Project 1: CE 340 Cryptography and Network Security

Term: Spring 2021-22

Strict deadline: 11.04.2022, 17:00 O'clock

Defined by: Süleyman KONDAKCI

STEPS OF IMPLEMENTATION OF A SINGLE-ROUND ENCRYPTION SCHEME:

You will implement the algorithm given in Fig. 1.

1) Your code will read a plaintext from a file block-by-block with block size of 8 characters. Please note that plaintext file must contain at least 10 lines of text with each line minimum 25 characters.
2) A permutation code of length 8 is chosen to be **IP = 6 4 2 8 7 5 3 1**. The 8-character text blocks will be permutated using the initial permutation (**IP)** code.
3) Use either BINARY values of characters from **Table 1** to encode characters so that each character will be 8 bits of length.

Table 1: Character encoding

A **00000001**	**B** **00000010**	**C** **00000011**	**Ç** **00000100**	**D** **00000101**	**E** **00000110**
F **00000111**	**G** **00001000**	**Ğ** **00001001**	**H** **00001010**	**I** **00001011**	**İ** **00001100**
J **00001101**	**K** **00001110**	**L** **00001111**	**M** **00010000**	**N** **00010001**	**O** **00010010**
Ö **00010011**	**P** **00010100**	**R** **00010101**	**S** **00010110**	**Ş** **00010111**	**T** **00011000**
U **00011001**	**Ü** **00011010**	**V** **00011011**	**Y** **00011100**	**Z** **00011101**	**.** **00011110**
, **00011111**	**(** **00100000**	**)** **00100001**	**!** **00100010**	**;** **00100011**	**:** **00100100**
‘ **00100101**	**“** **00100110**	**-** **00100111**	**?** **00101000**	**$** **00101001**	**@** **00101010**
% **00101011**	**a** **100000001**	**b** **100000010**	**c** **100000011**	**ç** **100000100**	**d** **100000101**
e **100000110**	**f** **100000111**	**g** **1…**	**ğ**	**h**	**ı**
i	**j**	**K**	**l**	**m**	**n**
o	**ö**	**p**	**q**	**r**	**s**
ş	**t**	**u**	**ü**	**v**	**y**
z	**Q**	**W**	**q**	**w**	**…**

4) Preform a Shift-right-rotate operation with 4 positions on the encoded characters.
5) Get 2 characters at a time from the encoded block: Characters at odd positions are placed into the left nibble and characters with even positions are placed into the right nibble.
6) Choose a 16-bit (2 characters) key and convert them to 16 bits, and put the bits into a 4x4 matrix (table), see **Fig.1**.
7) Now use the columns 3,1,0, 2 to generate **K_2** and use columns 0,1,3,2 to generate **K_3.**
   a. Concatenate columns 3 and 1, that is $x = (3||1)$, and concatenate columns 0 and 2 that is $y = (0||2)$. Here, *x* and *y* will now be 8 bits each. Now perform *XOR* on *x* and *y* to generate **K_2**, i.e., **K_2** = *(x XOR y)*
   b. Concatenate columns 0 and 1, that is $w = (0||1)$, and concatenate columns 2 and 3, that is $z = (2||3)$, and. Now perform *XOR* on *x* and *y* to generate, i.e., **K_3** = *(w XOR z).*
   c. Use **K_2** in function **SLR(3, K_2)** to generate **K_4**. Likewise, Use **K_3** in function **SRR(3, K_3)** to generate **K_5**.
8) Apply XOR together with **K_2** and **K_3** to encrypt the first part (stage 6, 8). Swap XOR’d partitions so that left becomes right and right becomes left.

9) Now, generate two new sub-keys (**K**$_4$ and **K**$_5$) by using **K**$_2$ and **K**$_3$ and the two rotate functions, see **Fig.1 (part 10).**
10) The swapped partitions are XOR'd with the new sub-keys.
11) Swap the result again and merge the results of swapping.
12) Finally, a cipher block is obtained by passing the resulting 8-character block through the reverse permutation **IP**$^{-1}$.
13) Encrypted blocks will be saved in a ciphertext file.
14) Finally, verify your encryption by decrypting the ciphertext file.

What to deliver:

1) **Source code:** You can choose to implement the algorithm in either of these languages **C, C++, Java, or Python**
2) **Screenshots of a sample run**
3) **Plaintext file used for the test**
4) **Ciphertext file obtained from the test**

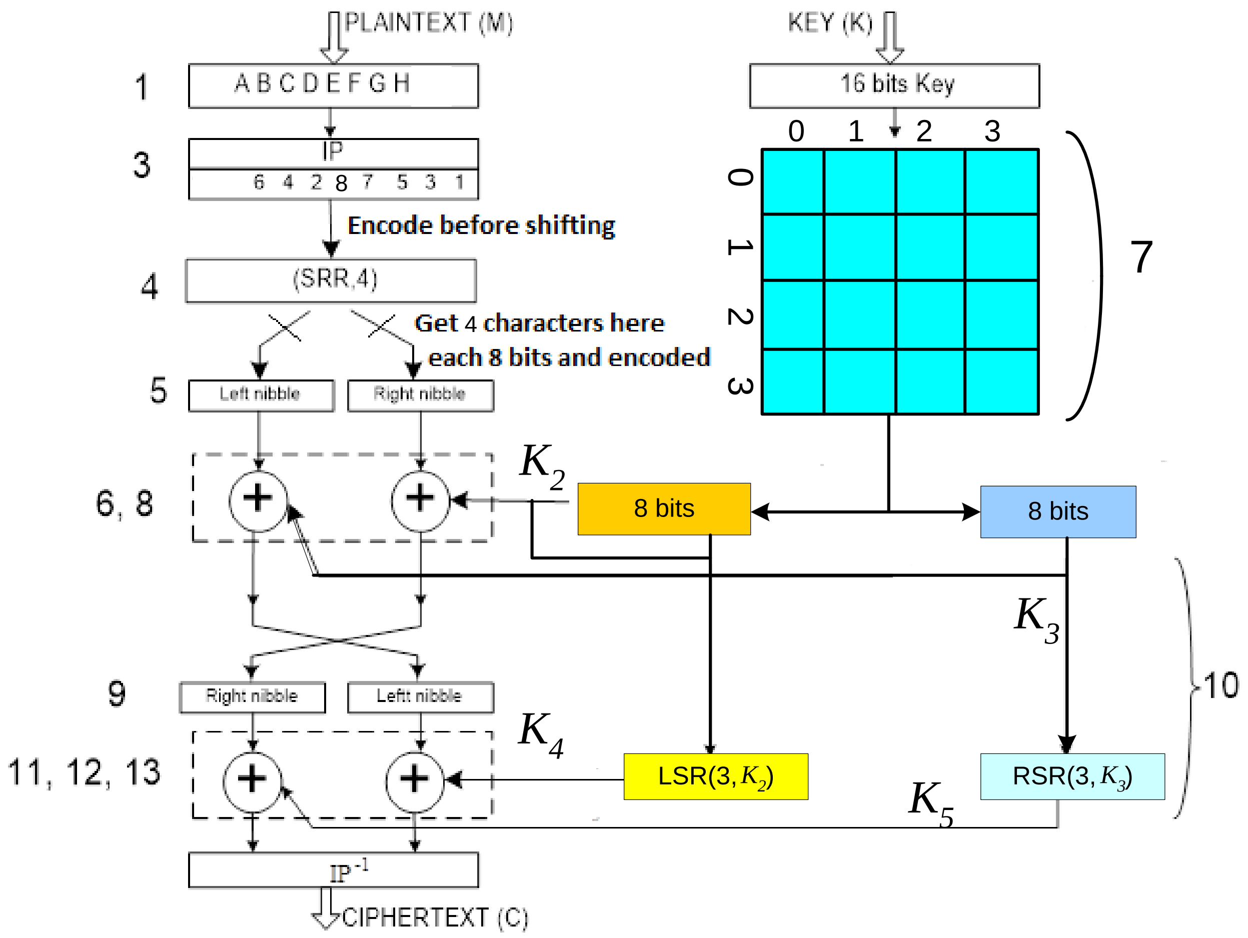

Fig.1 Block Diagram of the encryption process.