ADVANCED ENCRYPTION STANDARDS (AES)

1. Introduction
2. Definitions
3. Description of the cipher
4. implementations
5. Security
6. **Introduction**

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

It is available in different encryption packages and is a publicly accessible cipher approved by NSA to transmit top secret information.

1. **Definitions**
2. **Description of the Algorithm:**

AES is a Symmetric Key encryption algorithm implemented by Federal Information Processing Standards. 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

**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 ew 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.