Steganalysis Of Digital Image

Jyoti Auti[1] Shubhangi Bagal[2] Priyanka Bidave[3]
[1, 2, 3] Dept of Electronics And telecommunication
[1, 2, 3] Rajarshi Shahu College of Engineering, Pune University Maharashtra, India

Abstract*- In this system, we present the steganalysis of very recently proposed a steganographic method, the random LSB replacement technique using Diffie Hellman Key exchange protocol. We describe how the method is susceptible to detection, extraction and disabling attacks. This method is therefore insecure and should only be used for hiding information under the noncritical situation.*

Steganalysis is a technique for the detection of the secret information embedded in another image known as cover image and if possible the secret text is tried to recover. In this system techniques are used for the detection of the hidden data. Firstly, hiding is done by LSB replacement method in which the attacker knows about the cover image without the knowledge of the coding algorithm of the stego-image.

Keywords- Steganography, Steganalysis, MATLAB, QT designer, Cover image, Stego image

I. INTRODUCTION

Steganalysis is the study of detecting messages hidden using steganography. The main aim in the steganography is to hide the very existence message in the covered medium. The proposed method describes two steps for hiding the secret information by using the steganography based on matching key. The first step is to find the shared stego-key between the two communication parties over insecure networks by applying Diffie Hellman Key exchange protocol. The second step in the proposed method is that the sender uses the secret stego-key to select pixels that it will be used to hide. Each selected pixel is then used to hide 8 bits of binary information depending on the LSB replacement method.

The counter-technique of image steganography is known as image steganalysis. It begins by identifying the artifacts that exist in the suspect file which has formed as a result of embedding a message. The one which we are going to study as a part of our research is a Man-In-Middle attack. Our system will be capable of preventing the Man-In-Middle attack.

II. METHODOLOGY

Block Diagram:

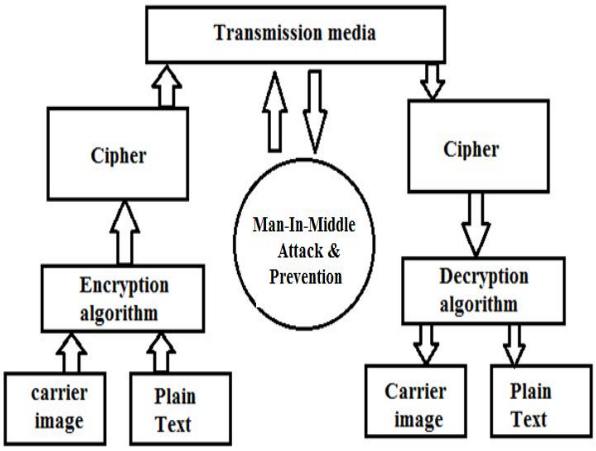


Fig1. Block Diagram

The above system consists of two unit, sender unit and receiving unit. The image and the text document are given to the encryption phase. The encryption algorithm is used for embedding the data into the image. The resultant image acting as a carrier image is transmitted to the decryption phase using the transmission medium. For extracting the message from the carrier image, it is sent to the decryption section. The plain text is extracted from the carrier image using the decryption algorithm.

The proposed method describes two steps for hiding the secret information by using the public steganography based on matching method.

Step 1: The first step is to find the shared stego-key between the two communication parties (Alice and Bob) over insecure networks by applying Diffie Hellman Key exchange protocol. As shown in Figure 5.2, DH key exchange protocol shows the technique for the key exchange between two parts (Alice and Bob) to get shared Stego-key values. Alice must generate the keys (public and private keys) and use her private keys to give the new public key and send it to Bob's side. Bob must obtain and issue new public keys. Then at the end of the protocol, each side recovers his/her received a public key to reach the shared values between them, that's mean Alice and Bob have arrived same sego-key value.

Step 2: Matching Method

The second step in this technique is that the sender uses the secret stego-key to select pixels that it will be used to hide. Each selected pixel is then used to hide 8 bits binary information depending on the matching method which is summarized in four cases as shown in Table 5.1 Since the 8 bits data will be compared with the selected pixel's bytes, red, green and blue values respectively to produce an array with 2 bit binary values as 00, 01, 10, and 11. Alice's side starts comparing to search the equality, where, she takes data value and compare it with the value of the red colour (± 7 – decimal value). As shown in Table 5.1, Case no. 1, if they are equal, then the value zero (00 – binary value) is set to the array. Case no. 2, if the data value and the red value are not equivalent then the value will be compared with the green colour, if they are equals (± 7 – decimal value) then the array is set to be one (01- binary value).

Case no. 3, if the data value and the green value are not equivalent then the value will be compared with the blue colour, if they are equals (± 7 – decimal value) then the value two (10 – binary value) is set to the array (refer to Figure 5.3). Finally Case no. 4, If in case the secret data didn't equal any of the previous three conditions then the LSB method is used to embed the data inside the selected pixel, and the value three (11 – binary value) is set to the array. In this case, the data value will be distributed as follows:

The first three bits of the data are replaced by the three least significant bits of the red byte.

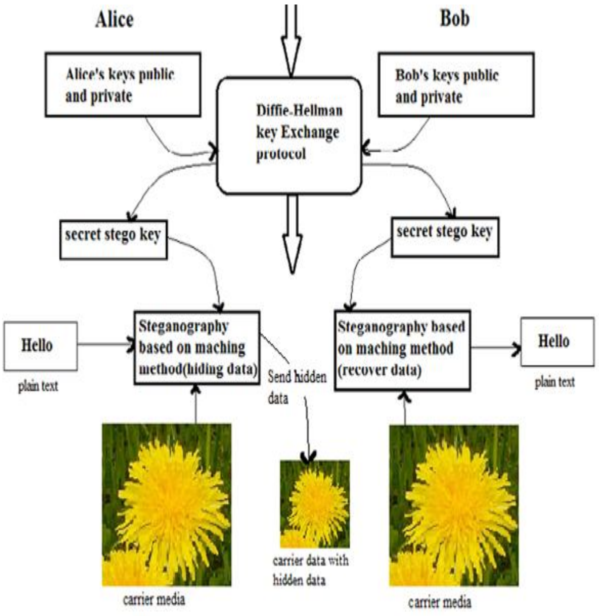

Alice
Bob
Alice's keys public and private
Diffie-Hellman key Exchange protocol
Bob's keys public and private
secret stego key
secret stego key
Hello
plain text
Steganography based on maching method(hiding data)
Send hidden data
Steganography based on maching method (recover data)
Hello
plain text
carrier media
carrier data with hidden data
carrier media

Fig2. The public key steganography protocol

Table1. The four main cases in the proposed public-key steganalysis

Case no.1	If 8 bit data= red(8 bit)	Then red value= 8 bit data
Case no.2	If 8 bit data= green(8 bit)	Then green value= 8 bit data
Case no.3	If 8 bit data= blue(8 bit)	Then blue value= 8 bit data
Case no.4	Otherwise	Use LSB method

Diffie-Hellman Key Exchange Protocol

I. Diffie-Hellman's Features-

The Diffie-hellman algorithm has two attractive feature

- Secret keys are created only when needed. There is no need to store secret keys

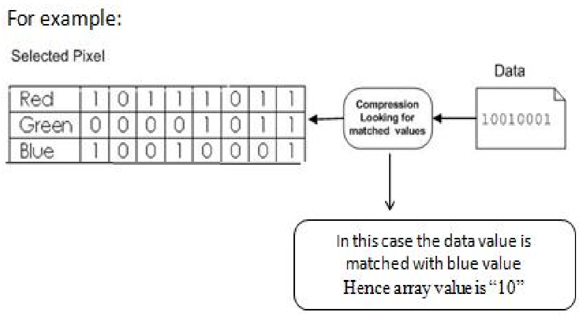

For example:
Selected Pixel
Red 1 0 1 1 1 0 1 1
Green 0 0 0 0 1 0 1 1
Blue 1 0 0 1 0 0 0 1
Compression Looking for matched values
Data
10010001
In this case the data value is matched with blue value
Hence array value is "10"

Fig3. A working example in the proposed public-key steganalysis

For a long period of time, exposing them to increased vulnerability.

- The exchange requires no preexisting infrastructure other than an agreement on the global parameters. However, there are a number of disadvantages to Diffie- Hellman algorithm:
- It does not provide any information about the identities of both parties. So it is vulnerable to impersonation attack.
- It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys.

It can not prevent replay attack.

It is subject to a MITM attack, in which a attacker C impersonates while communicating with A and impersonates A while communicating with B.

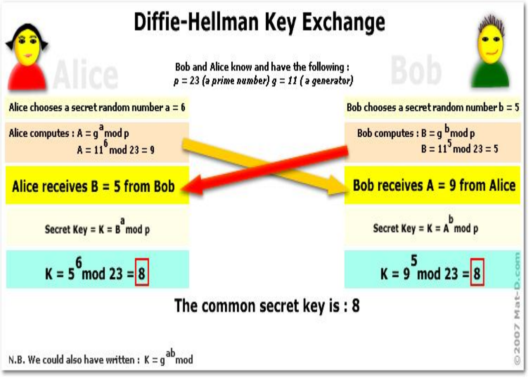

Diffie-Hellman Key Exchange
Alice
Bob
Bob and Alice know and have the following :
p = 23 (a prime number) g = 11 (a generator)
Alice chooses a secret random number a = 6
Bob chooses a secret random number b = 5
Alice computes : A = g^a mod p
A = 11^6 mod 23 = 9
Bob computes : B = g^b mod p
B = 11^5 mod 23 = 5
Alice receives B = 5 from Bob
Bob receives A = 9 from Alice
Secret Key = K = B^a mod p
Secret Key = K = A^b mod p
K = 5^6 mod 23 = 8
K = 9^5 mod 23 = 8
The common secret key is : 8
N.B. We could also have written : K = g^ab mod
©2007 Mat-D.com

Fig4. Difie-Hfellman Key Exchange

Man-in-the-middle Attack

Man-in-the-middle attack proceeds as follows:
step1: A C (B): α*XA* mod *q*
step2: C (A) B: α*XC* mod *q*
step3: BC (A): α*XB* mod *q*
step4: C (B) A: α*XC* mod *q*

User A generates a one–time private key *XA* calculates *YA*, and sends his public key *YA* in a message addressed to user B. The enemy attacker C intercepts A's message. Attacker can saves A's public key (*YA*) and generates a one–time private key *XC*, calculates *YC*, and sends his public key *YC* in a message to user B. This message is sent in such a way that it appears as though it was sent from A's host system. B receive C's message and stores C's public key with A's User ID.

Similarly, C sends a message to A with C's public key (*YC*), purporting to come from B. A calculates a secret key *K1* ((*YC*) *XA* mod *q*) based on *XA* and *YC*. C calculates *K1* ((*YA*) *XC* mod *q*) using *XC* and *YA*. So *K1* is shared by A and C. B calculates a secret key *K2* ((*YC*) *XB* mod *q*) based on *XB* and *YC*. C calculates ((*YB*) *XC* mod *q*) using *XC* and *YB*. So *K2* is shared by B and C. From now on user A thinks *K1* is shared with B, user B thinks *K2* is shared with A, but no key is shared by A and B actually. So C is able to relay messages from A to B and from B to A.

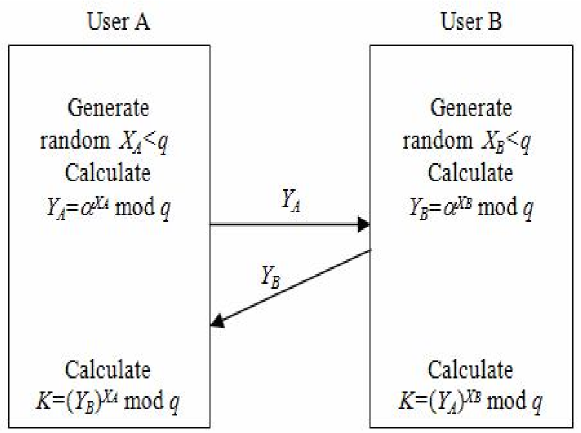

User A
User B
Generate random $X_A<q$
Calculate $Y_A=\alpha^{X_A}$ mod q
Generate random $X_B<q$
Calculate $Y_B=\alpha^{X_B}$ mod q
Y_A
Y_B
Calculate $K=(Y_B)^{X_A}$ mod q
Calculate $K=(Y_A)^{X_B}$ mod q

Fig 5. Diffie-Hellman key Exchange

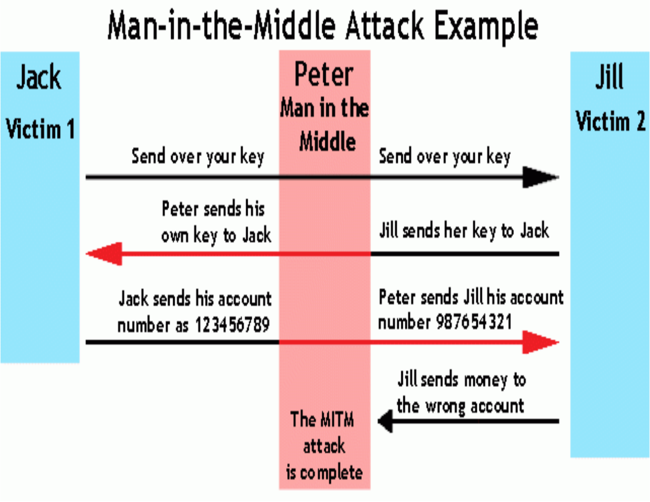

Man-in-the-Middle Attack Example
Jack
Victim 1
Peter
Man in the
Middle
Jill
Victim 2
Send over your key
Send over your key
Peter sends his own key to Jack
Jill sends her key to Jack
Jack sends his account number as 123456789
Peter sends Jill his account number 987654321
Jill sends money to the wrong account
The MITM attack is complete

Fig6. Example of MITM Attack

III. RESULT

Hence the designed system will be capable of preventing the Man-In-Middle attack as well as sending the text data over an image with an encryption algorithm. For designing the steganographic application, it consists of different phases of encryption, decryption and data transmission. This system for sending the personal data securely to the destination has been developed successfully. The design phase is the primary phase, which gives a brief idea about the different levels used for embedding the data into the image. We have designed the Steganography application which embedded the data into the image.

Calculations for hiding character in image

1) Binary calculations showing the first three bits of the data are replaced by the three least significant bits of the red byte.

R_MIX	CHARACTER	X=R_MIX|CHAR	Y=X&R'(C)
11111000	H(01001000)	11111000	00111000
11111000	E(01000101)	11111101	00111101
11111000	L(01001100)	11111100	01000100
11111000	L(01001100)	11111100	01000100
11111000	O(01001111)	11111111	01000111

Table 2

2) Binary calculations showing the second three data bits are replaced by the three least significant bits of the green byte.

G_MIX	CHARACTER	X=G_MIX|CHAR	Y=X&G'(C)
11000111	H(01001000)	11101111	00001110
11000111	E(01000101)	11000111	01000110
11000111	L(01001100)	11001111	01001000
11000111	L(01001100)	11001111	01001010
11000111	O(01001111)	11001111	01001000

3) Binary calculations showing the last two data bits are replaced by the two least significant bits of the blue byte

B_MIX	CHARACTER	X=B_MIX|CHAR	Y=X&B'(C)
00111111	H(01001000)	01111111	00000011
00111111	E(01000101)	01111111	00001000
00111111	L(01001100)	01111111	00001010
00111111	L(01001100)	01111111	00001010
00111111	O(01001111)	01111111	00001001

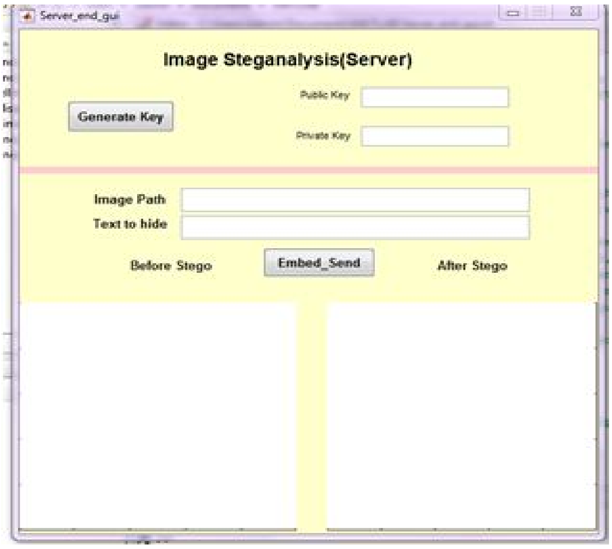
Table 3

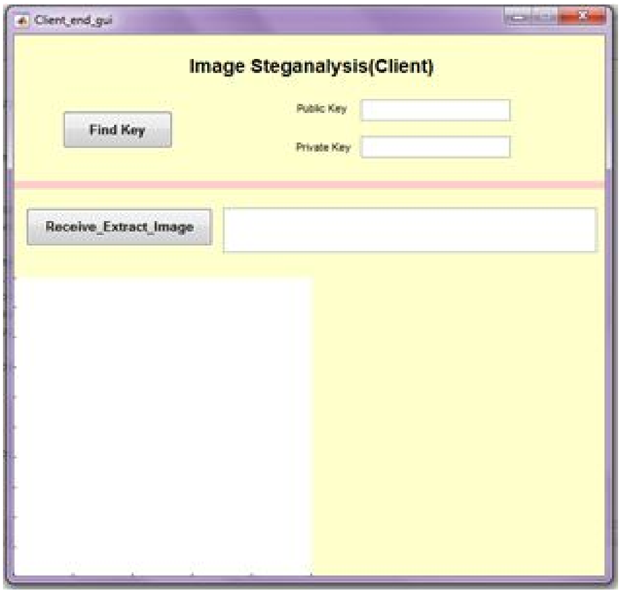
calculations for finding public and private key using Diffie-Hellman algoritham

SR. NO.	TYPE	PRIME NO.(p)	RANDOME NO.(g)	PUBLIC KEY	PRIVATE KEY
1	Server	23	9	6 ($X=g^a$ mod p)	**9($ka=Y^a$ mod p)**
2	Client	23	9	16 ($Y=g^b$ mod p)	**9($kb=X^b$ mod p)**

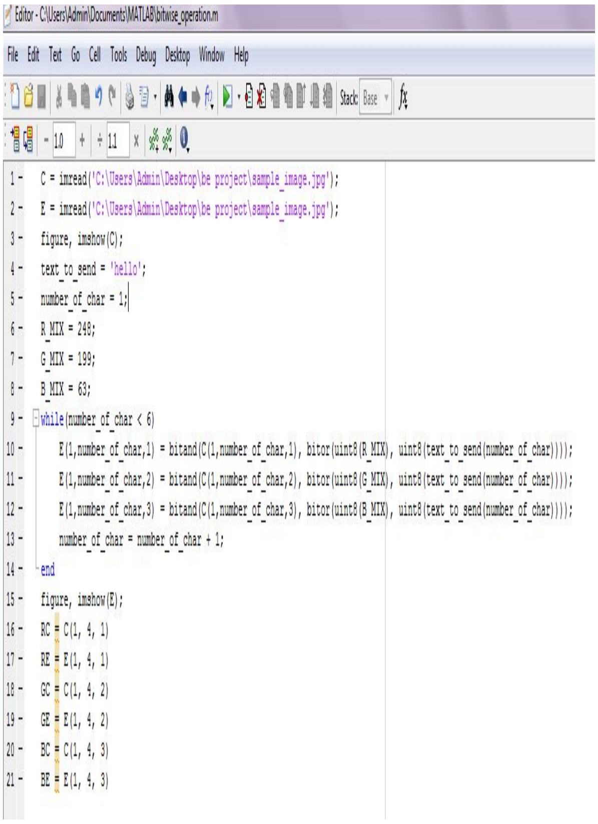
Table 4

First we have to create GUI (Graphical User Interface) using MATLAB, for server and client session.We have made the GUI required for the transmission and reception of text data in an image as given below:





Create program in MATLAB to hide secret message into sample image.


Editor - C:\Users\Admin\Documents\MATLAB\bitwise_operation.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
fx
1.0
1.1

```
C = imread('C:\Users\Admin\Desktop\be project\sample_image.jpg');
E = imread('C:\Users\Admin\Desktop\be project\sample_image.jpg');
figure, imshow(C);
text_to_send = 'hello';
number_of_char = 1;
R_MIX = 248;
G_MIX = 199;
B_MIX = 63;
while(number_of_char < 6)
    E(1,number_of_char,1) = bitand(C(1,number_of_char,1), bitor(uint8(R_MIX), uint8(text_to_send(number_of_char))));
    E(1,number_of_char,2) = bitand(C(1,number_of_char,2), bitor(uint8(G_MIX), uint8(text_to_send(number_of_char))));
    E(1,number_of_char,3) = bitand(C(1,number_of_char,3), bitor(uint8(B_MIX), uint8(text_to_send(number_of_char))));
    number_of_char = number_of_char + 1;
end
figure, imshow(E);
RC = C(1, 4, 1)
RE = E(1, 4, 1)
GC = C(1, 4, 2)
GE = E(1, 4, 2)
BC = C(1, 4, 3)
BE = E(1, 4, 3)
```

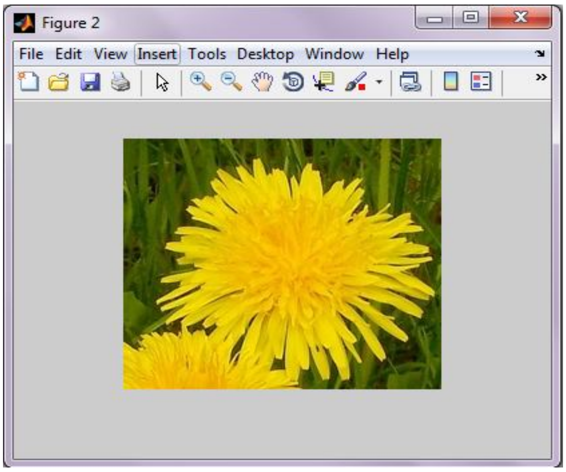
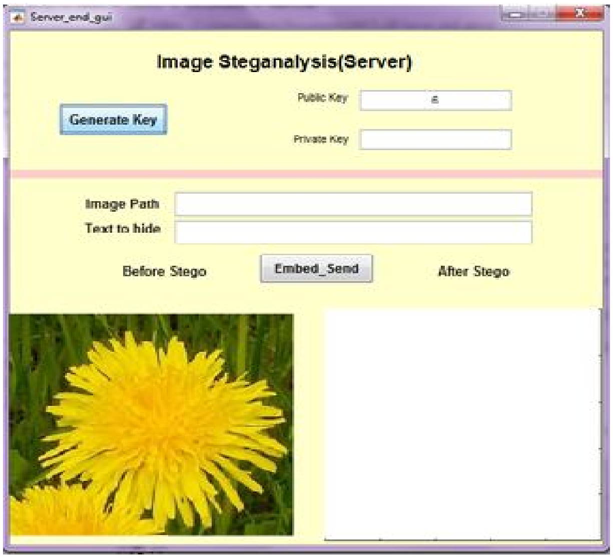


Fig7. Cover image with hidden data

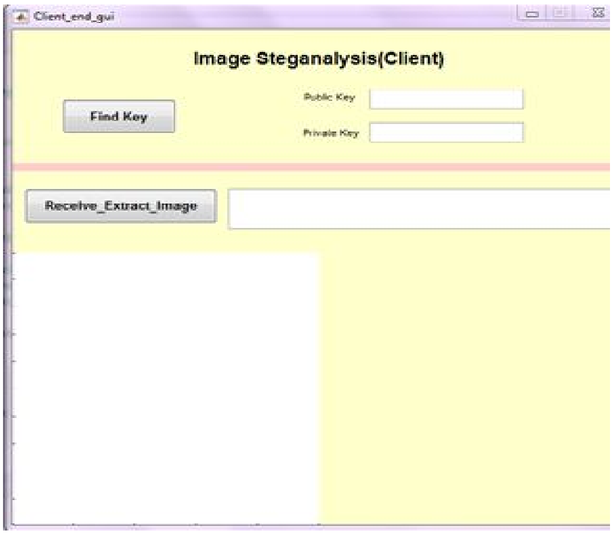
Generate public key using Diffie-Hellman Algorithm.

- Server chooses a random secret x,$1 <= x <= p-2$, and sends client g^x *mod pA-B* : g^x *mod p* and find public key.
- Client chooses a random secret y,$1 <= y <= p-2$, and sends server g^y*mod p*

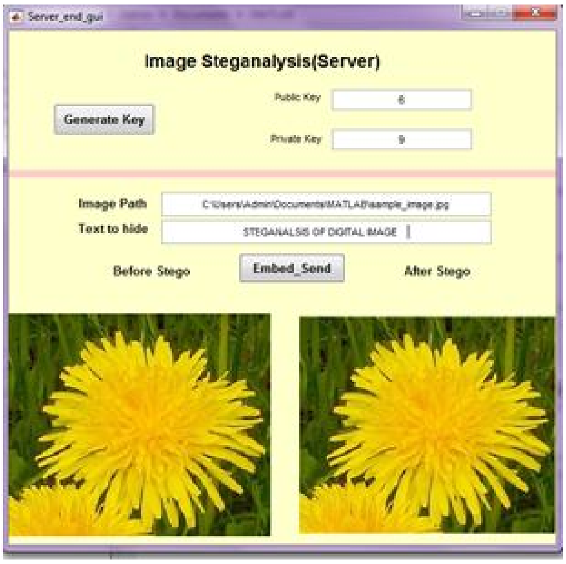
  B- A: g^y*mod p*
- Client receives *g^x* and computes the private key as *K = (g^x) ^y mod p*
- Server receives *g^y* and computes the private key as *K = (g^y) ^x mod p*



Public Key: 6



Step 5: If private keys are same then only message is to be transmitted.

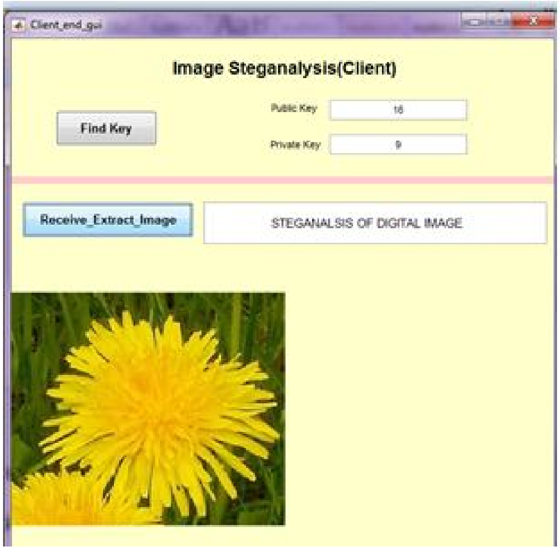


Public Key: 6

Private Key: 9

Image Path: C:\Users\Admin\Documents\MATLAB\sample_image.jpg

Text to hide: STEGANALSIS OF DIGITAL IMAGE



Public Key: 16

Private Key: 9

STEGANALSIS OF DIGITAL IMAGE

IV. FUTURE SCOPE

Future work includes embedding a secret message in another type of files like audio, video. Further work could include developing a YASS(yet another steganographic scheme) and strong encryption algorithms like AES or DES. GUI can be refined and made more user-friendly. Also, a command line version can be developed for this application so that it will suit the more experienced users.

V. CONCLUSION

Steganalysis has its place in security. Steganalysis can be used along with steganography to make a highly secure data. Formerly just an interest of the military, Steganography is now gaining popularity among the masses. Since we use three least significant bits of pixel values to store data, the data may be lost if any compression techniques are applied to stego-image, but after embedding the stego-image remains unchanged in its resolution as well in size. The tool worked fine with all type of images like .jpg, bmp, gif, tiff etc. It produces matching between the data bit parts and the selected pixels. 3 pixels were needed to hide 8 bit of data in case of normal LSB method but in this method, only one pixel is enough to hide 8 bit of data. So by using matching method size of the data that can be hidden in a pixel is increased by 3 times more than normal LSB method. The security of the in this steganalysis depends on Diffie Hellman public key exchange protocol. Hackers may use trial and error method to match the keys, so the tool is designed such that it should corrupt the sec data after three trials.

REFERENCE

[1] I. Avcibas, M. Kharrazi, N.D. Memon, and B. Sankur. Image steganalysis with binary similarity measures. *EURASIP Journal on Applied Signal Processing*, 17:2749–2757, 2005.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996.

[3] G. Cancelli, G. Doërr, I.J. Cox, and M. Barni. Detection of ±1 LSB steganography based on the amplitude of histogram local extrema. In *Proceedings IEEE, International Conference on Image Processing*, *ICIP 2008*, pages 1288–1291, San Diego, CA, October 12–15, 2008.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.
Software available at http:// www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] C. Chen and Y.Q. Shi. JPEG image steganalysis utilizing both intrablock and interblock correlations. In *Circuits and Systems, ISCAS 2008. IEEE International Symposium on*, pages 3029–3032, May 2008.

[6] Willian Stallings, Network Security Essentials: Applications and Standards, 2nd ed, Beijng: qinghua press, 2004.1, pp.75–77.

[7] Li Xin, An Improvement of Diffie-Hellman Protocol, Network & Computer Security, 2007,12, pp. 22–23.