Advanced Encryption Standard

May 2, 2016

Authors- Gopi.Para
Sudheer.Mandava

Table of Contents

1. Introduction:

This Advanced Encryption Standard (AES) specifies the Rijndael algorithm which takes the hexadecimal values of the plaintext with 128 bits as input, using keys lengths of AES-128, AES-192, and AES-256 bits.

2. Definitions:

AES	Advanced Encryption Standard
Array	An enumerated collection of identical entities (e.g., an array of bytes).
Bit	A binary digit having a value of 0 or 1.
Byte	A group of eight bits that is treated either as a single entity or as an array of 8 individual bits.
Plaintext	- Data input to the Cipher or output from the Inverse Cipher.
S-box	- Non-linear substitution table used in several byte substitutions

Transformations

Round Key - Round keys are values derived from the Cipher Key using the Key expansion routine

Nb - Number of columns (32-bit words) comprising the State. In this document Nb = 4

Nk - Number of 32-bit words comprising the Cipher Key. For this standard, Nk = 4, 6 or 8.

Nr - Number of rounds, Nr = 10 for 128 bit, 12 for 192 bit, 14 for 256 bit.

Rcon [] -The round constant word array.

RotWord() - Function that takes 4 byte word and performs cyclic permutation.

aesStateXOR - Performs the XOR operation on the state matrix and the Round keys in different rounds.

aesNibbleSub - Transforms the bytes in the matrix with the corresponding S-Box values.

aesShiftRow - Cyclically shifting the last three rows of the matrix by different offsets.

aesMixColumn - Mixes the column multiplies state matrix against the current state matrix.

3. Generating round keys:

In generating the round keys first we have to consider a key, consider 128 bit in this case that is splitted into four 32 bits columns indicated as w matrices.

For example consider

Key in Hex is 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75, this key is splitted into 4*32 bit ‘w’ matrices.

(a)w[0] = (54,68,61,74),w[1] = (73,20,6D,79),w[2] = (20,4B,75,6E),w[3] = (67,20,46,75)

(b) A circular byte left shift is performed on w [3] = (67, 20, 46,75)

(c)Substituting the ‘w’ matrix with s-box.

(d)Adding round constant to the obtained ‘w’ matrix.

(e) W [4] matrix is generated by XORing the ‘w’ matrix obtained after adding round constant with the w[0]. The obtained result is XORed with w [1] and so on till w [7]. This will produce the first round key.

→ In this way we can generate 10 round keys in case of 128 bit, 12 round keys in 192 bit, 14 round keys in 256 bit.

4. Encryption:

In encryption, the individual transformations aesNibbleSub (), aesShiftRow (), aesMixColumn (), and aesStateXOR () takes place and process the state.

4.1 aesNibbleSub ():

Substituting the bytes in the current state matrix obtained after state XORing with Round key with the bytes in the s-box.

4.2 aesShiftRow ():

The current state matrix is cyclically left shifted with offsets of 0,1,2,3 i.e. the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets) which causes diffusion of the bits over multiple rounds. The first row, r = 0, is not shifted

4.3 aesMixColumn ():

The Mix Columns () transformation operates on the State column-by-column which multiplies fixed matrix against current state matrix.

4.4 aesStateXOR ():

AesStateXOR () is nothing but adding round key. In the aesStateXOR () transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of 'Nb' words (based on the size of the key) from the key schedule.

5. Padding method: Pad with 0x80 followed by zero bytes (one and zeros padding)

Add a single padding byte value of 0x80 and then pad the balance with enough bytes of value zero to make the total length an exact multiple of 8 byes. If the single 0x80 byte makes the total we don't have to add any zeros bytes. This is known as one and zeros padding

Usage: AES input block = f o r _ _ _ _ _ _
(In Hex) = 66 6F 72 80 00 00 00 00
Key = 01 23 45 67 89 AB CD EF
AES output block = BE 62 5D 9F F3 C6 C8 40

6. Key Expansion:

The Key Expansion generates a total of Nb (Nr + 1) words. For this to be completed, we need 'Nb' words, and 'Nr' rounds require 'Nb' words of key data.

The first 'Nk' words of the expanded key are filled with the Cipher Key. Every following word, w[[i]], is equal to the XOR of the previous word, w[[i-1]], and the word Nk positions earlier, w[[i-Nk]]. For words in positions that are a multiple of Nk, a transformation is applied to w[[i-1]] prior to the XOR, followed by an XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word (RotWord()), followed by the application of a table lookup to all four bytes of the word (SubWord()).

7. Decryption:

7.1 InvShiftRows () transformation:

InvShiftRows () is the inverse of the Shift Rows () transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted.

7.2 InvNibbleSub () transformation:

InvSubBytes () is the inverse of the byte substitution transformation, in which the inverse S-box is applied to each byte of the State

7.3 InvMixColumns () transformation

InvMixColumns () is the inverse of the Mix Columns () transformation. InvMixColumns () operates on the State column-by-column,

7.4 aesStateXOR transformation:

This aesStateXOR is its own inverse, since it only involves an application of the XOR operation.

8. Working of Driver program:

Running from command line:

1. AEScipher.java, Driver file should be in a folder.
2. Test vector which contains key and plaintext should be in a testCase.txt file created in the same folder because the driver file accepts the standard input from the user.
3. Open the command window in the folder which you have all your files and the testCase.txt file.
4. Run from the command line as
   javac *.java- This will create the class files in your folder.
   java Driver<testCase.txt- this will produces the output showing the encryption and decryption.

9**. Authentication Strategy:** we highly recommend SHA-3 because it is more secure and ability to run on many types of devices, as well as its facility in performing better than SHA-2

10. Implementation Issues: An implementation of the AES algorithm shall support all three lengths specified 128,194,256. Firstly we have some issue regarding key length, block size and number of rounds. We figure out that with help of some research documents. The main conflict arise during the process of mixing columns and reversing them during the decryption

11. Further Enhancement:

1. Implementing an application and make it available in web as the best tool among all the applications we have now.

2. Speed up the process of encryption and decryption.

12. Conclusion: Various data messages were encrypted using different key sizes. The original data was properly retrieved via decryption of cipher text. The modifications brought in this code are tested and proved to be accurately encrypting and decrypting the data messages.

References:

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf
http://www.di-mgt.com.au/cryptopad.html
https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf
https://www.lri.fr/~fmartignon/documenti/systemesecurite/5-AES.pdf