I.J. Engineering and Manufacturing, **2018, 1, 1-15**
Published Online January 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijem.2018.01.01

Available online at http://www.mecs-press.net/ijem

Transmitting Security Enforcement By Text Encrypting and Image Hiding Technique using Combined Encrypt/Hide Keys

Mohammed Jawar Khami *

Basra technical institute, Southern technical university, Basra, Iraq.

Received: 13 May 2017; Accepted: 11 September 2017; Published: 08 January 2018

Abstract

Comparative study of cryptography and steganography techniques shows that they have some strong and weak points when they used alone. But as we know from soft computing techniques (neural, genetic, and fuzzy computing), that when combining (hybridizing), more than one techniques, by the suitable way to do a job, the outcome will be a better technique with more strong points and less weak points. Work of this paper represents an attempt to prove that combining cryptography with steganography techniques will result in hard transmitting system to break and thus enforcing security issues of secret text data transmitting over public channels. Matlab programs are written to encrypt plain text secret information following AES encrypt/decrypt algorithm with a key of 128 bits long and then hide/extract the text according to LSB insertion method with a key of 128 bits long too. System tests show that both techniques enforce each other and private data transmitting become more secure.

Index Terms: Cryptography, Steganography, Keys combining, AES algorithm, LSB hide/extract techniques.

1. Introduction

Nowadays, the Internet has made it possible for users to send and receive all types of digital data (Audio, image, text, video), from anywhere around the world. Also, from security point view, communication of private and secret information (electronic financial transactions, e-business applications, and secure video surveillance systems), is a major challenge and its complexity increases with the levels of sophistication [1]. Many attempts have been made to secure data transmitting over the internet either by making it illegible or unreadable through encoding or masking. In general, solutions for maintaining the secrecy of data transmission over an unsecured media such as the internet can have two approaches. These are cryptography and steganography. Regardless

* Corresponding author. +964 780 133 4525
E-mail address: mjkhami@stu.edu.iq, mjkhami@yahoo.com

many papers showed that neither cryptography nor steganography comes up with the ultimate solution for privacy preservation in open systems. However, the combination of steganography and cryptography can greatly increase the security of communication and is usually considered a good practice for securing security driven communication environments[2].

Cryptography is the process of converting original data into cipher copy so that the original data is not readable by the third party or making it difficult for intruders to extract the original data. [3], Steganography, on the other hand, is the process of hiding secret data in another clear covering media so there is no knowledge of secret data existence in that cover [4]. Comparative study of these two approaches shows that they have strong and weak points. Security point enforcements come out when combining (hybridizing), both of them in one system, the outcome will be harder than ever for any intruder to get the original data. Since when the steganography fails and the secret data can be detected, it is still of no use as it is encrypted and thus unreadable[5].

There are many algorithms and methods to accomplish data steganography [6-9] and cryptography [10-12]. And to combine them in one system, it requires selecting the best method from each one of them to suit and verify the intended needs of the applications. In this work, the encryption method is according to the advanced encrypting standard (AES) algorithm with a Private encrypting key of 128 bits long and the steganography technique is text-in-image (image hiding technique), hiding by least significant bit (LSB) insertion method [13-15]. Both methods (Encrypt/Hiding), are selected for many reasons such as their simplicity, difficult to break, and easily programmed.

2. Proposed System

The goal of this work is to design a software to Encrypt file of secret English plain text first and then embed it into gray or color cover image. The size of encrypted text (and thus the original secret plain text size), is bounded only by the size of the selected cover image. Many programming tricks have been included in writing this software. For example, at encryption stage and in order to make any attempt for breaking the encrypting technique much harder, the main encrypting key is generated by combining two different keys. First key (S-Key), is the one that must be given by the sender while the second key (R-Key), should be obtained from the receiver. Each key should be exactly 128 bits (16 characters) long, and so will the length of the combined key (SR-Key). By this way, an effort to decrypt the encrypted text and getting back the original secret plain text needs to know not just one key but both of them at the same time and Knowing the way by which they combined together, and this is less probable to happen.

At the embedding stage the following steps are included:

- Hide encrypted text at the least significant bit, in bit-plain-1, of the cover image starting at byte 1 of the image, seems to be easy to extract it back, this may look true but the difficulty comes out here by not letting intruders knowing the exact size of the embedded text and thus any attempt to read and reconfigure it back will be definitely difficult since, if the encrypted text size is unknown then decrypting it also will be difficult.
- The applied hiding technique uses a hiding key. This key is the sender key (S-Key) which will be embedded in the same cover image on bit-plain-2 of randomly selected bytes. The randomness of locations, where sender key is hidden, depends on setting the seed of system random number generator to the value of the receiver key (R-Key).

The proposed system can be described in steps at two locations as follow:

Sender location:

Step-1: Read input plain text,

- Read cover image data matrix,
- Read sender key string (S-Key), and,
- Read receiver key string (R-Key).

Step-2: Combine S-Key and R-key into the same length encrypting key (SR-Key).
Step-3: Encrypt input plain text according to AES encrypting algorithm and SR-Key.
Step-4: Hide encrypted text in cover image starting at byte 1 and bit-plain-1.
Step-5: Hide size of encrypted text in last 32 bytes of the cover image and in bit-plain-1.
Step-6: -Set system random number generator seed to the value of R-Key string,

- Use system random number generator to create a vector of 128 elements. All vector elements values are unique and in the range from 1 to (cover image size – 32).
- Hide S-Key in the cover image at byte locations equal to element values of the created vector and in bit-plain-2.

Step-7: Modified image with hidden text is ready to transmit.

Receiver location:

Step-1: -Read the received image data matrix, and
-Read receiver Key string (R-Key)
Step-2: Extract size value of encrypted text from the last 32 bytes of the received image and from bit-plain-1.
Step-3: Extract encrypted text from received image starting at byte 1 and from bit-plain-1.
Step-4: Set system random number generator seed to the value of R-Key string,

- Use system random number generator to create a vector of 128 unique elements values in the range of 1 to (cover image size – 32).
- Extract S-Key from cover image byte on locations equal to element values of the created vector and from bit-plain-2.

Step-5: Combine S-Key and R-key into decrypting key SR-Key of the same length.
Step-6: Decrypt the obtained encrypted text according to AES encrypting algorithm and SR-Key.
Step-7 Use the decrypted text as the retrieved secret plain text.

The proposed system is coded using Matlab programming language. Main program descriptive flow charts are depicted in Fig. 1. for the sender location and Fig. 2. for the receiver location. The written program can deal with image's files of type '.bmp', '.tif', '.png', and '.jpg'. The current plain text should be from English text writing. These programs could be edited and modified to be capable of encrypting/hiding Arabic and other writing languages. Main programs and functions are listed at the end of this paper (Appendix A). The only unlisted functions are the encrypt (cipher), and decrypt (decipher), functions. These functions have a standard form and can be easily downloaded from the internet.

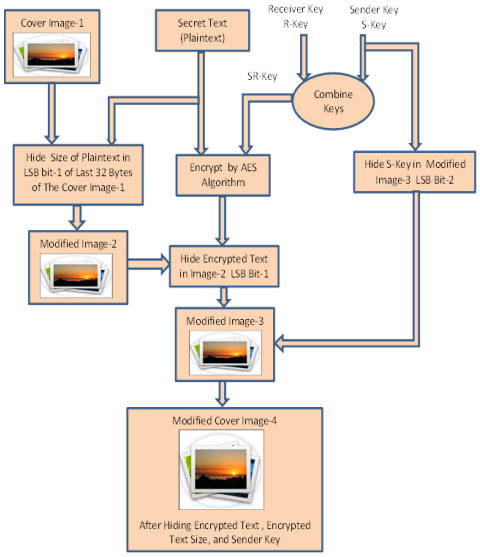

Fig.1. System Flow Diagram at Sender Location.

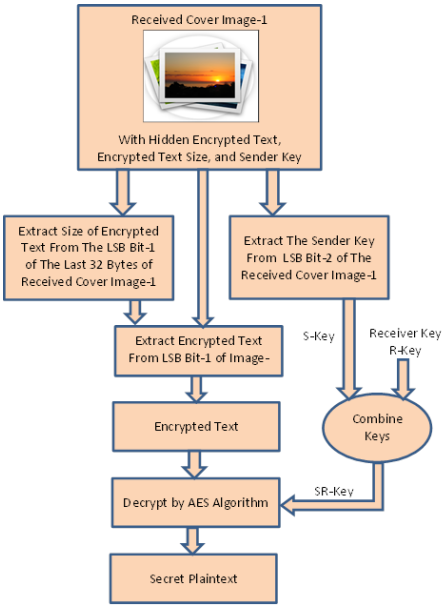

Fig.2. System Flow Diagram at the Receiver Location.

3. System Test and Implementation

The proposed system is tested, as shown in Fig.3., by using two files of English plain text. First one contains 4361 characters while the second has 215 characters. Also, the test is done with color and gray images of different types (bmp, tiff, png, and jpg images), with different images size. All peak signal to noise ratio

(PNSR), calculations have been written in Table 1. and they all satisfy the purposes of the hard visual detection of the embedded text. The only noticeable value that was shown in Table 1. is the big change in the row for the image of type ‘jpg'. Its size changes from 12.8 kB to 240 kB. That is because after the text is hidden in it, its type is changed deliberately from ‘jpg’ to ‘bmp’, in order to save the image without compression, (since ‘.jpg' image changes its size when re-saving it again due to compression processing property).

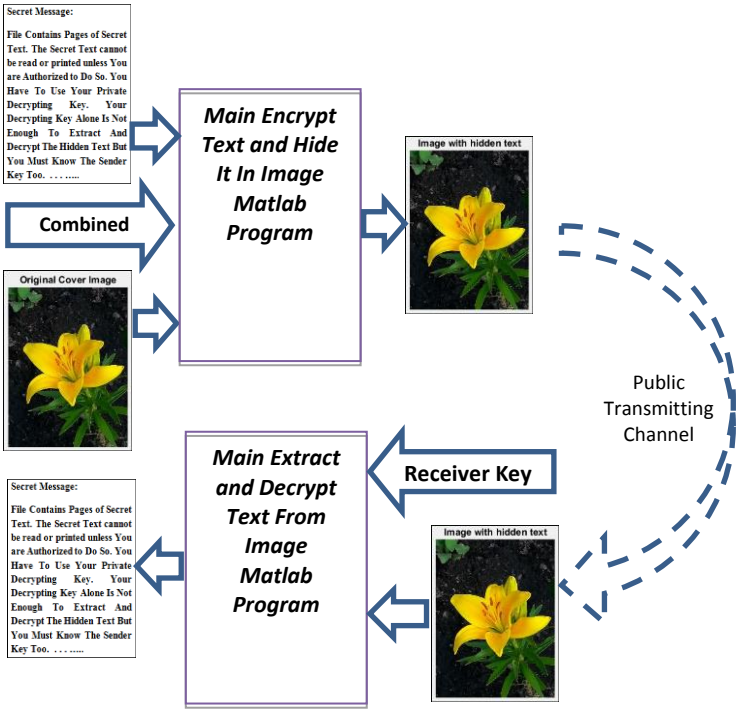

Fig.3. System Implementation.

Table 1. PSNR for Different Image Types and Hidden Text Sizes

Image Filename	Image Type	Image Color	Image RowXCol	Size (kB)	New Image Type	New Size (kB)	Text of 4361 Bytes	Text of 215 Bytes
							PSNR (db)	PSNR (db)
Autumn	Tiff	color	345X206	208	tiff	209	58.8741	70.6561
cameraman	Tiff	grey	256X256	64.5	tiff	63.5	53.7751	65.3126
Football	**Jpg**	color	320X256	**12.8**	**bmp**	**240**	59.5572	71.2441
Forest	Tiff	grey	447X301	120	tiff	120	56.9068	68.8658
lighthouse	Png	color	480X640	473	png	472	65.3014	77.0427
Fruit	Bmp	color	615X456	822	bmp	822	64.9221	76.5239

4. Conclusions

Combining AES encrypt/decrypt algorithm with 128 bits key, and hide/extract LSB insertion method with 128 bits hiding key too, give new and more secured transmitting technique. Sender and receiver Keys

combining make the process of breaking any one of them more difficult, since intruder need to know both keys to break the system. if one technique fails, the system will continue securing the secret information by the other technique. Experiments show that secret text size only bounded by cover image size. Noise added to the cover image, due to embedding process, is small and modifications on original images are hard to be detected for most image types (except for '.jpg' image type). This paper programs can be updated easily to encrypt-hide-extract-decrypt text writing other than English text. The programs can be used also in increasing security features in communication technology for smartphones since security issues for these devices are limited.

References

[1] Palak Mahajan, "Steganography: A data hiding technique", International journal of advanced research in computer science and software engineering, Vol. 4, Issue 11, Nov. 2014.
[2] Auqib Hamid Lone, Ab Waheed Lone, Moin Uddin, "A novel scheme for image authentication and secret data sharing", International journal of computer network and information security(IJCNIS), Vol.8, No.9, pp.10-18, 2016.DOI: 10.5815/ijcnis.2016.09.
[3] A. Anuradha, Hardik B. Pandit, "Review on information techniques: A comparative analysis", IJRET, Vol. 5, Issue 2, Feb. 2016, pp. 128-132.
[4] Ramanpreet Kaur, Baljit Singh, "Survey and analysis of various steganography techniques", IJESAT, Vol. 2, Issue 3, May-June 2012.
[5] Arvind Kumar, Km. Pooja, "Steganography – A data hiding technique", IJCA (0975-8887), Vol. 9, No. 7, Nov. 2010, pp. 19-23.
[6] Sahar A. El_Rahman, "A comprehensive image steganography tool using LSB scheme", IJIGSP, vol.7, no.6, pp.10-18, 2015.DOI: 10.5815/ijigsp.2015.06.02
[7] Sabyasachi Samanta, Saurabh Dutta, Goutam Sanyal,"A Novel approach of text steganography using nonlinear character positions (NCP)", IJCNIS, vol.6, no.1, pp.55-60,2014. DOI: 10.5815/ijcnis.2014.01.08
[8] Fahd Alharbi,"Novel high-quality data hiding system", IJIGSP, vol.5, no.7, pp.47-53, 2013.DOI: 10.5815/ijigsp.2013.07.07
[9] Ibrahim M. Hussain, M. Kamran Khan, Mohammad Naseem, Aisha Ajmal, Osama M. Hussain, "Improved bit plane splicing LSB technique for secret data hiding in images using linear congruent method", IJIGSP, vol.4, no.7, pp.1-14, 2012.
[10] P. Srinivasarao, P. V. Lakshmipriya, P. C. S. Azad, T. Alekhya, K. Raghavendrarao, K. Kishore, "A technique for data encryption and decryption", International journal of future generation communication and networking, vol. 7, no. 2 (2014), pp.117-126.
[11] Thomas Baigneres, Pascal Junod, Yi Lu, Jean, "A classical introduction to cryptography exercise book", Springer science & business media, 2006.
[12] Douglas Selent, "Advanced encryption standard", nSight: Rivier academic journal, Vol. 6, Num. 2, Fall 2010.
[13] Mohammed J. Khami, Lemya G. Shehab and Zeynab M. Jawar, "Matlab Coding For Text Steganography System By Using LSB Insertion Method With Key", Basrah journal of science (A) Vol.33 (2), 37-51, 2015.
[14] Champakamala .B.S, Padmini.K, Radhika .D. K., "Least significant bit algorithm for image steganography", International journal of advanced computer technology (IJACT) ISSN: 2319-7900 34 | Vol. 3, Issue. 4. August 25, 2014.
[15] Gabriel Macharia Kamau, Stephen Kimani, Waweru Mwangi, "An enhanced least significant bit steganographic method for information hiding", Journal of information engineering and applications ISSN 2224-5782 (print) ISSN 2225-0506 (online) Vol 2, No.9, 2012.

Authors' Profiles

Mohammed Jawar Khami (1953) is an assistant professor of computer science at Basra technical institute in the Southern technical university, Iraq. He received his B.Sc. (Hon.) in electrical and electronic engineering 1982 from Sunderland Polytechnic, England, UK. Also, he received his MSc (1989) and Ph.D. (2000) in computer science from college of science, Basra University, Iraq. His area of research includes pattern recognition, computer based device's control systems, data encryption, and data hiding. He is a former computer center manager and head of the computer systems department at Basra technical institute.

Appendix A. The proposed system Matlab software

A.1. Encrypt/Hide Program.

```
% EncryptHide.m
% Dr. Mohammed J. Khami
% Comp. Sys. Dep./ Basra Tech. Institute.
% mjkhami@stu.edu.iq, mjkhami@yahoo.com, mjkhami@gmail.com
% 29/04/2017
%
clc; clear; close all;fclose all; commandwindow; WD=cd();
% Set off warning for big size image.
warning ( 'off','images:initSize:adjustingMag') ;
%
% This program is to encrypt content of plain text file "filename.txt"
% by AES-algorithm with sender and receiver keys. And to save output
% encrypted-text in "En_filename.txt" file on the same path of the
% input plain text file. And then hide it in image.
% Note: Sender and Receiver keys have maximum lengths of 128 bits.

%% Input Cover image Path and Filename
[CoverImageFilename, CoverImagePth]=uigetfile({'Image file(*.png;*.tif;
   *.jpg;*.bmp)'},'Choose Cover Image To Encode.');
if isequal(CoverImageFilename,0) || isequal(CoverImagePth,0)
   return  % User canceled.
End

% Read CoverImage Data Matrix.
fmt=CoverImageFilename(end-2:end); % Get image type
CoverImage= imread([CoverImagePth, CoverImageFilename],fmt);

%% Prog. Section-2 : Get Encryption/Decryption PlainKey.
[S_key, R_key]=getplainkey5(); %Get S-key/R-Key (each of 128 bit long).
PlainKeyLength1=length(S_key); PlainKeyLength2=length(R_key);
% Make sure each key is 128 bit long.
```

```
if PlainKeyLength1<1 || PlainKeyLength1>16 ...
      || PlainKeyLength2<1 || PlainKeyLength2>16
   ttext='Error: Sender/Reciever Keys must be 1-to-16 characters Long.';
   uiwait(msgbox({ttext},'Error','error','modal'));
   cd(WD); clc; return;
end

% Combine S_Key with R_Key to Get SR_Key by calling (two_keys_in_one5)
SR_key=two_keys_in_one5(S_key, R_key);
PlainKey=SR_key; % Encryption key is ready.

% Prepare AES algorithm's Parameters, by calling standard function
% 'aes_init.m' loaded from the internet.
[s_box, inv_s_box, w, poly_mat, inv_poly_mat] = aes_init(PlainKey);

%% Read Plaintext Filename
[filen1, path1] =uigetfile({'*.txt';},'Choose Plain Text File: ');
if isequal(filen1,0) || isequal(path1,0)
   ttext='Error: Filename must not be empty';
   uiwait(msgbox({ttext},'Error','error','modal'));
   cd(WD); return % User cancelled.
end
PlainTextFileName=[path1,filen1]; FE1=fopen(PlainTextFileName,'r');

% Read Plaintext
plaintext=fread(FE1); PlainText=plaintext';

%% Divide the read Plaintext into slices of 16 characters wide.
PlainTextLength=length(PlainText);
loop_int=fix(PlainTextLength/16); loop_rem=mod(PlainTextLength,16);
if loop_rem>=1 && loop_rem<=15
   for i=loop_rem+1:16
      PlainText=[PlainText,' '];
   end
   loop_int=loop_int+1;
end
PlainText = double(PlainText);
for cy=1:loop_int
   PlainText1=PlainText((cy-1)*16+1:cy*16);
   % Encrypt one text slice at a time by calling cipher function.
   ciphertext = cipher (PlainText1, w, s_box, poly_mat);
   if cy==1
      EncText =ciphertext;
   else
      EncText =[EncText,ciphertext];
   end
end % End of Ciphering all file contents.
```

```
%% Hide encrypted text in CoverImage
% 1) Hide encrypted text in CoverImage by LSB in bit-plain-1 of image
%    data starting from the first byte from the CoverImage.
% 2) Calculate encrypted-text-size then hide size value at the last 32
%    bytes of the CoverImage (after converting text size value into
%    binary string of 32bits).
% 3) Hide Reciever key string in bit-plain-2 of randomly selected data
%    points of CoverImage.
% All above Hidings will be done by calling function:
%     "EmbedEncryptedTextInImageWithKeys.m".
%
ModifiedCoverImageH=EmbedEncryptedTextInImageWithKeys(CoverImage,EncText,
                                    S_key,R_key);
% End of hiding Process.
figure(1);
subplot(121);imshow(CoverImage,[]);title('Original Cover Image');
subplot(122);imshow(ModifiedCoverImageH,[]);title('Image with hidden text');
OutPutImageFilename=[CoverImagePth,'O_', CoverImageFilename];
% When image is of '.jpg' type it must be save as '.bmp' file.
if lower(OutPutImageFilename(end-2:end))=='jpg'
   OutPutImageFilename(end-2:end)='bmp';
end
imwrite(ModifiedCoverImageH,OutPutImageFilename);
fclose all; cd(WD);
% End of Encryption/Hide Program.
```

A.2. Extract/Decrypt program.

```
% ExtractDecrypt.m
% Dr.  Mohammed J. Khami
% Comp. Sys. Dep./ Basra Tech. Institute.
% mjkhami@stu.edu.iq , mjkhami@yahoo.com , mjkhami@gmail.com
% 29/04/2017
%
clc; clear; close all; fclose all;
% Set warning for big size image off.
warning ( 'off','images:initSize:adjustingMag');
%% Read the Modified Image Data File.
% Get the ModifiedCoverImageH Filename and its Path
[filen pth]=uigetfile({'O_*.*'},'Choose Modified Cover Image To Encode.');
if isequal(filen,0) || isequal(pth,0)
   return  % User cancelled.
end % if
ModifiedCoverImageH= imread([pth filen]); % Original cover image
close all;
% Gets Rows and Columns of input image.
[ImageRows,ImageCols,ImageClr]=size(ModifiedCoverImageH);
if ImageClr>1
```

```
   ModifiedCoverImage=ModifiedCoverImageH(:,:,1);
else
   ModifiedCoverImage=ModifiedCoverImageH(:,:);
end
RetrievedCoverImageInBinary=dec2bin(ModifiedCoverImage);
% Get Reciever R_Key
R_Key=getplainkey_receiver();
% RC_Hidden is row & col. number where we hide the secret text size.
RC_Hidden='';
for i=ImageRows*ImageCols-31:ImageRows*ImageCols
   RC_Hidden=[RC_Hidden,char(RetrievedCoverImageInBinary(i,8))];
end
Row_tr=bin2dec(RC_Hidden(1:16)); Col_tr=bin2dec(RC_Hidden(17:32));
RetrievedTextInBinary=RetrievedCoverImageInBinary(1:Row_tr*Col_tr,8);
RetrievedTextIn_ascii=bin2dec(reshape(RetrievedTextInBinary ,Row_tr,Col_tr));
RetrievedTextInChar= char(RetrievedTextIn_ascii)';

% Set seed of the random number generator to R_Key
rkey=0;
for i=1:16
   rkey=rkey+2^i*double(R_Key(i));
end
rng(rkey);
% Generate random byte locations
indx=randperm(size(ModifiedCoverImage,1)*size(ModifiedCoverImage,2)-32,128);
% Hide  S_Key in the randomly generated byte locations
S_Key_Bin='';
for i=1:128
   S_Key_Bin=[S_Key_Bin,RetrievedCoverImageInBinary(indx(i),7)];
end
S_Key='';
for i=1:8:128
   S_Key=[S_Key,char(bin2dec(S_Key_Bin(i:i+7)))];
end
CompositeKey=two_keys_in_one(S_Key, R_Key);
PlainKey=double(CompositeKey);

% Prepare AES algorithm parameters.
[s_box, inv_s_box, w, poly_mat, inv_poly_mat] = aes_init(PlainKey);
PlainText=RetrievedTextInChar; PlainTextLength=length(PlainText);
% Divide PlainTex into slices each of 16-characters.
loop_int=fix(PlainTextLength/16); loop_rem=mod(PlainTextLength,16);
if loop_rem>0 && loop_rem<16
   for i=loop_rem+1:16
      PlainText=[PlainText,' '];
   end
   loop_int=loop_int+1;
end
```

```
PlainText0 = double(PlainText);

for cy=1:loop_int
   if cy==1
      PlainText1=PlainText0((cy-1)*16+1:cy*16);
   else
      PlainText1=PlainText0((cy-1)*16+1:cy*16);
   end
   decText = inv_cipher (PlainText1, w, inv_s_box, inv_poly_mat);
   if cy==1
      DecText =decText;
   else
      DecText =[DecText,decText];
   end
end
DecText=char(DecText);
DecText
% End of Extract/Decrypt Program.
```

A.3. Getplainkey Function

```
function [SPlainKey, RPlainKey]=getplainkey()
% Function to returns sender key ‘SPlainKey' and Receiver key 'RPlainKey'
% using inputdlg() matlab command.
SPlainKey=''; RPlainKey=''; PlainKey='';
prompt = {'Sender (Encryption) Key?','Receiver (Decryption) Key?'};
dlg_title = 'Input Keys'; num_lines = 1; defaultans ={'Khami1953';''};
options='on';
PlainKey=inputdlg(prompt,dlg_title,num_lines,defaultans,options );
if isempty(PlainKey)
   return;
end
PlainKey1=char(PlainKey{1}); PlainKey2=char(PlainKey{2});
while true
   if PlainKey1(1)==char(32)
      PlainKey1=PlainKey1(2:end);
   else
      break;
   end
end
while true
   if PlainKey2(1)==char(32)
      PlainKey2=PlainKey2(2:end);
   else
      break;
   end
end
PlainKeyLength1=length(PlainKey1); PlainKeyLength2=length(PlainKey2);
```

```
if PlainKeyLength1<16
   for i=1:16-PlainKeyLength1
      PlainKey1=[PlainKey1,' '];
   end
end
if PlainKeyLength2<16
   for i=1:16-PlainKeyLength2
      PlainKey2=[PlainKey2,' '];
   end
end
SPlainKey=PlainKey1; RPlainKey=PlainKey2;
% End of function 'getplainkey()'.
```

A.4. Two_keys_in_one Function

```
function CompositeKey = (SenderKey ,RecieverKey)
% Function to convert two key strings each of 16 char's into one
% composite 16 char key.
%% Make sure both given keys are 16 char long.
s=length(SenderKey); r=length(RecieverKey);
if (s~=16)||(r~=16)
   CompositeKey='';
   return
end
a_bin=''; b_bin='';
for i=1:16
   a_bin=[a_bin,dec2bin(SenderKey(i),8)];
   b_bin=[b_bin,dec2bin(RecieverKey(i),8)];
end
ab=[];
for i=1:128
   ab(i)=xor(str2num(a_bin(i)),str2num(b_bin(129-i)));
end
nk=reshape(ab,16,8); CompositeKey=[];
for i=1:16
   CompositeKey(i)=(bin2dec(num2str(nk(i,1:8))));
end
% End of function 'two_keys_in_one(SenderKey,RecieverKey)'.
```

A.5. EmbedEncryptedTextInImageWithKeys function

```
function ModifiedCoverImageH =
EmbedEncryptedTextInImageWithKeys(CoverImage,Message,S_Key,R_Key)
% Function 'EmbedEncryptedTextInImageWithKeys' to do the following:
% 1) Hide encrypted text 'Message' in CoverImage by LSB in bit-plain-1 of
%    image data starting from the first byte from the CoverImage.
% 2) Calculate encrypted-text-size then hide size-value in the last 32
%    bytes of the CoverImage (after converting text size-value into
```

```
%     binary string of 32 bits).
% 3) Hide Reciever key 'R-Key' string in bit-plain-2 of randomly selected
%    data points of CoverImage.
%
MessageInDouble=Message;
% Convert text array into binary number of ii*jj rows and 8 columns.
MessageInBinary=dec2bin(MessageInDouble,8);
% Get size of the binary message
[MessageBinaryRows, MessageBinaryCols]=size(MessageInBinary);
% Combine MessageBinaryRows & MessageBinaryCols into one string in
% binary representation.
MessageBinaryRowsInBinary=dec2bin(MessageBinaryRows,16);
MessageBinaryColsInBinary=dec2bin(MessageBinaryCols,16);
MessageBinaryRows_ColsInBinary=[MessageBinaryRowsInBinary,MessageBinaryColsInBinary];
% Test if the CoverImage is Color image
[ImageROW,ImageCOL,ImageColor]=size(CoverImage);
if ImageColor>1
   % Change cover image to be gray image
   gray_cover_im=CoverImage(:,:,1);
else
   gray_cover_im=CoverImage;
end

% Convert gray_cover_im image into a binary representation.
CoverImageInBinary=dec2bin( gray_cover_im);

%% First: Hide Encrypted Text size "Row % Columns" in bit-plan-1
%  of the last 32 byte of CoverImage.
temp=0;
for i=ImageROW*ImageCOL-31:ImageROW*ImageCOL
   temp=temp+1;
   CoverImageInBinary(i,8) =MessageBinaryRows_ColsInBinary(temp);
end

%% Second: Hide all Encrypted Text in bit-plain-1 of last-32 bytes of CoverImage.
for i=1:MessageBinaryRows*MessageBinaryCols
   CoverImageInBinary(i,8) =MessageInBinary(i);
end

%% Third: Hide S_Key in bit-plain-2 of randomly select byte location of CoverImage.
% This requires first, putting S_Key in 128 bit string format
S_Key_Bin=''; DSKey=double(S_Key);
for i=1:16
   S_Key_Bin=[S_Key_Bin,dec2bin(DSKey(i),8)];
end
% then second set the seed of the random number generatore to R_Key.
rkey=0;
for i=1:16
```

```
   rkey=rkey+2^i*double(R_Key(i));
end
rng(rkey);
% Generate random byte locations
indx=randperm(size(CoverImage,1)*size(CoverImage,2)-32,128);
% Then third hide  S_Key in the randomly generated byte locations.
for i=1:128
   CoverImageInBinary(indx(i),7) =S_Key_Bin(i);
end
SKey='';
for i=1:128
   SKey=[SKey,CoverImageInBinary(indx(i),7)];
end
S_Key='';
for i=1:8:128
   S_Key=[S_Key,char(bin2dec(SKey(i:i+7)))];
end
% Convert CoverImageInBinary to unsigned 8-bit integer, after reshaping
% it as an array of (r) rows and (c) column as in the original cover
% image array.
ModifiedCoverImage=uint8(reshape(bin2dec(CoverImageInBinary),ImageROW,ImageCOL ));
if ImageColor>1
   ModifiedCoverImageH(:,:,1)=ModifiedCoverImage;
   ModifiedCoverImageH(:,:,2)=CoverImage(:,:,2);
   ModifiedCoverImageH(:,:,3)=CoverImage(:,:,3);
else
   ModifiedCoverImageH=ModifiedCoverImage;
end
% End of function 'EmbedEncryptedTextInImageWithKeys'.
```

A.6. Getplainkey_Receiver Function

```
function PlainKey=getplainkey_receiver()
% Function "PlainKey=getplainkey()" returns key-text 'PlainKey' of 16
% characters using inputdlg() matlab command.

PlainKey='';
while 1
   prompt={'Input Reciever Key (1 to 16) characters? '};
   name = 'Input Reciever Key'; defaultans = {' '};
   options.Resize ='on'; options.WindoStyle ='modal';
   options.Interpreter = 'tex';
   PlainKey=inputdlg(prompt,name,[1 40],defaultans,options );
   PlainKey=char(PlainKey);
   if isempty(PlainKey)
      return;
   end
   PlainKey=PlainKey;
```

```
   % Remove any leading spaces if in PlainKey.
   while true
   if PlainKey(1)==char(32)
      PlainKey=PlainKey(2:end);
   else
      break;
   end
end
   PlainKeyLength=length(PlainKey);
   if PlainKeyLength<1 || PlainKeyLength>16
      uiwait(msgbox([{'Error: Key length must be in'};{'between 1- to-
                  16 characters.'}],'Error','error','Modal'));
      cd(WD); clc; return;
   end
   break;
end
% Append spaces if it is not 16 char. long.
add_key=16-PlainKeyLength;
if PlainKeyLength<16
   for i=1:16-PlainKeyLength
      PlainKey=[PlainKey,' '];
   end
end
return
```

How to cite this paper: Mohammed Jawar Khami,"Transmitting Security Enforcement By Text Encrypting and Image Hiding Technique using Combined Encrypt/Hide Keys", International Journal of Engineering and Manufacturing(IJEM), Vol.8, No.1, pp.1-15, 2018.DOI: 10.5815/ijem.2018.01.01