**4.Literature Review**

**4.1. Background**

The earlier ciphers can be broken with ease on modern computation systems. The DES algorithm was broken in 1998. On the other hand, 3DES has three times as many rounds as DES and is correspondingly slower. It uses 64-bit block size which is less for more security, large block size is required. To mitigate the vulnerable aspects of DES and 3DES, NIST (National Institute of Standards and Technology) invited experts who work on encryption and data security to introduce an innovative block cipher algorithm to encrypt and decrypt data with powerful and complex structure. After performing various criteria and security parameters, they selected one of the five encryption algorithm that proposed by two Belgian cryptographers Joan Daeman and Vincent Rijmen. The original name of AES algorithm is the Rijndel algorithm. AES is intended to replace 3DES with high computational efficiency, 128-bit block size, and cryptanalysis resistance is strong against differential truncated differential, linear, interpolation and square attacks.

AES algorithm is of three types I.e., AES-128, AES-192 and AES-256. This classification is based on key used in the algorithm for encryption and decryption process in which numbers 128, 192 and 256 represent the size of the key in bits. But it only accepts a block size of 128 bits. The key size determines the security level as the size of the key increases the level of security increases. A number of AES parameters depend on key length. If the key size used is 128 then the number of rounds is 10. Likewise, for 192 and 256 bits it is 12 and 14 respectively.

**4.2. The AES ciphers**

AES is based on design principle known as substitution-permutation network which is efficient in both software and hardware. The input is a single 128-bit block for decryption and encryption and is known as the IN matrix. This block is copied into a state array which is modified at each stage of the algorithm and then copied to an output matrix.). Both the plaintext and key are depicted as a 128-bit square matrix of bytes. This key is then expanded into an array of key schedule words (the w matrix). It must be noted that the ordering of bytes within the in matrix is by column. The same applies to the w matrix.

**4.3. AES Encryption process:**

The key size used for an AES cipher specifies the number of transformations rounds that convert the input, called the plaintext into the final output, called the ciphertext. The number of rounds is as follows:

* 10 rounds for 128-bit keys.
* 12 rounds for 192-bit keys.
* 14 rounds for 256-bit keys.

Inside each round comprise of four sub-process. Each round consists of the four steps to encrypt 128-bit block. SubByte, ShiftRow, Mixcolumn and AddRoundKey transformations on the state block array in addition to an initial round key. The round function repetition of 10, 12 or 14 rounds depends on the key length.

1. **Substitute Bytes Transformation:**

The first stage of every round is initiated with SubBytes transformation. In the step, each byte in the matrix is updated using an 8-bit nonlinear substitution box (Rijndael S-box). The S-box used is derived from multiplicative inverse over GF (28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. According to diffusion and confusion Shannon’s principles for cryptographic algorithm design it has important roles to obtain much more security. For each round, each byte of the matrix is mapped into another byte by following way: the leftmost nibble of the byte is used to specify a particular row of the s-box and rightmost specifies the column. For example, in AES if we have hex 53 in the state, it has to replace to hex ED. ED created from the intersection of 5 and 3. For remaining bytes of the state have to perform this operation.

[fig: Table 1 AES S-box Table]

[Fig. 3 Substitute byte transformation]

1. **ShiftRows Transformation:**

The shiftRows step operates on rows of the state. The purpose of this step is to shift bytes of the state cyclically to the left in each row other than row number zero.

• The first row of state is not altered.

• The second row is shifted 1 byte to the left in a circular manner.

• The third row is shifted 2 bytes to the left in a circular manner.

• The fourth row is shifted 3 bytes to the left in a circular manner.

[fig ShiftRows transformation]

1. **MixColumn Transformation:**

In the mixColumn step, each byte of one row in matrix transformation multiply by each value (byte) of the state column. The results of these multiplication are used with XOR to produce a new four bytes for the next state. Together with shiftRows, mixColumn provides diffusion in the cipher.

[fig multiplication matrix]

During this operation, each column is multiplied by the known matrix that for 128-bit key is

[fig. MixColumns stage]

The multiplication operation is defined as: multiplication by 1 means leaving unchanged, multiplication by 2 means shifting byte to left and multiplication by 3 means shifting to left and then performing xor with the initial unshifted value. After shifting, a conditional xor with 0x11B should be performed if the shifted value is larger than 0xFF.

S'(0,j) = (2 • s0,j ) ⊕ (3 • s1,j ) ⊕ s2,j ⊕ s3,j

S'(1, j) = s0,j ⊕ (2 • s1,j ) ⊕ (3 • s2,j ) ⊕ s3,j

S'(2, j) = s0,j ⊕ s1,j ⊕ (2 • s2,j ) ⊕ (3 • s3,j )

S'(3, j) = (3 • s0,j ) ⊕ s1,j ⊕ s2,j ⊕ (2 • s3,j )

1. **Add Round Key Transformation:**

The last stage of each round of encryption is to add the round key. (In fact, this is also done before the first round.) Before the first round, the first two words (W0 and W1) of the expanded key are added. Both the 128 bits of the round key and the 128 bits of state data (also referred to as the input) are structured in a 4x4 matrix of bytes. The Round Key is derived from the Cipher Key by means of the key schedule. In this stage state and key are bitwise XORed. In the first round, W2 and W3 are added. In the last round, W4 and W5 are added. All additions are done modulo 2, that is, with XOR. AddRoundKey has the ability to provide much more security during encrypting data. This operation is based on creating the relationship between the key and the cipher text. The cipher text is coming from the previous stage. The AddRoundKey output exactly relies on the key that is indicated by users. Furthermore, in the stage the subkey is also used and combined with state. The main key is used to derive the subkey in each round by using Rijndael's key schedule. The size of subkey and state is the same. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

[fig. Add Round Key]

**AES Key Expansion:**

AES algorithm is based on AES key expansion to encrypt and decrypt data. It is another most important steps in AES structure. Each round has a new key. The key expansion routine creates round keys word by word, where a word is an array of four bytes. The routine creates 4x (Nr+1) words. Where Nr is the number of rounds.

The cipher key (initial key) is used to create the first four words. The size of key consists of 16 bytes (k0 to k15) as shown in Fig.8 that represents in an array. The first four bytes (k0 to k3) represents as w0, the next four bytes (k4 to k7) in first column represents as w1, and so on. We can use particular equation to calculate and find keys in each round easily as follows:

• K [n]: w[i] = k [n-1]: w[i] XOR k[n]: w[i]

This equation uses to find a key for each round rather than w0. For w0 we have to use particular equation that is different from above equation.

• K[n]: w0 = k [n-1]: w0 XOR SubByte (k [n-1]: w3>>8) XOR Rcon [i].

[fig. AES key expansion]

 AES Key Expansion Example

K1:

W0 = 0f 15 71 c9

W1 = 47 d9 e8 59

W2 = 0c b7 ad e8

W3 = af 7f 67 98

To find K2:

K2 = w0 = k1: w0 XOR SubByte (k1:w3>>8) XOR Rcon [2]

0f 15 71 c9 XOR SubByte (af 7f 67 98>>8) XOR Rcon [2]

Rcon [2] from Auxiliary function = 02 00 00 00

0f 15 71 c9 XOR SubByte (7f 67 98 af ) XOR 02 00 00 00

K2 = w0 = df q0 37 b0

K2: w1 = k1: w1 XOR k2: w0

47 d9 e8 59 XOR df q0 37 b0

K2: w1 = 98 49 df e9

K2: w2 = k1: w2 XOR k2: w1

In this example we have found W0 and W1. In a similar way we can find W2 and W3.

[fig: AES key expansion]

**4.3. AES Decryption process:**

The decryption is the process to obtain the original data that was encrypted. This process is based on the key that was received from the sender of the data. The decryption processes of an AES is similar to the encryption process in the reverse order and both sender and receiver have the same key to encrypt and decrypt data. The last round of a decryption stage consists of three stages such as InvShiftRows, InvSubBytes, and AddRoundKey.

[fig decryption flowchart]

1. **Inverse shiftRow Transformation:**

Inverse Shift Rows 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. The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb-shift (r, Nb) bytes, where the shift value shift (r, Nb) de pends on the row number.

[fig Inverse shift row transformation]

1. **Inverse substitute byte Transformation:**

Inverse Substitute Bytes is the inverse of the byte substitution transformation, in which the inverse S-box is applied to each byte of the State. It is reverse process of Substitute byte transform. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF (2^8). There is an inverse s-box table for substitute.

[fig Inverse s-box]

1. **Inverse mixColumn Transformation:**

Inverse Mix Columns is the inverse of the Mix Columns transformation. Inverse Mix Columns operates on the State column-by-column, treating each column as a four-term polynomial.