Advanced Encryption Standard (Rijndael): A Java Implementation

Security Algorithms and protocols

Course: MSCS_630L_231_16S
Prof. Pablo Rivas

By Siva Chintapalli,
Sourav Bhowmik

1. **Abstract:**

AES is a method to encrypt digital data. It was established as a standard by NIST(National Institute of Standards and Technology) in 2001. AES was developed by two cryptographers Joan Daemen and Vincent Rijmen. It is not an ideal block cipher which means that it can be broken at least theoretically. This paper describes how was the encryption and decryption algorithms of AES in Java was implemented and extended for different key lengths by the authors. It also suggests an authentication mechanism to be used along with AES for maximum security and a padding scheme.

2. **Contents:**

3. Introduction:

The implementation for the encryption algorithm in 128 bits was already available to the authors from a previously completed Lab assignment. The next set of tasks to be accomplished were:

- Extension of encryption algorithm for 192-bits and 256-bits.
- Implementation of AES decryption algorithm for 128-bits.
- Extension of the decryption algorithm for 192-bits and 256-bits.
- Implementation of a Padding scheme for the plain-text.

Prerequisite: Reader should be familiar with 128-bit encryption algorithm as mentioned in Lab 2 and Lab 3 of this course.

4. Extension of encryption algorithm for 192-bits and 256-bits:

The first step was to generate the Round Keys. The 128-bits implementation uses only 10 Round Keys, one for each of its 10 rounds. The 192-bits implementation has 12 rounds and the 256-bits implementation has 14 rounds. This means 12 and 14 Round keys have to be generated for 192-bits and 256-bits respectively.
The round keys are stored in the W[4][44] matrix for 128 bits. The size of W has been changed to W[4][52] for 192-bits and to W[4][60] for 256-bits.

Let us define Nw as number of bit words in the Cipher Key.
Thus, Nw=6 for 192-bits and
Nw=8 for 256-bits

Frame 1:

Matrix W is filled as:
The first Nw columns are same as the Cipher Key
If column index j is not divisible by Nw
W[j]= (W[j-Nw])XOR(W[j-1])

If column index j is divisible by Nw
Wnew= RconXOR (S-boxSubstitution (LeftShift (W[j-1])))
W[j]=(W[j-4])XOR(Wnew)

Only for Nw=8
If column index j is divisible by Nw/2 and not divisible by Nw
Wnew= S-boxSubstitution(W[j-1])
W[j]=(W[j-4])XOR(Wnew)

Everything else apart from what is mentioned above is exactly same as 128-bit encryption.

5. Implementation of AES decryption algorithm for 128-bits:

The decryption algorithm is the step by step backwards of the encryption algorithm. Basically, three inverse functions have been implemented to do the same, namely:

- InverseShiftRow()
- InverseSBoxSub()

- InverseColumnMix()

These are the inverse of the functions- aesShiftRow(), aesNibbleSub() and aesMixColumn() respectively.
Note: There is no need to inverse the aesStateXOR() function.
The function aesMixColumn() can be implemented by using a matrix that is available online or can be computed for each byte by making use of the matrix:
InverseGaloisMatrix=

0E 0B 0D 09
09 0E 0B 0D
0D 09 0E 0B
0B 0D 09 0E

where:

x*9=(((x*2)*2)*2)^x
x*11= ((((x*2)*2)^x)*2)^x
x*13=((((x*2)^x)*2)*2)^x
x*14=((((x*2)^x)*2)^x)*2

Note: x*2 is same as the one used in 128-bit encryption implementation.
The same set of Round Keys are used for decryption but in reverse order.

6. Extension of the decryption algorithm for 192-bits and 256-bits

The Round Key generation has been explained in section 3. Rest of the steps are same as Section 4.

7. High-level Summary of modifications made for extension to 192-bits and 256-bits:

	128	192	256
Size of W	4*44	4*52	4*60
No. of Rounds	10	12	14
Changes in Code	Nw=4	Nw=6	Nw=8, additional S-BoxSubstitution for j divisible by 4 and not divisible by 8.

Table 1:

8. Working of Driver.java:

The driver can accept input in 2 ways:

- From text file
- From Standard Input

From text File:

- The file should have exactly 4 lines.
- The first line should contain exactly one of '0', '1' or '2'. '0' implies 128-bit encryption is being followed. '1' implies 192-bit encryption is being followed. '2' implies 256-bit encryption is being followed.
- The second line should contain either 'e' or 'd'. 'e' implies encryption. 'd' implies decryption.
- The third line should contain the key.
- The fourth line should contain the message.

Any discrepancy from the above mentioned rules will lead to termination of the program with an appropriate prompt.
The program prints the output to standard output.

Sample text file input:

Frame 2:

```
1
e
000102030405060708090a0b0c0d0e0f1011121314151617
5468617473206D79204B756E67204675
```

This means that the file will be encrypted in 192-bit encryption with the specified key and plain-text.

From Standard Input:

The rules remain the same except for the fact that the user will have to enter the data manually. He/she would be guided with appropriate prompts for ease of use.

9. Padding Scheme:

A simple padding scheme has been used for messages that are not of length 128-bits(32 characters).

- The length of the message is first determined.
- Next, the no. of characters(say n) needed to make the message a multiple of 32 is determined.
- The message is padded n/2 times with the value n/2.

This padding scheme was chosen because it is very simple to understand and easy to implement.

10. Suggested Authentication Mechanism:

Authentication of messages is an important construction to detect modifications made to messages when they are in transit. Use of AES coupled with authentication from HMAC-SHA-3 would a very good security strategy. HMAC is not only efficient but also easy to implement. SHA-3 (Keccak) although relatively new has not yet been broken. So, it would be a good pick. SHA-2 is also a strong but old(and so possibly weakened) hash algorithm. Hence, it can also be used along with HMAC as a strategy.

11. Conclusion:

This document clearly specifies and enables the reader to understand and use the project developed by the authors. The project consists of full implementation of the Advanced Encryption Standard(AES) in 128, 192 and 256 bits in Java. AES is one of the top security encryption algorithms available to us. Additionally, an efficient and simple padding scheme has also been explained that has been used along with AES for achieving uniform block size. HMAC-SHA-3 is the suggested authentication algorithm to be used for prevention of tampering of messages.