A classical approach for hiding encryption key in the same encrypted text document

Prof. Dr. Taleb A. S. Obaid[1]
College of Computer Science and Information Technology,
Basra University,
Basra, Iraq.
tasobaid@gmail.com

Assist. Prof. Dr. Mohammed J. Khami[2]
Basra Technical Institute,
Southern Technical University,
Basra, Iraq.
mjkhami@yahoo.com,

Dr. Lemya Gh. Shehab[3]
Basra Technical Institute,
Southern Technical University
Basra, Iraq.
Lemyaaldawood@yahoo.com

Received October 22, 2017. Accepted for publication January 17, 2018
DOI: http://dx.doi.org/10.31642/JoKMC/2018/050104

Abstract *--- Data encrypting and steganography techniques each have its own merits and weaknesses. Crackers and hackers developed many software applications that can attack the weakness of these techniques and thus can reach to the meaning of the private information and its existence.*

In this paper, a simple technique is proposed to make use of encrypting and steganography merits simultaneously, and thus makes any attempt for breaking both techniques at the same time harder than applying each of them alone. The proposal uses English, Arabic, or mixed writing plain text. Encrypting the text by AES-encrypting algorithm with an encryption key of 128 bit long and then hiding the encrypted text with only one part of the encrypting key (sender key only), in another English/Arabic (cover), plain text. The text hiding technique is done according to the nonprintable characters of the Unicode character set method. The proposed technique is coded into a form of Matlab program. Experimenting with this paper technique shows how difficult to break the system and thus can applying it in emailing private documents between users connected to a virtual network on an insecure public channel.

Keywords --- encrypting and steganography techniques, hybrid systems, Unicode character set.

I. INTRODUCTION

Nowadays life depends completely on internet and computing, or on what is called the information technology (IT). However, the existence of defenselessness in IT has created some security issues that greatly affect the privacy of individuals [1,2]. Many techniques are used to protect privacy issues through the enforcement of security rules and requirements. The most common techniques to do this is either transforming private data into a different form or concealing its existence while it's on the way by a suitable hiding method [3,4].

Cryptography, watermarking, and steganography are examples of the most basic data encrypting and hiding techniques that are being used to address copyright management, protect information, and conceal secrets [5,6].

The central to cryptographic operations are cryptographic keys. A key is a piece of variable data that is fed as input into a cryptographic algorithm to perform encryption/decryption operations. In a well-

designed cryptographic scheme, the security of the system depends only on the security of the keys used.

The big matter facing the communication security is how the messages cannot be read by anyone except the intended recipient using a set of algorithms combined with keys to convert the original message (plain-text) to encrypted message (cipher-text) and convert it back in the intended recipient side to the original message (plain-text). The field that related to this problem is called cryptography (the science of writing secret codes), so, now a day's become more important to discourse this issue [7]. The real challenge facing the security of communications is to hide the key to prevent intruders from discovering it on the internet.

Steganography is the art and science of data hiding that makes data invisible by hiding (or embedding) them in another piece of data, known alternatively as the cover, the host, or the carrier. There is a need to hide secret information inside certain types of digital data. Storing, hiding, or embedding secret information in all types of digital data is one of the tasks of the field of steganography. Steganography can be used to prove copyright ownership, to identify attempts to tamper with sensitive data and to embed annotations, for more details see [8,9].

Practical cases prove that when encrypting used with steganography techniques before transmitting private and secret data on a public or insecure channels, the results show an enhancement in the security issues and make it difficult for the intruders to obtain the real meaning of such transmitted data. For this reasons, we are going to apply suitable types of both techniques in emailing text documents on a virtual network of users.

The proposal work uses new structure style to hide the encrypted text and its encrypting key within another plain text (cover text). But since both encrypting and steganography techniques require their own keys, thus these secret keys must be embedded in the same cover text before transmitting. How and where these keys are hidden this will be the main goal of our proposal.

II. PROPOSED SYSTEM LAYOUT

Figure (1) depicts a virtual network of users [A, B, C, D, E, F, G, and H], each user has his own Private key, [Key(A), Key(B), Key(C), Key(D), Key(E), Key(F), Key(G)]. The transmit/receive operations among users of this virtual network can be from **one-to-all** (Public-mode), and **one-to-one** (Private-mode). In public-mode all user must agree on using the same key in their transmit/receive operations while in private-mode any user must assign one private key to each user in the network to control (allow, deny), the delivering of his encrypted message (different keys to different users). However, user key can be changed whenever his owner finds it necessary to do so.

Even though, users use some sort of available secure mailing applications, that may exist on the internet like electronic mail (e-mail), they can use this paper method to increase security of their important data when transmit it over public channels and not depends on what such public emailing applications offer from the security point view.

The network needs not be made of physically connected nodes (users), instead, it may be made of a group of users apart from each other, and all may use public and insecure channels in mailing their secret and private text documents. For example, a network can be imagined in the case of big manufacturing company and some expert peoples who are living apart across the country and the company needs their suggestions and evaluations about a newly designed product in form of typed text reports.

The heart of the network management in this proposal depends on running the same copy of text document encrypt/decrypt program. The program works according to the advanced encryption standard (AES), algorithm with the use of 128-bits (16-characters) long encrypting key. In this paper, the underlying encrypt/decrypt key is made by combining two keys. The first key (128-bits long), is supplied by the sender (S-Key), while the second key (also 128-bit long), is obtained from the receiver (R-Key). The combined key (SR-Key), will be used in plaintext document encrypting process and when document encrypting is completed, only sender key must be hidden (according to a selected steganography technique), in the encrypted document before sending it to the other user (receiving user).

At the receiving end, the hidden sender key is extracted first and then combined with the receiver key (known to receiver since it is his key). The new combined key will be used to decrypt the encrypted text document (of course, after filtering the received encrypted text from the hidden key bytes). The workflow of this system can be described at two different locations, sending and receiving locations as in following:

Sending location:

1. **Sender private encrypting key**, S-Key, is Known to him. S-Key is 128 bits (or 16-characters), long.
2. **Receiver private encrypting key**, R-Key, could be obtained by asking the receiver for it. R-Key must be also 128 bits long.
3. **Combine** S-Key with R-Key to get new encrypting key SR-Key of 128 bits long too.
4. The sender selects the **plain text document** that he wants to encrypt.
5. **Encrypt** the selected plain text document by applying AES algorithm and SR-Key.
6. **Hide** only sender key, S-Key, in the encrypted text document by applying suitable steganography, **hiding-technique**.
7. **Send** the encrypted text document with hidden S-Key in it to the receiver side.

Receiving location:

1. **Read** the received encrypted document.
2. **Receiver private encrypting key**, R-Key, is Known to him. R-Key is 128 bits (or 16-characters), long.
3. **Sender encrypting key**, S-Key, has to be **extracted** from the received encrypted document by applying suitable steganography, **extracting-technique**.
4. **Filtering** the received encrypted text document from the hidden S-Key data. The outcome of this step is exactly like the encrypted text document before the sender hides his S-Key in it.
5. **Combine** S-Key with R-Key to get the original encrypting key, SR-Key (128 bit long).
6. **Decrypt** the encrypted text document by AES algorithm and SR-Key to obtain the original sent plain text document.

Key combining operation:

Combining of S-Key with R-Key is done by simple processing steps. These are:

Step-1: **Read** both S-Key and R-Key strings.

Step-2: **Make sure** that S-Key and R-Key have exactly 128 bits (or 16-characters) long. If any of them is less than 16 characters then append spaces to it, until it is 16-characters long. But if it is longer than 16 characters then issue an error message to notify that.

Step-3: **Convert** both keys into the binary system.

Step-4: **Reverse** R-Key binary sequence in such way that bit 128 becomes 1 and bit 127 becomes 2 and so on for all other bits. As shown in Figure(2).

Step-5: Execute simple **XOR** between S-Key and the reversed sequence of R-Key^{Rev} to obtain the combined encrypting key, SR-key.

Note: In step-5, reversing only the R-Key binary bit sequence is to reduce the probability of getting a combined key, SR-Key, with all its bit, are set to zero, since this is expected when executing XOR() function between two identical values.

Hiding/extracting techniques:

Steganography techniques followed in this paper work according to the use of the non-printable characters of the Unicode character set from [10,11]. In these techniques, the ASCI-value of any character of the combined key, SR-Key, first converted into its binary equivalent value, then replacing each '0' and '1' of it with the nonprintable characters of the code point of $200C_{hex}$ (or 8204_{dec}) and $200D_{hex}$ (or 8205_{dec}) respectively. For example the character 'A' with ASCI-value of $(65)_{Dec}$ or $(01000001)_{Bin}$ become (200C 200D 200C 200C 200C 200C 200C 200D). This new character representation can be hidden (appended to or inserted in), in any character string and no one can notice its existence neither can appear when printing the whole string.

Encrypted text filtering operation:

Since the received encrypted document has the sender key S-Key characters hidden in it thus, filtering operation is needed to get only the original encrypted message from the received encrypted text. From the computer programing side view, the applied filtering technique of this paper is done by setting any character of the received encrypted text with ASCI-value greater than 255 to 0 (note the character with ASCI of 0 is the null ' nothing ' character), and this means removing any hidden data from the received text document.

III. SYSTEM IMPLEMENTATION

System steps are written and encoded into a form of (m-file) Matlab encrypt/decrypt program (Appendix A). The program can deal with only English, only Arabic, or English mixed with Arabic plain text documents. Both sender and receiver must have a copy of this program to encrypt and decrypt messages in between them. An example of documents, before and after encryption, are shown in Figure(3). Some of the important processing steps and flow diagram, for an English only, Arabic only and English/Arabic mixed text documents are depicted in Figure(4) for both sending and receiving locations.

Conclusions

The central to cryptographic operations are cryptographic keys. In a well-designed cryptographic scheme, the security of the scheme depends only on the security of the used-keys. Our encrypting/hiding method concentrates on how encrypting and hiding keys transmitted on opened space without letting intruders know about them. The following points of conclusions show some important issues on our method:

1) The proposed encrypt/decrypt processing steps of this work can work efficiently with any type of encrypting/decrypting algorithms as long as these algorithms apply an encrypting key.
2) When this paper method is used in an opened and insecure transmitting media, such as a public electronic mail, it will add more security consideration points to what that public media already have.
3) Hiding one part of the encrypting key (sender encrypting key), into the encrypted document by the nonprintable characters of the Unicode character set has some advantages, and also some disadvantages too, over the use of other available hiding methods. Here is some of them:-
   a) Hiding by the Unicode method allows us to recover %100 same original encrypted text very easy and without any change in its content or its configuration. This type of recovery is very important at the decryption end. Since changing any character type or its position within the encrypted text will make it difficult to decrypt it right again.
   b) Filtering of the original encrypted text from what is hidden in it (sender key data), can be accomplished just by scanning the received encrypted text one character at a time and whenever a Unicode character is met, it must be replaced by a null character (a character with ASCI-value of zero).
   c) The disadvantage of using the Unicode character set method in hiding the sender key is denoted by the big change in encrypted text file size before and after hiding the key. The size of the encrypted text with the hidden key could be twice (or more), as the size of the same text before hiding sender key in it. The reasons for this is that any character of the encrypted text with the hidden key should be represented by two bytes instead of one, and also due to the application of the external encoding system (1 to 7 Unicode characters sequence for replacing each sender key character).
4) Added security consideration on transmitting text documents by this paper method, come from the following points:
   I. Additive securities come from the application of the encryption technique at the first place. Different encrypting techniques may have different difficulties and may require more complicated decrypted methods.
   II. In the second place, additive securities result from the applied steganography technique. These additive securities could come out as results of:
      A) Hiding secret text (sender key in our case), in a cover text (the encrypted text document), could be done in many ways. Hiding the encrypting key could be in a pre-specified and known location (static), or in randomly selected position (dynamic), from those of the cover text. Selection any way of these methods may add more difficulties in recovering hidden data back.
      B) Sender key could be hidden in random locations on the cover text according to static or dynamic key location production method (*Note: hiding key and encrypting key are not the same, as shown by their names*). Hence, should we use a static or dynamic production method in generating such key. And if dynamic key production method is chosen, then what criteria must one use as a seed for the random number generator to determine the locations of the key characters among those of the cover text.
      C) The hidden message in Unicode character set hiding method is obtained by encoding the secret text according to suitable encoding system before hiding it in the cover message. The selection of the encoding system could depend on special criteria such the use of language character sequence, (In this paper the letter-frequency sequence of the English secret message text), and the applied coding technique (Coding technique in this work is the Hoffman coding). Part of the obtained codebook is depicted in Table(1). Characters ordering and encoding techniques add more difficulties to recover the hidden text and thus better security.

5) Using encrypting and steganography techniques at the same time on one text data object produce an encrypting-steganography hybrid system. It enforces features and cancels drawbacks of both techniques.

REFERENCES

[1] Philip Brey, Adam Briggle,and Edward Spence, ”The Good Life in Technological Age”, New York, Routledge, Taylor & Francis, 2012

[2] G.Kalpana, P. V. Kumar and R.V. Krishnaiah, “A Brief Survey on Security Issues in Cloud and its Service Models", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 6, June 2015.

[3] Jayaram P, Ranganatha H R, Anupama H S, “Information hiding using Audio steganography”, The International Journal of Multimedia & Its Applications, Vol.3, No.3, August 2011

[4] Arvind Kumar, and Km. Pooja, “Steganography- A Data Hiding Technique”, International Journal of Computer Applications, Vol. 9, Issue 7, Nov. 2010.

[5] Palak Mahajan, ”Steganography: A Data Hiding Technique“, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 11, November 2014.

[6] C. C. Chang, P. Tsai, and M.H. Lin, "An Adaptive Steganography for Index-based images using Codeword Grouping", Advances in Multimedia Information Processing –PCM, Springer, Vol. 3333, pp 731-738, 2004.

[7] Melvin Hausner, “Notes on Number Theory and Cryptography (V55.0106) Quantitative Reasoning: 4 Cryptography “, New York University, 2002.

[8] Debnath Bhattacharyya, Asmita Haveliya, and Tai-hoon Kim, " Secure Data Hiding in Binary Text Document for Authentication", Applied. Mathematic. Information and Science, Vol. 8, No. 1, 371-378 (2014).

[9] L. Y. Por, B. Delina, “Information Hiding: A New Approach In Text Steganography”, 7th Wseas Int. Conf. On Applied Computer & Applied Computational Science (Acacos ’08), Hangzhou, China, (2008)

[10] M. J. Khami, “Unlimited size of English plaintext-in-text hiding algorithm”, International Journal Of Computer Science and Engineering, Vol. 6, Issue 1, Dec. – Jan. 2017; 89-96.

[11] M. j. Khami, “New rules-based approach for text data hiding", International Journal Of Computer Science and Engineering, Vol. 6, Issue 3, Apr May 2017; 1-20.

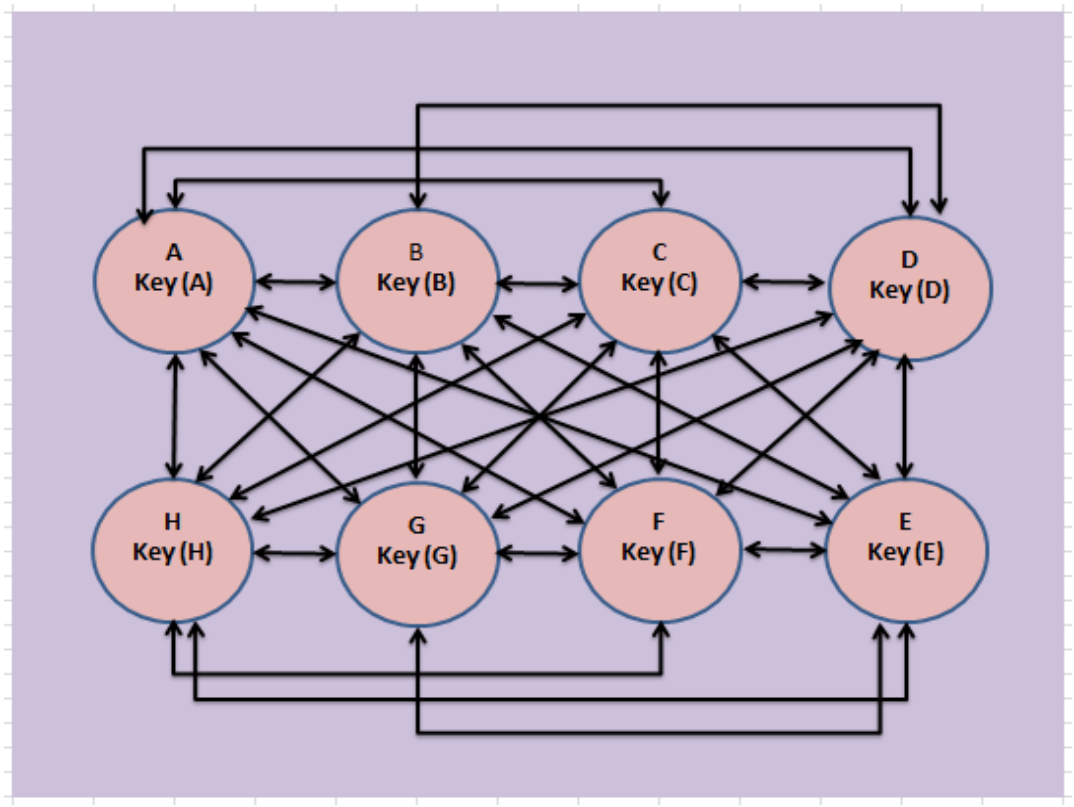


Figure (1) Virtual network of users.

S-Key = '0100101101101000011000010110110101 . . . . 0010000000100000'

1 128

R-Key = '0110110101101111011010000110000101 . . . . 0010000000100000'

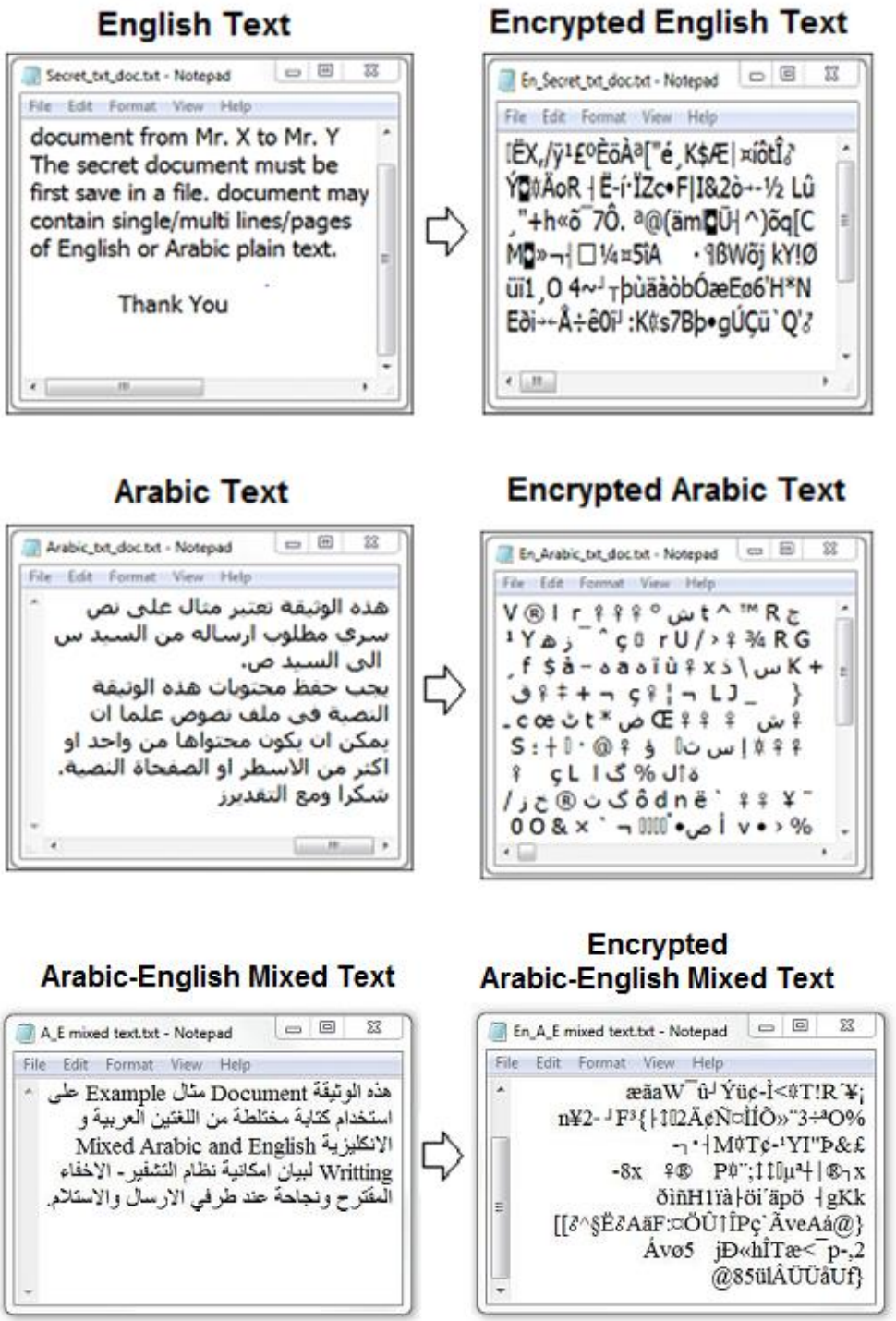
128 1

R-Key^{Rev} = '0000010000000100 . . . . 0100001100001011011110110101101101'

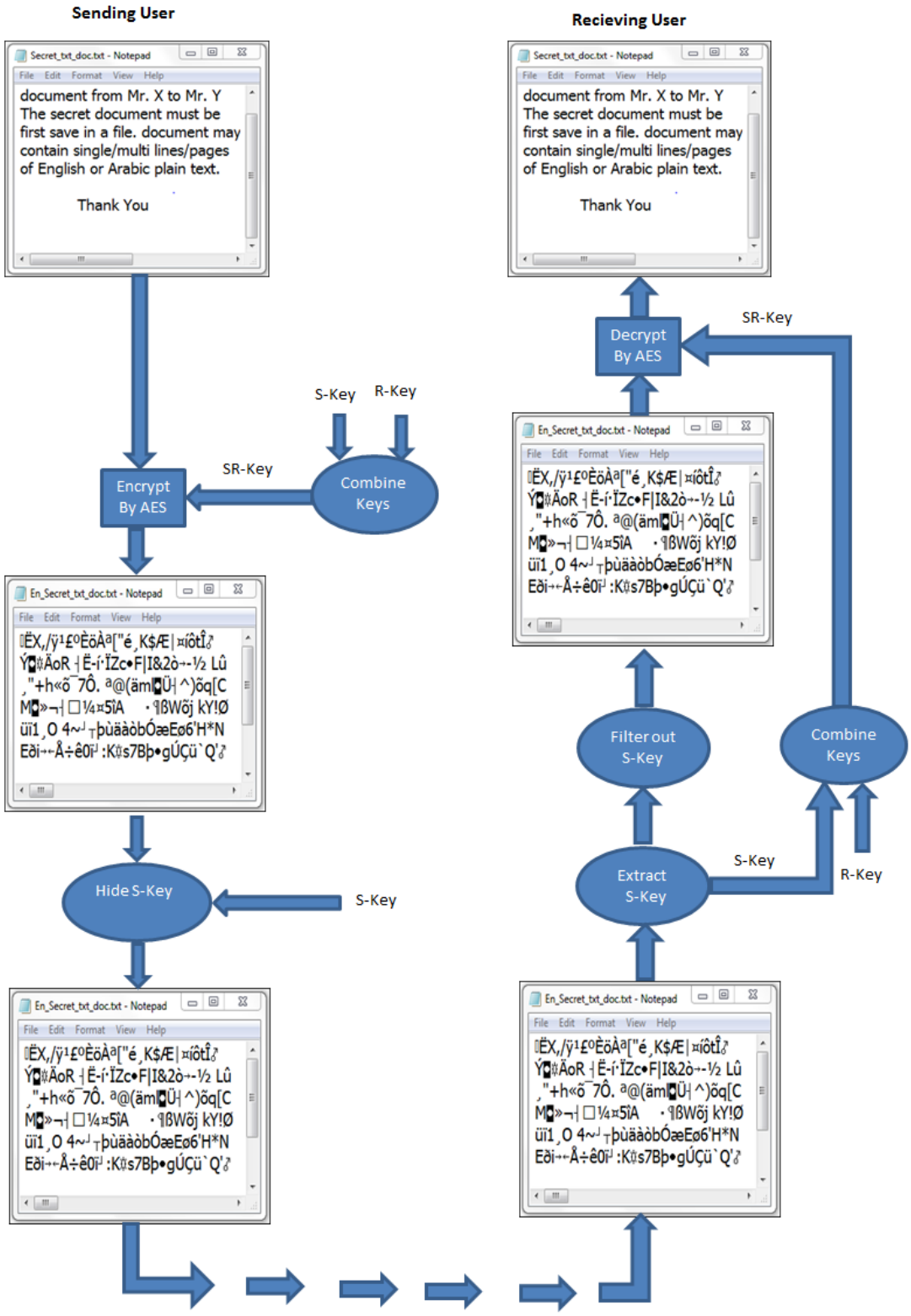
SR-Key = XOR(S-Key , R-Key^{Rev})

S-Key = '0100101101101000011000010110110101 . . . . 0010000000100000'
R-Key^{Rev} = '0000010000000100 . . . . 0100001100001011011110110101101101'
==========
SR-Key = '0100111101101100 . . . . . 1101011010010110'

Figure (2) Combining S-Key and R-Key into SR-Key.



Figure(3) An example of documents, before and after encryption.



Figure(4) System flow diagram at both sending and receiving locations.

Table(1) Part of system codebook.

Char	Binary Code	Unicode Equivalent Code
''	0	200C
e	1	200D
t	00	200C 200C
a	10	200D 200C
o	01	200C 200D
i	11	200D 200D
n	000	200C 200C 200C
s	001	200C 200C 200D
h	010	200C 200D 200C
r	011	200C 200D 200D
d	100	200D 200C 200C
l	101	200D 200C 200D
c	110	200D 200D 200C
u	111	200D 200D 200D
m	0000	200C 200C 200C 200C
w	0001	200C 200C 200C 200D
f	0010	200C 200C 200D 200C
g	0011	200C 200C 200D 200D
y	0100	200C 200D 200C 200C
..	....	....
..	....	....
}	011111	200C 200D 200D 200D 200D 200D
~	100000	200D 200C 200C 200C 200C 200C
..	....	....
..	....	....

Appendix A : System Matlab files

Program 1: main_aes_encrypt5.m

```
clc;
clear;
% This program is to encrypt the content of the plain text file
% "filename.txt" and save the encrypted text in "En_filename.txt"
% Note:
%        Sender and Receiver keys have maximum lengths of 128 bits(16_chars).
%
% Dr. Mohammed J. Khami
% March - 2017
% mjkhami@yahoo.com
fclose all;
WD=cd();
cr_lf_char=[char(13),char(10)];
%% Program section (1):
% A) Get Encryption/Decryption Key.
[SPlainKey, RPlainKey]=getplainkey5();
PlainKeyLength1=length(SPlainKey);
PlainKeyLength2=length(RPlainKey);
if PlainKeyLength1<1 || PlainKeyLength1>16 || PlainKeyLength2<1 || PlainKeyLength2>16
   ttext='Error: Sender & Receiver Keys must be 1-to-16 characters Long.';
uiwait(msgbox({ttext},'Error','error','modal'));
   cd(WD);
   clc;
   return;
end
CompositeKey=two_keys_in_one5(SPlainKey, RPlainKey);
PlainKey=CompositeKey; % Encryption key is ready.
% B)Initialize AES algorithm with PlainKey>
[s_box, inv_s_box, w, poly_mat, inv_poly_mat] = aes_init(PlainKey);
% C) Get 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');
%
if filen1(1:3)=='En_'
   ttext='Error: This file is allready encrypted and can not be encrypted agian.';

uiwait(msgbox({ttext},'Error','error','modal'));
   cd(WD);
   return % User cancelled.
end
if strcmp(filen1(1:3),'De_')
   filen2=filen1(4:end);
end
filen2=['En_',filen1];
EncryptedTextFileName=[path1 filen2];
FE2=fopen(EncryptedTextFileName,'w');
%
plaintext=fread(FE1);
% Divid Current PlainText into slices of 16 characters.
PlainText=plaintext';
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);
   ciphertext = cipher (PlainText1, w, s_box, poly_mat);
   if cy==1
      EncText =ciphertext;
   else
      EncText =[EncText,ciphertext];
   end
end % End of ciphering one compelet line of plaint text.
EncText=hide_key5(SPlainKey,EncText);
fwrite(FE2,EncText,'ubit16');
fprintf('\n\n\tEncrption of(%s) is saved in (%s)\n\n',filen1,filen2);
fclose all;
close all;
cd(WD);
% End of Encryption Program.
```

Program 2: main_aes_decrypt5.m

```
clc;
clear;
% This program is to decrypt the content of the encrypted text file
% "En_filename.txt" and save the recoverd text in "De_filename.txt"
% Note:
% En_filename.txt must be encrypted by "main_aes_encrypt5.m".
% The receiving key have maximum lengths of 128 bits (16_chars).
%
% Dr. Mohammed J. Khami
% March - 2017
% mjkhami@yahoo.com
%% Variables declaration.
WD=cd(); % Current Directory
cr_lf_char=[char(13),char(10)]; % Carriage-Return Symbole
[Letter, Letter_BineCode,Letter_UniCode]=array_definition5;
% Read file of encrypted text.
[filen1 pth1] =uigetfile({'En_*.txt';},'Choose Encrypted Text File: ');
if isequal(filen1,0) || isequal(pth1,0)
   ttext='Error: Filename must not be empty';
uiwait(msgbox({ttext},'Error','error','modal'));
   cd(WD);
   return % User cancelled.
end
EncryptTextFileName= [pth1 filen1];
FE1=fopen(EncryptTextFileName,'r');
if filen1(1:3)=='De_'
   ttext='Error: File contains plain text & can not be decrypted unless you change its name.';
uiwait(msgbox({ttext},'Error','error','modal'));
   cd(WD);
   return % User cancelled.
else
   filen2=['De_',filen1(4:end)];
end
DecryptTextFileName= [pth1,filen2];
FE2=fopen(DecryptTextFileName,'w');
EncryptedText=fread(FE1,'ubit16');
EncryptedText=EncryptedText';
[Plainkey,EncryptedText_Pure]=Extract_Hidden_Key5(EncryptedText);
SPlainKey=Plainkey;
RPlainKey=getplainkey_receiver();
CompositeKey=two_keys_in_one5(SPlainKey,RPlainKey);
PlainKey=double(CompositeKey);
[s_box, inv_s_box, w, poly_mat, inv_poly_mat] = aes_init(PlainKey);
PlainText=EncryptedText_Pure;
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
fwrite(FE2,DecText);
fprintf('\n\tRecovered plain text is saved in file(%s)\n',filen2);
FE3=fopen(filen2,'r');
fclose all;
close all;
cd(WD);
% End of Decryption Program.
```

Function 1: getplainkey5.m

```
function [SPlainKey, RPlainKey]=getplainkey5()
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});
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;
```

Function 2: two_keys_in_one5.m

```
function CompositeKey = two_keys_in_one5(SenderKey,ReceiverKey)
%% Make sure both given keys are 16 char long.
s=length(SenderKey);
r=length(ReceiverKey);
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(ReceiverKey(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
```

Function 3: hide_key5.m

```
function KeyStegoLine=hide_key5(ts,tc)
%% Variables declaration for steganography section.
[Letter, Letter_BineCode,Letter_UniCode]=array_definition5;
%% Start hiding algorithm
sizets=size(ts,2); % Determine size of current secret text line .
sizetc=size(tc,2); % Determine size of current cover text line .
rng(sizetc); % Set the random number generator to (sizetc)
IdxCurrentCoverLine=randperm(sizetc);
field=1;          % Special flag used in [tablesearch() and cellsearch()]
                  % functions.
StegoText= []; % Variable to hold the stegano text
%
%% Embedding one character from secret text line before it's
% corresponding character of the corresponding cover text line.
%
for ct=1: sizetc
   if IdxCurrentCoverLine(ct)<=sizets
      %
      % tablesearch() function is used to code each character of the
      % secret text line according to special coding scheme which
      % applies two characters (200C and 200D) of the Unicode
      % character set.
      %
      GivenCellString=ts(1, IdxCurrentCoverLine(ct));
      [CellName,CellBinCode, CellUniCode]=tablesearch5(GivenCellString, field);
      StegoText= [StegoText,' ', char(CellUniCode)];
   end
   %
   GivenString=tc(1, ct);
   StegoText= [StegoText,' ', dec2hex(double(GivenString),4)];
end
%
SizeStego=size(StegoText,2); % Get size of Stego text line.
%
```

```
        % Each character of the resultant stego
text must be coded in form of
        % two byte per character, i.e., as same as
the coding applied in
        % Unicode character set.
        %
        KeyStegoLine= [];
        for c=1:5: size(StegoText,2)
           T=StegoText(c+1:c+4);
           if ismember(' ',T)
              T=StegoText(c+2:c+5);
           end
           KeyStegoLine=          [KeyStegoLine,
char(hex2dec(T))];
        end
```

Function 4: array_definition5.m

```
        function
[Letter,Letter_BinCode,Letter_UniCode]=
array_definition5
        global Letter;
        global Letter_BinCode;
        global Letter_UniCode;
        % Creation of Letter,Letter_BinCode, and
Letter_Unicode
        Letter=[];
        Letter=[{'
'};{'e'};{'t'};{'a'};{'o'};{'i'};{'n'};{'s'};{'h'};{'r'};{'d'
};
        {'l'};{'c'};{'u'};{'m'};{'w'};{'f'};{'g'};{'y'};
{'p'};{'b'};{'v'};
        {'k'};{'j'};{'x'};{'q'};{'z'};{'A'};{'B'};{'C'};
{'D'};{'E'};{'F'};
        {'G'};{'H'};{'I'};{'J'};{'K'};{'L'};{'M'};{'N'
};{'O'};{'P'};{'Q'};
        {'R'};{'S'};{'T'};{'U'};{'V'};{'W'};{'X'};{'
Y'};{'Z'};{'!'};{""};
        {'#'};{'$'};{'%'};{'&'};{""};{'('};{')'};{'*'};
{'+'};{','};
        {'-
'};{'.'};{'/'};{'0'};{'1'};{'2'};{'3'};{'4'};{'5'};{'6'};
{'7'};
        {'8'};{'9'};{':'};{';'};{'<'};{'='};{'>'};{'?'};{'
@'};{'['};{'\'};
        {']'};{'^'};{'_'};{'`'};{'{'};{'|'};{'}'};{'~'};{c
har(13)};
        {char(10)};{char(9)};{char(11)};{char(12)
};{char(14)};{char(15)};
        {char(16)};{char(17)};{char(18)};{char(1
9)};{char(20)};{char(21)};
        {char(22)};{char(23)};{char(24)};{char(2
5)};{char(26)};{char(27)};
        {char(28)};{char(29)};{char(30)};{char(3
1)};{char(8)}; {char(7)};
        {char(6)};{char(5)};{char(4)};{char(3)};{
char(2)};{char(1)}];
        Letter_BinCode=[
        {'0'};{'1'};{'00'};{'10'};{'01'};{'11'};
        {'000'};{'001'};{'010'};{'011'};{'100'};{'10
1'};{'110'};{'111'};
{'0000'};{'0001'};{'0010'};{'0011'};{'0100'};{'010
1'};{'0110'};{'0111'};
{'1000'};{'1001'};{'1010'};{'1011'};{'1100'};{'110
1'};{'1110'};{'1111'};
{'00000'};{'00001'};{'00010'};{'00011'};{'00100'};
{'00101'};{'00110'};
{'00111'};{'01000'};{'01001'};{'01010'};{'01011'};
{'01100'};{'01101'};
{'01110'};{'01111'};{'10000'};{'10001'};{'10010'};
{'10011'};{'10100'};
{'10101'};{'10110'};{'10111'};{'11000'};
{'11001'};{'11010'};{'11011'};{'11100'};
{'11101'};{'11110'};{'11111'};
{'000000'};{'000001'};{'000010'};{'000011'};{'00
0100'};{'000101'};
{'000110'};{'000111'};{'001000'};{'001001'};{'00
1010'};{'001011'};
{'001100'};{'001101'};{'001110'};{'001111'};{'01
0000'};{'010001'};
{'010010'};{'010011'};{'010100'};{'010101'};{'01
0110'};{'010111'};
{'011000'};{'011001'};{'011010'};{'011011'};{'01
1100'};{'011101'};
{'011110'};{'011111'};{'100000'};{'100001'};{'10
0010'};{'100011'};
{'100100'};{'100101'};{'100110'};{'100111'};{'10
1000'};{'101001'};
{'101010'};{'101011'};{'101100'};{'101101'};{'10
1110'};{'101111'};
{'110000'};{'110001'};{'110010'};{'110011'};{'11
0100'};{'110101'};
{'110110'};{'110111'};{'111000'};{'111001'};{'11
1010'};{'111011'};
{'111100'};{'111101'};{'111110'};{'111111'}];
        Letter_UniCode=[];
        for i=1:size(Letter_BinCode,1)
           te=char(Letter_BinCode(i,1));
           dd=[];
           for j=1:size(te,2)
              if te(1,j)=='0'
                 if j>1
                    dd=[dd,' ','200C'];
                 else
                    dd=['200C'];
                 end
              else
                 if j>1
                    dd=[dd,' ','200D'];
                 else
                    dd=['200D'];
                  end
```

```
            end
        end
    Letter_UniCode=[Letter_UniCode;{dd}];
    End
```

Function 5: getplainkey_receiver.m

```
function PlainKey=getplainkey_receiver()
PlainKey='';
while 1
    prompt={'Input Receiver Key (1 to 16) characters? '};
    name = 'Input Receiver 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;
    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
add_key=16-PlainKeyLength;
if PlainKeyLength<16
    for i=1:16-PlainKeyLength
        PlainKey=[PlainKey,' '];
    end
end
return
```

Function 6: tablesearch5.m

```
function [CellName,CellBinCode, CellUniCode]=tablesearch5(GivenString, field)
global Letter;
global Letter_BinCode;
global Letter_UniCode;
CellName={};
CellBinCode={};
CellUniCode={};
if (field<1)||(field>3)|| isempty(GivenString)
    return
end
for i=1:size(Letter,1)
    if field==1
        id=strfind(Letter,GivenString);
    end
    if field==2
 id=strfind(Letter_BinCode,GivenString);
    end
    if field==3
 id=strfind(Letter_UniCode,GivenString);
    end
    if id{i,1}==1
        break
    end
end
if id{i,1}==1
    CellName=Letter(i);
    CellBinCode=Letter_BinCode(i);
    CellUniCode=Letter_UniCode(i);
else
    CellName={};
    CellBinCode={};
    CellUniCode={};
End
```

Note: Unlisted functions like aes_initi(), cipher() and inv_cipher() have standard forms and can be easily downloaded from the internet.