AES Extension from 128-bits to 192-bits & 256-bits

MSCS_630L_231_16S

Security Algorithms & Protocols

Created By

Dixita Sharegar

Saranya Radhakrishnan

Contents

Introduction

AES i.e. Advanced Encryption Standard is a specification by NIST to encrypt and decrypt electronic data established by the National Institute of Standards and Technology. It is a block cipher which uses a block length of 128 bits. AES allows for 3 different key lengths 128,192 and 256. AES is symmetric i.e. it uses the same secret key for encryption & decryption, both sender & receiver should share this key. All key lengths are considered good enough to protect classified information up to the "Top Secret" level.

AES 128 bit encryption was accomplished in the previous lab work for this course, which serves as a base for the subsequent decryption techniques implemented as mentioned in this paper. The tasks to be accomplished included:

- Encryption using 192 and 256 bit keys.
- Decryption for 128, 192 and 256 bit keys.
- Padding scheme.

Program Execution

Driver

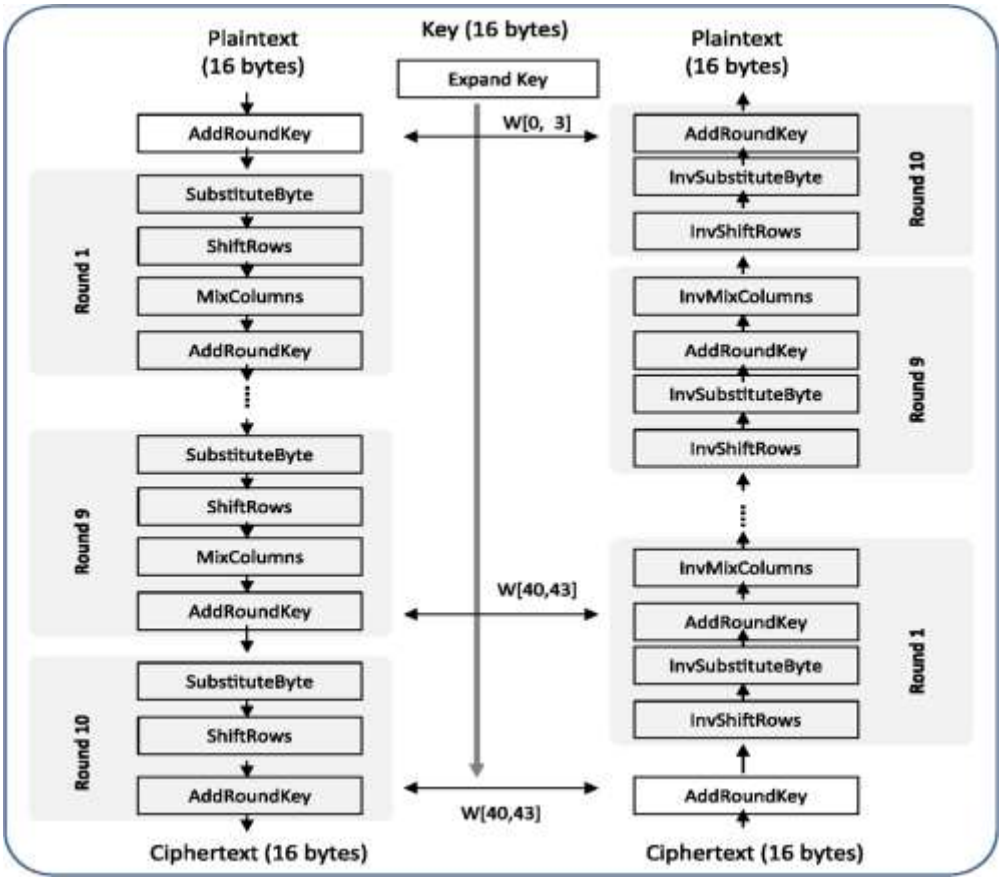
This file contains the main class, it accepts the key and input data. Here based on the key size, operations are performed on the data. Number of round transformations depends on the size of the key. Once the key & plaintext are taken as input, basic validation is done to check if data is in hex format. Next the number of rounds and number of columns is calculated and stored in the global variables. Now we break the plain text into blocks of 32 bytes and check if padding is needed. Once the text is padded, it is sent to the encryption function. This encrypted text is printed and then sent as an input to the decryption function. After obtaining the decrypted text, padding is removed and the original text is displayed.

Encryption

Encryption technique used for 192 and 256 bits was same as the one used for 128 bit. The algorithm mode is first set to encryption. First the input data is split into a 4x4 array. Now based on the number of rounds, different operations are performed on the round key and the round data. For the first round the plaintext is added (XOR operation performed) to the first round key. From the second round till the second last round the sequence of operations are the same i.e. Nibble substitution (Sbox substitution), row shifting, mix columns and add key. For the last round we perform Nibble substitution, row shifting and add key.

Decryption

Decryption technique was also the same for 128,192 and 256 bits. A set of reverse operations are performed to convert cipher text to plaintext. The round keys were used in the reverse order while decrypting data. First the algorithm mode is set to decrypt and the input data is split into a 4x4 array. In the first round, cipher text and first decryption round key are added (XOR). From the second round to the second last round, similar operations are performed in the following order: add round key, inverse mix column, inverse shift rows, and inverse nibble substitution. The last round includes add round key with the last decryption key.



GlobalObjects

This file contains common constants, variables and functions that are used by both encryption and decryption classes. It contains common functions like the following:

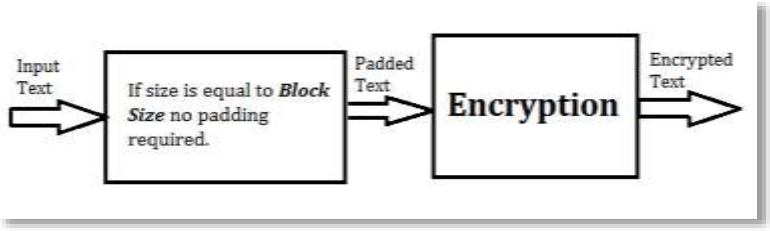
- generateDataMatrix: It splits the data into 4x4 array.
- generateKeyMatrix: It splits the key and generates an array based on the key size.
- aesStateXOR: It performs XOR operation on the input data and the key.
- aesRcon: Gets the Rcon value from the table.

It contains many such common functions and variables that are used by driver, encryption and decryption files.

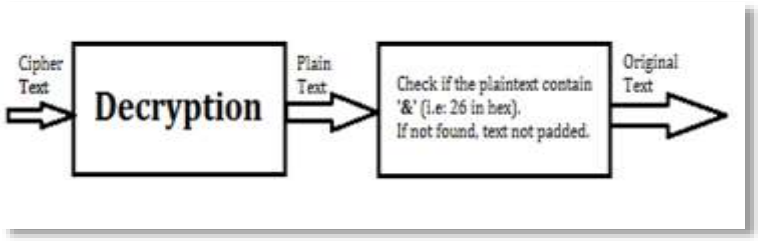
Padding

If block size is not a multiple of 32, then it needs to be padded. The end of the string is padded with hex value of '&' followed by number of bytes to pad. We have 3 conditions to be considered while
padding the text.

1) Input Size=Block size

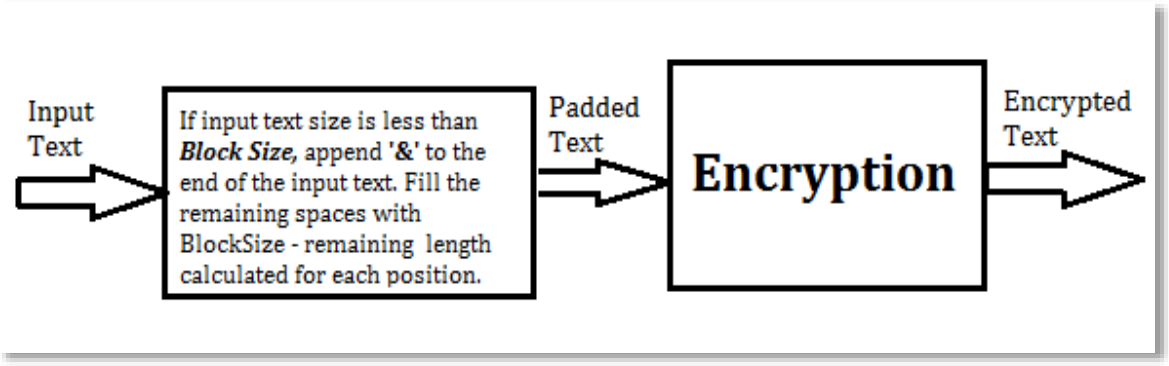


If the input size is equal to or a multiple of block length then no padding is required. Also after decrypting nothing needs to be removed from the decrypted message.

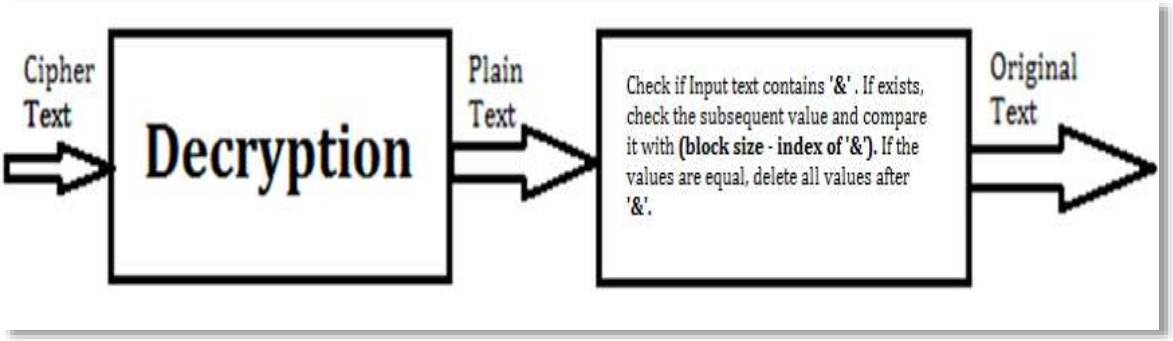


2) *Input size< Block Size*

Add hex value of '&' to the end of the plaintext and pad the remaining bits with the length of spaces left. For example if the plaintext is "00112233445566778899" which is 20 bytes, then 10 bytes need to be padded. So we first add hex value of '&' so the plaintext is "0011223344556778899**26**". Now the number of spaces left to pad is 10 so we pad as follows "00112233445566778899**2610**" .Now the number of spaces left is 8 so we add "00112233445566778899**261008**" and we finally get "00112233445566778899**2610080604**" as the padded text.

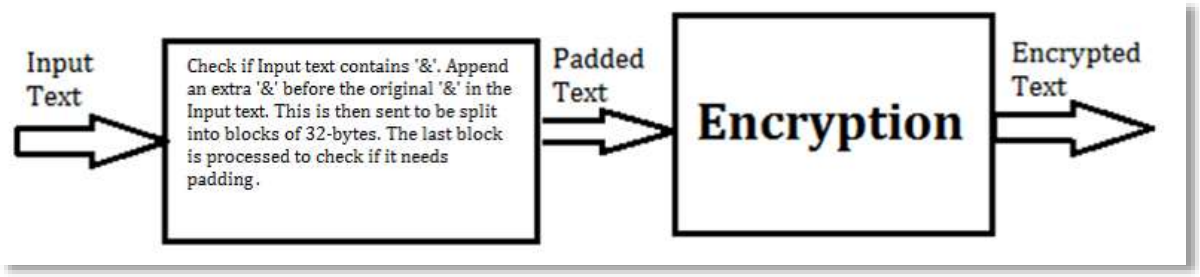


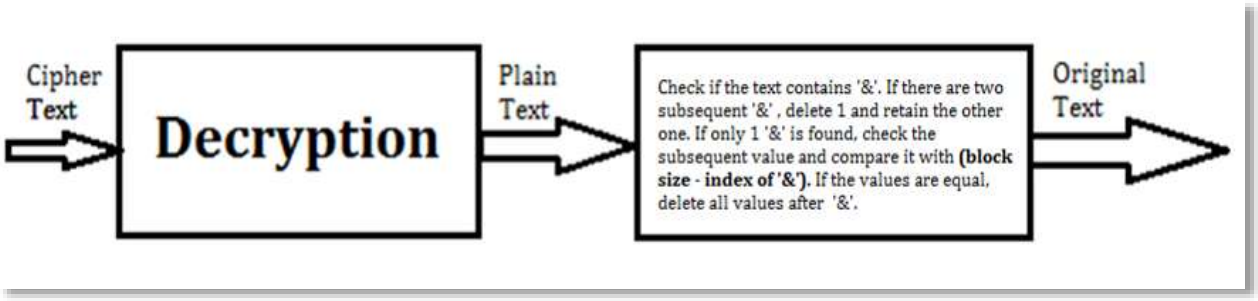
To remove padding we find the hex value of '&' i.e. "26". Then we check the consequent value which is 08 in our case. Then we subtract the block size (32)-from the position of "&" which is (22) and we get 10. So we fetch the string from 0 to 20.



3) *Input size> Block size*

If input text contains '&' then append '&' to it and then split it into blocks of 32. Last block is checked to see if it needs padding. While removing padding check for consequent '&' and delete one.





Input & Output

Ten test cases are created with key lengths varying from 128, 192 and 256 bits. It contains varying length input plaintext so that padding can be tested.

Suggested Authentication Mechanism: HMAC-SHA3

Authentication is important to know that the data has not been tampered during transmission. HMAC is very fast and secure. SHA3 provides high level of parallelism and is faster than SHA2 in all modern PCs. SHA3 provides sufficient number of rounds (24) for security. Even if an attacker uses 1 billion computers which perform 1 billion operations it would take 1.6×10^{61} years to evaluate permutation of 2^{288}.

Conclusion

This document gives a brief explanation of how AES 128,192 and 256 encryption and decryption have been performed. It also specifies the padding technique used with detailed steps of what operations are performed based on the block size. The authentication scheme that best suits AES is HMAC-SHA3.

References

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
http://www.slideshare.net/gecaccavale/sha3-keccak-sponge-function