Scilab Textbook Companion for Cryptography and Network Security by A. Kahate[1]

Created by
Akash Goel
B.Tech.
Computer Engineering
Delhi Technological University
College Teacher
None
Cross-Checked by
Spandana

May 27, 2016

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Cryptography and Network Security

Author: A. Kahate

Publisher: Tata McGraw-Hill, New Delhi

Edition: 3

Year: 2003

ISBN: 9789332900929

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes 4

2 Cryptography Techniques 5

3 Computer Based Symmetric Key Cryptographic Algorithm 35

4 Computer Based Asymmetric Key Cryptographic Algorithm 40

6 Internet Security Protocols 51

List of Scilab Codes

Exa 2.5 Fig 2pt5 A scheme for codifying messages 5
Exa 2.6 Fig 2pt6 Codification using the alphabet replacement scheme 6
Exa 2.8 Fig 2pt8 Example of a plaint text message being transformed into cipher text 7
Exa 2.11 Fig 2pt11 Example of breaking caesar cipher 9
Exa 2.13 Fig 2pt13 Attempts to break modified Caesar cipher text using multiple possiblities 9
Exa 2.14 Fig 2pt14 Polygram substitution 10
Exa 2.15 Fig 2pt15 Vignere Tableau 11
Exa 2.18 Fig 2pt18 Keyword matrix 11
Exa 2.25 Fig 2pt25 Encryption process in Playfair cipher 12
Exa 2.26 Fig 2pt26 Keyword matrix 14
Exa 2.33 Fig 2pt33 Practice example for playfair cipher 15
Exa 2.34 Fig 2pt34 Encryption and decryption of hill cipher . . 16
Exa 2.36 Fig 2pt36 Rail fence technique 20
Exa 2.38 Fig 2pt38 Example of simple columnar ransposition technique 21
Exa 2.40 Fig 2pt40 Example of simple columnar ransposition technique with multiple rounds 23
Exa 2.42 Fig 2pt42 Vernam cipher 25
Exa 2.43 Fig 2pt43 Encryption 27
Exa 2.44 Fig 2pt44 Decryption 28
Exa 2.51 Fig 2pt51 Number of parties and the corresponding number of lock and key pairs 29
Exa 2.54 Fig 2pt54 Diffie Hellman key exchange 30
Exa 2.56 Fog 2pt56 Man in the middle attack in Diffie Hellman key exchange 31

Exa 2.65 Fig 2pt65 Understanding key range 33
Exa 3.2 Fig 3pt2 Functioning of XOR logic 35
Exa 3.2.1.1 Example XOR operations 35
Exa 3.3 Fig 3pt3 Stream cipher 36
Exa 3.34 Fig 3pt34 Example of selection of S box output based on the input 37
Exa 3.81 Fig 3pt81 Key expansion example 38
Exa 4.4 Fig 4pt4 RSA algorithm example 40
Exa 4.5 Fig 4pt5 RSA Encryption scheme 41
Exa 4.5.1 ElGamal Key Generation 43
Exa 4.5.2 ElGamal Key Encryption 43
Exa 4.5.3 ElGamal Key Decryption 44
Exa 4.6 Fig 4pt6 RSA Encryption scheme 45
Exa 4.9.1 ElGamal Signature 46
Exa 4.9.2 ElGamal Signature verification 47
Exa 4.17 Fig 4pt17 Longitudinal redundancy check 48
Exa 4.18 Fig 4pt18 Simple message digest 49
Exa 4.62 Fig 4pt62 Knapsack algorithm for Public Key Encryption 49
Exa 6.54 Fig 6pt54 Base 64 encoding example 51
Exa 6.55 Fig 6pt55 Base 64 encoding mapping table 53
AP 1 Functions for Chapter 6 example codes 54
AP 2 Chapter 4 functions 55
AP 3 Functions for Chapter 2 example codes 55

Chapter 2

Cryptography Techniques

check Appendix AP 3 for dependency:

Chapter_2.sci

Scilab code Exa 2.5 Fig 2pt5 A scheme for codifying messages

```
1  // Substitutition scheme of Caesar cipher
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 a = ascii('A')
15 for i =0:25
16     printf("%c : %c\n",ascii(a+i),encrypt_caesar(
       ascii(a+i)))
```

```
17 end
18 // A scheme for codifying messages
19 //(replacing each alphabet with an alphabet three
     places down the line)
```

---

check Appendix AP 3 for dependency:

```
Chapter_2.sci
```

Scilab code Exa 2.6 Fig 2pt6 Codification using the alphabet replacement scheme

```
1  // Substitutition scheme of Caesar cipher
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 a = ascii('A')
15 pt = "I LOVE YOU"
16 printf("Plaintext:\n\t%s\n",pt)
17
18 //Encryption using encrypt_caesar function from
     dependency file
19 printf("Encrypted text:\n\t%s",encrypt_caesar(pt))
20
21 // A scheme for codifying messages
22 //(replacing each alphabet with an alphabet three
     places down the line)
```

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.8 Fig 2pt8 Example of a plaint text message being transformed into cipher text

```
1
2 // Move scilab to current file directory
3 [u,t,n] = file()
4 n = strcat(n)
5 file_name = basename(n)+fileext(n)
6 file_name = strcat(file_name)
7 ind=strindex(n,file_name)
8 path = part(n,1:ind-1)
9 chdir(path)
10
11 exec("Chapter_2.sci",-1)
12
13 pt = ["Hi Amit,",
14
15 "Hope you are doing fine. How about meeting at the
     train station this friday at 5 p.m. ? Please let
     me know if it is OK with you.",
16
17 "Regards.",
18
19 "Atul"]
20
21 disp("Plain-text message:")
22 disp("")
23 for i=1:length(length(pt))
24     printf("%s\n",strcat(pt(i)))
25 end
26
```

```
27 ct_full = list()
28 a = ascii('a')
29 z = ascii('z')
30 A = ascii('A')
31 Z = ascii('Z')
32
33
34 //Encryption using encrypt_caesar funtion from
      depenency file
35 for k = 1:length(length(pt))
36     ct = []
37     for i=1:length(pt(k))
38         x = ascii(part(pt(k,1),i:i))
39         if x>=A & x<=Z then
40             ct(k,i) = encrypt_caesar(part(pt(k),i:i)
                  )
41         elseif x>=a & x<=z then
42             c = convstr(part(pt(k),i:i),'u')
43             c = encrypt_caesar(c)
44             c = convstr(c,'l')
45             ct(k,i) = c
46         else
47             ct(k,i) = part(pt(k),i:i)
48         end
49     end
50     ct_full(k) = ct
51 end
52
53 disp("")
54 disp("Corresponding cipher-text message:")
55 disp("")
56 for i=1:length(ct_full)
57     printf("%s\n",strcat(ct_full(i)))
58 end
```

---

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.11 Fig 2pt11 Example of breaking caesar cipher

```
1 // Example of breaking Caesar cipher
2
3 // Move scilab to current file directory
4 [u,t,n] = file()
5 n = strcat(n)
6 file_name = basename(n)+fileext(n)
7 file_name = strcat(file_name)
8 ind=strindex(n,file_name)
9 path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 key = 3
15 a = ascii('A')
16 ct = "L ORYH BRX"
17 printf("Encrypted text:\n\t%s\n",ct)
18
19 //Decryption using function from dependency file
20 printf("Plaintext:\n\t%s",decrypt_caesar(ct))
```

check Appendix AP 3 for dependency:

```
Chapter_2.sci
```

Scilab code Exa 2.13 Fig 2pt13 Attempts to break modified Caesar cipher text using multiple possiblities

```
1 // Attempts to break moidified Caesar cipher text
    using multiple possiblities
```

```
2
3 // Move scilab to current file directory
4 [u,t,n] = file()
5 n = strcat(n)
6 file_name = basename(n)+fileext(n)
7 file_name = strcat(file_name)
8 ind=strindex(n,file_name)
9 path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 a = ascii('A')
15 ct = "KWUM PMZM"
16 printf("Encrypted text:\n\t%s\n",ct)
17 printf("Possible Plaintext:\n\t\n")
18
19
20 //Decryption using library function
21 printf("Attempt Number\n(Value of k)\n");
22 for key = 1:25
23     printf("\t%d. \t",key)
24     printf("%s\n",decrypt_caesar_general(ct,26-key))
       ;
25 end
```

Scilab code Exa 2.14 Fig 2pt14 Polygram substitution

```
1
2 //Polygram substitution
3
4 pt = ["HELLO" "HELL"]
5 ct = ["YUQQW" "TEUI"]
6
7 for i=1:length(length(pt))
```

```
8       printf("Plaintext: %s\n",pt(1,i))
9       printf("Ciphertext: %s\n\n",ct(1,i))
10  end
```

Scilab code Exa 2.15 Fig 2pt15 Vignere Tableau

```
1  // Vignere tableau
2
3  a = ascii('A')
4
5  // Print header
6  printf(" \t")
7  for i=1:26
8       printf("%c ",ascii(a+i-1))
9  end
10 printf("\n\n")
11 //end of header
12
13 for i=1:26
14      printf("%c\t",ascii(a+i-1))
15      for j=0:25
16           printf("%c ",ascii( a + modulo( i+j+key, 26
                ) ) )
17      end
18      printf("\n")
19 end
```

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.18 Fig 2pt18 Keyword matrix

```
1  //Keyword matrix for the example Fig 2.18
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 key = "PLAYFAIREXAMPLE"
15
16 mat = playfair_matrix(key)                    //
     calling matrix population function from the
     dependency file
17 [row,col] = size(mat)
18 for m=1:row
19     for n=1:col
20         printf("%c ",ascii(mat(m,n)))
21     end
22     printf("\n")
23 end
```

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.25 Fig 2pt25 Encryption process in Playfair cipher

```
1  //Encryption process in Playfair cipher
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
```

```
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 //Playfair cipher key
15 key = "PLAYFAIR EXAMPLE"
16 disp("Original plaintext:")
17 pt = "MY NAME IS ATUL."
18 disp(pt)
19
20 //Using functions from dependency file to reformat
      the input
21
22 pt = playfair_pt(pt)              // substituting J to
       I and handling duplicates
23 pt_digram = digram_array(pt)        // converting to
      digrams
24
25 disp("Plaintext message broken down into pair of
      elements:")
26 print_matrix(pt_digram,0)
27 disp("")
28 a = ascii('A')
29
30 key_matrix = playfair_matrix(key);
31 // mat contains ascii values of characters of
      playfair matrix
32 //Use "disp(mat)" to verify this
33 disp("Playfair Cipher Key matrix: ")
34
35 print_matrix(key_matrix,1)
36
37 //disp(pt_matrix)
```

```
38 ct_mat = encrypt_playfair(pt_digram,key_matrix)
39
40 disp("Playfair ciphertext:")
41 print_matrix(ct_mat,0)
```

---

check Appendix AP 3 for dependency:

Chapter_2.sci

Scilab code Exa 2.26 Fig 2pt26 Keyword matrix

```
1  //Keyword matrix for the example Fig 2.18
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 key = "PLAYFAIR EXAMPLE"
15 printf("Keyword:\n%s\n\n",key)
16 printf("Matrix:\n")
17
18 //calling matrix population function from dependency
       file
19 mat = playfair_matrix(key)
20
21
22 [row,col] = size(mat)
23 for m=1:row
24     for n=1:col
```

```
25          printf("%c ",ascii(mat(m,n)))
26      end
27      printf("\n")
28  end
29
30  disp("")
```

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.33 Fig 2pt33 Practice example for playfair cipher

```
1  //Practice example for playfair cipher
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 //Playfair cipher key
15 key = "HARSH"
16 disp("Original plaintext:")
17 pt = "MY NAME IS JUI KAHATE. I AM HARSHU''S SISTER."
18 disp(pt)
19
20 //using functions from dependency file to reformat
      the input
21
```

```
22 pt = playfair_pt(pt)              // substituting J to
       I and handling duplicates
23 pt_digram = digram_array(pt)       // converting to
      digrams
24
25 disp("Plaintext message broken down into pair of
      elements:")
26 print_matrix(pt_digram,0)
27 disp("")
28 a = ascii('A')
29
30
31 //Calling function to calculate the playfair matrix
      from the dependency file
32 key_matrix = playfair_matrix(key);
33
34 // mat contains ascii values of characters of
      playfair matrix
35 //Use "disp(mat)" to verify this
36 disp("Playfair Cipher Key matrix: ")
37
38 print_matrix(key_matrix,1)
39
40 //disp(pt_matrix)
41 ct_mat = encrypt_playfair(pt_digram,key_matrix)
42
43 disp("Playfair ciphertext:")
44 print_matrix(ct_mat,0)
```

Scilab code Exa 2.34 Fig 2pt34 Encryption and decryption of hill cipher

```
1 ////////////////////////////////////////
2 //                                    //
3 //(a) Encrytion scheme of hill cipher //
4 //                                    //
```

```
5  ////////////////////////////////////////
6
7
8  //PLaintext
9  pt = "CAT"
10
11 disp("Plaintext: ")
12 disp(pt)
13
14 l = length(pt)
15 pt = strsplit(pt)
16
17 a = ascii("A")
18 pt_mat = []
19
20 //Taking A=0,B=1,C=2,etc.
21 for i=1:l
22     pt_mat(i,1)=ascii(pt(i,1))-a
23 end
24
25 disp("Plaintext matrix:")
26 disp(pt_mat)
27
28 //Key matrix
29 key_mat = [6 24 1; 13 16 10;20 17 15]
30 disp("Encryption Key matrix:")
31 disp(key_mat)
32
33 //ciphertext matrix
34 ct_mat = key_mat * pt_mat
35
36 disp("Product: ")
37 disp(ct_mat)
38 [r,c]=size(ct_mat)
39
40 //Taking mod for correct conversion
41 for i=1:r
42     ct_mat(i,1) = modulo(ct_mat(i,1),26)
```

```
43 end
44
45 disp("Ciphertext matrix: ")
46 disp(ct_mat)
47
48 disp("Ciphertext: ")
49
50 //Conversion of code to letters
51 ct=[]
52 for i=1:r
53     ct(i,1) = ascii(ct_mat(i,1)+a)
54 end
55 ct = strcat(ct)
56 disp(ct)
57
58
59
60 ////////////////////////////////////////
61 //                                    //
62 //(b) Decrytion scheme of hill cipher //
63 //                                    //
64 ////////////////////////////////////////
65
66 //Ciphertext
67 disp("Ciphertext: ")
68 disp(ct)
69
70 l = length(ct)
71 ct = strsplit(ct)
72
73 a = ascii("A")
74 ct_mat = []
75
76 //Taking A=0,B=1,C=2,etc.
77 for i=1:l
78     ct_mat(i,1)=ascii(ct(i,1))-a
79 end
80
```

```
81 disp("Ciphertext matrix:")
82 disp(ct_mat)
83
84 //Key matrix for decryption (inverse of encryption
      key matrix)
85 key_mat = [8 5 10; 21 8 21;21 12 8]
86 disp("Decryption Key matrix:")
87 disp(key_mat)
88
89 //ciphertext matrix
90 pt_mat = key_mat * ct_mat
91
92 disp("Product: ")
93 disp(pt_mat)
94 [r,c]=size(pt_mat)
95
96 //Taking mod for correct conversion
97 for i=1:r
98     pt_mat(i,1) = modulo(pt_mat(i,1),26)
99 end
100
101 disp("Plaintext matrix: ")
102 disp(pt_mat)
103
104 disp("Plaintext: ")
105
106 //Conversion of code to letters
107 pt=[]
108 for i=1:r
109     pt(i,1) = ascii(pt_mat(i,1)+a)
110 end
111 pt = strcat(pt)
112 disp(pt)
```

---

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.36 Fig 2pt36 Rail fence technique

```
1  //Rail fence technique
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 disp("Original plaintext message:")
15 pt = "Come home tomorrow"
16 disp(pt)
17
18 //function from dependency file
19 pt = remove_spaces(pt)
20
21 ct = []
22 k=1
23
24 //Writing diagonally
25 for i=1:length(pt)
26     if modulo(i,2)==0 then
27         continue
28     end
29     ct(k,1) = part(pt,i:i)
30     ct(k,2) = part(pt,i+1:i+1)
31     k = k+1
32 end
```

```
33
34 ct = strcat(ct)
35 disp("")
36 disp("Ciphertext:")
37 disp(ct)
```

---

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.38 Fig 2pt38 Example of simple columnar ransposition technique

```
1  //Example of simple columnar ransposition technique
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 disp("Original plaintext message:")
15 pt = "Come home tomorrow"
16 disp(pt)
17 disp("")
18
19 //function from dependency file
20 pt = remove_spaces(pt)
21
22 l = length(pt)
23
```

```
col = 6

row = l/6
    if modulo(l,6)>0 then
        row=row+1
    end

//Conversion of plaintext into a message table
//function from dependency file
pt_mat = message_rectangle(pt,col)

disp("Plaintext message rectangle:")
printf("\n")
for i=1:col
    printf(" %d ",i)
end
disp(pt_mat)
disp("")

//Column read order
col_order = [4 6 1 2 5 3]
disp("Column order: ")
disp(col_order)
disp("")
k=1

ct=[]
//Convert to ciphertext
for n = 1:length(col_order)
    j = col_order(n)
    for i=1:row
        pos = (i-1)*col + j
        if pos>l then
            continue
        end
        ct(k)=pt_mat(i,j)
        k=k+1
    end
```

```
62 end
63 disp("Ciphertext:")
64 ct = strcat(ct)
65 disp(ct)
```

---

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.40 Fig 2pt40 Example of simple columnar ransposition technique with multiple rounds

```
1  //Example of simple columnar ransposition technique
       with multiple rounds
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci")
13
14 pt = "Come home tomorrow"
15 disp("Original plaintext message:")
16 disp(pt)
17
18 //function from dependency file
19 l = length(remove_spaces(pt))
20 //disp(l)
21
22 rounds = 2
23 col_order = [4 6 1 2 5 3]
```

```
col = 6
row = l/6
    if modulo(l,6)>0 then
        row=row+1
    end

for r=1:rounds
    printf("\nRound %d:",r)

    //function from dependency file
    pt_mat = message_rectangle(pt)

    disp("")
    disp("Plaintext:")
    disp(pt)
    disp("Plaintext message rectangle:")
    printf("\n")
    for i=1:col
        printf(" %d ",i)
    end
    disp(pt_mat)

    k=1

    ct=[]
    //Convert to ciphertext
    for n = 1:length(col_order)
        j = col_order(n)
        for i=1:row
            pos = (i-1)*col + j
            if pos>l then
                continue
            end
            ct(k)=pt_mat(i,j)
            k=k+1
        end
    end
    disp("Ciphertext:")
```

```
62      ct = strcat(ct)
63      disp(ct)
64      pt = ct
65      disp("")
66  end
67
68  disp("Final ciphertext:")
69  disp(ct)
```

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.42 Fig 2pt42 Vernam cipher

```
1  //Vernam cipher
2
3  // Move scilab to current file directory
4  [u,t,n] = file()
5  n = strcat(n)
6  file_name = basename(n)+fileext(n)
7  file_name = strcat(file_name)
8  ind=strindex(n,file_name)
9  path = part(n,1:ind-1)
10 chdir(path)
11
12 exec("Chapter_2.sci",-1)
13
14 a= ascii('A')
15
16 pt = "HOW ARE YOU?"                    //Plaintext
17 disp("")
18 disp("Original plaintext:")
19 disp(pt)
20
21 //function from dependency file
```

```
22 pt = remove_spaces(pt)                    //Processed
      plaintext for encryption
23
24 disp("")
25 disp("Plaintext:")
26 disp(pt)
27 disp(ascii(pt)-a)
28
29 disp("")
30 disp("One-time pad:")
31 otp = "NCBTZQARX"                          //OTP
32 disp(otp)
33 disp(ascii(otp)-a)
34
35 ct = []
36
37 for i=1:length(pt)                          //Encryption
      stage
38     ct(i) = ascii(part(pt,i:i)) + ascii(part(otp,i:i
          )) -2*a
39 end
40
41 disp("")
42 disp("Initial total:")
43 disp(ct')
44
45
46 disp("")
47 disp("Subtracting 26 if >25")
48 ct = modulo(ct,26)                          //Taking modulo
      26 to make range b/w 0-25
49 disp(ct')
50 ct = char(ct+a)'                            //Ciphertext
51
52 disp("")
53 disp("Ciphertext: ")
54 disp(strcat(ct))
```

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.43 Fig 2pt43 Encryption

```
1
2 //Encryption
3
4 // Move scilab to current file directory
5 [u,t,n] = file()
6 n = strcat(n)
7 file_name = basename(n)+fileext(n)
8 file_name = strcat(file_name)
9 ind=strindex(n,file_name)
10 path = part(n,1:ind-1)
11 chdir(path)
12
13 exec("Chapter_2.sci",-1)
14
15 pt = "Hello John"
16
17 disp("Plain-text message:")
18 disp(pt)
19
20 a = ascii('a')
21 z = ascii('z')
22 A = ascii('A')
23 Z = ascii('Z')
24
25
26 ct = []
27 for i=1:length(pt(k))
28     x = ascii(part(pt(k,1),i:i))
29     if x>=A & x<=Z then
30         //function from dependency file
```

```
31          ct(k,i) = encrypt_caesar_general(part(pt(k),
                i:i),1)
32      elseif x>=a & x<=z then
33          c = convstr(part(pt(k),i:i),'u')
34          c = encrypt_caesar_general(c,1)
35          c = convstr(c,'l')
36          ct(k,i) = c
37      else
38          ct(k,i) = part(pt(k),i:i)
39      end
40  end
41
42  ct = strcat(ct)
43  disp("")
44  disp("Cipher text")
45  disp(ct)
```

---

check Appendix AP 3 for dependency:

`Chapter_2.sci`

Scilab code Exa 2.44 Fig 2pt44 Decryption

```
1
2  //Decryption
3
4  // Move scilab to current file directory
5  [u,t,n] = file()
6  n = strcat(n)
7  file_name = basename(n)+fileext(n)
8  file_name = strcat(file_name)
9  ind=strindex(n,file_name)
10 path = part(n,1:ind-1)
11 chdir(path)
12
13 exec("Chapter_2.sci",-1)
```

```
14
15 pt = "Ifmmp Kpio"
16
17 disp("Plain-text message:")
18 disp(pt)
19
20 a = ascii('a')
21 z = ascii('z')
22 A = ascii('A')
23 Z = ascii('Z')
24
25
26 ct = []
27 for i=1:length(pt)
28     x = ascii(part(pt(1,1),i:i))
29     if x>=A & x<=Z then
30         //function from dependency file
31         ct(1,i) = decrypt_caesar_general(part(pt,i:i
               ),1)
32     elseif x>=a & x<=z then
33         c = convstr(part(pt,i:i),'u')
34         c = decrypt_caesar_general(c,1)
35         c = convstr(c,'l')
36         ct(1,i) = c
37     else
38         ct(1,i) = part(pt,i:i)
39     end
40 end
41
42 ct = strcat(ct)
43 disp("")
44 disp("Cipher text")
45 disp(ct)
```

Scilab code Exa 2.51 Fig 2pt51 Number of parties and the corresponding number of lock and key pairs

```
1
2 // Number of parties and the corresponding number of
      lock-and-key pairs
3
4 printf("Parties involved\tNumber of lock-and-key
     pairs required")
5
6 n = (2:5)
7
8 //disp(n)
9 num = factorial(n)
10 //disp(num)
11 den = factorial(2)*factorial(n-2)
12 //disp(den)
13
14 for i=1:length(num)
15     printf("\n\t%d\t\t\t%d\n",n(i),num(i)/den(i))
16 end
```

Scilab code Exa 2.54 Fig 2pt54 Diffie Hellman key exchange

```
1 // Diffie-Hellman key exchange
2
3 n = 11                        // Two prime numbers
4 g = 7                         //need not be kept secret
5 printf("n: %d\ng: %d\n",n,g)
6
7 x = 3                         // Alice's secret random
     number
8 A = modulo((g^x),n)           // Alice's message to Bob
9                               //A = 2
10
```

```
11 y = 6                          // Bob's secret random
      number
12 B = modulo((g^y),n)            // Bob's message to Alice
13                                //B  =  4
14
15 printf("x: %d\ny: %d\nA: %d\nB: %d\n",x,y,A,B)
16
17 K1 = modulo((B^x),n)           // Alice's key
18                                //K1  =  9
19
20 K2 = modulo((A^y),n)           // Bob's key
21                                //K2  =  9
22 printf('Alice''s Key %d\n',K1)
23 printf('Bob''s Key %d',K2)
24  // K1 = K2, thus both Alice and Bob have the same
      key
```

Scilab code Exa 2.56 Fog 2pt56 Man in the middle attack in Diffie Hellman key exchange

```
1
2 //Man-in-the-middle attack in Diffie-Hellman key
     exchange
3
4 n = 11                          //Large prime numbers
5 g = 7                           //which are public
6
7 printf("n: %d\ng: %d\n",n,g)
8
9 x_a = 3                         //Alice's x
10 x_t = 8                        //Tom's x
11 y_t = 6                        //Tom's y
12 y_b = 9                        //Bob's y
13
14 A_a = modulo(g^x_a,n)          //Alice's A
```

```
15 A_t = modulo(g^x_t,n)          //Tom's A
16 B_t = modulo(g^y_t,n)          //Tom's B
17 B_b = modulo(g^y_b,n)          //Bob's B
18
19 disp("Before intrusion by Tom: ")
20 disp("For Alice:")
21 printf("x: %d\nA: %d\n",x_a,A_a)
22 disp("For Tom:")
23 printf("x: %d\ty: %d\nA: %d\tB: %d\n",x_t,y_t,A_t,
   B_t)
24 disp("For Bob:")
25 printf("y: %d\nB: %d\n",y_b,B_b)
26
27 A_b = A_t                        //Substituting Tom's A
   as A for Bob
28 B_a = B_t                        //Substituting Tom's B
   as B for Alice
29 A_t = A_a                        //Changing Tom's A to
   Alice's A
30 B_t = B_b                        //Changing Tom's B to
   Bob's B
31
32 disp("After intrusion by Tom during exhange of keys:
    ")
33 disp("For Alice:")
34 printf("x: %d\nA: %d\tB: %d\n",x_a,A_a,B_a)
35 disp("For Tom:")
36 printf("x: %d\ty: %d\nA: %d\tB: %d\n",x_t,y_t,A_t,
   B_t)
37 disp("For Bob:")
38 printf("y: %d\nA: %d\tB: %d\n",y_b,A_b,B_b)
39
40
41
42 //Now, Tom can calculate separate keys for Alice and
    Bob
43
44 K1_a = modulo(B_a^x_a,n)          //Alice's key
```

```
45 K1_t = modulo(B_t^x_t,n)          //Tom's key for Bob
46 K2_t = modulo(A_t^y_t,n)          //Tom's key for
      Alice
47 K2_b = modulo(A_b^y_b,n)          //Bob's key
48
49 printf("\n\nKeys:\n")
50
51 disp("Alice''s key:")
52 printf("\tK1: %d\n\n",K1_a)
53 disp("Tom''s keys:")
54 printf("\nTo communicate with Alice\n\tK2: %d",K2_t)
55 printf("\nTo communicate with Bob\n\tK1: %d\n\n",
      K1_t)
56 disp("Bob''s key:")
57 printf("\tK2: %d",K2_b)
58
59 //We can see that K1_a == K2+t and K1_t == K2_b
60 //Thus, Tom can communicate with Alice using K2_t
      and with Bob using K1_t
61 //and easily carry out
```

Scilab code Exa 2.65 Fig 2pt65 Understanding key range

```
1
2 //Understanding key range
3
4 n = [2; 3]
5 states = []
6 for i=1:length(n)
7     printf("Bits: %d\n",n(i,1))
8     printf("No of states: %d",2^n(i,1));
9     disp("The states are:")
10    for j=0:2^n(i,1)-1
11        disp(dec2bin(j))
12    end
```

```
13      disp("")
14  end
```

Chapter 3

Computer Based Symmetric Key Cryptographic Algorithm

Scilab code Exa 3.2 Fig 3pt2 Functioning of XOR logic

```
1
2 //Functioning of XOR logic
3
4 printf("\tInput 1\tInput 2\tOutput\n\n")
5
6 for i=0:1
7     for j=0:1
8         printf("\t  %d\t  %d\t  %d\n",i,j,bitxor(i,j
              ))
9     end
10 end
```

Scilab code Exa 3.2.1.1 Example XOR operations

```
1
2 //Example XOR operations
```

```
3
4 A  = bin2dec("101")
5 printf("A: %s\n\n",dec2bin(A))
6 B  = bin2dec("110")
7 printf("B: %s\n\n",dec2bin(B))
8
9 C = bitxor(A,B)
10 printf("C: %3s\n\n",dec2bin(C))
11
12 disp("C XOR A")
13 disp(dec2bin(bitxor(C,A)))
14
15 disp("C XOR B")
16 disp(dec2bin(bitxor(C,B)))
```

Scilab code Exa 3.3 Fig 3pt3 Stream cipher

```
1
2 //Stream cipher
3
4 disp("In text format:")
5 disp("Plain text")
6 disp("Pay 100")
7 disp("")
8
9 disp("Cipher text")
10 disp("ZTU91 ^%D")
11 disp("")
12
13 disp("")
14
15 disp("In binary format:")
16
17 disp("Plain text")
18
```

```
19 pt = "010111001"
20
21 disp(pt)
22 disp("")
23
24 //convert to decimal
25 pt = bin2dec("010111001")
26
27 disp("XOR operation with the key")
28 key="100101011"
29 disp(key)
30
31 //convert key to decimal
32 key=bin2dec(key)
33 disp("")
34
35 //calculate cipher text
36 ct = bitxor(pt,key)
37 ct = dec2bin(ct)
38
39 disp("Cipher text")
40 disp(ct)
41 disp("")
```

Scilab code Exa 3.34 Fig 3pt34 Example of selection of S box output based on the input

```
1
2 x = [1 0 1 1 0 1]
3
4 disp("Row: ")
5 row = [x(1),x(6)]
6 printf("in binary - %d%d",row)
7
8 //Convert to decimal
```

```
9  printf("\nin decimal - %d",bin2dec(strcat([string(
       row)])))
10
11 disp("")
12
13 disp("Column: ")
14 col = x(2:5)
15 printf("in binary - %d%d%d%d",col)
16
17 //Convert to decimal
18 printf("\nin decimal - %d",bin2dec(strcat([string(
       col)])))
```

Scilab code Exa 3.81 Fig 3pt81 Key expansion example

```
1
2  //Key expansion example
3
4  n = 0:15
5  n = int8(n)
6  disp("Byte position(decimal)")
7  for i=1:length(n)
8      printf("%4d",n(i))
9  end
10
11 disp("")
12
13 disp("Value(hex)")
14
15 for i=1:length(n)
16     printf("  %s","0"+string(dec2hex(n(i))))
17 end
18
19 disp("")
20
```

```
21 for i=0:3
22     printf("\n\tW[%d]\t\t\t",i)
23     for j=1:4
24         printf("0%s\t",string(dec2hex(n(i*4+j))))
25     end
26
27 end
```

Chapter 4

Computer Based Asymmetric Key Cryptographic Algorithm

Scilab code Exa 4.4 Fig 4pt4 RSA algorithm example

```
1  //RSA algorithm example
2
3  p = 47
4  q = 71
5
6  n = p*q
7  z = (p-1)*(q-1)
8
9  e = 79                                    // E<N and E & Z
       are coprime
10
11 i=1
12 d = i
13
14 //Brute-force approach to find 'd'
15 while(1==1)
16     if modulo(i*e,z)==1 then              // (E*D)mod Z = 1
17         d=i
18         break
```

```
19      end
20      i=i+1
21  end
22
23  printf("%d",d)
24
25  //Public key: (n,e)
26  //Privae key: (n,d)
27  printf("\nPublic Key: (%d,%d)\nPrivate Key: (%d,%d)\
       n\n",n,e,n,d)
28
29
30  P = 688                                 //Plaintext
31  printf("Plaintext: %d\n",P)
32
33  C = 1                                   //Encrypted Text
34  for i = 1:e
35      C = modulo(C*P,n)
36  end
37
38  printf("Encrypted Text: %d\n",C)
39
40  P=1                                     //Decrypting the
       encypted text 'C'
41  for i = 1:d
42      P = modulo(P*C,n)
43  end
44
45  printf("Decrypted Text: %d\n",P)
```

Scilab code Exa 4.5 Fig 4pt5 RSA Encryption scheme

```
1  // RSA Encryption scheme
2
3  p = 7                              //Large prime numbers
```

```
4  q = 17                          //Small values taken here
       for convenience of calculation
5                                  //and explanation
6
7  n = p*q
8  z = (p-1)*(q-1)
9
10 e = 5                                    // e<n and e & z
       are coprime
11
12 i=1
13 d = i
14 while(1==1)                              //Calcualtion of
       'd' from e and z
15     if modulo(i*e,z)==1 then             // (e*d)mod z = 1
16         d=i
17         break
18     end
19     i=i+1
20 end
21
22 printf("\nPublic Key: (%d,%d)\nPrivate Key: (%d,%d)\
       n\n",n,e,n,d)
23
24 PT = 10                                   //Example
       plaintext
25
26 printf("Plaintext: %d\n",PT)
27
28 C = modulo(PT^e,n)
29 printf("\nEncrypted Text Code: %d\n\n",C)
30
31 PT=1
32 for i = 1:d
33     PT = modulo(PT*C,n)
34 end
35
36 printf("Decrypted Text: %d\n",PT)          //Conversion
```

to plain text in ASCII

Scilab code Exa 4.5.1 ElGamal Key Generation

```
1
2 //ElGamal Key Generation
3
4 p = 11
5 e1 = 2
6 d = 3
7
8 e2 = modulo(e1^d,p)
9
10 disp("Public key:")
11 printf("(%d,%d,%d)",e1,e2,p)
```

Scilab code Exa 4.5.2 ElGamal Key Encryption

```
1
2 //ElGamal Key Encryption
3
4 r = 4
5 pt = 7
6 e1 = 2
7 e2 = modulo(e1^d,p)
8
9 c1 = modulo(e1^r,p)
10 c2 = modulo(pt*e2^r,p)
11
12 disp("Cipher text")
13 printf("(%d,%d)",c1,c2)
```

check Appendix AP 2 for dependency:

Chapter_4.sci

Scilab code Exa 4.5.3 ElGamal Key Decryption

```
1
2 //ElGamal Key Decryption
3
4 // Move scilab to current file directory
5 [u,t,n] = file()
6 n = strcat(n)
7 file_name = basename(n)+fileext(n)
8 file_name = strcat(file_name)
9 ind=strindex(n,file_name)
10 path = part(n,1:ind-1)
11 chdir(path)
12
13 exec("Chapter_4.sci",-1)
14
15 p = 11
16 r = 4
17 pt = 7
18 d = 3
19 e1 = 2
20 e2 = modulo(e1^d,p)
21
22 c1 = modulo(e1^r,p)
23 c2 = modulo(pt*e2^r,p)
24
25 x =c1^d
26 x_inv = mod_inv(x,p)
27
28 pt = modulo(c2*x_inv,p)
29 disp("Original plaintext")
30 disp(pt)
```

Scilab code Exa 4.6 Fig 4pt6 RSA Encryption scheme

```
1  // RSA Encryption scheme
2
3  p = 7                          //Large prime numbers
4  q = 17                         //Small values taken here
       for convenience of calculation
5                                 //and explanation
6
7  n = p*q
8  z = (p-1)*(q-1)
9
10 e = 5                                  // e<n and e & z
       are coprime
11
12 i=1
13 d = i
14 while(1==1)                            //Calcualtion of
       'd' from e and z
15     if modulo(i*e,z)==1 then           // (e*d)mod z = 1
16         d=i
17         break
18     end
19     i=i+1
20 end
21
22 printf("\nPublic Key: (%d,%d)\nPrivate Key: (%d,%d)\
       n\n",n,e,n,d)
23
24 PT = 'F'                                 //Example
       plaintext in ASCII
25 P = ascii(PT)-ascii('A')+1              //Conversion of
       ASCII to integer code
26                                        //(A=1,B=2,C=3,
```

```
                                                    ... )
27 printf("Plaintext: %s\n",PT)
28 printf("Plain text Integer code: %d\n\n\",P)
29
30 C = 1
31 for i = 1:e
32     C = modulo(C*P,n)
33 end
34
35 C = modulo(C,n)
36 printf("Encrypted Text Code: %d\n\n",C)
37
38 P=1
39 for i = 1:d
40     P = modulo(P*C,n)
41 end
42
43 PT = ascii(ascii('A')+P-1)
44 printf("Decrypted Text Code: %d\n",P)    //Decryoted
     code
45 printf("Decrypted Text: %s\n",PT)        //Conversion
     to plain text in ASCII
```

---

check Appendix AP 2 for dependency:

`Chapter_4.sci`

Scilab code Exa 4.9.1 ElGamal Signature

```
1
2 //ElGamal Signature
3
4 // Move scilab to current file directory
5 [u,t,n] = file()
6 n = strcat(n)
7 file_name = basename(n)+fileext(n)
```

```
8  file_name = strcat(file_name)
9  ind=strindex(n,file_name)
10 path = part(n,1:ind-1)
11 chdir(path)
12
13 exec("Chapter_4.sci",-1)
14
15 e1 = 10
16 e2 = 4
17 p = 19
18 m = 14        //original message
19 d = 16
20 r = 5         //random number selected by sender
21
22 r_inv = mod_inv(r,p-1)       //inverse of r modulo (p
      -1)
23
24 s1 = modulo(e1^r,p)
25
26 temp = (m-d*s1)*r_inv
27 while temp<0                        //calculate modulus
      (p-1) for negative values
28     temp = temp+p-1
29 end
30 s2 = modulo(temp,p-1)
31
32 printf("The signature is: (%d,%d)",s1,s2)
```

Scilab code Exa 4.9.2 ElGamal Signature verification

```
1
2 //ElGamal Signature verification
3
4 e1 = 10
5 e2 = 4
```

```
m = 14
p = 19
s1 = 3
s2 = 4

v1 = modulo(e1^m,p)
disp("V1")
disp(v1)

v2 = modulo(e2^s1 * s1^s2,p)
disp("V2")
disp(v2)

disp("Since V1=V2, signature is valid")
```

Scilab code Exa 4.17 Fig 4pt17 Longitudinal redundancy check

```
//Longitudinal redundancy check

data = [ "11100100","11011101","00111001","00101001"
    ]
disp("Original data")
disp(data)
data = bin2dec(data)

lrc = 0.

for i=1:length(data)
    lrc = bitxor(lrc,data(i))
end

disp("LRC: ")

for i=1:7
```

```
18      if lrc<(2^(8-i)) then
19          printf("0")
20      else
21          printf("%s",dec2bin(lrc))
22          break
23      end
24  end
```

Scilab code Exa 4.18 Fig 4pt18 Simple message digest

```
1  // Simple message digest
2
3  n = 7391743                              //Message
4  printf("Original number is %d\n",n)
5
6  n_str = string(n)                        //Conversion of
       integer to string for easy access
7  l = length(n_str)
8  n_v = strsplit(n_str,1:l-1)              //String to
       vector of characters
9
10 d = 1
11 for i=1:l
12      d = d * ( ascii(n_v(i:i)) - ascii('0'))
            //
13      d = modulo(d,10)
14      i = i+1
15 end
16
17 printf("Message digest is %d\n",d)
```

Scilab code Exa 4.62 Fig 4pt62 Knapsack algorithm for Public Key Encryption

```
// Knapsack algorithm for Public Key Encryption

PT = [0 1 1 0 1 1; 1 1 1 0 0 0; 0 1 0 1 1 0]

disp("Plain text")
disp(PT)

K = [1 7 8 12 14 20]
disp("Knapsack:")
disp(K)

[row,col] = size(PT)
C = []
for i=1:row
    sum=0
    for j=1:col
        sum = sum+PT(i,j)*K(j:j)
    end
    C(i:i) = sum
end

disp("Cipher text:")
disp(C)
```

Chapter 6

Internet Security Protocols

check Appendix AP 1 for dependency:

Chapter_6.sci

Scilab code Exa 6.54 Fig 6pt54 Base 64 encoding example

```
1
2 //Base 64 encoding example
3
4 [u,t,n] = file()
5 n = strcat(n)
6 file_name = basename(n)+fileext(n)
7 file_name = strcat(file_name)
8 ind=strindex(n,file_name)
9 path = part(n,1:ind-1)
10 chdir(path)
11
12 //Get function to create encoding table
13 exec("Chapter_6.sci",-1)
14
15 enc = encoding_table()
16
17 inp = "001000110101110010010001"          // Input
```

```
18 disp("24-bit input:")
19 disp(inp)
20 disp("")
21 dec = []
22
23 for i=1:length(inp)/6                          //
      Convert to 6-bit packets stored as
24     str = part(inp,((i-1)*6+1):((i-1)*6+6)) //
         integers
25     dec(i)=0
26     for j=1:length(str)
27         if part(str,j:j)=='1' then
28             dec(i) = dec(i)+2^(6-j)
29         end
30     end
31 end
32
33 disp("Divided into 6-bit blocks:")
34 disp(dec2bin(dec'))
35 disp("")
36
37 disp("Decimal equivalents:")
38 disp(dec')                                      //
      Decimal equivalents
39 disp("")
40 dec_str = []
41
42 for i=1:length(dec)
43     dec_str(i) = (ascii(enc(dec(i)+1)))
44 end
45
46 disp("Map to base-64 encoding table (shown in Fig.
      6.55):")
47 disp(dec_str')                                  //
      Character values from encoding table
48 disp("")
49 dec_str = ascii(dec_str)
50
```

```
51 bin_str = dec2bin(dec_str)                    //
     Convert to ASCII binary
52 bin_str = string(bin_str)
53 bin_str = '0'+bin_str                          //
     Convert to 8-bit from 7-bit
54 disp("ASCII equivalent binary")
55 disp(bin_str)
```

check Appendix AP 1 for dependency:

`Chapter_6.sci`

Scilab code Exa 6.55 Fig 6pt55 Base 64 encoding mapping table

```
1
2 //Base-64 encoding mapping table
3
4 disp("Base-64 encoding mapping table")
5
6 [u,t,n] = file()
7 n = strcat(n)
8 file_name = basename(n)+fileext(n)
9 file_name = strcat(file_name)
10 ind=strindex(n,file_name)
11 path = part(n,1:ind-1)
12 chdir(path)
13
14 //Get function to create encoding table
15 exec("Chapter_6.sci",-1)
16
17 enc = encoding_table()
18
19 for i=0:63
20     printf("%d - %c\n",i,ascii(enc(i+1)))
21 end
```

Appendix

Scilab code AP 1 Functions for Chapter 6 example codes

```
1
2  ////////////////////////////////
3  //      base-64 encoding      //
4  ////////////////////////////////
5
6  function [enc]=encoding_table()
7      a = ascii('A')
8      enc = []
9
10
11     for i=0:25
12         enc(i+1) = i+a
13     end
14
15
16     for i=26:51
17         enc(i+1) = i+a+6
18     end
19
20     for i=52:61
21         enc(i+1) = i-52+ascii('0')
22     end
23
24     enc(63) = ascii('+')
25     enc(64) = ascii('/')
26
```

```
27 endfunction
```

Scilab code AP 2 Chapter 4 functions

```
1
2 //Euclid's extended algorithm to calculate inverse
     of n modulo p
3
4 function [ans]=mod_inv(n,p)
5     p_ = p
6     q = []
7     m = []
8     i=1
9     r = 1
10    while r>=0
11        if i<3
12            m(i,1) = i-1
13        else
14            m(i,1) = m(i-2,1) - m(i-1,1)*q(i-2,1)
15            if m(i,1)<0
16                m(i,1) = m(i,1)+p_
17            end
18            m(i,1) = modulo(m(i,1),p_)
19        end
20        if r==0
21            break
22        end
23        q(i,1) = int(p/n)
24        r = modulo(p,n)
25        p = n
26        n = r
27        i = i+1
28    end
29    ans = m(i,1)
30 endfunction
```

Scilab code AP 3 Functions for Chapter 2 example codes

```
1
2  ////////////////////////////////////
3  //          Caesar cipher         //
4  ////////////////////////////////////
5
6
7  //Generalised Caesar cipher encryption
8  function [ct] = encrypt_caesar_general(pt,key)
9      a = ascii('A')
10     l = length(pt)
11     ct = zeros(l)
12
13     for i =1:l
14         if isletter(part(pt,i:i)) then
15             ct(i) = a + modulo( ascii(part(pt,i:i))+
                   key-a, 26 )
16         else
17             ct(i) = ascii( part(pt,i:i) )
18         end
19     end
20     ct = char(ct)
21     ct = strcat(ct)
22 endfunction
23
24 //Caesar cipher encryption (key = 3  always)
25 function [ct] = encrypt_caesar(pt)
26     ct = encrypt_caesar_general(pt,3)
27 endfunction
28
29
30 //Generalised Caesar cipher decryption
31 function [pt] = decrypt_caesar_general(ct,key)
32     a = ascii('A')
33     key = 26-key
34     l = length(ct)
35     pt = zeros(l)
36
37     for i =1:l
```

```
38          if isletter(part(ct,i:i)) then
39              pt(i) = a + modulo( ascii(part(ct,i:i))+
                    key-a, 26 )
40          else
41              pt(i) = ascii(part(ct,i:i));
42          end
43      end
44      pt = char(pt)
45      pt = strcat(pt)
46  endfunction
47
48
49  //Caesar cipher decryption (key = 3  always)
50  function [pt] = decrypt_caesar(ct)
51      pt = decrypt_caesar_general(ct,3)
52  endfunction
53
54
55
56  ////////////////////////////////////
57  //          Playfair cipher           //
58  ////////////////////////////////////
59
60
61
62  //func to remove spaces from a string
63  function [mat]=remove_spaces(str)
64      mat=[]
65      k=1
66      for i=1:length(str)
67          if ~isletter(part(str,i:i)) then
68              continue
69          end
70          mat(k,1) = part(str,i:i)
71          k=k+1
72      end
73      mat = strcat(mat)
74  endfunction
```

```
75
76
77  //func to substitute I for J
78  function [mat]=i_to_j(str)
79      str = remove_spaces(str)
80      mat=[]
81      k=1
82      for i=1:length(str)
83          mat(k,1) = part(str,i:i)
84          if mat(k,1)=='J' then
85              mat(k,1) = 'I'
86          end
87          k = k+1
88      end
89      mat = strcat(mat)
90  endfunction
91
92  //func to insert X between repeating characters
93  function [mat]=handle_duplicates(str)
94      mat = []
95      l = length(str)
96      k = 1
97
98      for i=1:l
99          if i>1 & part(str,i:i)==part(str,i-1:i-1)
               then
100             mat(k,1)='X'
101             k=k+1
102         end
103         mat(k,1) = part(str,i:i)
104         k = k+1
105     end
106     mat = strcat(mat)
107 endfunction
108
109 //Matrix creation and population for Playfair cipher
110 //func to populate playfair matrix
111 function [mat]=playfair_matrix(key)
```

```
112
113     key = i_to_j(key)
114     a = ascii('A')
115     i = ascii('I')
116     j = ascii('J')
117     row = 5
118     col = 5
119     visited = zeros(26,1);
120     mat = ones(row,col);
121
122     len = length(key)
123
124     li=1
125     k=1
126
127     for m=1:row
128         for n=1:col
129             while li<=len & visited(ascii(part(key,
                    li:li)) - ascii('A')+1,1)~=0,
130                 li=li+1
131                 if part(key,li:li)=='I' & visited(j-
                        a+1)==1 | part(key,li:li)=='J' &
                        visited(i-a+1)==1 then
132                     li = li+1
133                 end
134             end
135             while k<=26 & visited(k,1)~=0
136                 k=k+1
137                 if k==i-a+1 & visited(j-a+1)==1 | k
                        ==j-a+1 & visited(i-a+1)==1 then
138                     k = k+1
139                 end
140             end
141             if li<=len then
142                 mat(m,n) = ascii(part(key,li:li))
143                 visited(ascii(part(key,li:li))-a
                        +1,1) = 1
144             else
```

```
145                     mat(m,n) = k+ascii('A')-1
146                     visited(k,1) = 1
147                 end
148
149             end
150         end
151
152 endfunction
153
154 //func to check and convert plaintext to suitable
      format for encipherment using playfair cipher
155 function [mat]=playfair_pt(pt)
156     mat = i_to_j(pt)
157     mat = handle_duplicates(mat)
158 endfunction
159
160 function [mat]=digram_array(pt)
161     k = 1
162     l = length(pt)
163     for i=1:l
164         if modulo(i,2)==0 then
165             continue
166         end
167         mat(k,1) = part(pt,i:i)
168         i=i+1
169         if i>l then
170             mat(k,2) = 'X'
171         else
172             mat(k,2) = part(pt,i:i)
173         end
174         k=k+1
175     end
176 endfunction
177
178 function []=print_matrix(mat,new_line)
179     [r,c] = size(mat)
180     t = type(mat)
181
```

```
182     for i=1:r
183         for j=1:c
184             if  t==[1] then          // real numbers
                   return 1, characters return 10
185                 printf("%c ",ascii(mat(i,j)))
186             else
187                 printf("%c ",mat(i,j))
188             end
189         end
190         printf("    ")
191         if new_line~=0 then
192             printf("\n")
193         end
194     end
195 endfunction
196
197 function [r,c]=find_letter(key_mat,a)
198     [row,col] = size(key_mat)
199     r = 0
200     c = 0
201     for i=1:row
202         for j=1:col
203             if ascii(key_mat(i,j))==a then
204                 r=i
205                 c=j
206                 break
207             end
208         end
209     end
210 endfunction
211
212
213 function [mat]=encrypt_playfair(pt_mat,key_mat)
214
215     [row,col] = size(pt_mat)
216     mat = []
217
218     for i=1:row
```

```
219         a = pt_mat(i,1)
220         b = pt_mat(i,2)
221         [r_a,c_a] = find_letter(key_mat,a)
222         [r_b,c_b] = find_letter(key_mat,b)
223 
224         if r_a==r_b then
225             c_a = modulo(c_a,5)+1
226             c_b = modulo(c_b,5)+1
227         elseif c_a==c_b then
228             r_a = modulo(r_a,5)+1
229             r_b = modulo(r_b,5)+1
230         else
231             temp = c_a
232             c_a = c_b
233             c_b = temp
234         end
235         mat(i,1) = ascii(key_mat(r_a,c_a))
236         mat(i,2) = ascii(key_mat(r_b,c_b))
237 
238     end
239 endfunction
240 
241 
242 ////////////////////////////////////
243 //      Transposition cipher      //
244 ////////////////////////////////////
245 
246 function [mat]=message_rectangle(str,col)
247     l = length(str)
248     row = l/6
249     if modulo(l,6)>0 then
250         row=row+1
251     end
252     //remove whitespace and non-alphabets from
        string
253     str = remove_spaces(str)
254     //Conversion of plaintext into a message table
255     mat = []
```

```
256     k=1
257     for i=1:row
258         for j=1:col
259             if k>l then
260                 break
261             end
262             mat(i,j) = part(str,k:k)
263             k=k+1
264         end
265     end
266 endfunction
267
268 ////////////////////////////////////
269 //    Diffie-Hellman Key Exchange    //
270 ////////////////////////////////////
271
272 function [key]=diffie_key(g,p,n)
273     key = modulo(g^p,n)
274 endfunction
```