CSE/IT-441 Cryptography & Applications, CS589 Cryptography – Fall 2017

Department of Computer Science, New Mexico Tech
Lecturer: Dongwan Shin

HOMEWORK 2 - Assigned: 10/2, **Due: 10/11** (**by midnight**)

Please **type your answers** and **submit** them along with your source code to Canvas.

Problem 1. [**5 points**] Consider a Feistel cipher composed of 16 rounds with block length 128 bits and key length 128 bits. Suppose that, for a given k, the key scheduling algorithm determines values for the first 8 round keys, k_1, k_2, ... k_8, and then sets,

$$k_9 = k_8,\ k_{10} = k_7,\ k_{11} = k_6,\ ...,\ k_{16} = k_1$$

Suppose you have a ciphertext c. Explain how, with access to an encryption oracle, you can decrypt c and determine m using just a single oracle query. This shows that such a cipher is vulnerable to a chosen plaintext attack. (An encryption oracle can be thought of as a device that, when given a plaintext, returns the corresponding ciphertext. The internal details of the device are not known to you and you cannot break open the device. You can only gain information from the oracle by making queries to it and observing its responses.)

Problem 2. [**18 points**] This problem provides a numerical example of encryption using a one-round version of DES. We start with the same bit pattern for the key and the plaintext, namely,

`In hexadecimal(h):`
F E D C B A 9 8 7 6 5 4 3 2 1 0

`In binary (b):`
1111 1110 1101 1100 1011 1010 1001 1000
0111 0110 0101 0100 0011 0010 0001 0000

a. Derive K_1, the first-round key (**b** & **h**).
b. Drive L_0, R_0 (**b**).
c. Expand R_0 to get $E[R_0]$ (**b**).[1]
d. Calculate $A = E[R_0] \oplus K_1$ (**b**).
e. Apply S-box substitutions on A and show 8 results from corresponding 8 S-boxes (**b**).
f. Concatenate the results of (e) to get a 32-bit result, B (**b**).
g. Apply the permutation to get P(B) (**b**).
h. Calculate $R_1 = P(B) \oplus L_0$ (**b**).
i. Write down the ciphertext (**b** & **h**).

Problem 3. [**7 points**] Suppose the DES F function mapped every 32-bit input R, regardless of the value of the input K (round key), to

1. 32-bit string of ones for odd number rounds and 32-bit string of zeroes for even number rounds

Using the following properties of the XOR operation,

[1] Use the tables in your textbook for solving c, e, and g.

$$\begin{gathered}(A \oplus B) \oplus C = A \oplus (B \oplus C)\\ A \oplus A = 0\\ A \oplus 0 = A\\ A \oplus 1 = \overline{A}\end{gathered}$$

, where 0 is an n-bit string of zeros and 1 is an n-bit string of ones.

a. What function would DES then compute?
b. What would the decryption look like?

Problem 4. [10 points] Let $\bar{s}$ denote the bitwise complement of a binary string s. (For example, $\overline{0101} = 1010$.) DES has the property that

$$DES_K(x) = \overline{DES_{\bar{K}}(\bar{x})}$$

for every key $K \in \{0,1\}^{56}$ and every input $x \in \{0,1\}^{64}$. This is called the key-complementation property. Under the chosen plaintext attack, it has been known that the key-complementation helps reduce the number of DES computations in an attempt for the exhaustive key search attack by a factor of two. Show how.

Problem 5. [10 points] Suppose that we use a block cipher to encrypt according to the rule

$$\begin{gathered}C_0 = IV \oplus E(P_0, K)\\ C_1 = C_0 \oplus E(P_1, K)\\ C_2 = C_1 \oplus E(P_2, K)\\ \ldots.\end{gathered}$$

What is the corresponding decryption rule? Are there any security advantages or disadvantages to this mode compared to CBC mode?

Programming Lab 1. [50 points] The document attached has a detailed description on a basic Substitution-Permutation Network (SPN) cipher. Implement both encryption and decryption functions of the SPN cipher in C. Note that instead of generating subkeys independently and unrelatedly, the key scheduling algorithm for your cipher uses left circular shift as follows:

$$K_i = (K_{i-1} << 4), \text{ where } K_0 = K$$

Your SPN cipher should operate like this.

/*************************
spn [-e] [-d] [text] [key]

OPTION
-e SPN Cipher Encryption
-d SPN Cipher Decryption

INPUT
key 16-bit key
text 16-bit plaintext or ciphertext

*************************/

2. A Basic Substitution-Permutation Network Cipher

The cipher that we shall use to present the concepts is a basic Substitution-Permutation Network (SPN). We will focus our discussion on a cipher, illustrated in Figure 1, that takes a 16-bit input block and processes the block by repeating the basic operations of a round four times. Each round consists of (1) substitution, (2) a transposition of the bits (i.e., permutation of the bit positions), and (3) key mixing. This basic structure was presented by Feistel back in 1973 [15] and these basic operations are similar to what is found in DES and many other modern ciphers, including Rijndael. So although, we are considering a somewhat simplified structure, an analysis of the attack of such a cipher presents valuable insight into the security of larger, more practical constructions.

2.1 Substitution

In our cipher, we break the 16-bit data block into four 4-bit sub-blocks. Each sub-block forms an input to a 4×4 S-box (a substitution with 4 input and 4 output bits), which can be easily implemented with a table lookup of sixteen 4-bit values, indexed by the integer represented by the 4 input bits. The most fundamental property of an S-box is that it is a nonlinear mapping, i.e., the output bits cannot be represented as a linear operation on the input bits.

For our cipher, we shall use the same nonlinear mapping for all S-boxes. (In DES all the S-boxes in a round are different, while all rounds use the same set of S-boxes.) The attacks of linear and differential cryptanalysis apply equally to whether there is one mapping or all S-boxes are different mappings. The mapping chosen for our cipher, given in Table 1, is chosen from the S-boxes of DES. (It is the first row of the first S-box.) In the table, the most significant bit of the hexadecimal notation represents the leftmost bit of the S-box in Figure 1.

input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Table 1. S-box Representation (in hexadecimal)

2.2 Permutation

The permutation portion of a round is simply the tranposition of the bits or the permutation of the bit positions. The permutation of Figure 1 is given in Table 2 (where the numbers represent bit positions in the block, with 1 being the leftmost bit and 16 being the rightmost bit) and can be simply described as: the output *i* of S-box *j* is connected to input *j* of S-box *i*. Note that there would be no purpose for a permutation in the last round and, hence, our cipher does not have one.

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
output	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Table 2. Permutation

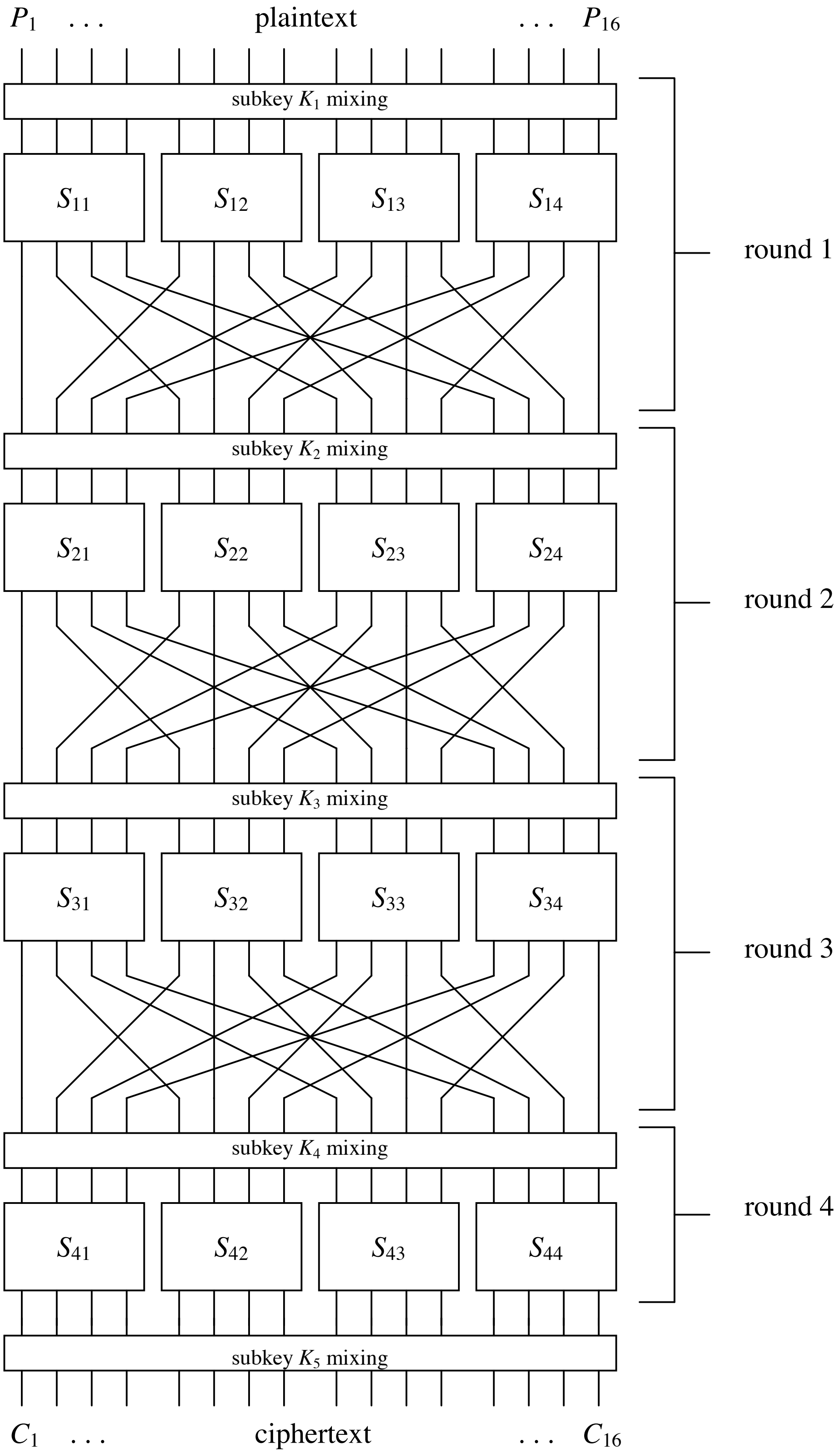

Figure 1. Basic Substitution-Permutation Network (SPN) Cipher

2.3 Key Mixing

To achieve the key mixing, we use a simple bit-wise exclusive-OR between the key bits associated with a round (referred to as a subkey) and the data block input to a round. As well, a subkey is applied following the last round, ensuring that the last layer of substitution cannot be easily ignored by a cryptanalyst that simply works backward through the last round's substitution. Normally, in a cipher, the subkey for a round is derived from the cipher's master key through a process known as the key schedule. In our cipher, we shall assume that all bits of the subkeys are independently generated and unrelated.

2.4 Decryption

In order to decrypt, data is essentially passed backwards through the network. Hence, decryption is also of the form of an SPN as illustrated in Figure 1. However, the mappings used in the S-boxes of the decryption network are the inverse of the mappings in the encryption network (i.e., input becomes output, output becomes input). This implies that in order for an SPN to allow for decryption, all S-boxes must be bijective, that is, a one-to-one mapping with the same number input and output bits. As well, in order for the network to properly decrypt, the subkeys are applied in reverse order and the bits of the subkeys must be moved around according to the permutation, if the SPN is to look similar to Figure 1. Note also that the lack of the permutation after the last round ensures that the decryption network can be the same structure as the encryption network. (If there was a permutation after the last substitution layer in the encryption, the decryption would require a permutation before the first layer of substitution.)