52	37561	0.15	99	37620	0.31	2754	2754	100
11	624	2496	40321	52	40379	0.14	99	40438	0.29	3313	3313	100
12	531	2124	37480	52	37538	0.15	99	37597	0.31	2773	2773	100
13	913	3652	49346	52	49404	0.12	99	49463	0.24	4862	4862	100
14	1104	4416	54471	52	54529	0.11	99	54588	0.21	6087	6087	100
15	573	2292	39901	52	39959	0.15	99	40018	0.29	3280	3280	100
16	928	3712	47987	52	48045	0.12	99	48104	0.24	4932	4932	100
17	908	3632	47277	52	47335	0.12	99	47394	0.25	4518	4518	100
18	564	2256	38253	52	38311	0.15	99	38370	0.31	2874	2874	100
19	571	2284	37959	52	38017	0.15	99	38076	0.31	3042	3042	100
20	900	3600	51968	52	52026	0.11	99	52085	0.23	5132	5132	100
21	592	2368	40696	52	40754	0.14	99	40813	0.29	3310	3310	100
22	567	2268	38566	52	38624	0.15	99	38683	0.30	2974	2974	100
23	577	2308	39798	52	39856	0.15	99	39915	0.29	3237	3237	100
24	1179	4716	54186	52	54244	0.11	99	54303	0.22	6111	6111	100
25	572	2288	38940	52	38998	0.15	99	39057	0.30	3162	3162	100
26	558	2232	39713	52	39771	0.15	99	39830	0.29	3146	3146	100
27	557	2228	39861	52	39919	0.15	99	39978	0.29	2997	2997	100
28	542	2168	38354	52	38412	0.15	99	38471	0.31	2975	2975	100
29	771	3084	45958	52	46016	0.13	99	46075	0.25	4405	4405	100
30	540	2160	37656	52	37714	0.15	99	37773	0.31	2823	2823	100
31	559	2236	37818	52	37876	0.15	99	37935	0.31	2880	2880	100
32	502	2008	37692	52	37750	0.15	99	37809	0.31	2880	2880	100
33	735	2940	44621	52	44679	0.13	99	44738	0.26	3899	3899	100
34	497	1988	37552	52	37610	0.15	99	37669	0.31	2782	2782	100
Average	**651.59**	**2606.35**	**41598.18**	**52**	**41656.18**	**0.14**	**99**	**41715.18**	**0.29**	**3526.24**	**3526.24**	**100**

Each message was tested against 34 cover files using text steganography based on Unicode characters utilizing 3 or 4 letters representing 2-bit, 3-bit, or 4-bit binary resulting in 100% message similarity for the inserted message and the extracted message. Apart from that, it also produced good invisibility because there was no visible difference between the appearance of the cover file and the stego file (a file that had an inserted message) as could be seen in Figure 14.



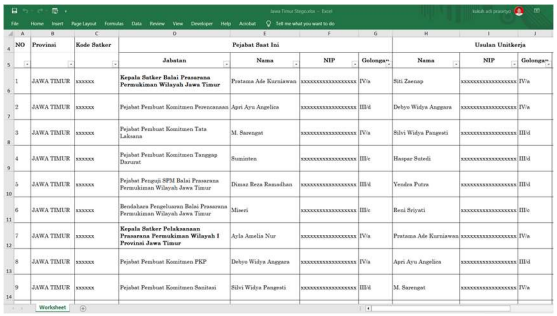


Figure 14: Example of Appearances from Cover File (above) and Stego File (below)

4.2 Result from Steganography Performance Evaluation

The evaluation that had been carried out was comparing the results of the average parameters from Capacity, Size Increasing Ratio (SIR), and Security Ratio which could be seen in Table 9.

Table 9: Evaluation Results of Capacity, Size Increasing Ratio (SIR), and Security Ratio

Letter	Represented Binary	Capacity (bits)	SIR for 52 Bytes Message (%)	SIR for 99 Bytes Message (%)	Security Ratio (%)
a, n, e	2-bit	1061.24	0	0	100
	3-bit	1591.85	0	0	100
	4-bit	2122.47	0.14	0.29	100
a, n, e, i	2-bit	1303.18	0	0	100
	3-bit	1954.76	0	0	100
	4-bit	2606.35	0.14	0.29	100

If the Capacity was greater, the steganography algorithm was better. If the Size Increasing Ratio (SIR) is smaller, the steganography algorithm was better. If the Security Ratio was greater, the steganography algorithm was better too.

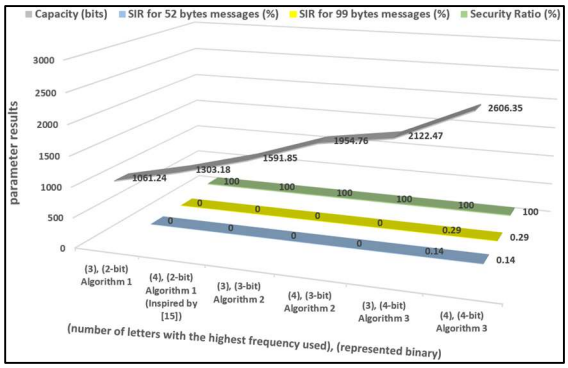


Figure 15: Chart of the Difference in Results for Each Parameter

In figure 15, you could see the differences in text steganography results in the excel file for each parameter. The Security Ratio parameter had the same results for all of them. The three best results when viewed from the Capacity parameter were the steganography Algorithm utilizing 4 letters to represent 4-bit binary, the steganography Algorithm utilizing 3 letters to represent 4-bit binary, and the steganography Algorithm utilizing 4 letters to represent 3-bit binary. However, if the Size Increasing Ratio (SIR) parameter was seen, the only one that had result of 0% from the best three was the steganography Algorithm utilizing 4 letters to represent 3-bit binary.

SIR from the steganography Algorithm utilizing 3 or 4 letters to represent 4-bit binary did not have a value of 0% because there were letter characters worth 1 byte which were replaced by Unicode from other characters that were not worth 1 byte. This occurred because of the limited number of Unicode characters worth 1 byte according to Table 4. In contrast to the steganography Algorithm utilizing 4 letters to represent 3-bit binary which had 0% for SIR because it replaced letter characters with value of 1 byte with Unicode from other characters which also had value of 1 byte according to Table 3.

4.3 Result from Comparison of Proposed Method with Similar Research

The proposed method here was the steganography Algorithm based on Unicode characters utilizing 4 letters to represent 3-bit binary. In figure 16, result of the proposed method which used 235 Bytes message with Sumatera Utara.xlsx as cover file could be seen.

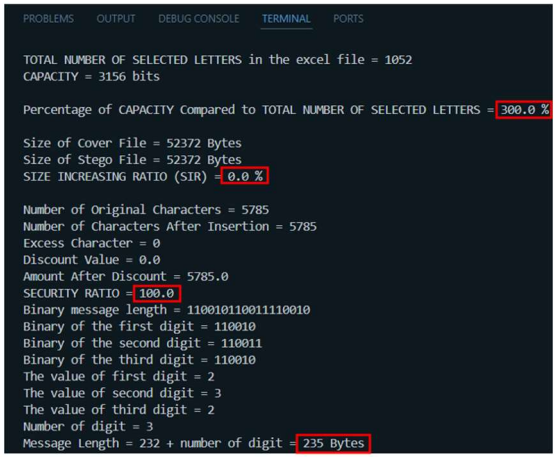


Figure 16: Result of the Proposed Method which used 235 Bytes Message

Comparison from proposed method with previous research about Percentage of Capacity Compared to The Number of Selected Letters, Size Increasing Ratio (SIR), and Security Ratio which could be seen in Table 10.

Table 10: Comparison of Proposed Method with Previous Research

Method	Percentage of Capacity Compared to The Number of Selected Letters (%)	Size Increasing Ratio for 235 Bytes Message (%)	Security Ratio (%)
[14]	200%	18.4	100
[15]	200%	0	100
Proposed Method	300%	0	100

The Security Ratio parameter had the same results for all of them which was 100%. This happened because there were no adding characters but changing characters in [14], [15], and proposed method. When viewed from the Size Increasing Ratio parameter, [14] had Size Increasing Ratio not equal to 0%. This was a weakness because it could

make other people suspected that a message had been inserted. On the other hand, [15] and the proposed method had Size Increasing Ratio equal to 0%. So, the two best results when viewed from the Size Increasing Ratio parameter were [15] and the proposed method. However, if the Percentage of Capacity Compared to The Number of Selected Letters parameter was seen, the one that had better result from the best two was the proposed method. This was a strength for the proposed method because it could make the inserted message larger in size. This was also a novelty compared to previous research.

5. CONCLUSIONS

Conclusion obtained from this research include:

1. The way to insert a mark in an Indonesian Excel file that had confidential information texts was to give a message as a mark where the message was first converted into binary and then inserted using the Latin letters that appeared most frequently in Indonesian (a, n, e, and i). These letters were made to have different Unicode characters but with the same Latin letter appearance to represent certain binaries, for example 2-bit binary, then upgraded to 3-bit binary or 4-bit binary.
2. The message as a mark in the confidential information file was successfully inserted and the best Steganography Algorithm from the experiments that had been carried out was the Steganography Algorithm utilizing 4 letters to represent 3-bit binary which has an average 0% for Size Increasing Ratio (SIR), 100% for Security Ratio, and 1954.76 bits for Capacity (Capacity increased compared to steganography based on Unicode characters utilizing 3 letters to represent 3-bit binary, 3 letters to represent 2-bit binary and 4 letters to represent 2-bit binary).
3. The proposed method was better than similar research that had been carried out previously. The proposed method was better than research [14] in terms of Size Increasing Ratio and Percentage of Capacity Compared to The Number of Selected Letters and better than research [15] in terms of Percentage of Capacity Compared to The Number of Selected Letters.

REFERENCES:

[1] APJII. Survei APJII 66.8 Persen Warga RI Ogah Ganti Password Akun 2023. https://apjii.or.id/berita/d/survei-apjii-668-persen-warga-ri-ogah-ganti-password-akun (accessed June 27, 2023).

[2] Google Indonesia. Survei baru di Indonesia mendapati hampir 2 dari 3 orang pernah mengalami pelanggaran data pribadi atau mengenal orang yang pernah mengalaminya, tetapi 92% tetap kurang menjaga keamanan sandi 2021. https://indonesia.googleblog.com/2021/11/survei-baru-di-indonesia-mendapati.html (accessed June 27, 2023).

[3] Thabit R, Udzir NI, Yasin SM, Asmawi A, Roslan NA, Din R. A Comparative Analysis of Arabic Text Steganography. Applied Sciences 2021;11:1–32. https://doi.org/10.3390/app11156851.

[4] Baawi SS, Mokhtar MR, Sulaiman R. A Comparative Study on The Advancement of Text Steganography Techniques in Digital Media. ARPN Journal of Engineering and Applied Sciences 2018;13:1854–63.

[5] Khodher MAA, Khairi TWA. Review: A Comparison Steganography Between Texts and Images. 5th International Scientific Conference of Al-Khwarizmi Society, vol. 1591, IOP Publishing Ltd; 2020. https://doi.org/10.1088/1742-6596/1591/1/012024.

[6] Begum F, Suthoju GR. Types of Steganography for Secure Data Maintenance. Annals of RSCB 2021;25:2144–59.

[7] Baawi SS, Mokhtar MR, Sulaiman R. New Text Steganography Technique Based on A Set of Two-Letter Words. J Theor Appl Inf Technol 2017;95:6247–55.

[8] Martono, Irawan. Penggunaan Steganografi dengan Metode End of File (EOF) pada Digital Watermarking. Jurnal TICOM 2013;2:36–42.

[9] Nugroho PD, Wathoni M. Pengamanan Text Dengan Teknik Steganografi Menggunakan Metode Least Significant Bit (LSB). Jurnal IPSIKOM 2015;3.

[10] Salim Y, Azis H. Sistem Penanda Kepemilikan File Dokumen Menggunakan Metode Digital Watermark Pada File Penelitian Dosen Universitas Muslim Indonesia. Jurnal ILKOM 2017;9:161–6. https://doi.org/10.33096/ilkom.v9i2.125.161-166.

[11] Tupen DVPR, Sridaryanto WE, Hakim L. Penerapan Least Significant Bit untuk Penyisipian Penanda Pada Gambar. Jurnal

Infomedia 2020;5:10–6. https://doi.org/10.30811/jim.v5i1.1577.

[12] Khuzaifi A, Fauziah, Fitri I. Teknik Steganography untuk Menyisipkan Pesan pada Sebuah Citra Menggunakan Metode Least Significant Bit (LSB). Jurnal Teknologi Informasi Dan Komunikasi 2022;6:417–23. https://doi.org/10.35870/jtik.v6i3.461.

[13] Saber AS, Awadh WA. Steganography in Ms Excel Document Using Unicode System Characteristics. Journal of Basrah Researches 2013;39:10–9.

[14] Rahma AMS, Bhaya WS, Al-Nasrawi DA. Text Steganography Based On Unicode of Characters in Multilingual. International Journal of Engineering Research and Applications (IJERA) 2013;3:1153–65.

[15] Baawi SS, Nasrawi DA, Abdulameer LT. Improvement of "text steganography based on unicode of characters in multilingual" by custom font with special properties. IOP Conf Ser Mater Sci Eng, vol. 870, Institute of Physics Publishing; 2020, p. 1–10. https://doi.org/10.1088/1757-899X/870/1/012125.

[16] Shah A, Saidin AZ, Taha IF, Zeki AM, Bhatti Z. Similarities and Dissimilarities between Character Frequencies of Written Text of Melayu, English, and Indonesian Languages. International Conference on Advanced Computer Science Applications and Technologies, IEEE Computer Society; 2013, p. 192–4. https://doi.org/10.1109/ACSAT.2013.45.

[17] Alshamsi A, Albaloushi S, Alkhoori M, Almheiri H, Ababneh N. Enhancing Arabic Text Steganography Based on Unicode Features. International Journal of Computing and Digital Systems 2022;11:685–93. https://doi.org/10.12785/ijcds/110155.

[18] Thabit R, Udzir NI, Yasin SM, Asmawi A, Gutub AAA. CSNTSteg: Color Spacing Normalization Text Steganography Model to Improve Capacity and Invisibility of Hidden Data. IEEE Access 2022;10:65439–58. https://doi.org/10.1109/ACCESS.2022.3182712.

[19] Al-Nofaie S, Gutub A, Al-Ghamdi M. Enhancing Arabic Text Steganography for Personal Usage Utilizing Pseudo-spaces. Journal of King Saud University - Computer and Information Sciences 2021;33:963–74. https://doi.org/10.1016/j.jksuci.2019.06.010.

[20] Alanazi N, Khan E, Gutub A. Inclusion of Unicode Standard Seamless Characters to Expand Arabic Text Steganography for Secure Individual Uses. Journal of King Saud University - Computer and Information Sciences 2022;34:1343–56. https://doi.org/10.1016/j.jksuci.2020.04.011.

[21] Tommy, Siregar R, Lubis I, E AMarwan, H AMahmud, Harahap M. A Simple Compression Scheme Based on ASCII Value Differencing. International Conference on Mechanical, Electronics, Computer, and Industrial Technology, vol. 1007, Institute of Physics Publishing; 2018. https://doi.org/10.1088/1742-6596/1007/1/012022.

[22] Zaynalov N, Narzullaev U, Muhamadiev A, Qilichev D. About Using Unicode To Hide Information In A Text Document. AIP Conf Proc, vol. 2365, American Institute of Physics Inc.; 2021. https://doi.org/10.1063/5.0058952.

[23] Zaynalov NR, Narzullaev UKh, Muhamadieva AN, Rahmatullaev IR, Buranova RK. Combining Invisible Unicode Characters to Hide Information in a Text Document. International Journal on Informatics Visualization 2020;4:148–53.

[24] Al-Shakarchy ND, Al-Shahad HF, Al-Nasrawi DA. Cryptographic System Based On Unicode. J Phys Conf Ser, vol. 1032, Institute of Physics Publishing; 2018. https://doi.org/10.1088/1742-6596/1032/1/012049.