Scientific Journal Impact Factor (SJIF) : 3.518
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

STEGANOGRAPHY USING AES

Enhance the security which will not be easy to break

Avishek gupta[1], Rabinath Jha[2] and Samrat Bose[3]
[1,2,3] *Computer Science and Engineering, Abacus Institute of Engineering & Management*

Abstract— This project focuses on designing a system in order to secure information form attacker using two security layers: 1.Cryptography 2.Steganography.This is an approach taken to enhance the security which will not be easy to break. The cryptographic algorithms which are commonly in use are RSA and AES. AES is the new standard after DES (the previous encryption standard); AES is a symmetric algorithm and nowadays it is widely used to encrypt and send data via internet, however for ultimate security we are combining both cryptography and steganography in one system. Our propose is to provide a high level of cracking complexity naturally it will result security enhancement.

Keywords— Cryptography, Steganography, AES, Encryption, Decryption, PPM FILES

I. INTRODUCTION

This is an approach taken to enhance the security which will not be easy to break. The cryptographic algorithms which are commonly in use are RSA and AES. AES is the new standard after DES (the previous encryption standard); AES is a symmetric algorithm and nowadays it is widely used to encrypt and send data via internet, however for ultimate security we are combining both cryptography and steganography in one system. Our purpose is to provide a high level of cracking complexity naturally it will result security enhancement. Unlike DES, AES was designed such that an efficient software implementation is possible. A straightforward implementation of AES which directly follows the data path description, such as the description given in this chapter, is well suited for 8- bit processors such as those found on smart cards, but is not particularly efficient on 32-bit or 64-bit machines, which are common in today's PCs. In a na¨ ıve implementation, all time-critical functions (Byte Substitution, ShiftRows, MixColumn) operate on individual bytes. Processing 1 byte per instruction is inefficient on modern 32-bit or 64-bit processors. However, the Rijndael designers proposed a method which results in fast software implementations. The core idea is to merge all round functions (except the rather trivial key addition) into one table look-up. This results in four tables, each of which consists of 256 entries, where each entry is 32 bits wide. These tables are named a T-Box. Four table accesses yield 32 output bits of one round. Hence, one round can be computed with 16 table look-ups. On a 1.2-GHz Intel processor, a throughput of 400 Mbit/s (or 50 MByte/s) is possible. The fastest known implementation on a 64-bit Athlon CPU achieves a theoretical throughput of more than 1.6 Gbit/s. However, conventional hard disc encryption tools with AES or an opensource implementation of AES reach a perfomance of a few hundred Mbit/s on similar platforms.

II. METHODS

2.1 Cryptography

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication‖. Thus, cryptography is a technique to hide or represent information so nobody but authorized parties can decode it. Encryption is hiding the information whereas decryption is retrieving the information back from the encrypted data.

2.1 Steganography

Steganography is the hiding of a secret message within an ordinary message and the extraction of it at its destination. Steganography takes cryptography a step farther by hiding an encrypted message so that no one suspects it exists. Ideally, anyone scanning your data will fail to know it contains encrypted data.

2.3 AES (Advance Encryption Standard)

In 1997 NIST called for proposals for a new Advanced Encryption Standard (AES). Unlike the DES development, the selection of the algorithm for AES was an open process administered by NIST. In three subsequent AES evaluation rounds, NIST and the international scientific community discussed the advantages and disadvantages of the submitted ciphers and narrowed down the number of potential candidates.

Within the call for proposals, the following requirements for all AES candidate submissions were mandatory:

- block cipher with 128 bit block size.
- three key lengths must be supported: 128, 192 and 256 bit.
- security relative to other submitted algorithms.
- efficiency in software and hardware.

2.4 Encryption

Firstly program asks user to enter text which user want to encrypt.
Then text given by user is saved or copied to "plain.txt".

Then it pops up menu which asks user to select key size or simply level of security as AES supports 256,192,128 bit keys and according to the security level user have to enter 32B(max),24B(med),16B(min) characters

Now AES encryption process starts which convert text from "plain.txt" to cipher text and after that it writes every character of cipher text in "Ccipher.txt"

2.5 Decryption

Firstly LSB extraction takes place in which every last bit of RBG (after converting it to binary) is extracted in combination of 8 bit which is ultimately used to form a single character. Similarly all binaries is extracted from RBG values and converted to their respective character and every single character is copied to "ncipher.txt".

2.6 AES Decryption

Firstly it asks user to select security level or simply key length (256b(32B),192b(24B),128b(16B)) used to encrypt text.

Then decryption process starts which converts text from "ncipher.txt" into their original text. then Original text is ultimately copied to "nplain.txt".

2.7 PPM (Portable PixMap)

A PPM file consists of two parts, a header and the image data. The header consists of at least three parts normally delineated by carriage returns and/or linefeeds but the PPM specification only requires white space. The first "line" is a magic PPM identifier, it can be "P3" or "P6" (not including the double quotes!). The next line consists of the width and height of the image as ASCII numbers. The last part of the header gives the maximum value of the colour components for the pixels, this allows the format to describe more than single byte (0..255) colour values. In addition to the above required lines, a comment can be placed anywhere with a "#" character, the comment extends to the end of the line.

The following are all valid PPM headers:

- Header example 1: P6 1024 788 255
- Header example 2: P6 1024 788
  # A comment 255

- Header example 3: P3 1024(the image width) 788(the image height)
  # A comment 1023

The format of the image data itself depends on the magic PPM identifier. If it is "P3" then the image is given as ASCII text, the numerical value of each pixel ranges from 0 to the maximum value given in the header. The lines should not be longer than 70 characters.If the PPM magic identifier is "P6" then the image data is stored in byte format, one byte per colour component (r,g,b). Comments can only occur before the last field of the header and only one byte may appear after the last header field, normally a carriage return or line feed. "P6" image files are obviously smaller than "P3" and much faster to read. Note that "P6" PPM files can only be used for single byte colours.While not required by the format specification it is a standard convention to store the image in top to bottom, left to right order. Each pixel is stored as a byte, value 0 == black, value 255 == white. The components are stored in the "usual" order, red - green - blue.

2.7.1 Steganography process (writing to ppm file)

after formation of "Ccipher.txt" steganography process starts which takes a single character at a time convert those characters into binary.
Then LSB substitution takes place in Opic.ppm file(ppm file consist of ASCII values which represents RGB combination for every pixel) in which every single binary bit of character from MSB to LSB is copied to last bit of subsequent RGB ASCII values.
As we have changed last bit of RGB according to binaries of characters taken from "Ccipher.txt" ASCII values of RGB combination changes.
Then we create "Epic.ppm" in which we write both modified and non modified ASCII values.

2.7.2 C Code for LSB substitution in .PPM file

```
void stegno_encrypt() {
FILE *fc,*fp,*ft;
char buff[50]; int i,size;
fc = fopen("Ccipher.txt","r"); fp = fopen("prac.ppm","r"); ft = fopen("tinter.ppm","w"); rewind(fc);
rewind(fp);
for(i=0;i<3;i++)
{
fgets(buff,50,(FILE*)fp);
fputs(buff,ft);
}
int k; //in for loop
int kc; //chracter passed to function int kp; //same here as above
char bin_c[8];//for storing binaries char bin_p[8];
//reading no of characters
fseek(fc,0,2); //placing the file pointer at the EOF size=ftell(fc);
fseek(fc,0,0); //Placing the file pointer at the BOF
for(k=0;k<size;k++)
{
kc = fgetc(fc);
Ascii_Bin(bin_c,kc);
int m;
for(m=0;m<8;m++)
printf("%c",bin_c[m]);
int j; for(j=0;j<8;j++)
{
int u;
for(u=0;u<2;u++)
{
```

```
int kv; fscanf(fp,"%d",&kv); fprintf(ft,"%d\n",kv);
}
fscanf(fp,"%d",&kp);
printf("%d\n",kp); Ascii_Bin(bin_p,kp);
for(u=0;u<8;u++) printf("%c",bin_p[u]);
//comparing binaries
if(bin_c[j]!=bin_p[7])
{
bin_p[7]=bin_c[j];
}
int p = Bin_Ass(bin_p);
fprintf(ft,"%d\n",p);
//printf("\n%d",p);
}
}
int ex; while(fscanf(fp,"%d",&ex)!=EOF)
{
fprintf(ft,"%d\n",ex);
}
// printf("\n binaries: %s",bin); fclose(fp);
fclose(fc);
fclose(ft);
}
```

III. EXPERIMENTAL RESULTS

We demonstrate our working principle of AES Encryption and Decryption with the Graphical View.

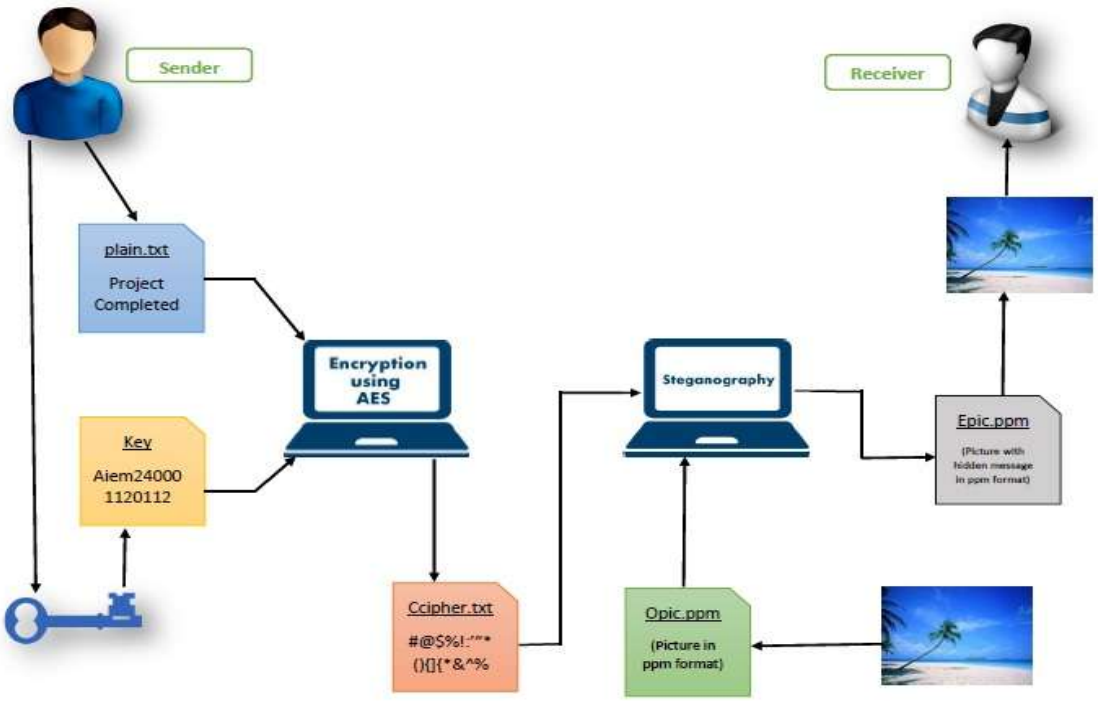

Fig. 3: (a) Graphical view of Working Principle AES Encryption and Decryption

3.1 AES Encryption

Firstly program asks user to enter text which user wants to encrypt. Then text given by user is saved or copied to "plain.txt". Then it pops up menu which asks user to select key size or simply level of security as AES supports 256,192,128 bit keys and according to the security level user have to enter 32B(max),24B(med),16B(min) characters .Now AES encryption process starts which convert text from "plain.txt" to cipher text and after that it writes every character of cipher text in "Ccipher.txt" .

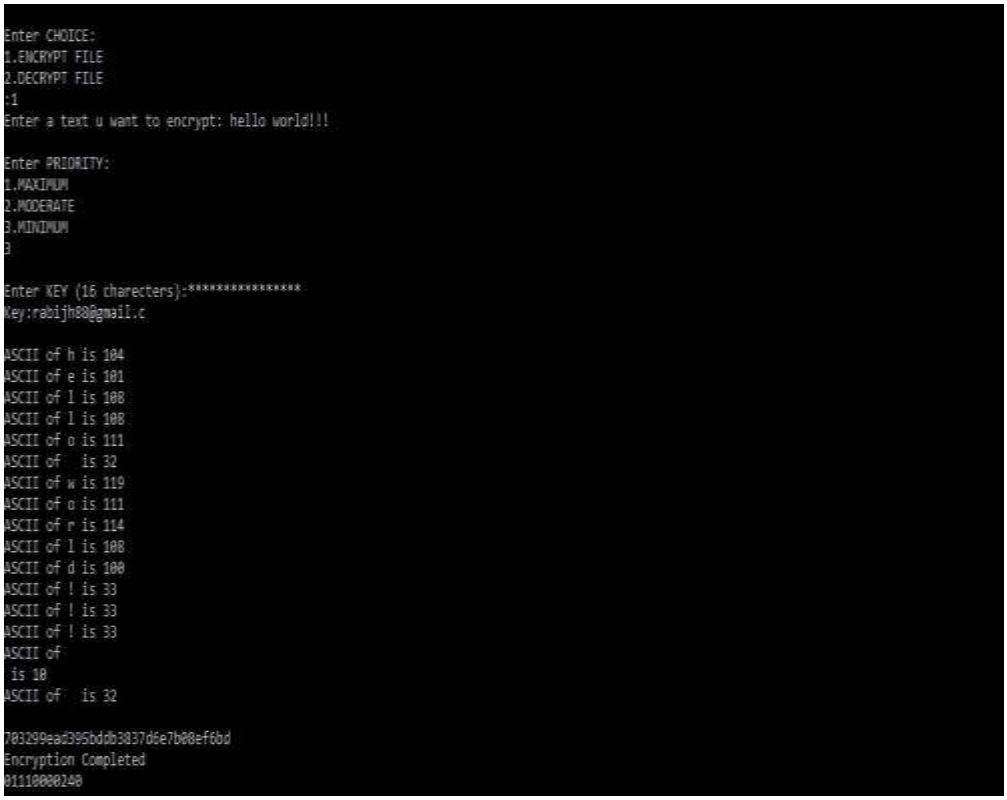

Fig. 3.1: (a) Graphical view of AES Encryption

3.2 AES Decryption

Firstly it asks user to select security level or simply key length (256b (32B), 192b (24B), 128b (16B)) used to encrypt text. Then decryption process starts which converts text from "ncipher.txt" into their original text. And then Original text is ultimately copied to "nplain.txt".

Fig. 3.2: (a) Graphical view of AES Decryption

3.3 Steganography process (writing to ppm file)

After formation of "Ccipher.txt" a steganography process start which takes a single character at a time convert those characters into binary. Then LSB substitution takes place in Opic.ppm file(ppm file consist of ASCII values which represents RGB combination for every pixel) in which every single binary bit of character from MSB to LSB is copied to last bit of subsequent RGB ASCII values. As we have changed last bit of RGB according to binaries of characters taken from "Ccipher.txt" ASCII values of RGB combination changes. Then we create "Epic.ppm" in which we write both modified and non modified ASCII values.

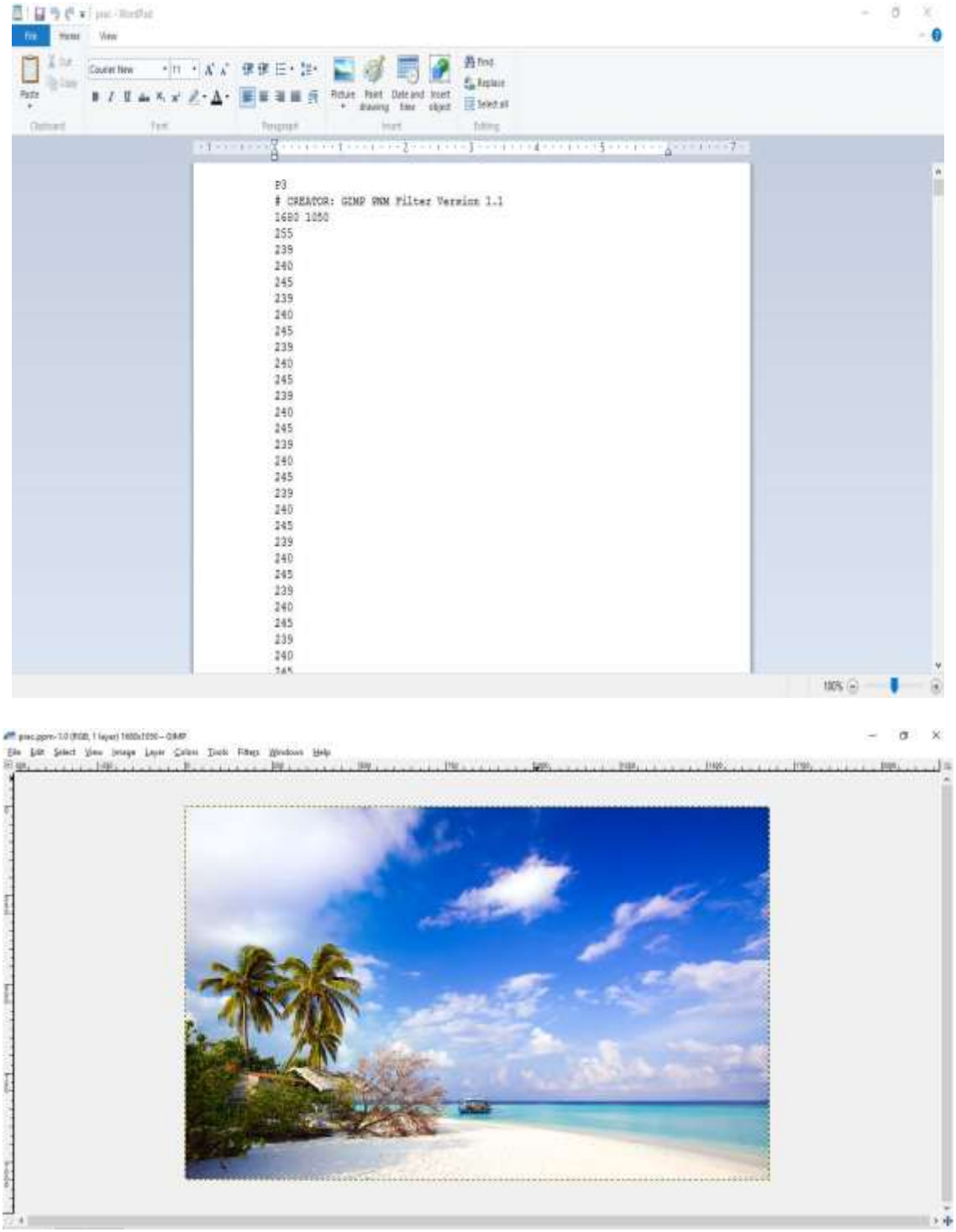

Fig. 3.3: (a) **Steganography process (writing to ppm file)** *(b) Desired Output after process*

3.4 Steganography process (reading from ppm file)

Firstly LSB extraction takes place in which every last bit of RBG (after converting it to binary) is extracted in combination of 8 bit which is ultimately used to form a single character.
Similarly all binaries is extracted from RBG values and converted to their respective character and every single character is copied to "ncipher.txt".

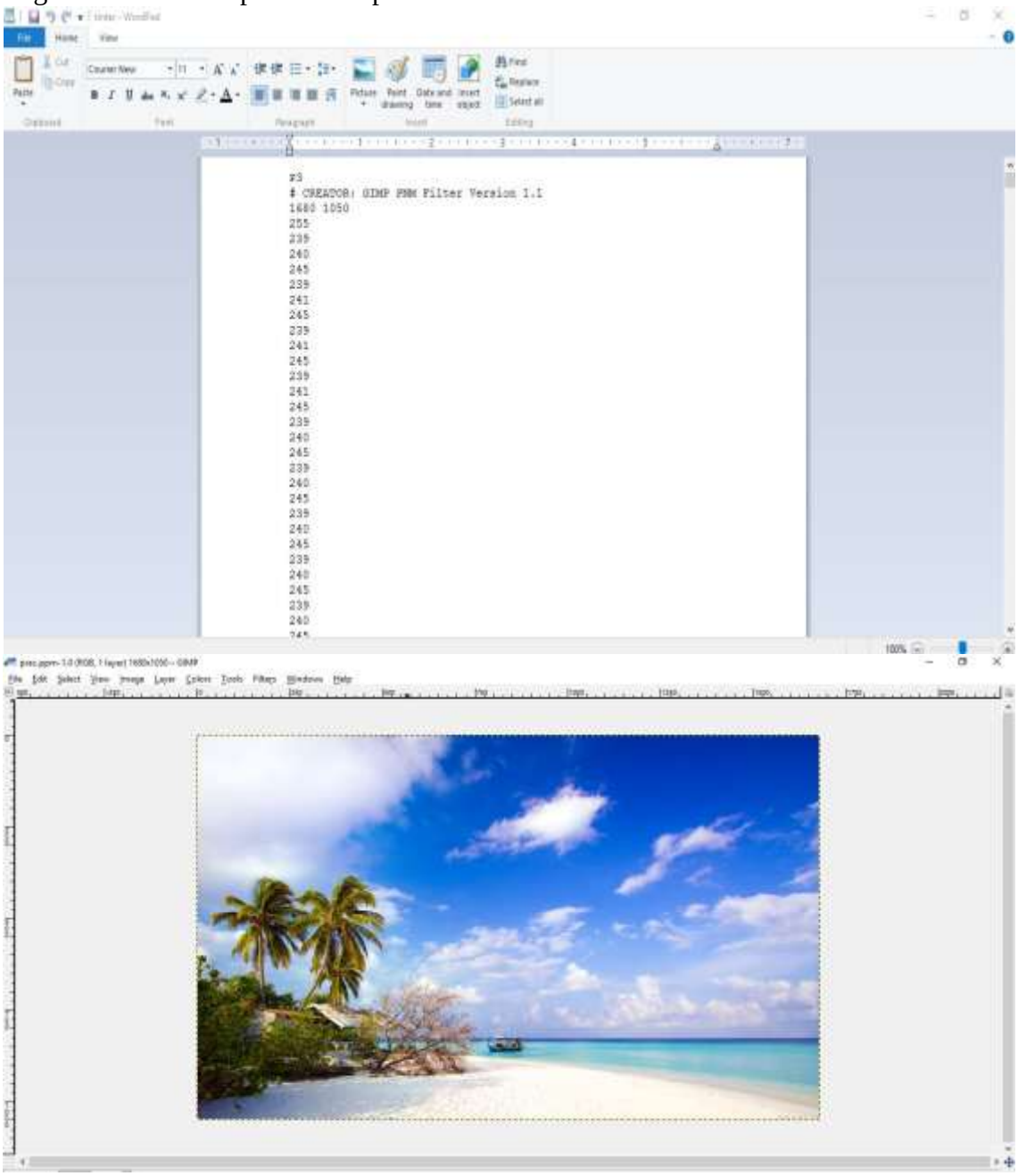

Fig. 3.3: (a)* Steganography process (reading from ppm file) *(b) Desired Output after process

IV. CONCLUSIONS

This project focuses on designing an system in order to secure information form attacker using two security layer – Cryptography and Steganography. This is an approach taken to enhance the security which will not be easy to break. AES is a symmetric algorithm and nowadays it is widely used to encrypt and send data via internet, however for ultimate security we are combining both cryptography and steganography in one system. Our propose is to provide a high level of cracking complexity naturally it will result security enhancement.

V. FUTURE SCOPE

The proposed system can be extended to standard video coding systems such as those using MPEG and other video formats. All the existing costly encryption products will have no use in future if the video encryption also invented with royalty free open source software. Therefore it will be the most flexible and cheaper solution.

REFERENCES

[1] Swati Paliwal and Ravindra Gupta, "A Review of Some Popular Encryption Techniques", International Journal of Advanced Research in Computer Science and Software Engineering Research Paper, Volume 3, Issue 2, February 2013, ISSN: 2277 128X .
[2] William Roche, "The Advanced Encryption Standard, The Process, Its Strengths and Weaknesses", University of Colorado, Denver, Spring 2006 Computer Security Class, CSC 7002, Final Paper May 6, 2006.
[3] Simar Preet Singh, and Raman Maini "Comparison of Data Encryption Algorithms" International Journal of Computer Science and Communication Vol.2, No. 1, January-June 2011, pp. 125-127[12]B. Gladman's AES related home page.
[4] Menezes, P. van Oorschot, and S. Vanstone,Handbook of Applied Cryptography, CRC Press, New York, 1997, p. 81-83.
[5] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone (1996). Handbook of Applied Cryptography. CRC Press. pp. 228–233. ISBN 0-8493-8523-7.
[6] ISO, JTC 1/SC 27 (2006). "ISO/IEC 10116:2006 - Information technology --Security techniques --Modes of operation for an n-bit block cipher". ISO Standards catalogue.
[7] NIST Computer Security Division's (CSD) Security Technology Group (STG) (2013). "Current modes". Cryptographic Toolkit. NIST. Retrieved April 12, 2013.

