Universidad Carlos III de Madrid

Implementing a simple cipher

IT Security – Lab work
Year 2010/2011

José María de Fuentes / Iván Rivera

Work description

Encryption is one of the most common cryptographic mechanisms. Encryption mechanisms can be classified using different criteria:

- **Key type:** there are symetric ciphers (DES, AES, etc.) and asymmetric ones (RSA).
- **Mode of operation:** there are block ciphers (DES, etc.) and stream ones (RC4, etc).

In this assignment, students will have to implement a simple symmetric block cipher. For this purpose, a reduced versión of the Feistel scheme (Figure 1) will be implemented. Recall that many ciphers are based on this generic scheme.

This document explains the internals of a Feistel network. The work to be done by the students is also explained, along with the evaluation criteria and hand-in specific requirements.

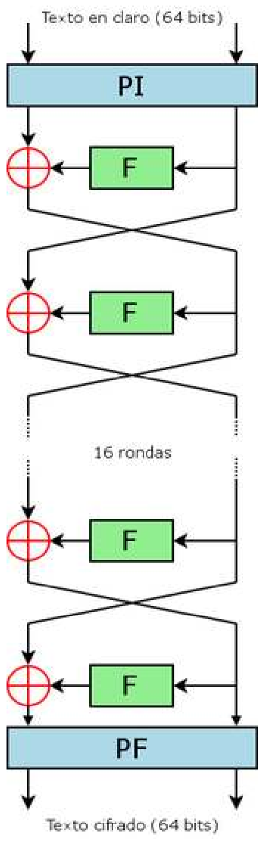

Figure 1. Feistel scheme

Encrypting in a Feistel network

For ciphering using a Feistel network, the message is divided in two blocks of the same length, called L_0 (left block) and R_0 (right block). Afterwards, a number ‘n’ of rounds are executed. The operations performed in each round are:

$L_i = R_{i-1}$

$R_i = L_{i-1}$ XOR $F(R_{i-1}, K_i)$

Where

- i denotes the round number (being i = 1 for the first round),
- F refers to a function that will be described throughout this document, and
- K_i denotes the subkey to be used in round i, which is derived from the initial encryption key. This derivation process is described later in the document.

The encrypted text is the concatenation of both L_n and R_n.

Descrypting in a Feistel network

To decipher a ciphertext using a Feistel network, the same rounds used during the encryption process have to be executed. The operations to be performed in those rounds are:

$R_{i-1} = L_i$

$L_{i-1} = R_i$ XOR $F(L_i, K_i)$

The decrypted text is the concatenation of both L_0 and R_0.

If the student analyzes carefully the expressions of both encryption and decryption, he/she will discover an intrinsic advantage of Feistel networks**: Feistel networks are reversible by definition – decryption only requires to perform the same operations used in the encryption process, but applying the keys in reverse order.**

What does the F function look like?

The F function (Figure 2) is applied to **half a block** (that is, it is applied to L_i or R_i).

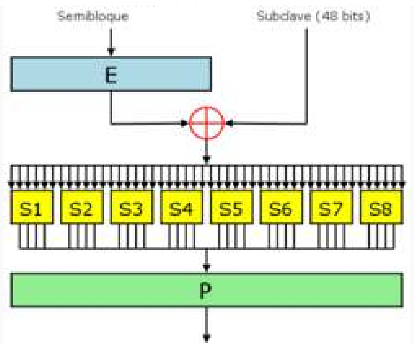

Figure 2. Main phases of the F function

The four phases can be described as follows:

1. *Expansion (E)* — the block is expanded to a greater number of bytes, essentially by duplication of some bytes.
2. *Mix* — the result of the expansion is then combined with the *current round subkey* using a XOR function. In each round a different subkey is employed. Subkeys are derived from the initial key through a key generation process.
3. *Substitution* (S1…S8) — after the mix operation, some bytes are substituted according to a given transformation. This is one of the strongest security points of Feistel networks and other ciphers based on this scheme, like the DES cipher.
4. *Permutation (P)* — finally, the resulting bits are re-ordered according to a fixed permutation.

Work to be done by students

Students must implement a **SIMPLIFIED version** of the Feistel networks previously described. These simplifications, which **only apply to the mandatory version of the work**, are the following ones:

- The block length will be fixed, and it will be 8 BYTES.
- The password will always be 4 CHARACTERS (bytes) long.
- The text to be encrypted will always be a multiple of the block length.
- In the F function:
    - No expansion will be required (in principle).
    - The substitution phase will consist on adding 9 to each byte. NO S-BOXes will be implemented.
    - There will not be permutation phase.

- The subkey generation algorithm will consist of left-shifting groups of 4 bits, as many as indicated by the round number (as an example, for round 2, a circular left-shifting of 2*4 = 8 bits = 1 byte will be performed).
- The maximum number of rounds will be 8.

Teachers will bring a Java base file which will contain some implemented methods. This file will also contain others to be implemented (mandatory) and others that can be implemented or improved from its basic version (see Evaluation criteria). The following table describes each method:

Method header	Explanation	Implemented	Mandatory	Optional
`static byte [] cifrarECB/descifrarECB (byte [] bytes)`	Encryption/Decryption in ECB mode.		X	
`byte [] ejecutarRonda(byte [] r, byte[] l, byte [] subclave)`	Calls other methods in the correct order, in order to implement a round (as described in this document).		X	
`byte [] generarSubclave (byte [] clave, int vez)`	Shifts (circular shift) to the left groups of 4 bits, as many groups as it is indicated by parameter 'vez'		X	
`byte [] ejecutarFuncionF (byte [] bloque, byte [] subclave)`	It outputs the result of applying the F function to a block using a subkey. This F function is implemented taking into account the simplifications previously described.		X	
`byte [] ejecutarPermutacionF (byte [] bloque)`	It outputs the result of applying a fixed permutation of the input value.			X
`static byte [] sustituir (byte [] bloque)`	It outputs the result of the substitution phase of the F function, taking into account the described simplifications.		X	X
`static byte [] expandirBloque(byte [] bloque)`	It outputs the result of the expansion phase of the F function.			X
`static byte [] cifrarCBC/cifrarOFB/… (byte [] bytes)` `static byte [] descifrarCBC/descifrarOFB/… (byte [] bytes)`	Implementation of different encryption/decryption modes.			X
`byte [] ejecutarXOR (byte [] a, byte [] b)`	Byte-to-byte XOR operation.	X		
`byte [] obtenerBloque (byte [] texto, int inicio, int fin)`	It outputs the array of bytes of the block defined by the start position indicated by 'inicio' and the end position indicated by 'fin'	X		
`Byte [] ficheroArray (String nombreFichero)`	Dumps the contents of the file on an array of bytes	X		

Method header	Explanation	Implemented	Mandatory	Optional
`void arrayFichero (byte [] bytes)`	Dumps the bytes stored in parameter 'bytes' to the output file	X		
`byte [] mitadIzquierdaBloque (byte[] bloque)`	It outputs the left half of the given block.	X		
`byte [] mitadDerechaBloque (byte[] bloque)`	It outputs the right half of the given block.	X		

IMPORTANT: It is not permitted to modify any header of the methods to be implemented. Any modification on them (headers) will mean that the delivery will not be considered and thus the practice will not be evaluated.

Practice work hand in

The result of this practice work must be the Java file "Cifrador.java" with the corresponding methods implemented, as it has been described in this document.

Hand in will be made through Aula Global 2. The delivery due date is the day in which the reduced group has a class session during the week of March 21st 2011.

Evaluation criteria

The evaluation will be based on the following criteria:

Criterion	Value (% over the total value)
Encryption of a file F. Decryption of the result in order to get the original file F.	50%
Cryptographic improvements over the expansion, substitution and permutation phases.	Up to +40%
Implementing at least two other different encryption modes (CBC, OFB, etc.)	+25%
Dealing with customizable block length or password length. Offering support to files which length is not multiple of the block length.	+10%
Execution time measurements using input files with different length.	+5%

Remarks:

- **Criteria 2, 3, 4 and 5 will only be considered if criterion 1 is fulfilled.**
- If the student implements the cryptographic improvements of criterion 2, a short document (**up to** 2 sheets) must be made in order to describe which the claimed improvements are. In that case, a single ZIP file will be delivered via Aula Global 2, containing both the document and the file "Cifrador. java"
- The delivered file ("Cifrador. java") must compile without any modification on the source code. In case a file does not compile, it will not be considered and thus not evaluated.