SWE 314 Assignment

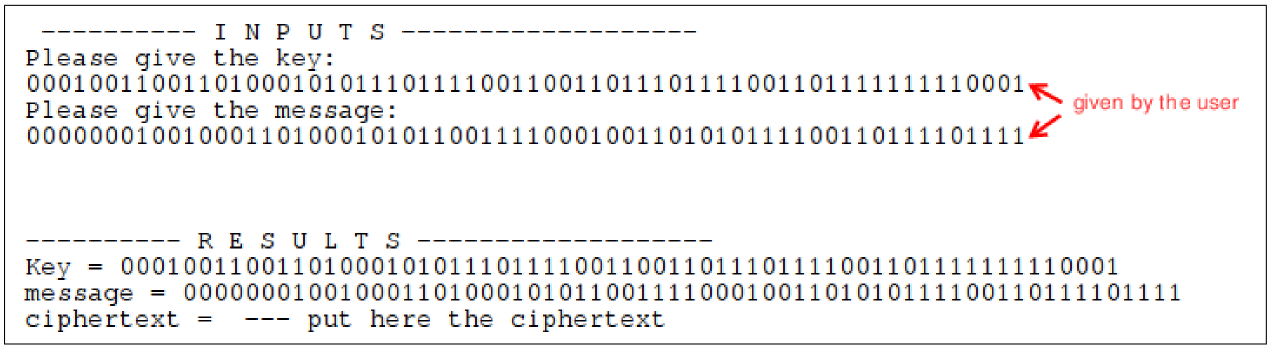
Dr. Mohsen Denguir

Date: October 7, 2021
Deadline: October 24, 2021

1 Description

You are asked to write a program that implement the Data Encryption Standard (DES) algorithm in Java.

As shown in Figure 1, the program must ask for a key and a (plaintext) message, and then display the ciphertext.



```
 ---------- I N P U T S --------------------
Please give the key:
0001001100110100010101110111100110011011101111001101111111110001
Please give the message:
0000000100100011010001010110011110001001101010111100110111101111



---------- R E S U L T S --------------------
Key = 0001001100110100010101110111100110011011101111001101111111110001
message = 0000000100100011010001010110011110001001101010111100110111101111
ciphertext =  --- put here the ciphertext
```

Figure 1: Program execution

The details of the algorithm were seen during the lectures.

The DES encryption process is shown in Figure 2. It applies an initial permutaion to the 64-bit input message. This is followed by sixteen rounds and a final permutation.

The DES encryption process uses sixteen keys, $K_1, ...K_{16}$, one for each round. The process used for the generation of these keys is shown in Figure 3. It takes the 64-bit key given by the user as input and generates sixteen 48-bit keys.

At each of the sixteen rounds, the DES enryption process evaluates a function F that has two arguments: the right half of the current round data (32 bits) and the 48-bit key corresponding to the current round. The process used to evaluate the function F is shown in Figure 4

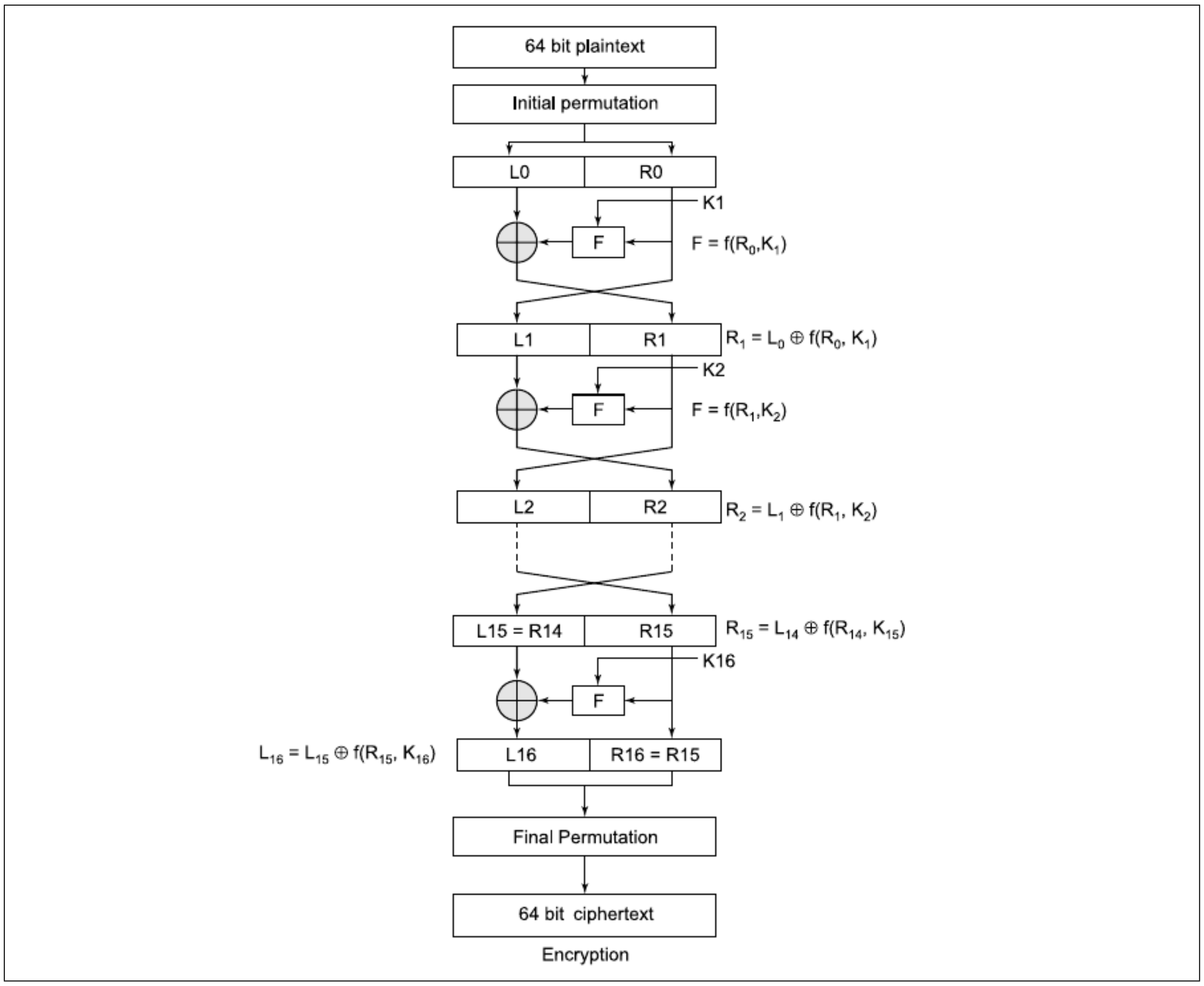


Figure 2: DES encryption process

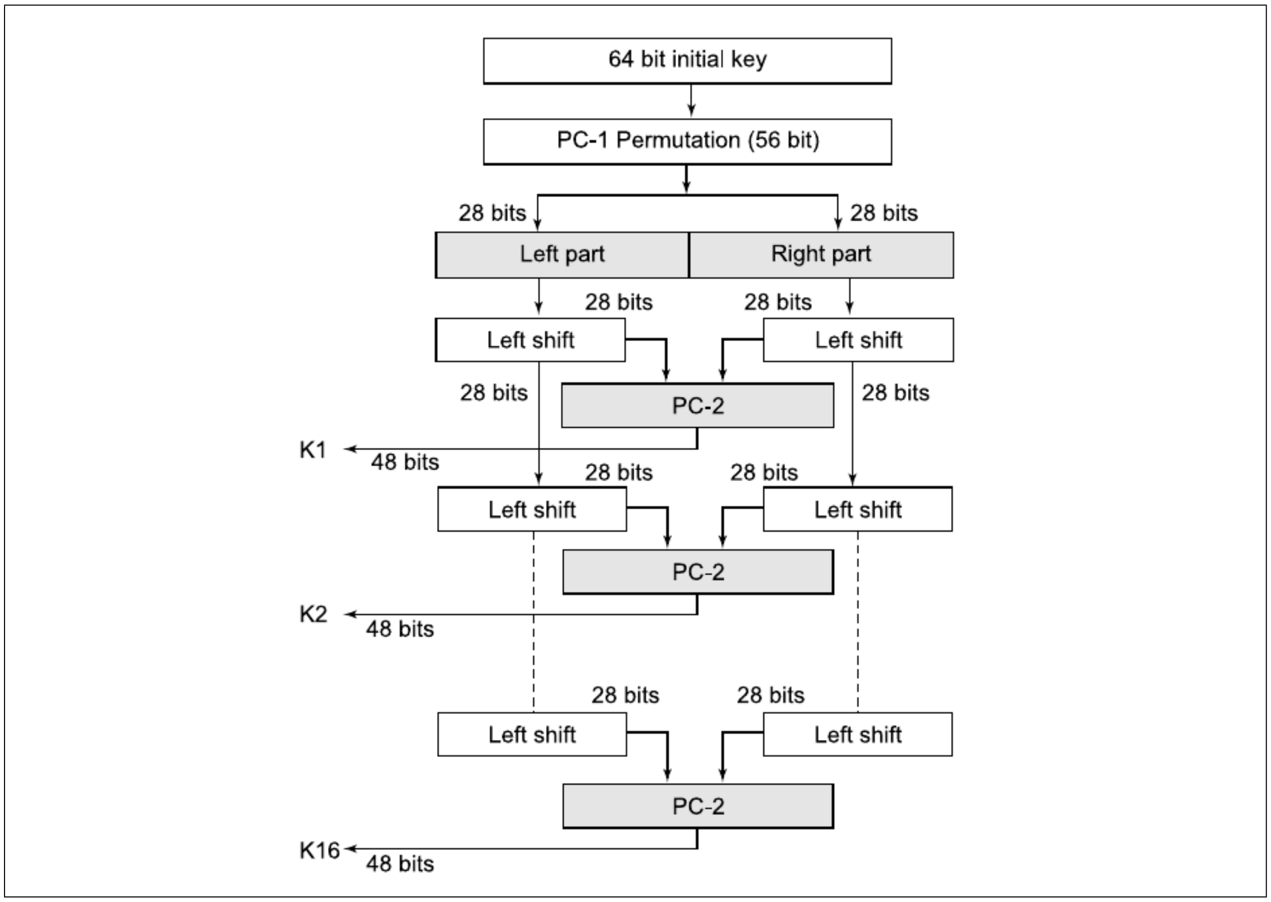


Figure 3: DES keys generation

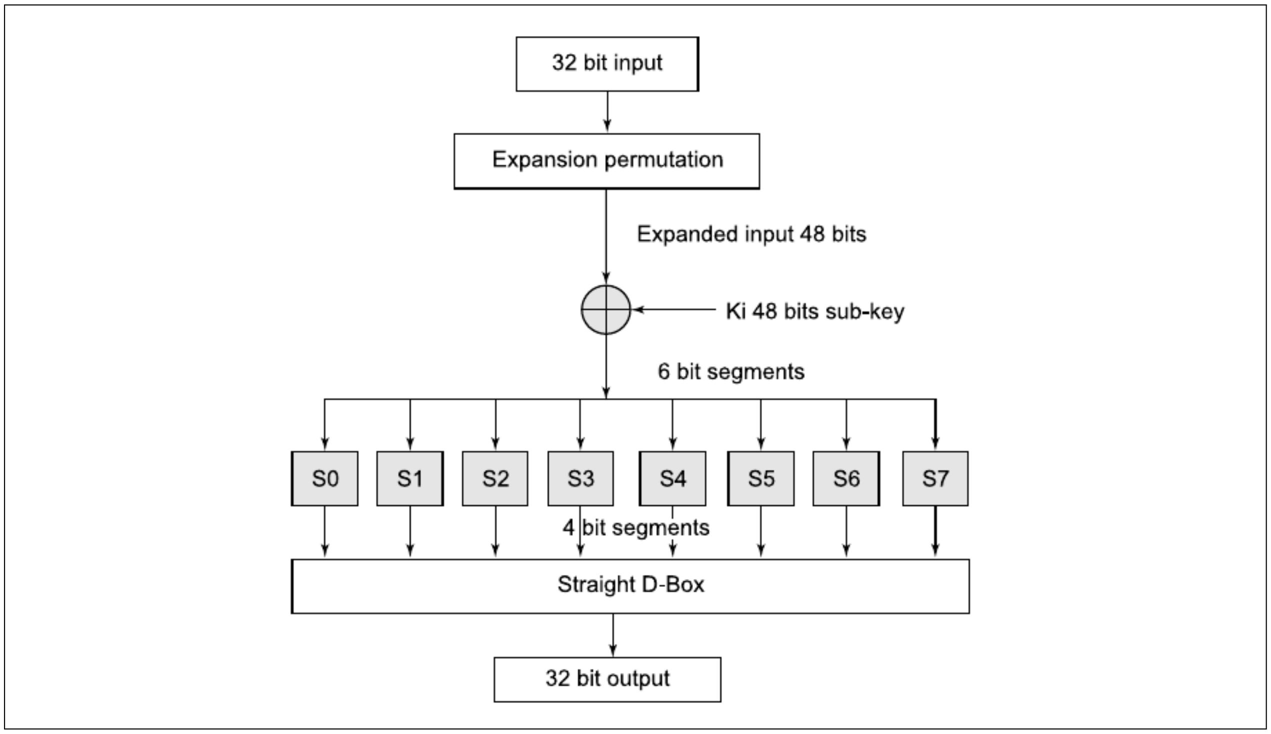


Figure 4: DES function

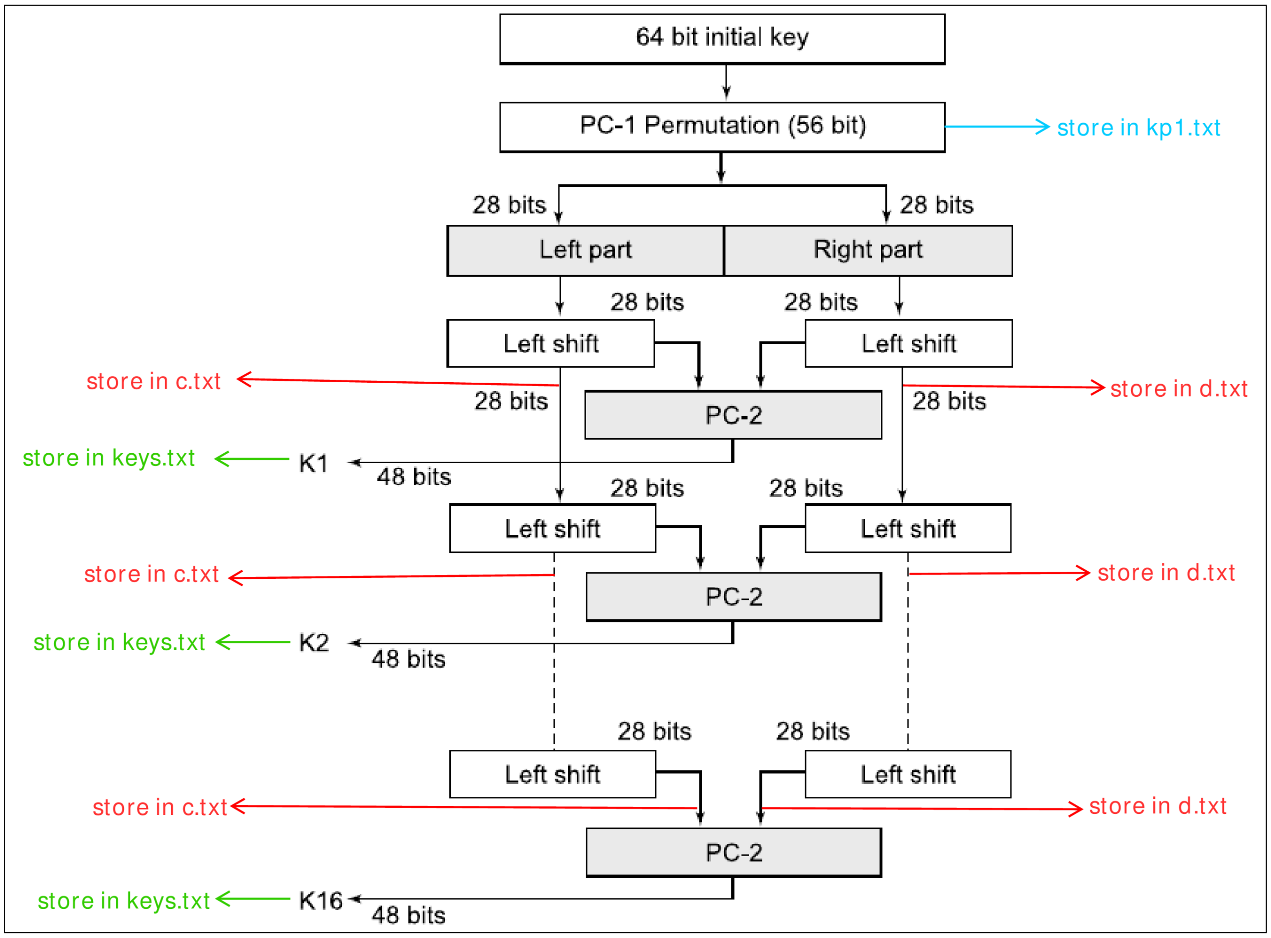


Figure 5: Keys generation: intermediate results

2 Intermediate results

You have to store the intermediate results in files as explained below.

2.1 Intermediate results of the keys generation

As shown in Figure 5 the intermediate results of the keys generation have to be stored in four files: `kp1.txt`, `c.txt`, `d.txt`, and `keys.txt`. The following table gives the details of the stored information

File	Stored information	Note
`kp1.txt`	the result of the PC1-permutation	Contains one 56-bit row
`c.txt`	the left halves of the intermediate keys	Contains sixteen 28-bit rows: one for each of the sixteen rounds.
`d.txt`	the right halves of the intermediate keys	Contains sixteen 28-bit rows: one for each of the sixteen rounds.
`keys.txt`	the generated keys (results of the PC2-permutation)	Contains sixteen 48-bit rows: one for each of the sixteen rounds.

2.2 Intermediate results of the evaluations of the function F

As shown in Figure 6 the intermediate results of the sixteen evaluations of the function F have to be stored in eleven files: `ep.txt`, `xored.txt`, `s0.txt`, `s1.txt`, ..., `s7.txt`, and `f.txt`. The following table gives the details of the stored information

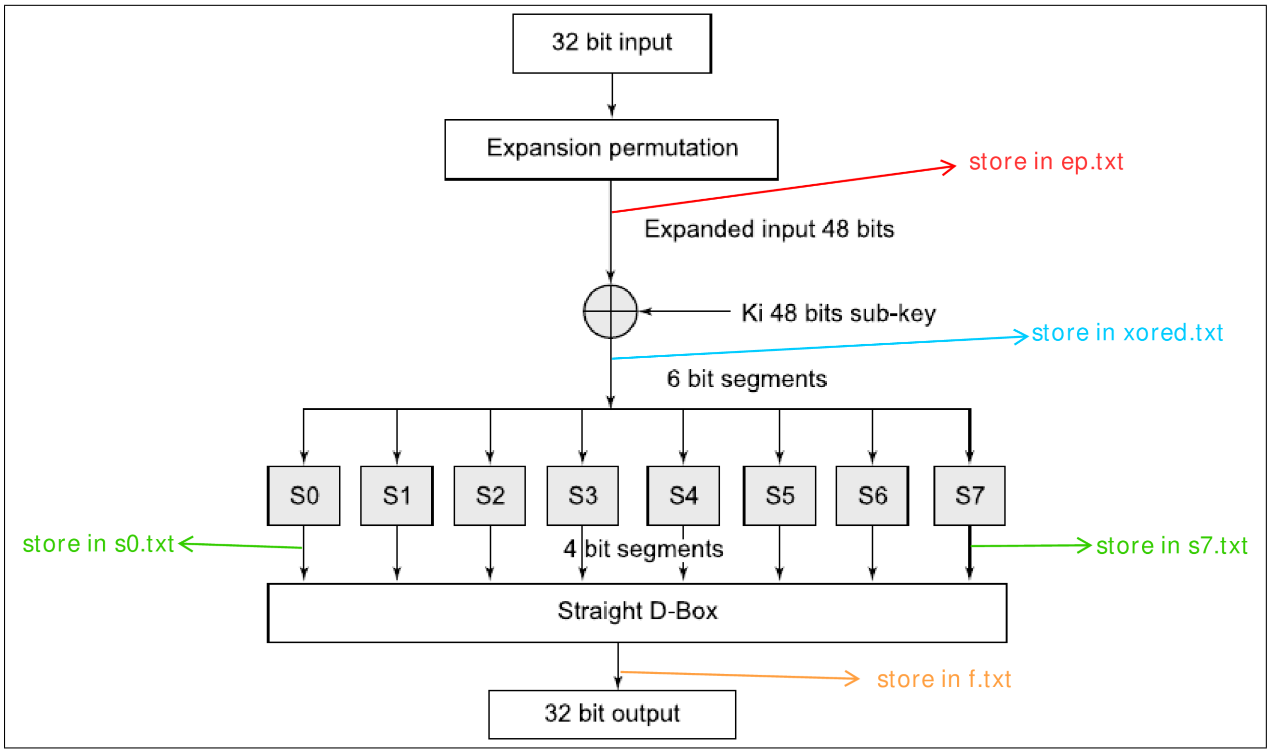


Figure 6: Evaluations of F: intermediate results

File	Stored information	Note
`ep.txt`	the result of the expansion permutation	Contains sixteen 48-bit rows: one for each of the sixteen evaluations
`xored.txt`	the result of the XOR operation	Contains sixteen 48-bit rows: one for each of the sixteen evaluations.
`s0.txt`	the result of the substitution S0	Contains sixteen 4-bit rows: one for each of the sixteen evaluations.
…	…	…
`s7.txt`	the result of the substitution S7	Contains sixteen 4-bit rows: one for each of the sixteen evaluations.
`f.txt`	the result of the evaluation of the function F	Contains sixteen 32-bit rows: one for each of the sixteen evaluations.

2.3 Intermediate results of the DES encryption process

As shown in Figure 7 the intermediate results of the DES encryption process have to be stored in five files: `ip.txt`, `l.txt`, `r.txt`, `fxored.txt`, and `fp.txt`. The following table gives the details of the stored information

File	Stored information	Note
`ip.txt`	the result of the initial permutation	Contains one 64-bit row
`l.txt`	the left halves of the intermediate messages	Contains seventeen 32-bit rows: L0, and one for each of the sixteen rounds.
`r.txt`	the right halves of the intermediate messages	Contains seventeen 32-bit rows: R0, and one for each of the sixteen rounds.
`fxored`	the result of the XOR operations	Contains sixteen 32-bit rows: one for each of the sixteen rounds.
`fp.txt`	the result of the encryption process	Contains one 64-bit row

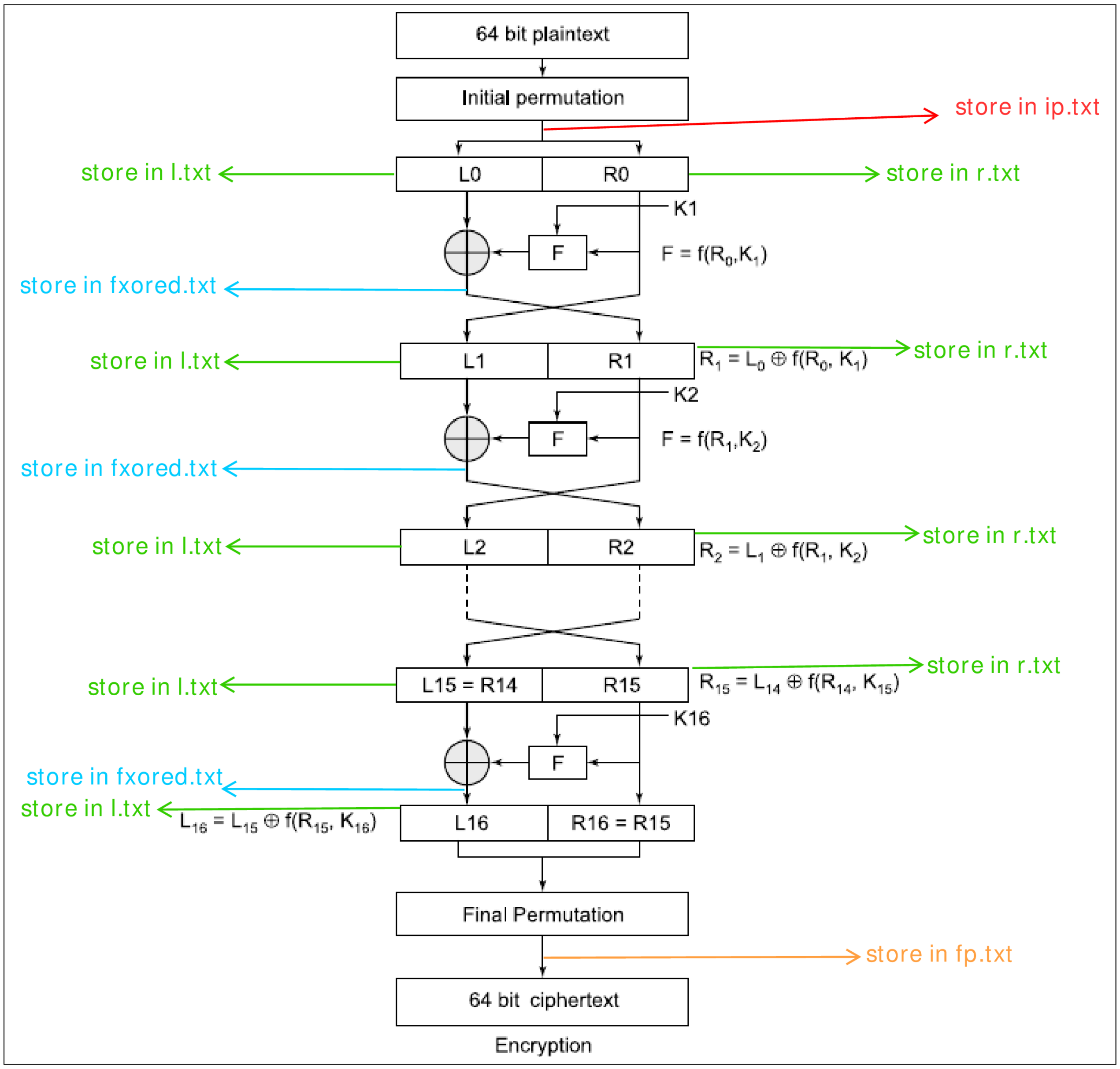


Figure 7: DES encryption process: intermediate results

3 Provided Material

A ZIP archive is provided. Its content is detailed here.

Folder `src`

The folder `src` contains a Java package, `swe314des`, with two classes:

1. `Main.java` contains a skeleton program that behaves as expected in Figure 1.
2. `IO.java` contains the necessary code needed for storing intermediate results.
   This code creates automatically a folder called `reults` and, inside this folder, all the (empty) files listed in the previous section.
   You don't have to create any file by yourself.
   The class `IO` has one public static method, `write`, that can be called from anywhere in your program. This method takes two arguments, the name of a file and a string, and writes the string to the file.
   For example, the statement

   ```
   IO.write("kp1.txt", "0111100110101" + "\n");
   ```

   will write the string `0111100110101` to the file `kp1.txt`.
   (don't forget the end-of-line `"\n"`)

Folder `sampleResults`

. The folder `sampleResults` contains the files that store the intermediate results of the execution of the DES encryption process with the following input:

key = 0001001100110100010101110111100110011011101111001101111111110001
message = 0000000100100011010001010110011110001001101010111100110111101111

4 Deliverables

No later than Sunday, October 24, you have to submit a ZIP archive that contains the following (upload to a shared folder that will be communicated to you later):

1. a text file that contains the names and the student IDs of the members of the team
2. all the Java source code
3. the results folder that contains the intermediate results of the execution of your program with a given key and given message of your choice (give this key and message in a separate text file)