

RPC-Secret: Image Steganography Tool using Randomized GB-LSB and Parity Check

Tun Wen Kit¹, Kamaruddin Malik Mohamad^{1*}

¹Faculty of Computer Science & Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author Designation

DOI: <https://doi.org/10.30880/aitcs.2023.04.01.001>

Received 20 July 2022; Accepted 27 May 2023; Available online 30 June 2023

Abstract: Due to the insecure data transmission between two entities on the network channel, it causes data leakage of sensitive information. To overcome such problems, image steganography tools can help to hide the existence of sensitive data without the attacker's knowledge. In this project, an image steganography tool called RPC-Secret is developed using randomized Green Blue-Least Significant Bit (GB-LSB) and Parity Check. Randomized GB-LSB will randomly select the Least Significant Bit (LSB) of the green or blue layer from each pixel, and Parity Check will determine whether the LSB of chosen layer needs to be replaced. The tool is developed using Object-Oriented Analysis and Design (OOAD) methodology, C and C# programming languages. The RPC-Secret tool hides messages with a maximum size of 50 bytes in bitmap (BMP) images with an average peak signal-to-noise ratio (PSNR) value of 56.2563db. RPC-Secret can hide messages without affecting the image quality too much, which means that secret messages are not easily detected.

Keywords: Image Steganography, Randomized GB-LSB, Parity Check

1. Introduction

Image steganography is a process where a message will be embedded into an image that is normally known as a cover-image, whereby the message can be in the format of text, image, audio or video [1]. As the use of multimedia and the Internet increases, how to protect information has become an important issue. In order to ensure that communication is as secure as possible, steganography is gradually being valued and is beginning to be widely used in business and government.

Due to the large amount of data transmitted every second on the Internet, it is considered insecure [2]. This issue occurs because the number of people using the Internet is increasing thus causing an increase in data transmission in the communication channel, which provides more attack opportunities for attackers. The issue of protecting consumers' important data security in cloud computing has attracted much attention [3]. Cloud computing has been widely used in business to provide consumers with services for storing sensitive data. The communication channels through which most consumers

*Corresponding author: malik@uthm.edu.my

2023 UTHM Publisher. All rights reserved.

publisher.uthm.edu.my/periodicals/index.php/aitcs

upload sensitive data to the cloud do not provide any protection, so data leakage is prone to occur. In addition to cloud computing, e-medical systems also use insecure channels to exchange sensitive patient information, which has attracted the attention of attackers [4]. If an attacker tampered with the patient's sensitive data, it may cause the patient to be treated incorrectly or, more seriously, to death.

This study is therefore attempting to propose a tool called RPC-Secret, which can hide and extract secret data within a bitmap (BMP) image for secure data transmission. The objective of this project is to design, develop and test the proposed tool and it focuses on BMP cover-image type and plaintext input type only. This tool is expected to perform data hiding by using the randomized Green Blue-Least Significant Bit (GB-LSB) and Parity Check algorithm on a BMP image. The significance of this tool is to propose a new technique to secure the communication channel during the data transmission. Two existing tools will be chosen and compared with the proposed tool in terms of peak signal-to-noise ratio (PSNR).

For the rest of this paper, it is organized as follows. Section 2 will discuss the related work of image steganography. Next, Section 3 describes the methodology chosen used for project development. Section 4 presents the result and discussion for the implementation and testing of the tool and Section 5 discusses the conclusion and future work.

2. Related Work

This section discusses the related work of image steganography, existing image steganography tools and comparative study of existing tools with proposed tools.

2.1 Image Steganography

Image steganography aims to hide information within the cover-image with the minimum visual deformation and the maximum payload capacity to create confidential communication. The two classes of image steganography techniques are the spatial domain and the transform domain.

Spatial domain steganography technique is a method where the information will be hidden into the cover-image by replacing the Least Significant Bit (LSB) of the image pixel value directly [5]. Work by [6] proposed a colored image steganography to embed information based on the reference and two LSB algorithm versions. The version from two LSB algorithms is a plain LSB and conditional-based LSB versions. Kocak [7] has proposed a Couple Layered Security Model (CLSM) using a hybrid structure of cryptography and steganography. He has used a 128-bits keyword to encrypt the message for cryptography and the encrypted message will be embedded using the LSB algorithm into two bits for G and B of the red, green and blue (RGB) channels. Swain et al [8] proposed an adaptive steganography method that combines directional pixel value differencing (PVD) and LSB. The k-bit information is inserted into the upper left pixel from the 2x2 pixel block, and then the upper left base pixel and other remaining pixels in the horizontal and vertical edge directions will go through the PVD embedding process.

Transform domain steganography technique is a process of first transforming the cover-image from the spatial domain to the transform domain, and then embedding information with its frequency, and then transforming the cover-image into the spatial domain after embedding the information [9]. Gayatri proposed an image steganography technique that combines LSB with XORing and Discrete Cosine Transform (DCT)[10]. In this technique, the secret image will be hidden in the cover image, but first, the colour RGB image will be converted to a grayscale image, and then the secret image will be hidden using XOR transformation, LSB technique and DCT technique. Nevriyanto et al. [11] proposed a combination of Discrete Wavelet Transform (DWT) and singular value decomposition algorithm. A text file is used to convert into an image as a watermark and then embed it into the cover-image. An algorithm based on Finite Ridgelet Transform (FRT) and DWT is proposed in [12]. Ridgelet

coefficients of each colour channel of the cover-image are obtained from colour cover-image through FRT algorithms while different wavelet coefficients obtained through single-level DWT algorithms.

Table 1: Comparison between spatial domain and transform domain

	Spatial domain	Transform Domain
Computational Complexity	Low	High
Robustness	Low	High
Advantages	Easy to implement, high payload capacity	More robust against compression, cropping, and image processing on the image
Disadvantages	Low resistance to image processing, vulnerable to steganalysis attacks	High computational time, low payload capacity
Examples	LSB, PVD	DCT, DWT

Table 1 shows the comparison table between the spatial domain and the transform domain. The computational complexity of the spatial domain is low, while the computational complexity of the transform domain is high. It means that the spatial domain requires less time and resources to run the algorithm, while the transform domain requires more time and resources to run the algorithm. The robustness towards the geometric distortion of the spatial domain is low whereas the transform domain is high. The advantages of the spatial domain are this technique is easy to implement and provides high payload capacity while that of the transform domain provides more robust against the compression, cropping, and image processing on the image. The disadvantages of the spatial domain are low resistance to image processing and being vulnerable to steganalysis attacks while that of the transform domain requires high computational time and provides low payload capacity. Examples of the spatial domain are Least Significant Bit (LSB) and Pixel Value Differencing (PVD) while that of the transform domain is discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT).

2.2 Existing Image Steganography Tool

Xiao Steganography is a tool developed by Nakasoft in Venezuela [13]. It is free software that was released in 2005. This tool supports multiple encryption algorithms with password-protected such as Rivest Cipher 2 (RC2), RC4, Data Encryption Standard (DES), Triple DES, Triple DES 112, and Secure Hash Algorithm (SHA), Message-Digest Algorithm 2 (MD2), MD4, and MD5. The supported image format is BMP, which accepts any type of file to hide in the image and save the target file. To read the hidden information from the target file, it also provides an extraction process, which cannot be extracted with any other software. The steganographic algorithm that is used by the tool is the basic method which is LSB substitution.

OpenStego is an image steganography tool with two major functions which are data hiding and a beta version for watermarking [14]. It provides hide data and extracts data under the data hiding function. It accepts any type of file to hide within the image format of BMP, Joint Photographic Experts Group (JPEG), Portable Network Graphic (PNG), and GIF. User must select one of the encryption algorithms which is Advanced Encryption Standard 128 (AES-128) or AES-256 and a password to complete the data hiding process. It can also encrypt and compress data and supports a plugin-based architecture. Currently, the steganographic algorithms applied by this tool are LSB and random LSB.

StegoMagic is a free image steganography tool that supports any type of files such as text file, wave file, bit map files in 24 bits, or bit map files that is 256-colour or 8-bit) to hide within the image [15]. The image format supported is BMP and it applies the DES encryption algorithm and password

protection to encrypt data before data hiding. The steganographic algorithm used is LSB. Except for text files, the size of the carrier file will remain the same after the data is hidden.

2.3 Comparison of Existing Tools

This subsection discusses the comparison of three existing image steganography tools. The comparison table of existing tools with proposed tool is listed in terms of login modules, conceal data types, supported image formats, and steganography algorithms.

Table 2: Comparison between existing tools with the proposed tool

Features/ Tools	Xiao Steganography 2.6.1	OpenStego	StegoMagic 1.0	Proposed tool - RPC-Secret
Login module	No	No	No	Yes
Conceal Data Type	Any File Type	Any File Type	Any File Type	Plaintext
Image format supported	BMP	BMP, JPEG, PNG, GIF	BMP	BMP
Steganographic Algorithm	LSB Substitution	LSB and Random LSB	LSB	Randomized GB-LSB and Parity Check

Table 2 shows a comparison of existing tools with proposed tools based on the features they have. For the first feature, the existing tools do not have a login module, but the proposed tool will have a login module to login. The conceal data type that these existing tools support are any file type but the proposed tool support plain text. The image format supported by Xiao Steganography 2.6.1 and StegoMagic 1.0 can support BMP which is the same as the proposed tool while OpenStego can support BMP, JPEG, PNG, and GIF. Xiao Steganography 2.6.1 and StegoMagic 1.0 have used the Least Significant Bit (LSB) Substitution as their steganographic algorithm. For the proposed tool, a randomized GB-LSB and Parity Check algorithm is used.

Figure 1 illustrates the pseudocode for the proposed encoding algorithm. It starts with the input of the message, cover-image and stego-key. After that, it checks if the message exceeds 50 bytes, the cover-image is BMP, and the stego-key size is 8 bits. If the condition is met, the message is converted to binary in ASCII format, and then the BMP pixel data offset is searched. BMP is converted to Red-Green-Blue layer and then to binary. The LSBs of each layer will form a group and determine the parity of the group. The stego-key bits are then collected to decide whether to select the LSB of the green or blue layer. Next, the message bits are collected and compared with the group parity to decide whether to replace the bit. Finally, three layers are collected and the image after embedding the message is returned.

```

Begin
  input messages, cover-image, and stego-key
  if message sizes > 50 then print "Maximum text size is 50 bytes."
  else if cover-image != BMP then print "Cover-image is only for BMP image format."
  else if stego-key size > 8 then print "The key size is 8 bit only.(8 digit)"
  else
    convert message into binary format using ASCII code format
    search for BMP pixel data offset value of 54
    for i=0; i<message length; i++
      convert cover-image into three layers (R-G-B)
      convert each layer into binary
      collect LSB of the three layers and form a group
      determine the parity of the group
        if the number of 1 in group = odd then parity = odd
        if the number of 1 in group = even then parity = even
      collect stego-key bit
        if bit value is 0 then choose LSB of Green layer
        if bit value is 1 then choose LSB of Blue layer
      collect message bit
        if message bit = 0 && parity = even || message bit = 1 &&
        parity = odd
          then the LSB of the chosen layer reverse
        else
          the LSB of the chosen layer remain unchanged
      collect three layers and return the image after embedding the message
    end_for
  end_if
  output stego-image
End

```

Figure 1: Randomized GB-LSB and Parity Check algorithm

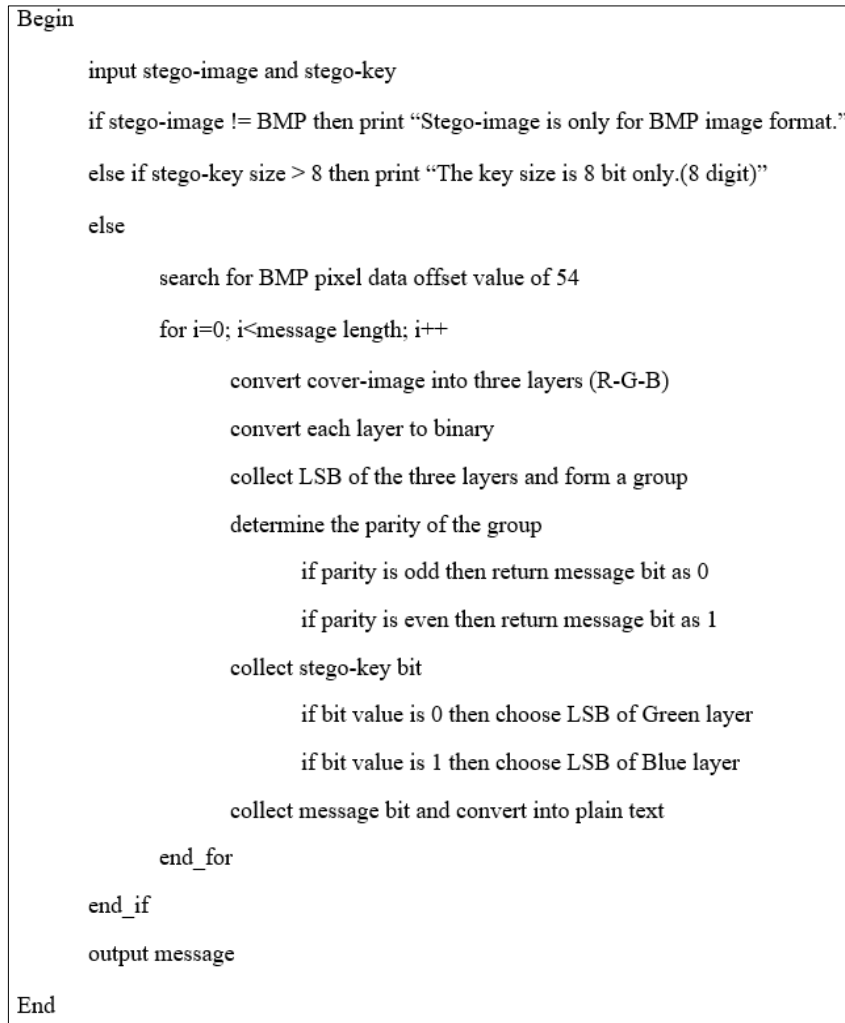


Figure 2: Randomized GB-LSB and Parity Check algorithm

Figure 2 illustrates the pseudocode for the proposed decoding algorithm. It starts with the input of the stego-image and stego-key. After that, it checks if the stego-image is BMP and the stego-key size is 8 bits. If the condition is met, the BMP pixel data offset is searched. BMP is converted to Red-Green-Blue layer and then to binary. The LSBs of each layer will form a group and determine the parity of the group. The stego-key bits are then collected to decide whether to select the LSB of the green or blue layer. Finally, the message bits are collected and converted into plaintext.

Randomized-LSB and Parity Checker technique

For example,
 Message bits: C (alphabet): 01100011
 User enters stego-key (8 bits): 10010011

1. 1 (LSB of blue layer from the pixel chosen)
2. 0 (LSB of green Layer from the pixel chosen)

The stego-key entered by the user is used to decide whether the LSB of the green or blue layer is to be chosen. Action to reverse the bit or not on the chosen layer will depend on the parity checker method.

Pixel	Color component	Component Bit	Group Bit	Parity	Message Bit	Resulting Component Bit
1 th	R	1011000 <u>1</u>	100	Odd	0	1011000 <u>1</u>
	G	0110001 <u>0</u>				0110001 <u>0</u>
	B	1110001 <u>0</u>				1110001 <u>0</u> (No change)
2 th	R	1010001 <u>0</u>	010	Odd	1	1010001 <u>0</u>
	G	0001110 <u>1</u>				0001110 <u>0</u>
	B	1100100 <u>0</u>				1100100 <u>0</u>
3 th	R	1101000 <u>0</u>	010	Odd	1	1101000 <u>0</u>
	G	0101001 <u>1</u>				0101001 <u>0</u>
	B	1010001 <u>0</u>				1010001 <u>0</u>
4 th	R	0100100 <u>0</u>	011	Even	0	0100100 <u>0</u>
	G	1011100 <u>1</u>				1011100 <u>1</u>
	B	0101011 <u>1</u>				0101011 <u>0</u>
5 th	R	0011010 <u>0</u>	000	Even	0	0011010 <u>0</u>
	G	0101010 <u>0</u>				0101010 <u>1</u>
	B	0100101 <u>0</u>				0100101 <u>0</u>
6 th	R	1010101 <u>0</u>	001	Odd	0	1010101 <u>0</u>
	G	1010110 <u>0</u>				1010110 <u>0</u> (No change)
	B	1010101 <u>1</u>				1010101 <u>1</u>
7 th	R	0100100 <u>1</u>	101	Even	1	0100100 <u>1</u>
	G	0001100 <u>0</u>				0001100 <u>0</u>
	B	0010001 <u>1</u>				0010001 <u>1</u> (No change)
8 th	R	0000111 <u>1</u>	100	Odd	1	0000111 <u>1</u>
	G	1100000 <u>0</u>				1100000 <u>0</u>
	B	0001110 <u>0</u>				0001110 <u>1</u>

Figure 3: Randomized GB-LSB and Parity Check algorithm

Figure 3 illustrates the randomized GB-LSB and Parity Check algorithm proposed. The randomized GB-LSB will randomly choose either LSB of Green or Blue layer from each pixel and then use parity check to substitute the LSB of that layer. The randomness of selecting a layer from each pixel to substitute its LSB depends on the stego-key entered by the user. An 8-bit stego-key entered by the user in which 0 represents the Green Layer is chosen while 1 represents Blue layer is chosen. The LSB of each pixel which contains three layers (Red, Green, Blue) will be collected and formed into a group of three bits. The LSB of the Green or Blue layer will be substituted or remain unchanged depending on the parity check method. Odd parity or even parity will be checked with message bit and decide what actions will be performed on randomly chosen GB-LSB.

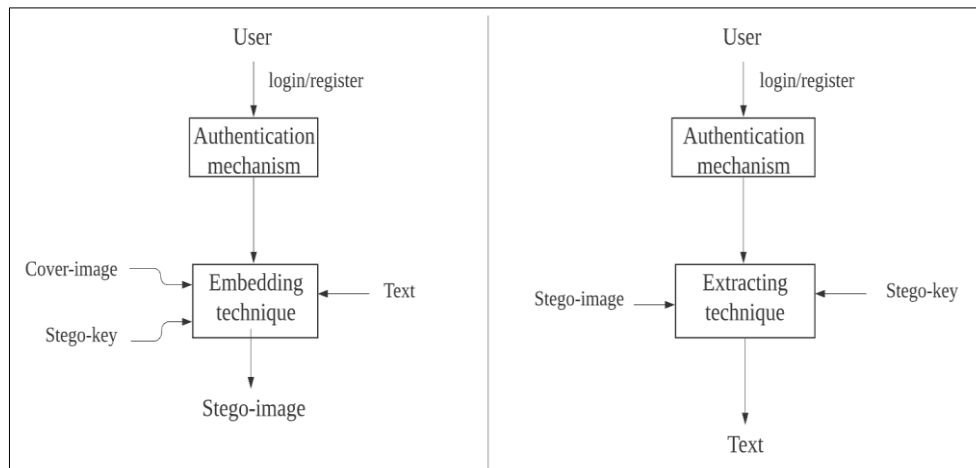


Figure 4: Operation process of the RPC-Secret tool

Figure 4 illustrates the operation process of the RPC-Secret tool. For the embedding process, the user needs to login into the tool, and then enter plaintext. After that, the user needs to select the BMP image as the cover-image for data hiding. For the extraction process, the user also needs to log in and browse the stego-image to extract secret information. User needs to enter a symmetric key for the embedding and extraction process to ensure that the communication channel between the sender and the receiver is secure.

3. Methodology

This section discusses the method chosen for designing and developing the project, the Object-Oriented Analysis and Design (OOAD) method. This methodology is divided into five phases which are object-oriented analysis, object-oriented design, object-oriented implementation, object-oriented testing and object-oriented maintenance.

3.1 Object-Oriented Analysis

In this phase, the features of RPC-Secret are determined. The functional and non-functional requirements of this proposed tool will be collected through the information provided by the user. For the requirements analysis process, four steps need collect and analyze requirements which are eliciting requirements, analyzing requirements, requirement modelling and review and retrospective.

Table 3: Functional requirement analysis

Module	Functions
Registration	User has to register as a new user for the tool. An alert message is prompted if an invalid input is detected
Login	User login with a valid username and password. An alert message is prompted if an invalid input is detected
Steganographic encoding	User input plaintext, cover-image and stego-key to embed messages within the cover-image. User is prompted to save the stego-image into the local files
Steganographic decoding	User input stego-image and stego-key to extract the messages from the stego-image

Table 3 shows the functional requirements analysis of the proposed tools. For the registration module, the user needs to register as a new user to use the tool. If invalid input is detected, an alert message will be prompted to remind the user to enter a valid input. Next, for the login module, the user needs to log in with a valid username and password. If invalid input is detected, an alert message shows

up to remind the user to enter the correct username and password. For the steganography encoding, the user will input the plaintext, cover-image and stego-key to embed messages within the cover-image. User is prompt to save the results which are stego-image into the local files. For the steganography decoding, user will input stego-image and stego-key to extract the messages that are hidden in the stego-image.

Table 4: Non-functional requirement analysis

Module	Functions
Performance of the encoding and decoding process	The tool can generate the result of encoding and decoding at a short time
Usability	The tool is simple to use, easy to learn and easy to understand. The user interfaces action and elements are consistent to reduce the users from making mistakes
Security	Users can access the tool with the correct username and password. The password should consist of alphanumeric characters, and the password length should be between 8 and 15.

Table 4 shows the non-functional requirement analysis of the proposed tool. First, the performance of encoding and decoding should be good enough, because it can generate results in a shorter time, so users don't have to wait for a long time. For usability requirements, the tool should be easy to use, easy to learn, and easy to understand. User interface actions and elements should be consistent to prevent users from making any mistakes. Third, for the security requirement, users must use the correct username and password to access the tool. The password should contain characters and numbers and the password length should be between 8 and 15.

3.2 Object-Oriented Design

In this phase, the designs that will be included are the general architecture of the tool, class diagrams, use case diagrams, sequence diagrams, and activity diagrams. Use case diagram of the proposed tool will be shown in Appendix A (Figure 11). First, users need to register or login into the tool. Next, the user must select whether the steganographic encoding or steganographic decoding. For the steganographic encoding, user must enter plaintext, cover-image and stego-key to produce stego-image. For the steganographic decoding, user must input stego-image and stego-key to extract the plaintext. If no operation is performed, the user can log out of the tool.

Activity diagram of the proposed tool will be shown in Appendix A (Figure 12). Activity is started by opening the RPC-Secret tool. The tool asks for authentication from users to enter username and password. If users do not have an account, user needs to register an account and after the registration succeeds, the user will be redirected to the login interface. If the correct username and password are entered, it will go to the home interface or else it will keep staying on the login interface. If user selects steganographic encoding, user goes to the stegoEncode interface, else will go to the stegoDecode interface. At the stegoEncode interface, user needs to input plaintext with a text size of not more than 50 bytes, the cover-image format is BMP and stego-key size is 8 bits to proceed to steganographic encoding, or else it will loop for the input of plaintext, cover-image and stego-key. At the stegoDecode interface, user need to input stego-image in BMP format and the stego-key size is 8 bits to proceed to steganographic decoding or it will loop for the input of stego-image and stego-key. Users can terminate the tool by logout of the tool.

3.3 Object-Oriented Implementation

In this phase, the object of the class and the relationship between the classes are coded and implemented using the programming language chosen. The implementation of the proposed tool will be divided into

two parts which are C and C# programming language. An early version of the tool is called a prototype and is made in the early stages of the product life cycle for experimental purposes.

3.4 Object-Oriented Testing

The testing of the proposed tool is done by applying the test plan. The test plan will include user acceptance testing. The Likert scale is used to evaluate user satisfaction towards the tool functionality. The second part of the test plan is a security check for the security requirements of the tool. The security check is tested in the proposed tool. As a result of the test, each test either passes or fails.

3.5 Object-Oriented Maintenance

The maintenance of the tool is done by writing the documentation as the way of how the tool performs. The improvements that need to be done for future work are specified in the documentation as well. The errors and bugs that were found out during the testing for the tool are listed for a future fix. Due to the limitation of time, object-oriented maintenance will not be accomplished.

4. Results and Discussion

This section presents the implementation and testing that was conducted for the RPC-Secret tool.

4.1 Implementation

Implementation process starts from the interface designs and database. The coding for the tool was done using C and C# programming languages and Integrated Development Environment (IDE) chosen is Microsoft Visual Studio 2019. The tool uses Microsoft SQL Server 2019 LocalDB as the database.

```

71. //hiding the message
72. for (int i = 24; i < height * width * 3; i += textLen * 24) {
73.     for (int k = 0; k < textLen; k++) {
74.         int* msgbit = charToBinary(message[k]);
75.
76.         for (int j = 0; j < 8; j++) {
77.             int parity = 0;
78.             int index = i + 24 * k + j * 3;
79.             for (int n = 0; n < 3; n++) {
80.                 // Selecting last bit after converting char to binary
81.                 parity += imageArray[index + n] % 2;
82.             }
83.             unsigned char* pixelToUpdate = &imageArray[index + 49 - stegoKey[j]];

```

Figure 5: Code implementation of Randomized GB-LSB

Figure 5 illustrates the code implementing randomized GB-LSB to hide the message. Line 72 illustrates a for loop to loop the variable “i” for the image pixel data. For the increment of this loop, it adds the value of message length times 24 to the variable “i”. Since each character will have 8 bits in binary and each pixel that consists of three layers will take only one bit to replace so the iteration will be equal to the total length of text multiplied by 24. The for loop on line 73 is used to loop the variable “k” for the text length. On line 74, an integer pointer to the “msgbit” variable is declared, which points to the “charToBinary” function implemented on the message array. The for loop on line 76 is used to loop through the stego-key value while the for loop on line 79 is used to create a group bit for the last bit of each colour component. Line 83 states a character pointer is declared as the “pixelToUpdate” variable, pointing to the actual address of the “imageArray” variable.

```

84.          //Determine the parity is even or odd
85.          parity = parity % 2;
86.          if (parity == msgbit[j]) {
87.
88.              if (pixelToUpdate[0] % 2 == 0)
89.                  pixelToUpdate[0]++;
90.              else
91.                  pixelToUpdate[0]--;
92.          }

```

Figure 6: Code implementation of Parity Check

Figure 6 illustrates a code implementation of parity check on a message entered by the user. On line 85, it uses a parity variable equal to parity modulus 2 to determine the parity of the group of bits consisting of the last bit of each colour component. If the remainder of the parity value is 0, its parity is even, and if the parity value is 1, its parity is odd. Line 86 states that if the parity bit is equal to the value of the message bit array, there are two cases, the parity bit is even and the message bit is 0 or the parity bit is odd and the message bit is 1. Line 86 states that if the parity bit is equal to the value of the message bit array, there are two cases, the parity bit is even and the message bit is 0 or the parity bit is odd and the message bit is 1. Line 88 states that if the parity of the “pixelToUpdate” variable is even or called 0, the value of the “pixelToUpdate” variable will become 1, and the parity is odd or called 1, and the value of the “pixelToUpdate” variable will become 0.

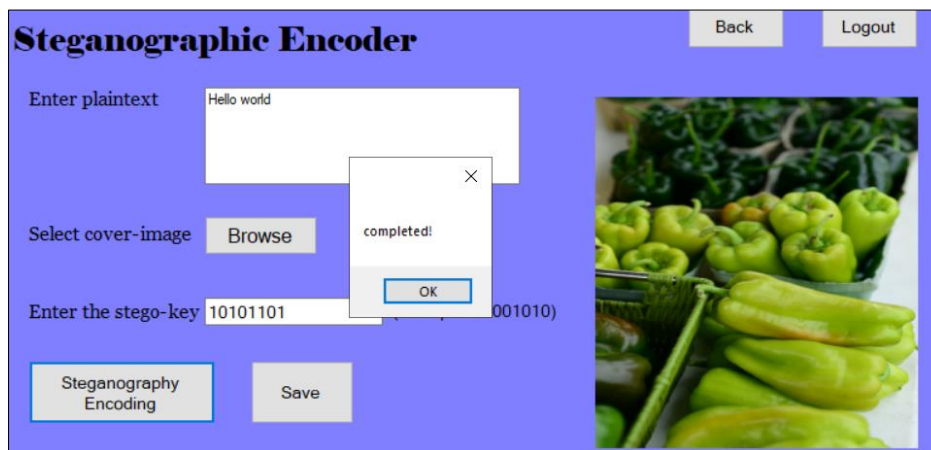


Figure 7: Hiding message within the image

Figure 7 illustrates the result of the user hiding the image within the image on the Steganography Encoding module. From here, the user can enter up to 50 bytes of plaintext, a BMP image, and an 8-bit stego-key. After that, the user can click the Steganography Encoding button to complete the message embedding process within the image. Lastly, user can click the Save button to save the stego-image to a local file.



Figure 8: Extract message from image with different stego-key

Figure 8 illustrates the results of users extracting messages from images using different stego-key on the Steganographic Decoding module. It can be found that the plaintext extracted from the image with different stego-key only displays part of the message. It means that only users who know the correct secret key can obtain the secret message hidden in the image.

Table 5: Result of PSNR value







Name	Cover-image	Stego-image	PSNR (db)	Average PSNR (db)
Grape			56.2879	
Rabbit			56.2385	56.2563
Waterfall			56.2424	

Table 5 shows the results of the PSNR value for three different BMP images generated by the RPC-Secret. The cover-image and the stego-image are included in the table, and it is difficult to see any difference between these two images by looking at them with the bare eye.

4.2 Functional Testing

Functional testing will be conducted according to the test plan developed for the four modules based on different test case assumptions.

Table 6: Functional testing

No.	Module	Test Case	Expected Result	Result
1.	Login	i. Enter valid username and password.	Login successful.	Success
		ii. Enter incorrect username and password.	An error message is displayed: "Incorrect username or password."	Success
		iii. Either the username or password fields are empty.	An error message is displayed: "Incorrect username or password."	Success
		iv. Click Login button.	Account authentication is carried out.	Success
		v. Click Register button.	Redirect to Registration module.	Success
2.	Registration	i. All the input fields are filled in with the correct format.	Register successful.	Success
		ii. All input fields are empty.	An error message is displayed: "All input fields cannot be blank."	Success
		iii. Same username is registered.	An error message is displayed: "Username already exists."	Success
		iv. Click Submit button.	Account registration is carried out.	Success
		v. Click Back button.	Redirect to Login module.	Success
3.	Steganography Encoding	i. Enter plaintext more than 50 bytes.	An error message is displayed: "Maximum text size is 50 bytes."	Success
		ii. Browse cover-image other than BMP image format.	An error message is displayed: "Cover-image is only for BMP image format."	Success
		iii. Enter stego-key more than 8 bits.	An error message is displayed: "The key size is 8 bits only."	Success
		iv. Click Steganography Encoding button.	A message box is displayed: "Completed!"	Success
		v. Click Save button.	A save file dialogue is pop up and a message box is displayed: "Completed!" when the image is saved.	Success
		vi. Click Logout button.	Logout and redirect to Login module.	Success
		vii. Click Save button.	A save file dialogue is pop up and a message box is displayed: "Completed!" when the image is saved.	Success
		viii. Click Logout button.	Logout and redirect to Login module.	Success
4.	Steganography Decoding	i. Browse stego-image other than BMP image format.	An error message is displayed: "Stego-image is only for BMP image format."	Success
		ii. Enter stego-key more than 8 bits.	An error message is displayed: "The key size is 8 bits only."	Success
		iii. Click Steganography Decoding button.	Plaintext box will show the hidden message.	Success
		iv. Click Logout button.	Logout and redirect to Login module	Success

Table 6 shows the summary of functional testing results for all modules. The test plan conducts different test cases for the Login module, Registration module, Steganography Encoding module, and Steganography Decoding module. By summarizing the functional test results, the tool is performed as expected based on the tests performed, and all results were successful.

4.3 User Acceptance Testing

User acceptance testing will be conducted on five UTHM students to collect their responses towards the performance of the proposed tool. A google form will be created and sent to random students, which will stop accepting responses one week after it is sent. The total number of volunteers to test the tool was five.

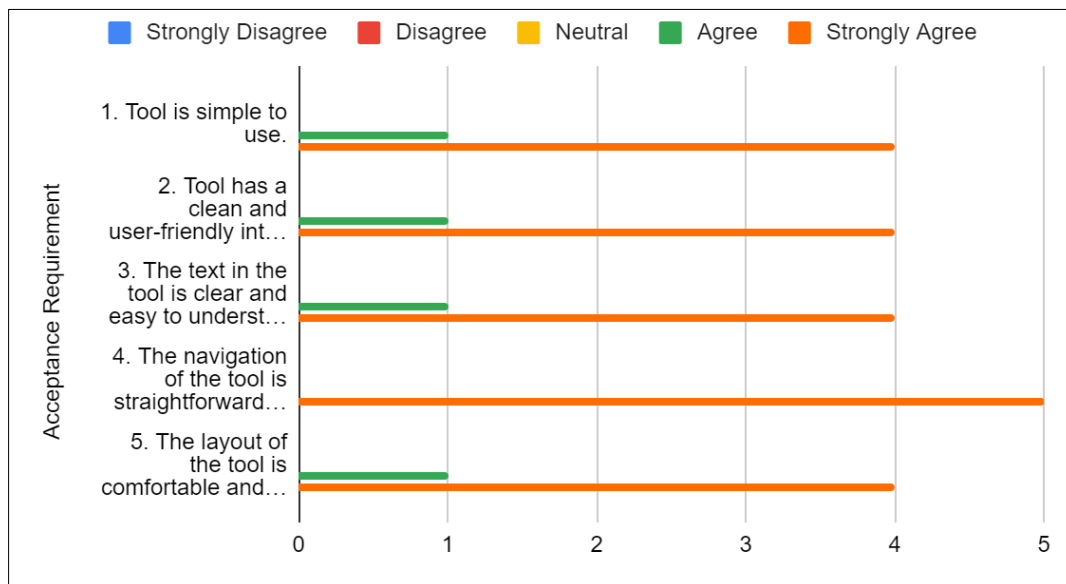


Figure 9: Bar chart of user acceptance testing

Figure 9 illustrates a bar chart of user acceptance testing of the proposed tool. Four volunteers strongly agreed, and one volunteer agreed that the tool is simple, has a clean and user-friendly interface, the text in the tool is clear and easy to understand, and the layout of the tool is comfortable and neat. Five volunteers strongly agreed that the navigation of the tool is straightforward and logical.

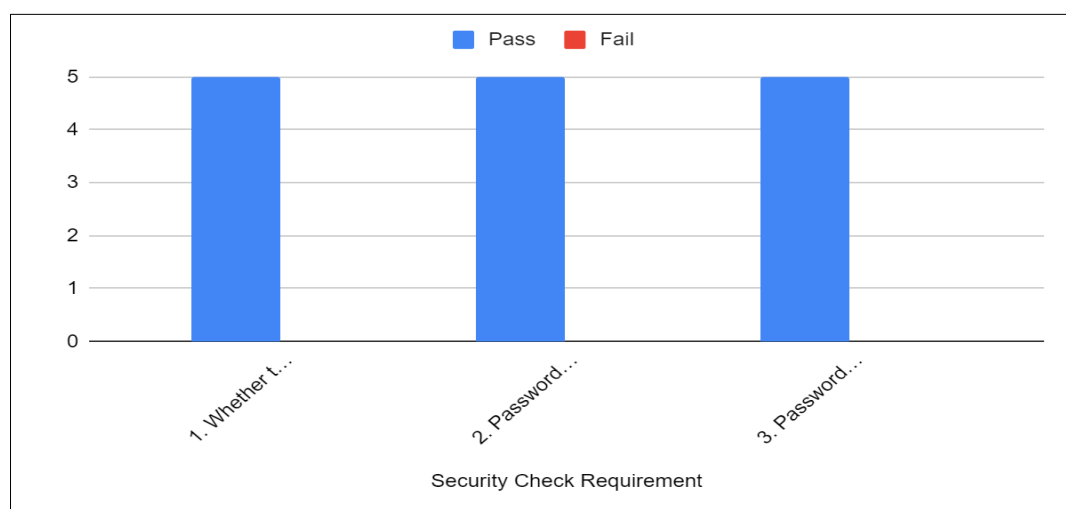


Figure 10: Bar chart of security check testing

Figure 10 illustrates the bar chart of security check testing of the proposed tool. For the security requirement, whether the username or password is wrong or empty, the error message "Incorrect username or password" will be displayed, and the result is Pass. The second security requirement, which is password complexity is enforced, and the result is still the same is Pass. Lastly, the security requirement for whether the password length is enforced is tested and all volunteers give the result are Pass.

5. Conclusion

This section summarizes the overall development, implementation, and testing of the tool. It includes the achievement of objectives, the advantages and disadvantages of RPC-Secret.

The objectives of the project which are to design RPC-Secret tool for image steganography by using the randomized GB-LSB and Parity Check algorithm, to develop RPC-Secret tool, and to test the tool's functionality and user acceptance towards the RPC-Secret tool are achieved. Users can hide secret information in an inconspicuous place instead of on paper and extract secret information at any time if needed.

There are several limitations that can be found in the RPC-secret tool, which is the tool Supports up to a message size of 50 bytes only. Besides, it supports plaintext and BMP images only for the steganography process.

There are some improvements that can be implemented for future works of the RPC-Secret tool, where the tool can allow users to hide message sizes of at least 200 bytes and support other conceal data types such as images and videos, as well as cover-images such as PNG and JPG. It is recommended that the tool be able to support other operating systems such as Mac and Linux platforms.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Appendix A

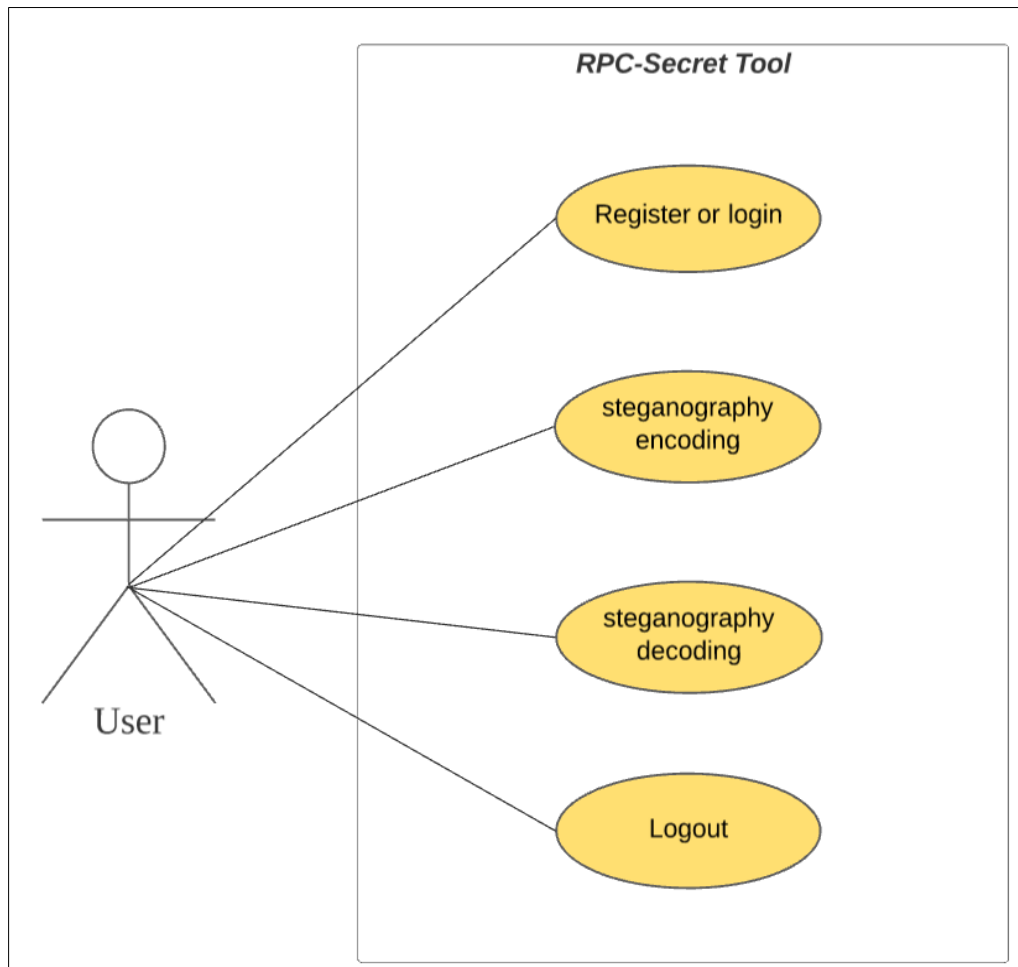


Figure 11: Use case diagram of the proposed tool

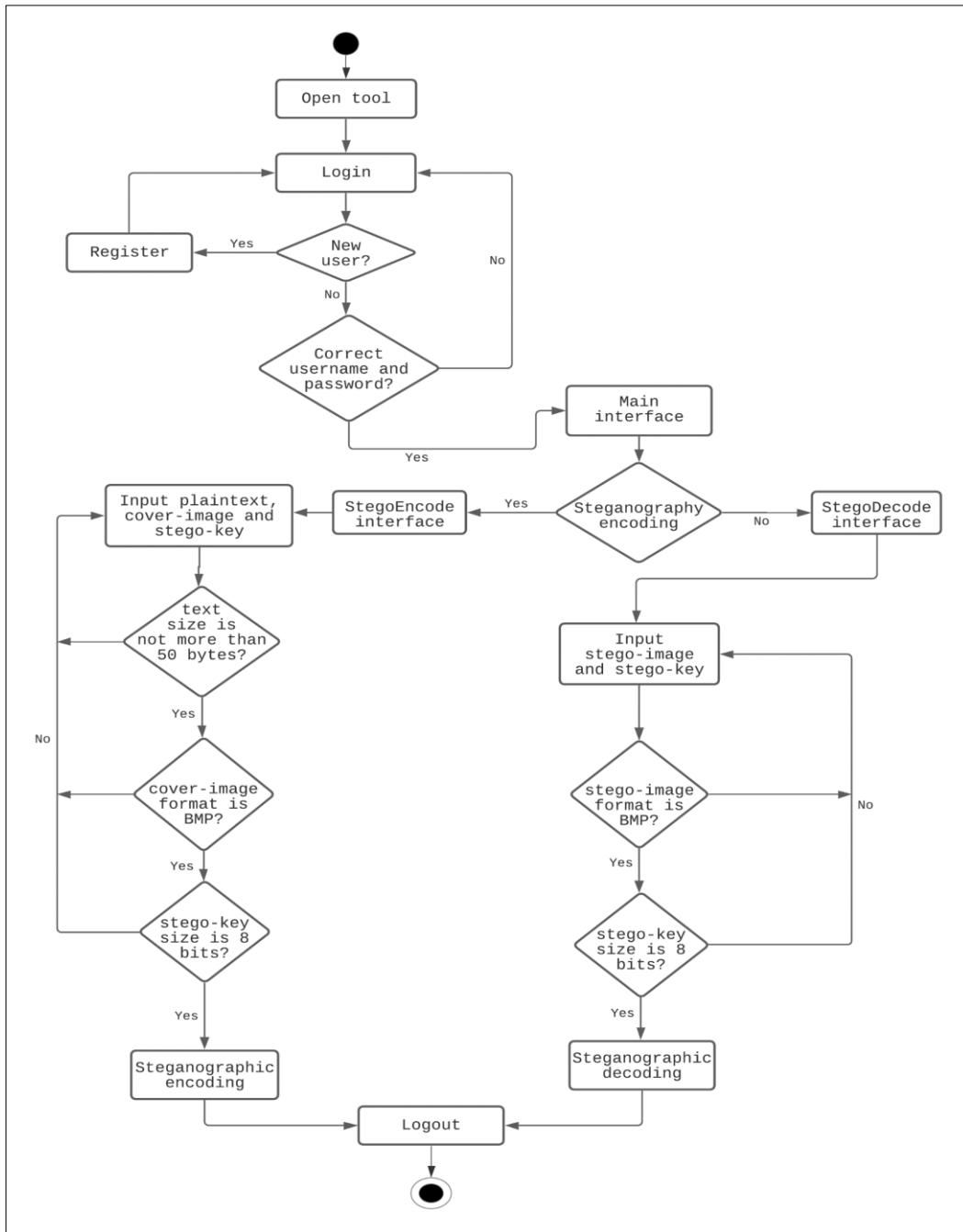


Figure 12: Activity diagram of the proposed tool

References

- [1] Abuali, M. S. & Rashidi, C. B. M. & Salih, M. H. & Raof, R. A. A. & Hussein, S. S. Digital image steganography in spatial domain a comprehensive review. *Journal of Theoretical and Applied Information Technology*. 2019. 97(19): 5081–5102.
- [2] Pranesh, R. & Vigneshwaran, M. & Harish, V. & Manikandan, G. A new approach for secure data transmission. *Institute of Electrical and Electronics Engineers*. March 18, 2016. Nagercoil, India: International Conference on Circuit, Power and Computing Technologies. 2016. pp. 1–4.
- [3] Raipurkar, K. V. & Deorankar, A. V. Improve data security in cloud environment by using LDAP and two way encryption algorithm. *Institute of Electrical and Electronics Engineers*. March 18, 2016. Indore, India: Symposium on Colossal Data Analysis and Networking. 2016. pp. 1–4.
- [4] Giri, D. & Obaidat, M. S. & Maitra, T. SecHealth: An Efficient Fog Based Sender Initiated Secure Data Transmission of Healthcare Sensors for e-Medical System. *Institute of Electrical and Electronics Engineers*. December 4, 2017. Singapore: GLOBECOM 2017 - 2017 IEEE Global Communications Conference. 2017. pp. 1–6.
- [5] Hashim, M. & Rahim, M. S. & Alwan, A. A. A review and open issues of multifarious image steganography techniques in spatial domain. *Journal of Theoretical and Applied Information Technology*. 2018. 96(1): 956–977.
- [6] Manaseer, S. & Aljawawdeh, A. & Alsoudi, D. A New Image Steganography Depending On Reference & LSB. *International Journal of Applied Engineering Research*. 2017. 12(1): 1950–1955.
- [7] Kocak, C. Clsm: Couple layered security model a high-capacity data hiding scheme using with steganography. *Image Analysis and Stereology*. 2017. 36(1): 15–23.
- [8] Swain, G. A Steganographic Method Combining LSB Substitution and PVD in a Block. *Procedia Computer Science*. June 1, 2016. Netherlands: Elsevier. 2016. pp. 39–44.
- [9] Saleh, M. A. Image Steganography Techniques - A Review Paper. *International Journal of Advanced Research in Computer and Communication Engineering*. 2018. 7(1): 52–58.
- [10] Bobade, G. G. & Patil, A. G. Testing of image steganography with use of LSB and DCT techniques. *International Journal of Innovative Technology and Exploring Engineering*. 2019. 8(10): 3694–3697.
- [11] Nevriyanto, A. & Sutarno, S. & Siswanti, S. D. & Erwin, E. Image Steganography Using Combine of Discrete Wavelet Transform and Singular Value Decomposition for More Robustness and Higher Peak Signal Noise Ratio. *Institute of Electrical and Electronics Engineers and Institution of Engineering and Technology*. October 2, 2018. Pangkal, Indonesia: 2018 International Conference on Electrical Engineering and Computer Science. 2018. pp. 147–152.
- [12] Thanki, R. & Borra, S. A color image steganography in hybrid FRT–DWT domain. *Journal of Information Security and Applications*. 2018. 40(1): 92–102.
- [13] Xiao Steganography. (2007). *Softonic*. Retrieved on October 10, 2021, from <https://xiao-steganography.en.softonic.com>
- [14] OpenStego. (2017). *OpenStego*. Retrieved on October 10, 2021, from <https://www.openstego.com/>
- [15] StegoMagic. (2005). *downloadsouce*. Retrieved on October 10, 2021, from <https://www.downloadsouce.net/1755462/stegomagic/>