CSE508 : Cmputer Network Security - Assignment 3

Due: Apr 21, 2024 at 11:59 PM [KST]

Overview

It is recommended that this project be done by pairs of students. You can, of course, choose to do it individually, but it is obviously going to be more work. Writing ciphers can be tedious due to the detailed nature of the required mathematical operations.

Plan on starting the work on this assignment right away.

You are permitted to discuss the problem, but don't go to the level of sharing code.

Baby Rijndael Specification

Rijndael was selected as the Advanced Encryption Standard by the NIST (National Institute of Standards and Testing) in the US back in 2000. It is a symmetric cipher with 128, 192, or 256 bit keys. Being selected as this standard made it the successor to DES (The Data Encryption Standard). The specification for AES is in FIPS197.
For this assignment, you will implement a reduced cipher similar in structure to Rijndael. Baby Rijndael was developed by Dr Clifford Bergmann at Iowa State University. The following describes the task for this assignment.

Task

You will write an implementation of Baby Rijndael that can encrypt and decrypt in ECB (Electronic Code Book) mode as well as implementing CBC (Cipher Block Chaining) mode.

Baby Rijndael is a scaled down version of the real Rijndael (AES) cipher. The description of the cipher and sample data is available documents included in this assignment post on Brightspace. Your job is to implement the cipher in any programming language you choose. I recommend you implement encryption in the form of a function

`babyr_enc(block,key)`

that encrypts a single block.

Of course you should implement a function for decryption as well. I recommend

`babyr_dec(block,key)`

For each of these, *block* is a 16 bit integer and *key* is also a 16 bit integer. It returns a 16 bit integer which is the ciphertext (or plaintext) for the corresponding input.

Note that this method can be called repeatedly on 16 bit blocks to encrypt an entire file!

You can implement your cipher in any language I have access to build. My preferred are C, Java, or Python. If you have a preference other than those, please check with me first.

Your program should take as input:

- a direction (encrypt or decrypt)
- a mode (ecb or cbc)
- a filename as input
- a key (written as 0xnnnn)
- an iv (written as 0xnnnn) – if cipher block chaining is used

It should write an output file with the name (filename).out. So if the user specified an input file name of mydatafile.txt. Your code should write out the file mydatafile.txt.out.

The specification from Dr Bergmann has a single example of a 1 block encryption. You can use this as a 'test vector' to test your encryption and decryption primitives [methods that encrypt one block] (Note, I will provide additional test vectors) If those work, then you can write code around your primitives to use the two different modes. Following that, you can complete your implementation by writing a main that reads the inputs from the user and subsequently calls the primitive operations with each block of an input file.

Provide a test main routine which either asks for the above inputs from the console or accepts them as command line parameters. It should then encrypt (or decrypt) the file and write the resulting output in the new filename as described above. Include a README.txt file with your submission to describe how the program is built and run!
Obviously if I encrypt a file and then decrypt with the same key, te final output file should match the original.
Hints:
Remember you are writing binary files since encryption will produce no printable characters. However, you should also use binary output when decrypting (it will still show up as good ascii values if they are written correctly.)
Input should be divisible by 2 since the block size is 16 bits. However, if it is not, you should pad the final block with a single character 0xff. Remember on decryption, you will need to strip this padding if one single byte at the end is equal to 0xff. This is a simplistic padding scheme. More realistic padding schemes involve using pseudo randomly generated data.

Submission

Your submission will be in the form of C, Java, python or other language's source code. For languages like C, please include a short build file or a makefile to make it easy to compile and link or if it is a language like python, give instructions on how to run the encryption and decryption (arguments, how to specify input data and key, etc.) You can place descriptions of how to run code into the README.txt file mentioned above.

You should create a zip file of all your encryption/decryption code as well as the main method for testing.

Submission will be on Brightspace, and the details follow.

Please follow this procedure for submission:

1. Place the deliverable source files into a folder by themselves. The folder's name should be CSE508_assign3_<studentName>_<studentId>.

2. Compress the folder and submit the zip file.
    a. On Windows, do this by clicking the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Brigthspace.
    b. On Mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Brightspace.

3. Note: The following procedure may vary slightly.
    a. Navigate to the course Brightspace site. Click **Assignments** in the top navbar menu.
    b. Look under the category 'Assignments'. Click **Assignment3.**
    c. Scroll down and under **Submit Assignment**, click the **Add a File** button**.**
    d. Click **My Computer** (first item in list).
    e. Find the zip file and drag it to the 'Upload' area of the presented dialog box.
    f. Click the **Add** button in the dialog.
    g. You may write comments in the comment box at the bottom.
    h. Click **Submit**. ← Be sure to do this so I can retrieve the submission!