CS4600

Cryptography and Information Security

AES (ECB) Algorithm Implementation Using C++ or Java

(using a 128 bit key)

The purpose of this project is to ensure a complete understanding of the workings of the most popular encryption standard in the world, Advanced Encryption Standard. It is strongly recommended that you use C++.

You will implement a program called, “AESEncrypt128” which will take as input a string which contains the absolute file path of a file to be encrypted. This string will be passed to the program as a command line argument. Your program should use a 128bit key for encryption.. You will implement AES using the ECB mode of operation.

Your program will prompt the user for a key. This key will be a 32 digit hex value. You should be expecting either upper or lowercase input for the key and you should allow whitespace in the key.

For example, your program will be executed with the following command:

```
AES128Encrypt file_to_be_encrypted.txt
```

It must prompt for the encryption key then encrypt the file and store it with the same filename but with the file extension, “.enc”; e.g., “file_to_be_encrypted.enc” given the above command line example. This file should be created in the same directory as the plaintext file.

The last block read from the plaintext file must be padded. Use PKCS5 padding to accomplish this. See me if you have questions

PKCS5 Padding

- If the block length is B then add N padding bytes of value N to make the input length up to the next exact multiple of B.
- If the input length is already an exact multiple of B then add B bytes of value B. Thus padding of length N between one and B bytes is always added in an unambiguous manner.
- After decrypting, check that the last N bytes of the decrypted data all have value N with $1 < N \leq B$. If so, strip N bytes, otherwise throw a decryption error.
- For AES128, B = 128b / 8 = 16B

Examples of PKCS5 padding for block length B = 8:

3 bytes: FDFDFD --> FDFDFD0505050505
7 bytes: FDFDFDFDFDFDFD --> FDFDFDFDFDFDFD01
8 bytes: FDFDFDFDFDFDFDFD --> FDFDFDFDFDFDFDFD0808080808080808

The resulting file should be encrypted using the standards discussed in class--this implies that when I decrypt it with a program using the same standard, I should get the same plaintext back that I used when I encrypted with your program.

This project will require some research on your part. Most of the layers in each round are straight forward, but there are shortcuts to make those that are not, simpler.

MOST IMPORTANT: You must do your own work. The only source code that can be included in this project that is not yours is the source code that I provide with the project. Any academic dishonesty will result in the application of the maximum penalty--no excuses will be accepted. If you have any question about academic dishonesty, see Cal Poly’s policy or contact the department of Student Conduct & Integrity.