DES ENCRYPTION

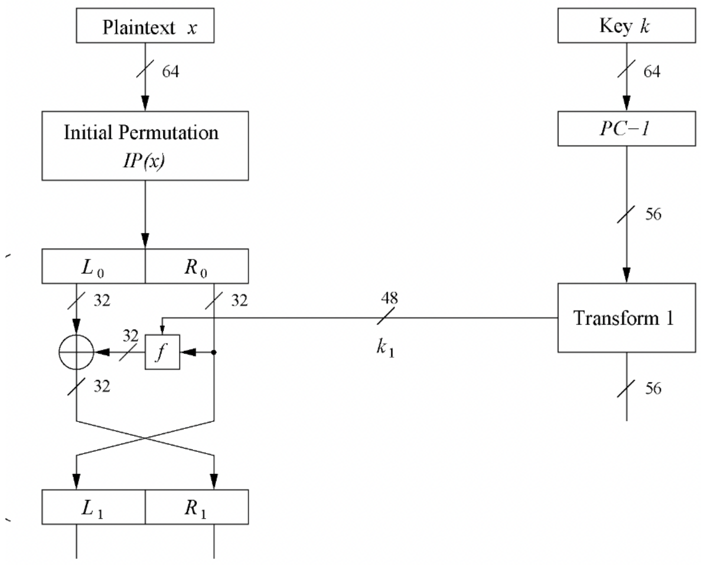
Data Encryption Standard (DES) is an encryption algorithm that encrypts blocks of size 64-bit. It is the most popular encryption standard for the past 30 years and is still widely used. However, nowadays 64-bit DES is considered unsafe due to its small key size (48 bit). For this reason, 3DES is used today, which is basically similar to DES, but it runs 3 times on one block and it uses a 128-bit key.

As said, DES encrypts blocks of size 64-bit, and it uses a 48-bit key. However, the initial key is 64-bit, but 8 bits are not used (parity bits) and further 8 bits get removed with key permutation. Each block goes through 16 identical rounds, but each round uses a different key, that is derived from the original key.

DES is a symmetrical encryption algorithm, which means that it uses the same key for encryption and decryption, and to decrypt a message we just perform operations in inverse.

DES algorithm works as follows:

- Plaintext of 64 bits goes through initial permutation, where bits are swapped based on the initial permutation table.
- Then 64-bit block is split into 32-bit halves L_i and R_i
- R_i is fed into the f function, the output which is then XORed with L_i
- The left and right halves are swapped
- Therefore $L_i = R_{i-1}$; $R_i = L_{i-1}$ XOR $f(R_{i-1}, k_i)$ where k is the key
- L and R are swapped after 16 rounds again, followed by the final permutation

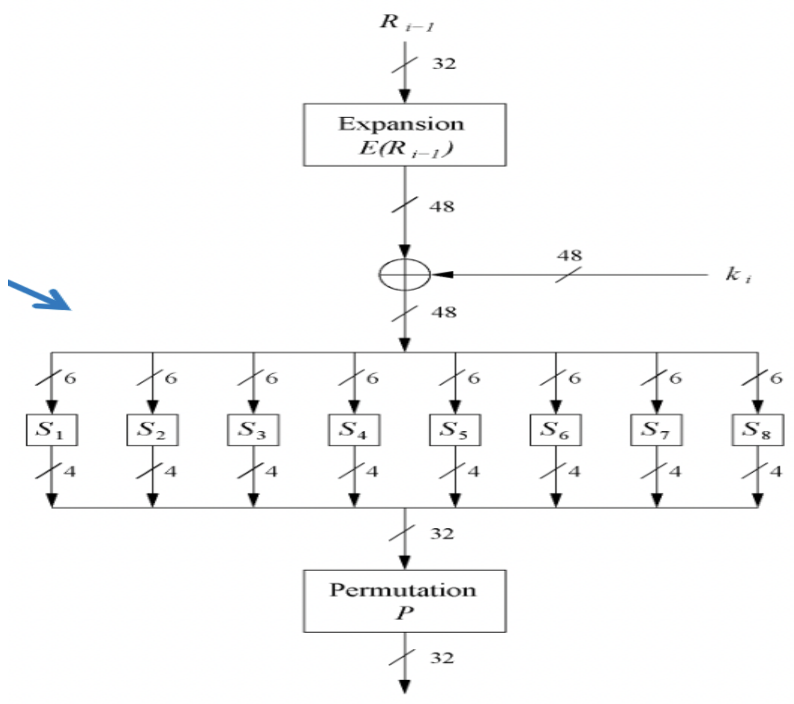


The f-Function

f-Function inputs are R_{i-1} and K_i

4 steps:

- Expansion
- XOR with round key
- S-Box substitution
- Permutation



From here the following classes are created and implemented:

InputHandler, Split, InitialPermutation, Round, RoundManager, FFunction, SBox (SBoxOne, SBoxTwo, … , SBoxEight), Key, Xor, OutputHandler, and Manager (with the main method)

Firstly, InputHandler has interface IInputHandler which specifies the method String getBinary(). InputHandler class implements this interface, reads the file, converts it to binary representation, and returns a string that represents the file in binary.

The Split class has interface ISplit with the method Vector<String> getBlocks(). Split implements this interface, and splits the file into 64 bit blocks, and returns the Vector of Strings that contain 64-bit blocks.

InitialPermutation class has interface IInitalPermutation with the method Vector<String> getPermutedBlocks(). Since initial permutation works on blocks, this class inherits from Split. It just adds a private variable Vector<String> permutedBlocks and implements the method from the interface.

Xor class has interface IXor with method String xor(char a, char b). This class implements the interface and it is a simple class that returns the calculation of XOR between two bits.

ISBox interface specifies method String get(String block). This interface is implemented by class SBox, and SBoxOne through SBoxEight. SBox class uses Directory Design Pattern to manage SBoxes.

Class Key generates a random 64-bit key, which gets reduces to 48-bit key, and from this key 16 subkeys are generated. Since for each 64-bit block the same 16 keys are used in the exact same order, this class uses Singleton Design Pattern to ensure that only one instance of Key object exists, which manages the keys.

FFunction class has an interface IFFunction which specifies the method String get(String block). FFunction implements this interface and it uses composition to compose the Key object, XOR object, and SBox object.

Class Round has an interface IRound which has the method String get(String left, String right). Round implements this interface and it uses composition to compose the XOR object and FFunction object.

RoundManger class has interface IRoundManager which specifies method Vector<String> getEncryptedBlocks(). RoundManager composes the Round object, FinalPermutation object, InputHandler object, and InitialPermutation object. This class manages 16 rounds, and ensures that each block of input goes through 16 rounds, and final permutation to get encrypted blocks.

OutputHandler has interface IOutputHandler which specifies method void write(Vector<String> encryptedBlocks). Simply this class writes to an output file the encrypted blocks.

Finally, the Manager class contains the main method.

Team members: Hanadi Jusufovic