COMP90043: Cryptography and security:

Week 4 Workshop Activity

Activity: Working of Fiestel's algorithm for encryption and decryption.

Both encryption and decryption iteratively runs sixteen rounds of the same inner algorithm. The only difference is that the decryption rounds use a reversed key order. Verify that the decryption is the inverse operation of the encryption.

Plaintext: $\mathbf{LE_0 \| RE_0}$

Output of the 16th round (Encryption): $\mathbf{LE_{16} \| RE_{16}}$

$LE_{16} = RE_{15}$ $\quad\quad$ $RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$

$LD_1 = RD_0 = LE_{16} = RE_{15}$

$RD_1 = LD_0 \oplus F(RD_0, K_{16})$

$= RE_{16} \oplus F(RE_{15}, K_{16})$

$= LE_{15} \oplus F(RE_{15}, K_{16}) \oplus F(RE_{15}, K_{16})$

$\mathbf{LD_1 = RE_{15}}$ **and** $\mathbf{RD_1 = LE_{15}}$

Output of 1st round of decryption : $\mathbf{RE_{15} \| LE_{15}}$

$LE_i = RE_{i-1}$ $\quad\quad$ $RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$

$RE_{i-1} = LE_i$

$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i)$

Output of the 16th round (Decryption): $\mathbf{RE_0 \| LE_0}$

After the final32-bit swap: $\mathbf{LE_0 \| RE_0}$ **=** Plaintext

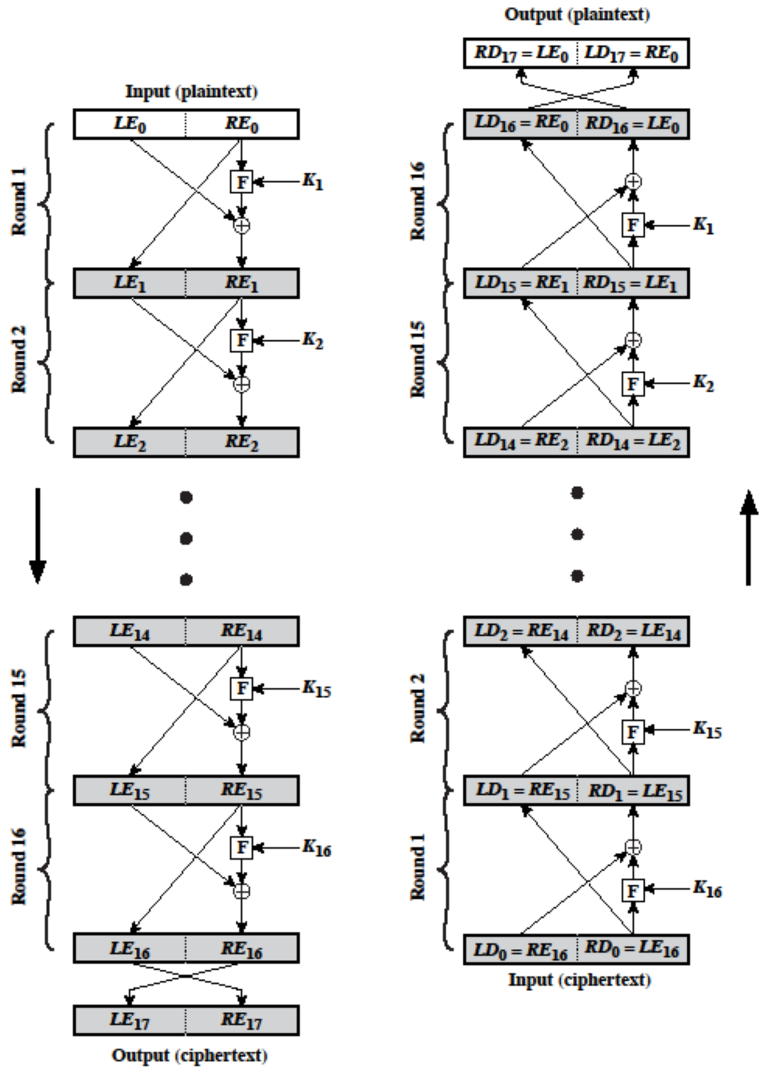

Figure 3.3 Feistel Encryption and Decryption (16 rounds)

- **6.8:** Only the plaintext unit corresponding to the ciphertext character is affected. In OFB method, the bit errors in transmission do not propagate. For example, if a bit error occurs in C_1, only the recovered value of P_1 is affected; subsequent plaintext units are not corrupted.

- **6.9:** Let message M1 have plaintext blocks $P1_j$ and ciphertext blocks $C1_j$. Similarly for message M2. If the same IV and key are used in Ofb mode for both messages, then both messages have the same output blocks O_j. Suppose an attacker can observe the ciphertext blocks for M1 and M2 and that the attacker knows the exact contects of $P1_q$. Then,

$$\begin{aligned} C1_q &= P1_q \oplus O_q && \text{by definition of OFB} \\ C1_q \oplus P1_q &= P1_q \oplus O_q \oplus P1_q && \text{add to both sides} \\ O_q \oplus P1_q \oplus P1_q &= C1_q \oplus P1_q && \text{rearrange} \\ O_q &= C1_q \oplus P1_q && \text{cancel terms} \end{aligned}$$

$$\begin{aligned} C2_q &= P2_q \oplus O_q && \text{by definition of OFB} \\ C2_q \oplus O_q &= P2_q \oplus O_q \oplus O_q && \text{add to both sides} \\ P2_q &= C2_q \oplus O_q && \text{add to both sides} \end{aligned}$$

- **6.5:** In some modes, the plaintext does not pass through the encryption function, but is XORed with the output of the encryption function. The math works out that for decryption in these cases, the encryption function must also be used.

- **6.1.a:** If the IVs are kept secret, the 3-loop case has more bits to be determined and is therefore more secure than 1-loop for brute force attacks.

- **6.1.b:** For software implementations, the performance is equivalent for most measurements. One-loop has two fewer XORs per block. Three-loop might benefit from the ability to do a large set of blocks with a single key before switching. The performance difference from choice of mode can be expected to be smaller than the differences induced by normal variation in programming style.

  For hardware implementations, three-loop is three times faster than one-loop, because of pipelining. That is: Let P_i be the stream of input plaintext blocks, X_i the output of the first DES, Y_i the output of the second DES and C_i the output of the final DES and therefore the whole system's ciphertext.

  In the 1-loop case, we have:

$$\begin{aligned} X_i &= DES(XOR(P_i, C_{i-1})) \\ Y_i &= DES(X_i) \\ C_i &= DES(Y_i) \end{aligned}$$

  where C_0 is the single IV.

  If P_1 is presented at $t = 0$ (where time is measured in units of DES operations), X_1 will be available at $t = 1$, Y_1 at $t = 2$ and C_1 at $t = 3$. At $t = 1$, the first DES is free to do more work, but that work will be: $X_2 = DES(XOR(P_2, C_1))$ but C_1 is not available until $t = 3$, therefore X_2 can not be available until $t = 4$, Y_2 at $t = 5$ and C_2 at $t = 6$.

  In the 3-loop case, we have:

$$\begin{aligned} X_i &= DES(XOR(P_i, X_{i-1})) \\ Y_i &= DES(XOR(X_i, Y_{i-1})) \\ C_i &= DES(XOR(Y_i, C_{i-1})) \end{aligned}$$

  where X_0, Y_0 and C_0 are three independent IVs.

  If P_1 is presented at $t = 0$, X_1 is available at $t = 1$. Both X_2 and Y_1 are available at $t = 4$. X_3, Y_2 and C_1 are available at $t = 3$. X_4, Y_3 and C_2 are available at $t = 4$. Therefore, a new ciphertext block is produced every 1 tick, as opposed to every 3 ticks in the single-loop case. This gives the three-loop construct a throughput three times greater than one-loop construct.

- **6.2:** Instead of $CBC[CBC(CBC(X))]$, use $ECB[CBC(CBC(X))]$. The final IV was not needed for security. The lack of feedback loop prevents the chosen-ciphertext differential cryptanalysis attack. The extra IVs still become part of a key to be determined during any known plaintext attack.

COMP90043: Cryptography and security:

Week 4 Workshop Activity

1) What is reversible mapping? What is irreversible mapping? Think about why Fiestel’s algorithm works for any function F, even for the irreversible ones.

In reversible mapping every value in the output has one to one relationship with the input values. In case of irreversible mapping one output can be associated with multiple inputs.

Operations of Fiestel algorithm do not depend on the round function F. In the decryption phase the round function in XORed with itself and as we know that $A \oplus A = 0$, the algorithm works for any round function F. However, a good and complex F function generally means greater resistance to cryptanalysis

2) What is the difference between a block cipher and a stream cipher?

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.

3) What is a product cipher?

In a product cipher, two or more basic ciphers are performed in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.

4) What is the difference between diffusion and confusion? How diffusion and confusion is achieved in Fiestel’s encryption algorithm?

In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. Confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by the use of a complex substitution algorithm.

COMP90043: Cryptography and security:

Week 4 Workshop Activity

In Fiestel's cipher, diffusion is achieved by repeatedly interchanging the two halves of the data followed by applying Exclusive OR on the left part and the output of the round function F. The effect is that bits from different positions in the original plaintext contribute to a single bit of ciphertext. Confusion is achieved by using the round function with sub-keys in each round.

5) What parameters and design choices determine the actual algorithm of a Feistel Cipher?

Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed.

Key size: Larger key size means greater security but may decrease encryption/decryption speed.

Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security.

Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

Round function: Again, greater complexity generally means greater resistance to cryptanalysis.

Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.

Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalysis, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength.

Homework:

The following are a list of questions for students to attempt at home to get a better grasp of the concepts discussed during the workshop.

1. Complete any questions that were not completed during the workshop.
2. What is avalanche effect? Why it is desired in encryption algorithms?
3. Write the block diagram for DES decryption algorithm.