

Secret File Sharing Techniques using **AES algorithm**

C. Navya Latha
Garima Agarwal
Anila Kumar GVN

200201066
200305032
200305002

1. Feature Overview

The Advanced Encryption Standard (AES) feature adds support for the new encryption standard AES, with Cipher Block Chaining (CBC) mode, to IP Security (IPSec). The National Institute of Standards and Technology (NIST) have created AES, which is a new Federal Information Processing Standard (FIPS) publication that describes an encryption method. AES is a privacy transform for IPSec and Internet Key Exchange (IKE) and has been developed to replace the Data Encryption Standard (DES). AES is designed to be more secure than DES: AES offers a larger key size, while ensuring that the only known approach to decrypt a message is for an intruder to try every possible key. AES has a variable key length-the algorithm can specify a 128-bit key (the default), a 192-bit key, or a 256-bit key.

The project aims at implementing the AES algorithm, such that whenever we need to share a secret file it can be encrypted using this algorithm and transmitted to the destination and decrypted at the destination side to get the original file.

2. Benefits

AES is a cryptographic algorithm that protects sensitive, unclassified information. It has the following characteristics:

1. Resistance against all known attacks
2. Speed and code compactness on a wide range of platforms
3. Design Simplicity

3. AES Structure

Following are the characteristics of the AES structure:

- (i) AES Algorithm processes the entire data block in parallel during each round using substitutions and permutations.
- (ii) The key that is provided as input is expanded into an array of forty-four 32bit words. Four distinct words serve as a round key for each round.
- (iii) Four different stages are used , one of permutation and three of substitution

Substitute Bytes : Uses an S-box to perform a byte-to-byte substitution of the block

Shift Rows : Simple Substitution

Mix columns : A substitution that makes use of Arithmetic cover

Add round key : A simple bitwise XOR of the current block with a portion of the expanded key

- (iv) For both encryption and decryption, the cipher begins with an Add Round Key stage, followed by nine rounds that each includes all four stages, followed by the tenth round of three stages.
- (v) Only the Add Round Key stage makes use of the key .and thus provides security
- (vi) Each stage is reversible.
- (vii) The decryption algorithm makes use of the expanded key in reverse order.
- (viii) The final round of the encryption and decryption consists of only three stages. This is a consequence of the particular structure of AES and is required to make cipher reversible.

4. Implementation Overview

This project is intended to develop a simple utility like WordPad but uses AES algorithm for document exchange in an encrypted manner rather than in plain text. Only authenticated users of the utility or users allowed by the document owner can view that document.

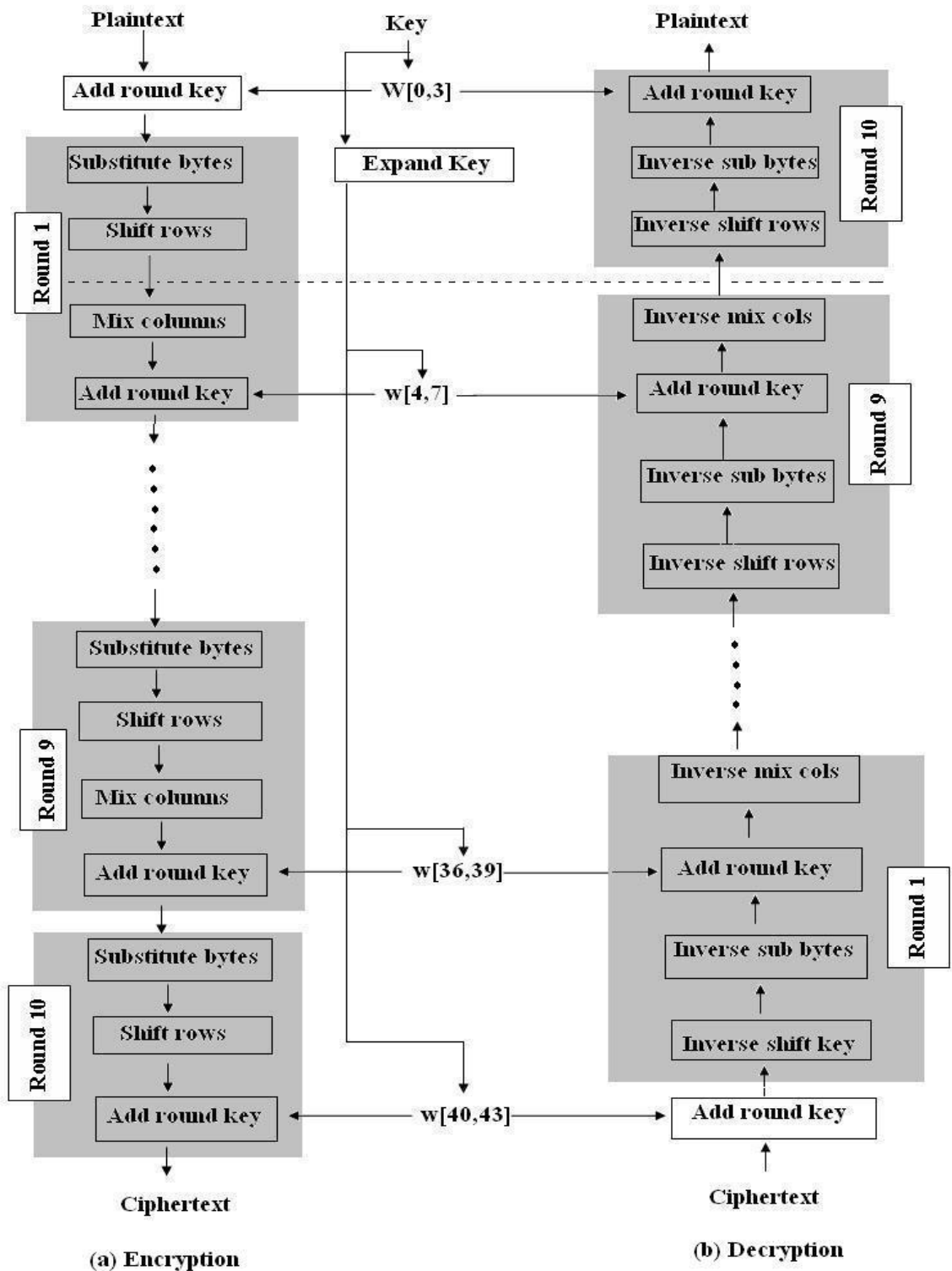
Implementing the AES algorithm will consists of the functions that take care of the four stages used in all the rounds of the algorithm as mentioned above. AES technique can be implemented using a key length of 128 bits or 192 bits or 256 bits having 10 rounds, 12 rounds, 14 rounds respectively. We will be dealing with 128 bit key length having 10 rounds since this is the most commonly used technique. An attack against 128-bit key AES requires 2^{128} operations which will be quite infeasible. Secondly the mathematical structure of AES is very neat which cannot lead to any attacks.

5 Environment

The Algorithm and the utility are going to be developed in Java which makes the whole setup platform independent.

The figure given below shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. The 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and the total key schedule is 44 words for the 128-bit key.

Following the figure is a brief description of each of the four stages used in AES. This is followed by a brief discussion of key expansion. For each stage we described forward (encryption) algorithm and inverse (decryption) algorithm, and the rationale of the stage.



Block Diagram of AES Algorithm

Substitute Bytes Transformation

Forward and Inverse Transformation

The forward substitute byte transformation is simple table lookup. AES defines 16 X 16 matrixes of byte values, called an S-box, which contains a permutation of all possible 256 8-bit values. Each individual byte of State is mapped into new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value.

The inverse substitute byte transformation makes use of the inverse S-box which is constructed by applying the inverse of the transformation of the actual equation followed by taking the multiplicative inverse of $GF(2^8)$.

Shift Row Transformation

Forward and Inverse Transformations

In forward shift row transformation the first row of the State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed.

The inverse shift row transformation performs the circular shifts in the opposite direction for each of the last three rows, with a one-byte circular right shift for the second row and so on.

Mix Column Transformation

Forward and Inverse Transformations

The forward mix column transformation operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column. Each element in the product matrix is the sum of products of elements of one row and one column. In this case, the individual additions and multiplications are performed on $GF(2^8)$.

During the inverse mix column transformation, the inverse matrix times the forward transformation matrix equals the identity matrix.

Add Round Key Transformation

Forward and Inverse Transformations

In the forward add round key transformation the 128 bits of State are bitwise XORed with the 128 bits of the round key. The operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key; it can also be viewed as a byte-level operation.

The inverse add round key transformation is identical to the forward add round key transformation, because the XOR operation is its own inverse.

AES Key Expansion

Key Expansion Algorithm

The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial Add Round Key stage and each of the 10 rounds of the cipher.

The key is copied into the first four words of the expanded key. The remaining of the expanded key is filled in four words at a time. Each added word $w[i]$ depends on the immediately preceding word $w[i-1]$, and the word four positions back, $w[i-4]$. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used.