Final Project

Extending AES from 128-bit to 192-bits and 256-bits.

MSCS 630- Doc. Pablo Rivas.

Due Date: 05/02/2016.

Authors:

Murali Krishna kandula (200-62-382)

Bhargav Uppalapati (200-62-353)

Contents

[Introduction: 3](#_Toc450084694)

[Program Execution: 3](#_Toc450084695)

[ Encryption 3](#_Toc450084696)

[o Round Key Generation 3](#_Toc450084697)

[Decryption: 4](#_Toc450084698)

[Input and Output: 4](#_Toc450084699)

[Authentication Strategy: 4](#_Toc450084700)

[Improvements and Extensions: 5](#_Toc450084701)

[Difficulties encountered: 5](#_Toc450084702)

[References: 5](#_Toc450084703)

## Introduction:

AES (Advanced Encryption Standard) also known as Rijndael is a type of encryption standard. AES is a symmetric key algorithm which means same key is used both for Encryption and decryption. The algorithm created as part of the final project encrypts and decrypts a given piece of text.

One important part of AES is how it produces keys on every round of encryption type, these are known as round keys. With the help of the round keys a given input text is converted into a cipher text.

To decrypt a given cipher text, AES accepts the cipher text as a message and uses the same key used for encryption to decrypt the message.

## Program Execution:

The main functionality of AES is to encrypt a message at the sender’s side and decrypt at the receivers side.

* Encryption:

The encryption process starts of by running the Driver.java file. The file accepts input plain text which is of 128-bits and a key which may be of 128, 192, 256-bits size. Larger the key, greater the encryption. Text and key are passed as inputs to the aes () function. The input key is divided by 8 so that we can determine number of round generations and size of matrix which stores round keys. In the case of 128 bit key length, 10 round keys are generated. For 192- bit key length, 12 round keys are generated. Finally 14 round keys are generated for 256-bit key length.

* + Round Key Generation: We take the input key and make it first 4 columns in the round key matrix, for the other 40 columns we do the following operations:

If the round key matrix column index is a multiple of the 4 or 6 or 8 then the values in the columns are shifted by 1 position. Corresponding values to these shifted elements are pulled from S\_BOX table. Next depending upon the round number, R\_CON values are substituted. Next step is to perform XOR operation between the present round constant obtained and the values pulled from S\_BOX table, the output is stored in a temporary matrix. The last step is to perform XOR operation between the temporary matrix and the previously generated round key.

If the input key matrix column index is not a multiple of the present column index, we XOR last column with past fourth column with respect to the column index.

* Round key generation when key size is 256-bits: The round key generations in 256-bit key encryption differs a little from 128 and 192bit key encryption. If round key matrix column index is a multiple of 4 and if input key matrix index is 8 then corresponding values from S\_BOX are pulled for the previous column, then next step is to XOR previously obtained values with round key matrix’s present column index- the length of the input key matrix.

After obtaining the round keys, the next step is to encrypt the plain text with the round keys. In the next step the elements of the input text are XORED with the first round key which is state XOR.

Now the values from the previous step are stored in a 4x4 matrix and the values are substituted with the corresponding S\_BOX values. The next step is to perform shifting operation on the previously obtained matrix. The next step is to do mix column operation, this is a combination of matrix addition and matrix multiplication in the Galois field**.**

## Decryption:

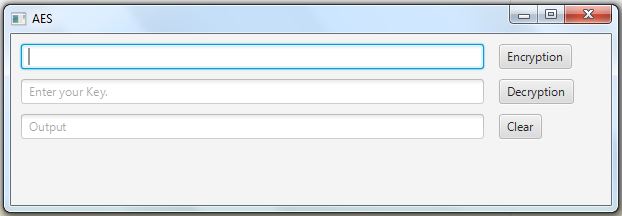
The encryption process starts of by running the Driver.java file. The file accepts input as cipher text which is of 128-bits and a key which may be of 128, 192, 256-bits size. AES is a symmetric key algorithm which means same key is used both for Encryption and decryption. Text and key are passed as inputs to the aes () function. The input key is stored in an array and it is divided by 8 so that we can determine round generations and size of matrix which stores round keys.

In the first round of decryption the following happens:

AES encryption is top down approach where first key is placed directly in the matrix and next steps are followed. In decryption last round key is given as input to the round key matrix. The next step is XORING cipher text and the last round key which was generated earlier and this is stored in a temporary matrix. Next step is implementing reverse nibble substitution where corresponding values to the temporary matrix are pulled from Inverse S\_BOX table. Next step is implementing inverse shift rows which performs right shift of elements in matrix by one position. This is the end of the first round.

The resultant matrix from the previous step is XORED with the key which is last but one. Mix column operation is performed on the XORED result, this is a combination of matrix addition and matrix multiplication in the Galois field, but to help with the multiplication look up tables with pre calculated values are present. The next step is to perform right shift on the matrix obtained from the previous step and Next step is implementing reverse nibble substitution. This process is continued until last but one round the last round would be do a XOR between previously generated reverse nibble substitution and the first round key. The resultant output is the plain text.

# Input and Output:



# Authentication Strategy:

We suggest using SHA 3.

* But Why SHA 3?
* The reason being the dissimilarity between SHA 2 and SHA 3, makes SHA 3 more secure.
* Less susceptible to collision- finding attacks.

# Improvements and Extensions:

Every project has a scope for improvements and extensions.

* Our AES can be improved by using a better padding strategy instead of using PKCS7.
* Implementing an authentication strategy is a good extension to our AES.
* Existing code can always be refined to make it efficient.
* UI can be improved to make it look better and user friendly, a more good extension would be add more functionalities.

# Difficulties encountered:

There were few hurdled that needed to be crossed before achieving the desired output. Initially there was difficulty figuring out how to encrypt a message using 256-bit key. Next hurdle was figuring out how to decrypt the cipher text, this was an arduous task but a doable one. Integrating everything into an UI was tricky but it was done. The implementation of AES was a huge learning curve in the field of encryption systems. The task was completed on time as a team.

# References:

https://en.wikipedia.org/wiki/Advanced Encryption Standard

https://en.wikipedia.org/wiki/Rijndael S-box

https://en.wikipedia.org/wiki/Rijndael\_mix\_columns

http://www.angelfire.com/biz7/atleast/mix\_columns.pdf

http://www.inconteam.com/software-development/41-encryption/55-aes-test-vectors