ADVANCED ENCRYPTION STANDARDS (AES)

1. Introduction
2. Terms used
3. Description of the cipher
4. Padding
5. Security
6. **Introduction**

**AES** is a secure block cipher. **AES** is a symmetric-key algorithm which means that same key can be used both for encryption and decryption. **Advanced Encryption Standard** is a US government standard that has been implemented to overcome the standard DES. **AES** uses a different structure than **DES**.

It is available in different encryption packages and is a publicly accessible cipher approved by National Secret Agency (NSA) to transmit most secure information.

1. **Terms Used:**
   1. Plain text- a message which has to be encrypted
   2. Cipher text- the encrypted message
   3. Cipher key- a secret key used to generate round keys
   4. Round keys- derived key from the key-expansion algorithm.
   5. RCON- It is a matrix having ten round constants. Every value is an element of AES-GF256 field.
   6. S-box: A lookup table used to grab the elements from, to substitute in the SubBytes() and InvSubBytes() functions.
2. **Description of the Algorithm:**

AES is a Symmetric-Key encryption algorithm implemented by Federal Information Processing Standards(FIPS). It is widely believed that it is secure, efficient and reliable.

It is a Rijndael algorithm specification, which uses cipher keys of lengths 128, 192 and 256 bits to process data blocks of size 128 bits. This encrypts 128-bit block (plaintext) to produce a 128-bit block (cipher text), or decrypts the 128-bit block (cipher text) into a 128-bit (plaintext).

AES uses keys of sizes 128,192 and 256 bits, so the encryption/decryption using cipher keys are denoted as AES-128, AES-192 and AES-256 respectively.

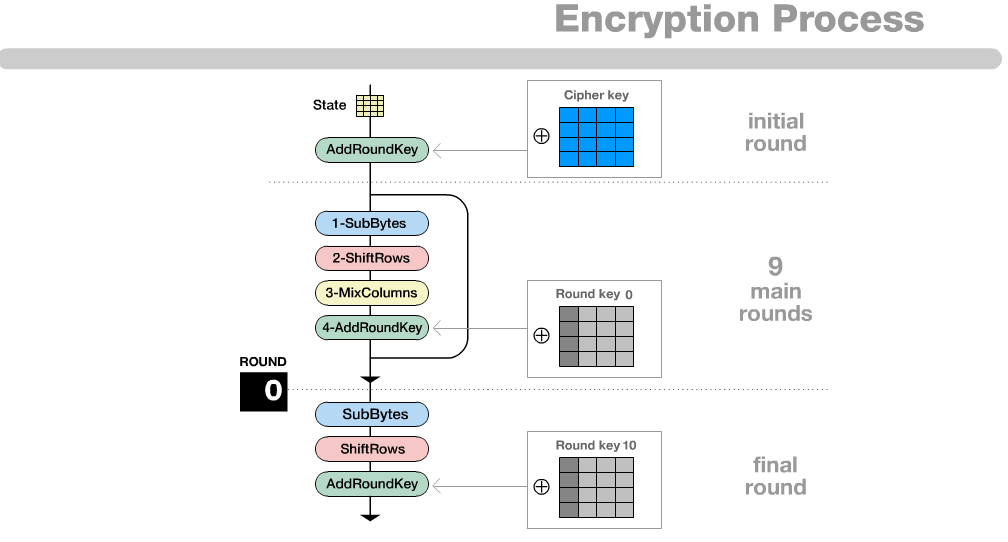
AES-128, AES-192 and AES-256 processes data blocks using 10, 12 and 14 iterations or rounds. These iterations are pre-defined sequences of transformations.

For the encryption/decryption process we need to generate the round keys which can be done using "Key Expansion" algorithm.

In encryption process, there are a series of steps/transformations involved to get the cipher text.

* 1. SubBytes
  2. ShiftRows
  3. MixColumns
  4. AddRoundKey

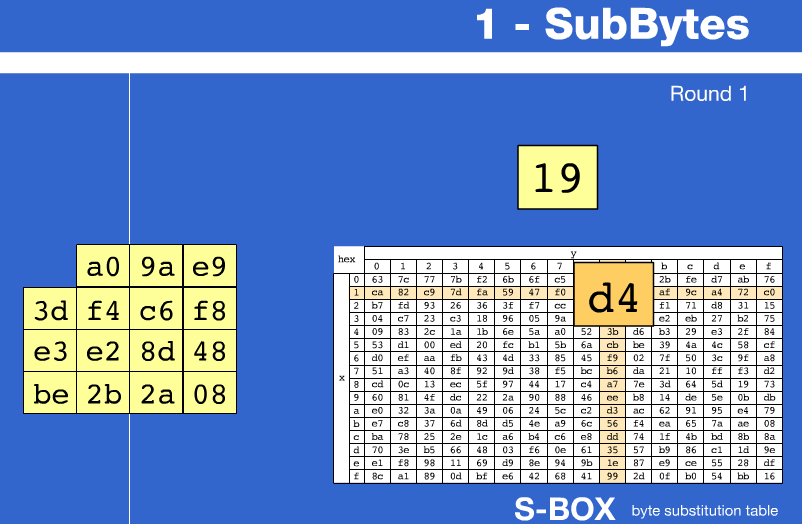
Of all the rounds/transformations, the last round has no MixColumn()/Inv MixColumns() to the final round in both the encryption and decryption processes.



**3a. SubBytes() transformation:**

In this step, we use a lookup table of 16 \* 16 size to get a replacement byte for a given byte in the input state array.

The lookup table is created by Galois Field(28). We replace all the 16 bytes of state array using each byte as an index to the lookup table and replace that value from that table. We can restore the original elements of the state array by using inverse SubBytes transformation.



**3b. ShiftRows() transformation:**

As the name suggests, in this transformation we shift the elements in the rows according to the value of the row they are present in. We consider a 4\*4 matrix for this step.

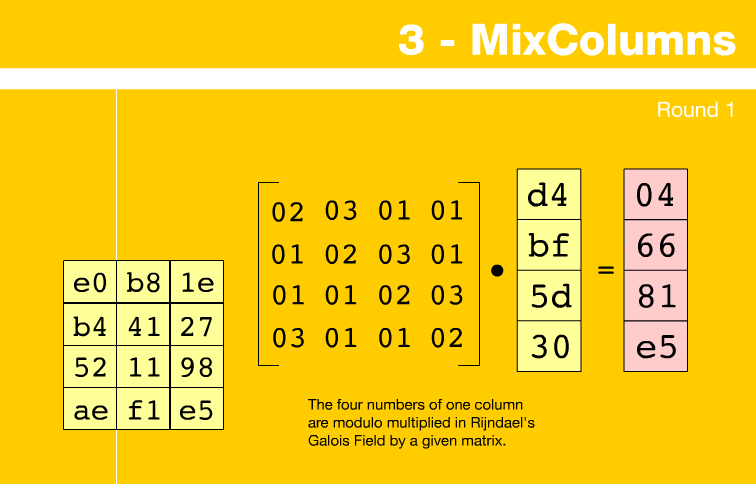
First row will be unchanged, the second row shifts to the left by one byte, the third row shifts by two bytes, the fourth row shifts to the left by three bytes.

* 1st row: no change
* 2nd row: left shift by 1 byte
* 3rd row: left shift by 2 bytes
* 4th row: left shift by 3 bytes

**3c. MixColumns() transformations:**

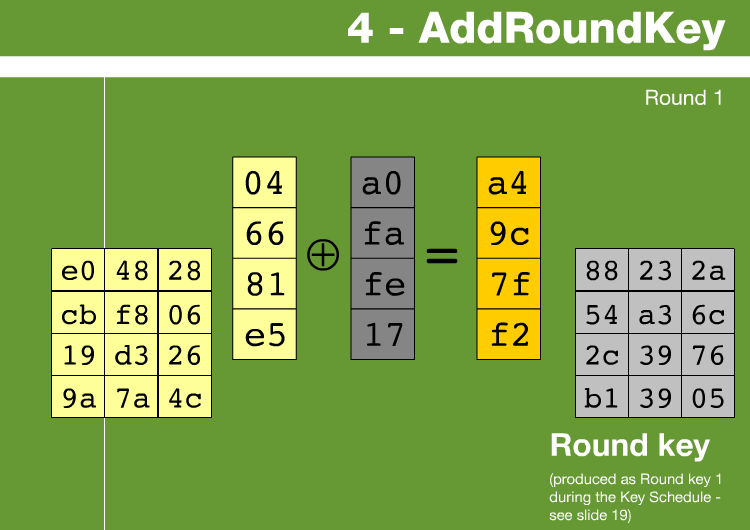
It is a 16-byte to 16-byte transformation which operates on 4\*4 matrix input. The state arrays each column is processed to produce a new column by matrix multiplication.

This operation takes each column of state array and replaces it with a new column generated by matrix multiplication.



**3d. AddRoundKeys() transformations:**

In this transformation, we just XOR the values of the state array with corresponding round key bytes.

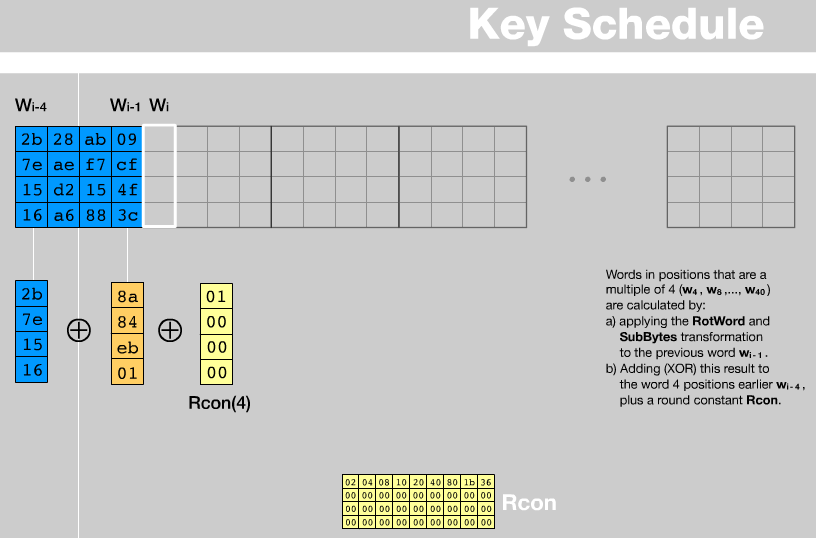


**4. Key Expansion:**

AES uses a cipher key whose length is 128,192 and 256 bits. This expands the cipher key into 10, 12, 14 round keys using “Key Expansion” algorithm.

It uses a key schedule to expand a key into a number of different keys.

It can be executed prior to encryption/decryption phase.



**AES Decryption:**

Here we use the same key used for encryption, generate it and save in reverse order.

Round keys and cipher text undergo transformations such as XOR, Inverse\_Sbox, Shifting rows and mix columns, based on the round of iteration.

This process involves the steps inverse of the encryption process.

1. InvAddRoundKey() : In this step we accept the key from input and generate round key as similar to the process of encryption and then store them in reverse order.Which involves operations such as exclusive-or,shifting bits,s\_box subsitution,r\_con substitution.

2. Once the round keys are ready and the cipher to be decrypted are saved in matrices, we start the decryption process by keeping track of number of rounds to be processed. Keeping count of rounds the cipher and round keys undergo exclusive-or, nibble\_subsitution(inverse s\_box matrix) and mix columns calculations.

3.Nibble\_subsitiion is similar to that of s\_box value subsitution, but in this we refer to inverse\_sbox matrix and pickup the corresponding value.

4. Mix column computations are calculated based on the pre computed lookup values of galois field. In last round the preprocessed message and last key are XOR'ed to get the decrypted text.

5.Once the computation has started and is getting iterated for each round key the counter is kept in track and for the last round of decryption the cipher and the round key undergo only exclusive-or and returns the decrypted output.

**Padding:**

As we know AES processes the data in blocks of 32 bytes, so in order to process the data the input plaintext should be a multiple of the block size. If the size of the input plaintext is not an exact multiple then we need to do ‘**Padding**’ before encrypting by adding a padding string and reverse in decrypting i.e., remove the padded string before decrypting. This padded message is encrypted and decrypted, but before the plain text is shown to user the padded bits are removed.

Our driver class accepts any size of input which is less than or equal to 32, pads it so that input size reaches 32 bytes. After padding the input text and key is passed to Aescipher class’s process Input method. The class accepts inputs, generates round keys and encrypts the text.

The encrypted text is passed on to driver class as cipher which is in turn given to process-input of Aesdecipher class. It accepts the key and cipher, generates round keys, decrypts the message and passed to Driver class as plain-text and prints it.Our driver can accept 128,192,256 bit keys and successfully perform encryption and decryption.

**Security:**

AES was designed to face any kind of attack. Some of the attacks that have been tested:

1. AES is more secure because of its larger key size.
2. Of all attacks till date, known attacks were on 7/8/9 rounds for 128/192/256-bit keys respectively.
3. The key sizes in decimal terms are:

a. 3.4 x 1038 possible 128-bit keys,

b. 6.2 x 1057 possible 192-bit keys,

c. 1.1 x 1077 possible 256-bit keys,

And it is said that it takes 149 trillion years to break the 128-bit AES key.

We suggest HMAC SHA-256 authentication strategy for this algorithm.