Jack Koefoed

CSIII

5/25/16

How to install and run:

Go to <https://www.visualstudio.com/products/visual-studio-community-vs> and click on download. This will install the free version of visual studio 2015 to your machine.

Once the IDE has been installed, go to file🡪open🡪project/solution then open the Rijndael.sln file. This will load all needed files.

The start button on the top bar will run the application. The right side toolbar accesses all files associated with the project.

Acronyms used in my comments:

AES—Advanced Encryption Standard, also known as Rijndael—the algorithm being used.

ECB—Electronic Codebook—A block cipher mode of operation where every block is encrypted and decrypted independently of one another. It is weak because identical blocks are encrypted identically.

CBC—Cipher Block Chaining—A block cipher mode of operation where every block is encrypted by XORing the previous ciphertext with the next plaintext before encryption, to create interdependency between the blocks and ensuring that one different byte in the plaintext affects all subsequent bytes.

CTS—Cipher Text Stealing—A system that avoids padding by initially padding the last n bytes of the last block with 0s to get a full block, which will become identical to the second to last ciphertext block due to the CBC XOR. The last block then is encrypted, and the two blocks are swapped to yield the full ciphertext. To decrypt, the process is reversed, but instead of having the XOR function do the copying, bytes must be manually copied.

How to look at the program:

Start with the Rijndael class. This class handles all events created from the UI. If the class opens in design mode, the underlying code can be accessed either by right clicking and clicking on View Code, or through the F7 key.

Next, move to the AES class. This class handles overarching program execution, including the creation of the key schedule. Focus primarily on the encrypt and decrypt methods, along with the key schedule method.

3rd, focus on the BitMatrix class. This class contains the methods for manipulating each block, and stores each block in an array of byte arrays. Efficiency is important here.

4th, look at the GF28 class. This is a class that does finite field arithmetic. I am aware that the static method that physically performs the arithmetic is fairly slow, which is why I precompute every product upon program launch and make each multiplication a single table lookup.

5th, look at the sBox class. This class contains the Rijndael S-Box, used for the key schedule and the SubBytes step. This class is similarly slow because it goes through every value, 0-255 and checks every value, 0-255, for the multiplicative inverse. I also precompute the entire table.

Lastly, take a look at the AlgorithmTester class. This is a class that runs the underlying algorithm in ECB mode, rather than CBC mode, meaning that there is no initialization vector, and compares it against test vectors found online.

Tests to run—All test files can be found in the directory with the project, in the TEST\_LIBRARY folder.

Encrypt/Decrypt the small text file, with working matching passwords. Should return an identical file, without any padding on the tail end.

Encrypt/Decrypt the 1 block text file. Should be no spaces after the k.

Encrypt/Decrypt the video. Make sure it can be opened afterwards.

Enter in an invalid or read-only path name. Should print error messages to the GUI.

Attempt to decrypt a non-encrypted file—Should say that it’s a bad file/password.

Attempt to decrypt an encrypted file with a non-encrypted password—Should say that it’s a bad file/password.

Attempt to encrypt a file while it is being encrypted, use the largest video file in the library—should block operating twice on the same file simultaneously.

Try any other ways of forcing an exception.

Test whether the algorithm works on a single block:

Run the Algorithm Tester class--

First, right click on the Rijndael project icon (the one 2nd from the top)

Go to “Properties”.

Change the startup object to encryption.AlgorithmTester.

Change the output type to console Application.

If the algorithm outputs the expected outputs on encryption, and it is identical after encrypting and decrypting, then the decrypt function must work.

Rubric:

**/ 40 : Total Score**

* / 19 : **External Correctness**
  + / **12**: **Encryption/Decryption Works**
    - /4: Encryption algorithm works on a single block
    - /4: Text files are encrypted + decrypted with identical inputs and outputs.
    - /4: Video/Pictures can be encrypted + decrypted with identical input and output
  + **/7: User input/output**
    - /1: Rejects incorrect password robustly and displays error message.
    - /2: Program is multi-threaded and does not lock up when running encryption.
    - /1: Rejects invalid file path names robustly and displays error message.
    - /1: Program prompts on shutdown if still operating on a file, but if prompt comes up and encryption is running, all threads close.
    - /1: Program rejects empty path names and passwords under 8 characters.
    - /1: Program rejects non-matching passwords in the two password boxes when encrypting, but not decrypting.
    - /1: Browse button allows file selection.
* / **15** : **Internal Correctness**
  + /3 Key generation:
    - /2 Key generated is random
    - /1 Key is hashed in a secure manner
  + /2 Threading
    - /2 UI threading is implemented properly.
  + /8 Time Efficiency
    - /1 Ciphertext stealing is implemented efficiently
    - /1 GF(2^8) table is implemented efficiently
    - /1 sBox is implemented efficiently
    - /2 Data is efficiently moved through Bit Matrices.
    - /1 Mix Columns is implemented efficiently
    - /1 Shift rows is implemented efficiently
    - /1 Add Round Key is implemented efficiently
  + /2 Memory efficiency
    - /2 Memory is not unnecessarily allocated and the GC is used appropriately
* / **6** : **Style/Documentation**
  + / 1 : Coding standards are adhered to: Header is at top of file, code is neatly indented and spaced
  + / 3 : Good coding style, code is readable: "self descriptive" identifiers are chosen, no "Magic Numbers" (or magic numbers are commented and explained), and tricky code is commented
  + / 2 : Documentation is present for methods and class variables, and a description exists for each method, parameter and return type as required