Implementation of a symmetric block cipher

Coursework project for 17COP514

The Algorithm

The following cipher is a Substitution-Permutation block cipher, like the AES cipher. It is not a cipher in actual use, it was designed for this coursework. It is considerably weaker than AES.

The blocks of plaintext and ciphertext have 64 bits, i.e. 8 bytes. The key is also 64 bits, i.e. 8 bytes. The input of the encryption algorithm is loaded into a 64-bit state which is processed throughout the algorithm. The final value of the state is returned as the output.

There are three basic operations:

1. Substitution: each byte in the state is substituted using the S-box of the AES cipher.
2. Permutation: For this step it may help to visualise the 8 bytes of the block as a two dimensional array of 8x8 bits, each of the bytes being one row of the array. Each column of this array is shifted downwards, circularly, by 0,1,2,…7 positions. That is:

b_{00}	b_{01}	b_{02}					
b_{10}	b_{11}	b_{12}					
b_{20}	b_{21}	b_{22}					
b_{30}	b_{31}	b_{32}					
b_{40}	b_{41}	b_{42}					
b_{50}	b_{51}	b_{52}					
b_{60}	b_{61}	b_{62}					
b_{70}	b_{71}	b_{72}					

   will become

b_{00}	b_{71}	b_{62}					
b_{10}	b_{01}	b_{72}					
b_{20}	b_{11}	b_{02}					
b_{30}	b_{21}	b_{12}					
b_{40}	b_{31}	b_{22}					
b_{50}	b_{41}	b_{32}					
b_{60}	b_{51}	b_{42}					
b_{70}	b_{61}	b_{52}					

3. Compute the bitwise XOR between the state and the 64-bit (8 bytes) round key.

The encryption starts with a bitwise XOR with a key and proceeds in 5 rounds as follows:

state XOR key0

Round 1:
Substitution
Permutation
state XOR key1

Round 2:
Substitution
Permutation
state XOR key2

Round 3:
Substitution
Permutation
state XOR key3

Round 4:
Substitution
Permutation
state XOR key4

Round 5:
Substitution
(no permutation in this round)
state XOR key5

The six round keys key0, key1,…, key5 are obtained from the original 64-bit key as follows: key0 is the original key; each new round key is obtained from the previous round key (i.e. key1 is obtained from key0, key2 is obtained from key1 etc). Denoting by b0,b1,…b7 the bytes of the previous key, from left to right, the bytes of the new key will be:
b0 XOR b1,
Sbox(b1),
b2 XOR b3,
Sbox(b3),
b4 XOR b5,
Sbox(b5)
b6 XOR b7,
Sbox(b7).

The Implementation

Programming language: C is preferred, but one can also use Python, Java etc. The program can be command-line; no GUI (Graphical User Interface) is needed.

Your task is to implement as many of the following as you can (note that the project is broken down into independent tasks where possible, so that work can be more easily shared between the students in the group, and can also be assessed independently):

1. The encryption algorithm for one plaintext block; the input plaintext and the key can be read from the user or even hardcoded in the program; output on the screen.
2. The decryption algorithm for one ciphertext block; inputs/outputs as for task 1.
3. Encryption and decryption of a plaintext consisting of several blocks, in each of the following five modes of operation: ECB, CBC, OFB, CFB, CTR. Inputs/outputs can be as in point 1 above or as in point 4 below for plaintext and ciphertext. The key, IV and counter can be read from the user or hardcoded in the program.
4. Input plaintext to be read from files for encryption and output ciphertext to be written in a file for decryption; it is sufficient to deal with files of moderate size, such that the whole file can be read into memory if needed. Ideally you should be able to read files of any type, interpreting them as a sequence of bits or as a sequence of bytes or ASCII characters. If you cannot read all file types, just concentrate on one single file type.
5. Using padding for the last block when needed and correctly removing the padding after decryption. There are several padding methods that allow for correct removing of the padding, you can use any of them.
6. Measuring the time performance of encryption and of decryption (number of byes encrypted/decrypted per second) in ECB mode, and in the other modes.

The Report

Length of report: flexible, guideline of 10-20 pages.

- Describe the encryption algorithm and the decryption algorithm.
- Describe your implementation (all the points above), any problems encountered and how they were overcome, any improvements.
- Use your program to encrypt some small amount of data (for example 256 bits) and write the results in your report.
- Discuss the performance: analysis of the time performance measurements, any other performance related comments, possible improvements. Try to obtain reliable time measurements.
- Some analysis of the security of this cipher. This can include theoretical analysis and/or experimental analysis. The tutorial questions regarding AES can serve as a guide.

Submission

- Prepare and submit electronically your report (.pfd or Word document), the source code files and executables.

- It is assumed, by default, that all members of the group contributed equally. If this is not the case, the report should clearly indicate the percentage that each team member contributed. In case of disagreement, each member should instead e-mail me with their opinion about the percentage contribution of each member. Note that some of the marks will be redistributed according to the percentage contributed by each member.

Marking

The group coursework (program and report) will form 35% of the final mark.

- The program will be allocated 20 marks. It will be assessed for correctness, completeness (the tasks 1-6 described above), good programming style, efficiency.
- The report will be allocated 15 marks. It will be assessed for completeness (covering the points above), clarity, good organisation.

Each member's understanding of the project topic and the program will also be assessed individually, forming 5% of the final module mark:

- Each group will have about 30 minutes to demonstrate their program in the lab; you should demonstrate correctness by showing that encrypting and then decrypting with the same key produces the plaintext you started with.
- Each person in the group should be present and prepared to answer questions about their part of the program and about the algorithm in general.

Advice

- Share the work equally among you, as much as possible. Ideally each person should contribute to both the programming and the report writing. Each member should familiarise themselves with the whole program and report.
- Bit operations are needed and will be taught in the module.
- Lab sessions will be allocated specifically for working on the project, and a teaching assistant will be available for giving advice.