ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 3, Issue 8, August 2015

# RAKS. M. PK 2.0, an Efficient Methodology to Determine the Steganography Signature of Steganography Tools in Case of RGB Image

Er. Rakesh Kumar[1], Pratik Kumar[2]

Computer Science Researcher, Dept. of Computer Science & Engineering, Ranchi University, India[1]

B. Tech. Final Year, Dept. of Computer Science & Engineering, CIT, Ranchi University, India[2]

**ABSTRACT:** As we all know, Steganography is being used in terrorist activities these days. Terrorist used to hide information behind carrier image in form of text or image which may be transferred sent or received through Social Networking Sites such as Facebook, Tweeter, Linkedin, etc. but unfortunately, no social networking sites have control over such Stego Image transfer. In this paper, we have discussed why RAKS. M. PK v2.0 is better and efficient over other techniques to detect whether an image is stego or normal by explaining the working mechanism of RAKS. M. PK 2.0. With the help of this research we put your effort to focus on those issues that make it difficult to find the Steganography Signature of the different Steganography tools. Also, we have tried to find out the Steganography Signature of "Securengine Pro" by the analysis of different graphs, patterns and study of prepared analysis table.

**KEYWORDS**: Steganography, RAKS. M. PK 2.0, Steganography Signature, Known Cover Attack, Security Threats.

## I. INTRODUCTION

Nowadays, a number of Steganography tools are available and use of Steganography is increasing day by day where Cryptography is not so very efficient. People tries to communicate with one another secretly. Law enforcement authorities have concerns in the trafficking of illicit materials through web page images, audio, and other files. It is very important to have the knowledge of methods of detecting hidden information and understanding the overall structure of this technology as it is crucial in uncovering these activities. People are using Steganography to avoid these policies and to send the message secretly

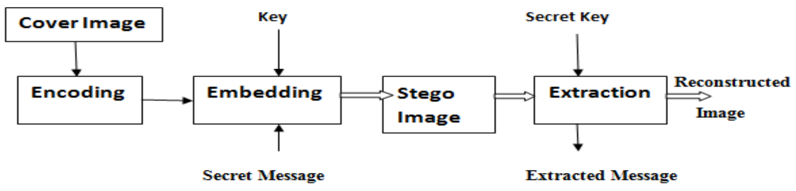


Fig. 1.1

In this paper, we have given all give a brief definition of Steganography and Steganalysis in general to provide a good understanding of these two terms but more importantly, we will talk about how to detect the existence of hidden information in case of Known Cover Attack and try to find out the Steganography Signature of "Securengine Pro" so that it will lead to uncover the Steganography Signature of various Steganography tools.

## II. TECHNIQUES THAT ARE PART OF STEGANOGRAPHY[1]

*Watermarking*:

a. Protects copyright owners of digital documents by hiding a signature in the information in a way that even a modified part of the document conserves the signature.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 3, Issue 8, August 2015

b. Prevents discovery by marking in a hidden and unique way every copy of a confidential document.

*Cover Channel*:

a. Allows people to communicate secretly by establishing a secret communication protocol.

b. Allows non-authorized communication through authorized communication of a firewall.

## III. INITIAL RESEARCH WORK

We have used bitmap image of size 20x20 in our research. It has been observed that when Steganography is done on a RGB image, there occurs some changes in its pixel values which leads to small distortions in the image. Using Securengine Pro v1.0, we performed Steganography on a gray scale 20x20 bitmap image, the text file we used contained only letter 'a', and the password used was 'aaaa'. We have found after multiple research that each time a Steganography is performed on a bitmap image using same text file and the password, even then every time changes occurs in different pixel values without any fixed pattern detected in the change of pixel positions used for hiding the bit values. We arrived at a conclusion that these changes in the pixel positions actually depends on the System time. In our next research we first froze the system time and performed the above research again. Disappointingly we noticed that still the pixel positions used during Steganography are not constant. Satisfied though, we found that the changes occurring now are quite less as compared when the system time was not frozen.

## IV. PROPOSED WORK

a. First of all we froze the system time by using the following code[2]:

(Please Refer Text Box 1)

b. We used Securengine Pro v1.0 to make a no. of Stego images. For this purpose we used a text file that contains "a" as text, "aaaa" used as password. AES algorithm is used for encrypting the text to create Stego images using the Securengine Pro v1.0.

c. We followed three methodologies for the analysis to find some common properties. For this purpose we developed two algorithms of our own i.e. RMP 1.1[3] and RMP 2.1[3].

*C++ code to freeze the system time*

```
#include<dos.h>
#include<stdio.h>
void main()
{

        int i, p_hr, p_min, p_sec,
        p_humd;
        struct time t;
        gettime(&t);
        printf("system time freezed
        … !!!");
        for(i=0; ; i++)
        settime(&t);

`
```

Text Box 1

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 3, Issue 8, August 2015

***Algorithm RMP 1.1*:**

Step I: Read a Normal RGB image as well as its corresponding Stego image in MATLAB.
Step II: Read The three Matrices obtained for the RGB Image.
Step III: Add the three Matrices of the normal and the stego image.
Step IV: Change the resultant Matrices into vertical matrix.
Step V: Transpose the Vertical matrix into horizontal matrix.
Step VI: Finally plot both the graphs of the matrices obtained, which overlap each other.

*MATLAB Code for RMP 1.1*:

```
i=imread('C:\Users\Dell\Desktop\Normal_RGB_image.bmp')
j=imread('C:\Users\Dell\Desktop\Stego_RGB_image.bmp')
a1=double(i(:,:,1))
a2=double(i(:,:,2))
a3=double(i(:,:,3))
x=a1+a2+a3
x1=x(:)
x2=x1'
b1=double(j(:,:,1))
b2=double(j(:,:,2))
b3=double(j(:,:,3))
y=b1+b2+b3
y1=y(:)
y2=y1'
stairs(x2,'red');
hold on;
stairs(y2,'green')
```

***Flow Chart for RMP 1.1:***

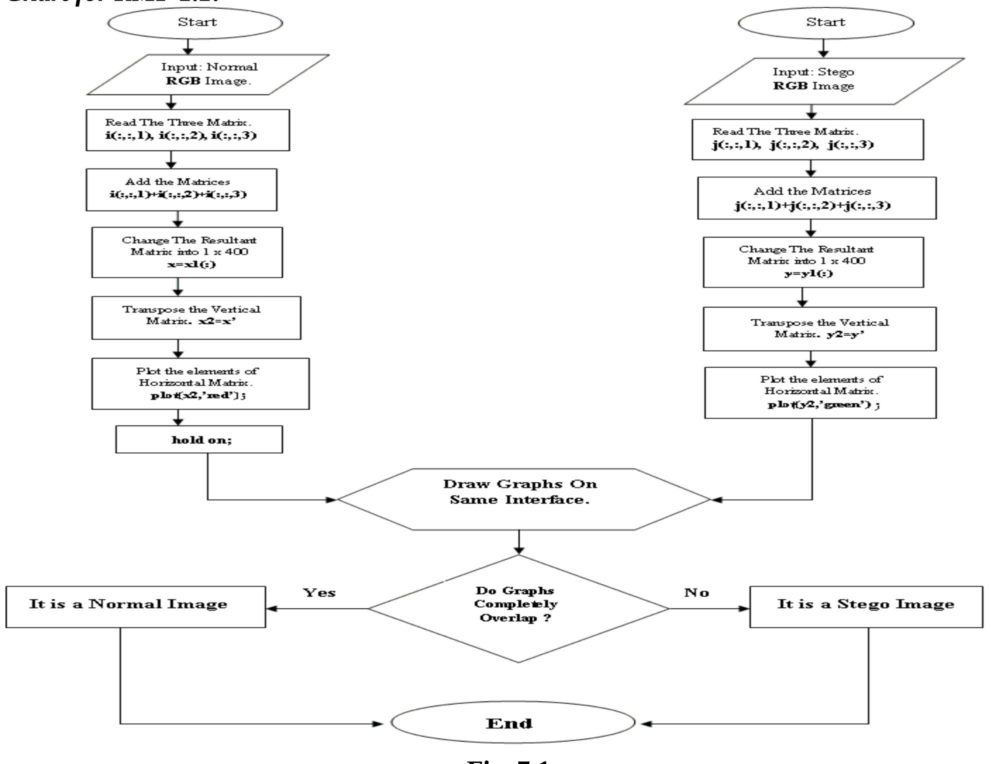

Start
Input: Normal RGB Image.
Read The Three Matrix. i(:,:,1), i(:,:,2), i(:,:,3)
Add the Matrices i(:,:,1)+i(:,:,2)+i(:,:,3)
Change The Resultant Matrix into 1 x 400 x=x1(:)
Transpose the Vertical Matrix. x2=x'
Plot the elements of Horizontal Matrix. plot(x2,'red');
hold on;
Start
Input: Stego RGB Image
Read The Three Matrix. j(:,:,1), j(:,:,2), j(:,:,3)
Add the Matrices j(:,:,1)+j(:,:,2)+j(:,:,3)
Change The Resultant Matrix into 1 x 400 y=y1(:)
Transpose the Vertical Matrix. y2=y'
Plot the elements of Horizontal Matrix. plot(y2,'green');
Draw Graphs On Same Interface.
Do Graphs Completely Overlap ?
Yes
No
It is a Normal Image
It is a Stego Image
End

**Fig. 7.1**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

Vol. 3, Issue 8, August 2015

***Algorithm RMP 2.1*:**

Step I: Read a Normal RGB image as well as its corresponding Stego images in MATLAB.
Step II: Add the three matrices obtained from the Normal RGB image and similarly add the three matrices obtained from its Stego RGB image.
Step III: Now find the difference by subtracting the resultant matrix of Stego image from that of normal image and store the result in another matrix of same dimension.
Step IV: If the resultant matrix is a zero matrix (i.e. all the elements of that very matrix is zero) then it will be a normal image otherwise it will be a Stego image.

*MATLAB Code*:

```
i=imread('C:\Users\Dell\Desktop\Normal_RGB_image.bmp')
j=imread('C:\Users\Dell\Desktop\Stego_RGB_image.bmp')
a1=double(i(:,:,1))
a2=double(i(:,:,2))
a3=double(i(:,:,3))
x=a1+a2+a3
b1=double(j(:,:,1))
b2=double(j(:,:,2))
b3=double(j(:,:,3))
y=b1+b2+b3
z=x-y;
```

If z is a zero matrix, it's a normal image otherwise it's a Stego image.

## V. RESULT AND ANALYSIS

We applied algo RMP 1.1 several times taking into consideration different RGB Images and obtained various different graphs.Both the graphs of Normal image and its Stego image are overlapped each other and difference between their graph pattern is clearly visible.
The Red graph pattern and the Green graph pattern corresponds to Normal and Stego image respectively.
Some of the obtained graphs are shown below:

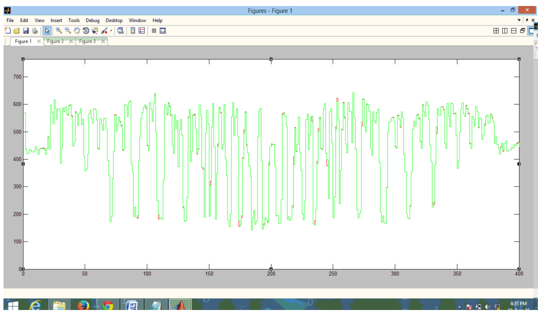


**Fig: 8.1**

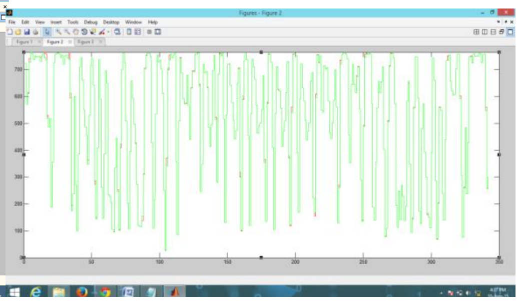


**Fig. 8.2**

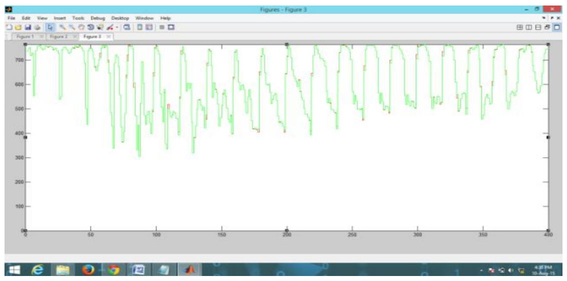


**Fig. 8.3**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

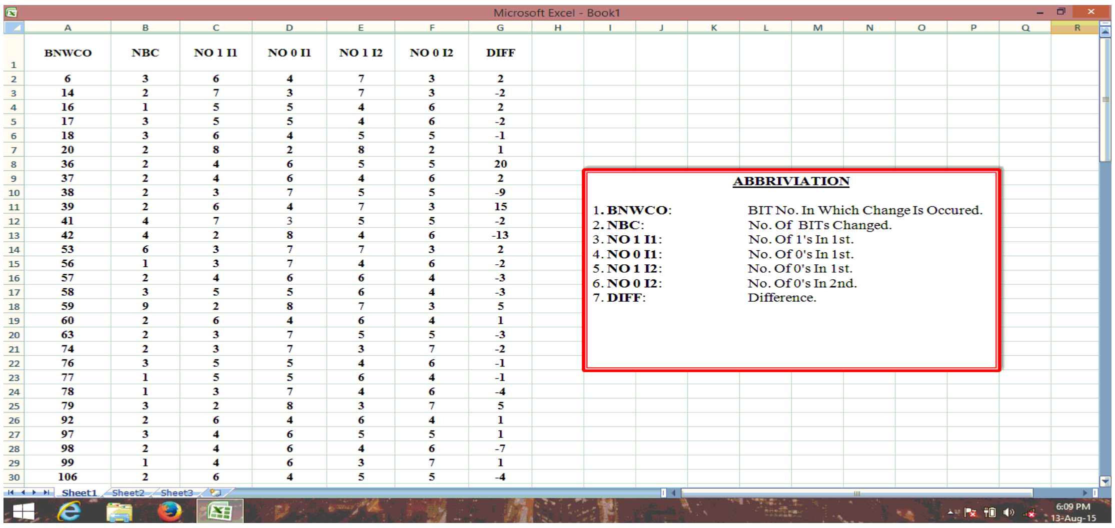
Vol. 3, Issue 8, August 2015

## VI. CONCLUSION

After the analysis of the above methodologies we came into the following conclusions:

**a.** ***Conclusion* 1:** Our Algorithm RAKS. M. PK 2.0 is found to be very efficient in case of Known Cover Attack for RGB image.

**b.** ***Conclusion* 2:** From *Analysis Table* as given below, we conclude that the change occurring in the pixel value is only upto a maximum of nine bits (i.e. 1st bit, 2nd bit, 3rd bit, 4th bit, 5th bit, 6th bit, 7th bit, 8th bit and the 9th bit).



| BNWCO | NBC | NO 1 I1 | NO 0 I1 | NO 1 I2 | NO 0 I2 | DIFF |
|---|---|---|---|---|---|---|
| 6 | 3 | 6 | 4 | 7 | 3 | 2 |
| 14 | 2 | 7 | 3 | 7 | 3 | -2 |
| 16 | 1 | 5 | 5 | 4 | 6 | 2 |
| 17 | 3 | 5 | 5 | 4 | 6 | -2 |
| 18 | 3 | 6 | 4 | 5 | 5 | -1 |
| 20 | 2 | 8 | 2 | 8 | 2 | 1 |
| 36 | 2 | 4 | 6 | 5 | 5 | 20 |
| 37 | 2 | 4 | 6 | 4 | 6 | 2 |
| 38 | 2 | 3 | 7 | 5 | 5 | -9 |
| 39 | 2 | 6 | 4 | 7 | 3 | 15 |
| 41 | 4 | 7 | 3 | 5 | 5 | -2 |
| 42 | 4 | 2 | 8 | 4 | 6 | -13 |
| 53 | 6 | 3 | 7 | 7 | 3 | 2 |
| 56 | 1 | 3 | 7 | 4 | 6 | -2 |
| 57 | 2 | 4 | 6 | 6 | 4 | -3 |
| 58 | 3 | 5 | 5 | 6 | 4 | -3 |
| 59 | 9 | 2 | 8 | 7 | 3 | 5 |
| 60 | 2 | 6 | 4 | 6 | 4 | 1 |
| 63 | 2 | 3 | 7 | 5 | 5 | -3 |
| 74 | 2 | 3 | 7 | 3 | 7 | -2 |
| 76 | 3 | 5 | 5 | 4 | 6 | -1 |
| 77 | 1 | 5 | 5 | 6 | 4 | -1 |
| 78 | 1 | 3 | 7 | 4 | 6 | -4 |
| 79 | 3 | 2 | 8 | 3 | 7 | 5 |
| 92 | 2 | 6 | 4 | 6 | 4 | 1 |
| 97 | 3 | 4 | 6 | 5 | 5 | 1 |
| 98 | 2 | 4 | 6 | 4 | 6 | -7 |
| 99 | 1 | 4 | 6 | 3 | 7 | 1 |
| 106 | 2 | 6 | 4 | 5 | 5 | -4 |

**ABBRIVIATION**

| | |
|---|---|
| 1. **BNWCO**: | BIT No. In Which Change Is Occured. |
| 2. **NBC**: | No. Of BITs Changed. |
| 3. **NO 1 I1**: | No. Of 1's In 1st. |
| 4. **NO 0 I1**: | No. Of 0's In 1st. |
| 5. **NO 1 I2**: | No. Of 0's In 1st. |
| 6. **NO 0 I2**: | No. Of 0's In 2nd. |
| 7. **DIFF**: | Difference. |

**c.** ***Conclusion 3:***

By using following formula we determine the percentage of distortion in the Pixels values, on the basis of which we conclude that the maximum distortion in pixels will not exceed 25%.

$$\text{Percentage of distortion (P)} = \frac{\textit{The No.Of Bits In Which Change Is Occured}}{\textit{Total No.Of Pixels Of the Original Image}} \text{ X } 100$$

***d.*** ***Conclusion 4:***

If

$$A_1 = \begin{pmatrix} A_{11}, A_{12}, \ldots\ldots, A_{1n}; \\ A_{21}, A_{22}, \ldots\ldots, A_{2n}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots. \ldots\ldots\ldots\ldots\ldots\ldots.. \\ A_{m1}, A_{m2}, \ldots.., A_{mn} \end{pmatrix}$$

and

$$B_1 = \begin{pmatrix} B_{11}, B_{12}, \ldots\ldots.., B_{1n}; \\ B_{21}, B_{22}, \ldots\ldots., B_{2n}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots... \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ B_{m1}, B_{m2}, \ldots.., B_{mn} \end{pmatrix}$$

If ‘$A_1$’ and ‘$B_1$’ are the RGB matrix of the Normal and Stego image respectively, then it is not possible in every case that elements of $A_1$ will be always greater than the elements of $B_1$, which is possible in the case of gray scale image[3].

## REFERENCES

[1] SANS Institute InfoSec Reading Room: http://www.sans.org/reading-room/whitepapers/stenganography/steganalysis-detecting-hidden-information-computer-forensic-analysis-1014
[2] FREAK SENSE, “How to freeze computer time using simple computer code’: http://freaksense.com/how-to-freeze-computer-time-using-simple-computer-code/
[3] Er. Rakesh Kumar, Pratik Kumar, “RAKS. M. PK 1.0, An Efficient Methodology To Determine The Steganography Signature Of Steganography Tools.”, IJCSC, DOI: 10.090592/IJCSC.2015.623, Vol. 6, no. 2, pp. 253-257, April – Sep 2015.

## BIOGRAPHY

Er. Rakesh Kumar. (Computer Science Researcher, Department of Computer Science, R.U, Ranchi, India). He has done B. Tech in CSE from CIT, Ranchi in 2015. He has designed his own programming language namely RAKs. M. PLUS which will be better and faster than c++ in many respects. He is nominated for *Honorary Doctorate* for his successful research works at *I.N.O.U, Lucknow.* He gave his valuable contribution in the field of Information security. He also proposed a new methodology to find the steganographic signature of different steganography tools. He is currently working on ‘Virus pot & Information security’. Image Processing, Cryptography, Information security and Data mining are his research areas of interest.

Pratik Kumar. (B. Tech. Final Year, Department of Computer Science, CIT, R.U, Ranchi, India) . He is a student of B. Tech in Computer Science and Engineering (Final year), CIT,Ranchi with a good Academic skills and have a keen interest towards the research works in the field of Computer Science. His research areas of interest includes Digital Image Processing, Wireless Sensor Network and Signal processing. He is a Co-author in the Original Research article published in International Journal Of Computer Science and Communication, entitled as “RAKS. M. PK 1.0 an efficient methodology to determine the Steganographic signature of Steganography tools”.