Sample problems

1. (Plaintfair cipher) Decrypt the ciphertext

OPNBNMDYNBDLCDXIWNMECRYDNIONP

if the keyword is DECRYPTION.

Solution.

The 5x5 matrix has the following view:

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
<th>C</th>
<th>R</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>T</td>
<td>I/J</td>
<td>O</td>
<td>N</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>F</td>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>K</td>
<td>L</td>
<td>M</td>
<td>Q</td>
<td>S</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

Repeatedly applying the three rules the given ciphertext can be decrypted in the following way:

Ciphertext: OPNBNMDYNBDLCDXIWNMECRYDNIONP
Plaintext: inthiscasethesecretisdecryption

2. (Hill cipher) Encrypt the plaintext PLAINTEXT with the key

\[
K = \begin{pmatrix}
5 & 1 & 6 \\
4 & 9 & 8 \\
3 & 10 & 12
\end{pmatrix}
\]

Solution.

\[
\begin{align*}
C_1 &= (5p_1 + 6p_2 + 9p_3) \mod 26 \\
C_2 &= (4p_1 + 9p_2 + 8p_3) \mod 26 \\
C_3 &= (3p_1 + 10p_2 + 12p_3) \mod 26
\end{align*}
\]

By substituting \(p_1=15, \ p_2=11, \ p_3=0, \ p_4=8, \ p_5=13, \ p_6=p_9=19, \ p_7=4, \) and \(p_1=23,\) we obtain

\[
\begin{pmatrix}
C_1 \\
C_2 \\
C_3
\end{pmatrix} = \begin{pmatrix}
5 & 1 & 6 \\
4 & 9 & 8 \\
3 & 10 & 12
\end{pmatrix} \begin{pmatrix}
15 \\
11 \\
0
\end{pmatrix}
\]
Thus,

\[ C_1 = 5 \times 15 + 1 \times 11 + 6 \times 0 = 75 + 11 + 0 = 86 \mod 26 = 8 \]

\[ C_2 = 4 \times 15 + 9 \times 11 + 8 \times 0 = 60 + 99 + 0 = 159 \mod 26 = 3 \]

\[ C_3 = 3 \times 15 + 10 \times 11 + 12 \times 0 = 45 + 110 + 0 = 155 \mod 26 = 25 = Z \]

Similarly,

\[
\begin{pmatrix}
C_4 \\
C_5 \\
C_6
\end{pmatrix} = \begin{pmatrix}
5 & 1 & 6 \\
4 & 9 & 8 \\
3 & 10 & 12
\end{pmatrix}
\begin{pmatrix}
8 \\
13 \\
19
\end{pmatrix}
\]

or

\[ C_4 = 5 \times 8 + 1 \times 13 + 6 \times 19 = 40 + 13 + 114 = 167 \mod 26 = 5 = F \]

\[ C_5 = 4 \times 8 + 9 \times 13 + 8 \times 19 = 48 + 117 + 162 = 327 \mod 26 = 15 = P \]

\[ C_6 = 3 \times 8 + 10 \times 13 + 12 \times 19 = 24 + 130 + 228 = 382 \mod 26 = 18 = S \]

Finally,

\[
\begin{pmatrix}
C_7 \\
C_8 \\
C_9
\end{pmatrix} = \begin{pmatrix}
5 & 1 & 6 \\
4 & 9 & 8 \\
3 & 10 & 12
\end{pmatrix}
\begin{pmatrix}
4 \\
23 \\
19
\end{pmatrix}
\]

or

\[ C_7 = 5 \times 4 + 1 \times 23 + 6 \times 19 = 20 + 23 + 114 = 157 \mod 26 = 20 = U \]

\[ C_8 = 4 \times 4 + 9 \times 23 + 8 \times 19 = 16 + 207 + 162 = 385 \mod 26 = 25 = Z \]

\[ C_9 = 3 \times 4 + 10 \times 23 + 12 \times 19 = 12 + 230 + 228 = 470 \mod 26 = 18 = C \]

The plaintext encrypted results IDZFPSUZC.

3. Suppose that we are using a Vigenere scheme with 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the given message. Given the ciphertext

YZRDTJVAROHOPHSGXGGXQTJWISIQPJSNRPZKOSNYOZX
find the key that yields the following plaintext:

MR MUSTARD WITH THE CANDLESTICK IN THE HALL

Solution.

As we know the key letter and plaintext letter identify the row, and the column, respectively. Hence, if the ciphertext and plaintext letters are given, the key letter can found easily. As it follows from the above table the key used to encrypt the plaintext

MR MUSTARD WITH THE CANDLESTICK IN THE HALL

is

MISS SCARLET WITH THE KNIFE IN THE SHOWROOM

4. (Rail fence technique) Use rail fence technique to decrypt the following message

ESENEIERNIAUIESTATRMDTRAENNVRIY

Solution.

In the rail fence technique, the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. Implementing this technique to the given ciphertext will result

ESENENIERNIERNIAUIESTATRMDTRAENNVRIY
That is, the plaintext is EASTERNMEDITERRANEAN UNIVERSITY.

5. Given ciphertext IYRMVLNSTLAEDTRSCEEIGIYSANLSII and the key 3 6 1 2 5 4, use transposition technique based on writing the message in a rectangle, row by row, and reading the message off, column by column, with permuting the order of columns to decrypt it.

Solution.

We first split the given ciphertext in six blocks, then write the blocks column-wise according to the given key, then read the message in. As a result we obtain the following table.

<table>
<thead>
<tr>
<th>3 6 1 2 5 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A S I N G L</td>
</tr>
<tr>
<td>E C Y L I N</td>
</tr>
<tr>
<td>D E R S Y S</td>
</tr>
<tr>
<td>T E M I S T</td>
</tr>
<tr>
<td>R I V I A L</td>
</tr>
</tbody>
</table>

Hence, the message is ASINGLECYLINDERSYSTEMISTRIVIAL.

6. (S-DES) Using S-DES, decrypt the string 10100010 using the key 0111111101 by hand. Show intermediate results after each function ($IP, F_k, SW, F_k, IP^{-1}$). Then decode the first 4 bits of the plaintext string and ciphertext string to letters and the second 4 bits to another letters where we encode A through P in base 2 (i.e. A=0000, B=0001, ..., P=1111).

Solution.

S-DES Key Generation

<table>
<thead>
<tr>
<th>Action</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>P10</td>
<td>0111111101</td>
<td>1111110011</td>
</tr>
<tr>
<td>LS-1</td>
<td>11111</td>
<td>11111</td>
</tr>
<tr>
<td></td>
<td>10011</td>
<td>00111</td>
</tr>
<tr>
<td>P8</td>
<td>1111100111</td>
<td>01011111</td>
</tr>
<tr>
<td>LS-2</td>
<td>11111</td>
<td>11111</td>
</tr>
<tr>
<td></td>
<td>00111</td>
<td>11100</td>
</tr>
<tr>
<td>P8</td>
<td>1111111100</td>
<td>11111100</td>
</tr>
</tbody>
</table>
S-DES Encryption

8-bit plaintext: 10100010 (KC)
IP 10100010 00110001
E/P
Excusive-OR
S0 1101 11
S1 1101 00
P4 1100 1001
Excusive-OR
SW 10100001 00011010
E/P
Excusive-OR
S0 1101 11
S1 1101 00
P4 1100 1001
Excusive-OR
IP⁻¹ 00101010 00111000
8-bit ciphertext: 00111000

S-DES Decryption

8-bit ciphertext: 00111000 (DH)
IP 00111000 00101010
E/P
Excusive-OR
S0 1010 10
S1 1001 10
P4 1010 0011
Excusive-OR
SW 00011010 10100001
E/P
Excusive-OR
S0 1101 11
S1 1101 00
P4 1100 1001
Excusive-OR
IP⁻¹ 00110001 10100010
8-bit plaintext: 10100010

Blowfish
Blowfish is a symmetric block cipher developed by Bruce Schiner. Blowfish was designed to have the following characteristics:

- **Fast**: Blowfish encrypts data on 32-bit microprocessor at a rate of 18 clock cycles per byte.
- **Compact**: Blowfish can run in less than 5K of memory.
- **Simple**: Blowfish’s simple structure is easy to implement and eases the task of determining the strength of the algorithm.
- **Variable secure**: The key length is variable and can be as long as 448 bits. This allows a tradeoff between higher speed and higher security.

Blowfish encrypts 64-bit blocks of plaintext into 64-bit blocks of ciphertext. Blowfish is implemented in numerous products and has received a fair amount of security.

**Subkey and S-box generation**

Blowfish makes use of a key that ranges from 32 bits to 448 bits (1 to 14 32-bit words). That key is used to generate 18 32-bit subkeys and four 8x32 S-boxes containing a total of 1024 32-bit entries. The total is 1042 32-bit values, or 4168 bytes.

The keys are stored in a $K$-array:

$$K_1, K_2, ..., K_j, \quad 1 \leq j \leq 14$$

The subkeys are stored in the $P$-array:

$$P_1, P_2, ..., P_{18}$$

There are four $S$-boxes each with 256 32-bit entries:

$$S_{1,0}, S_{1,1}, ..., S_{1,255}$$

$$S_{2,0}, S_{2,1}, ..., S_{2,255}$$

$$S_{3,0}, S_{3,1}, ..., S_{3,255}$$

$$S_{4,0}, S_{4,1}, ..., S_{4,255}$$

The steps in generating $P$-array and $S$-boxes are as follows:
1. Initialize first the \( P \)-array and then the four \( S \)-boxes in order using the bits of the fractional part of the constant \( \pi \). Then the leftmost 32 bits of the fractional part of \( \pi \) become \( P_1 \), and so on. Note that in binary \( \pi \) has the following view

\[
\begin{align*}
(00001010) & (00100100001111110110101010001000) \\
(00001010) & (00100100001111110110101010001000) \\
(00001010) & (00001010011101111010100100110111) \\
(00001010) & (00001010011101111010100100110111) \\
& \ldots
\end{align*}
\]

Thus, \( P_1 = 00100100001111110110101010001000 \)
\( P_2 = 10000101110101010100010011011100 \)
\( S_1, 0 = 11010001011000110000101110101100 \)
\( S_1, 1 = 10011000110111011110101101101100 \)
\( \ldots \)

2. Perform a bitwise XOR of the \( P \)-array and the \( K \)-array, reusing words from the \( K \)-array as needed. For example, for the maximum length key (14 32-bit words), \( P_1 = P_1 \oplus K_1 \), \( P_2 = P_2 \oplus K_2 \), \ldots, \( P_{14} = P_{14} \oplus K_{14} \), \( P_{15} = P_{15} \oplus K_1 \), \ldots, \( P_{18} = P_{18} \oplus K_1 \).

3. Encrypt the 64-bit block of all zeros using the current \( P \)- and \( S \)-arrays; replace \( P_1 \) and \( P_2 \) with the output of the encryption.

4. Encrypt the output of step 3 using the current \( P \)- and \( S \)-arrays and replace \( P_3 \) and \( P_4 \) with the resulting ciphertext.

5. Continue this procedure to update all elements of \( P \) and then, in order, all elements of \( S \), using at each step the output of the continuously changing Blowfish algorithm.
Blowfish encryption

Plaintext (64 bits)

Ciphertext (64 bits)

Round 1

\[ P_1 \oplus F \oplus P \]

LE_0 \quad 32 \text{ bits} \quad 32 \text{ bits} \quad RE_0

Round 16

\[ P_1 \oplus F \oplus P \]

LE_{16} \quad 32 \text{ bits} \quad RE_{16}

\[ P_{18} \oplus P_{17} \]

LE_{17} \quad 32 \text{ bits} \quad RE_{17}
Blowfish decryption

Round 1

\[ LD_0 \oplus F \oplus P_{38} \]

\[ RD_0 \]

\[ LD_1 \]

\[ RD_1 \]

Round 16

\[ LD_{16} \oplus F \oplus P_3 \]

\[ RD_{16} \]

\[ LD_1 \]

\[ RD_{17} \]

Plaintext (64 bits)
The update process can be summarized as follows:

\[
P_1, P_2 = E_{P, S}[0]
\]
\[
P_3, P_4 = E_{P, S}[P_1 \| P_2]
\]
\[
\vdots
\]
\[
P_{17}, P_{18} = E_{P, S}[P_{15} \| P_{16}]
\]
\[
S_{1,0}, S_{1,1} = E_{P, S}[P_{17} \| P_{18}]
\]
\[
\vdots
\]
\[
S_{4,254}, S_{4,255} = E_{P, S}[S_{4,252} \| S_{4,253}]
\]

where \(E_{P, S}[Y]\) is the ciphertext produced by encrypting \(Y\) using Blowfish with the arrays \(S\) and \(P\).

A total of 521 executions of the Blowfish encryption algorithm are required to produce the final \(S\)- and \(P\)-arrays. Accordingly, Blowfish is not suitable for applications in which the secret key changes frequently. Further, for rapid execution, the \(S\)- and \(P\)-arrays can be stored rather than rederived from the key each time the algorithm is used. This requires over 4 KByte of memory.

Detail of a single Blowfish algorithm
Encryption and decryption

Blowfish uses two primitive operations:

- **Addition**: Addition of words, denoted by \(+\), is performed modulo \(2^{32}\).
- **Bitwise exclusive-OR**: The operation is denoted by \(\oplus\).

It is important that these two operations do not commute. This makes cryptanalysis more difficult.

The following two figures are illustrating the encryption and decryption algorithms. The plaintext is divided into two 32-bit halves \(LE_0\) and \(RE_0\). We use the variables \(LE_i\) and \(RE_i\) to refer to the left and right half of the data after round \(i\) has completed. The resulting ciphertext is contained in the two variables \(LE_{17}\) and \(RE_{17}\).

Decryption is easily derived from the encryption algorithm. In this case, the 64-bits of ciphertext are initially assigned to the two one-word variables \(LD_0\) and \(RD_0\). We use the variables \(LD_i\) and \(RD_i\) to refer to the left and right half of the data after round \(i\). As with most block ciphers, Blowfish decryption involves using the subkeys in reverse order. However, unlike most block ciphers, Blowfish decryption occurs in the same algorithmic direction as encryption, rather than reverse.

S-box design

One of the most intense areas of research in the field of symmetric block ciphers is that of S-box design. Here we mention some of the general principles. In essence, we would like any change to the input vector to an S-box to result in random-looking changes to the output. The relationship should be nonlinear and difficult to approximate with linear functions.

One obvious characteristic of the S-box is its size. An \(n \times m\) S-box has \(n\) input bits and \(m\) output bits. DES has 6x4 S-boxes. The Blowfish has 8x32 S-boxes. Larger S-boxes, by and large, are more resistant to differential and linear cryptanalysis. On the other hand, the larger the dimension \(n\), the larger the lookup table. Thus, for practical reasons, a limit of \(n\) equal to about 8 to 10 is usually imposed.

S-boxes are typically organized different from the manner used in DES. An \(n \times m\) S-box typically consists of \(2^n\) rows and of \(m\) bits each. The \(n\) bits of input select one of the rows of the S-box, and the \(m\) bits in that row are the output. For example, in an 8x32 S-box, if the input is 00001001, the output consists of the 8 bits in row 9 (if the first row is labeled row 0).