Securing Files Using AES Algorithm

Aditya Rayarapu, AbhinavSaxena, N.Vamshi Krishna,Diksha Mundhra

Department of Computer Science Engineering,
Jawharlal Nehru Technological University
VignanaBharathi Institute of Technology,
Ghatkesar, Hyderabad, India.

Abstract—**Nowadays information systems play a vital role for organizations and individuals, in which the security is given the high priority. Often, solutions are developed for very similar problems over and over again. In this paper, the files are encrypted and stored in the disk by using a secret key and asking for the same secret key while decrypting. The algorithm used in this system is Advance Encryption Standard (AES). AES-128, AES-192 and AES-256 represents the encryption key sizes (128 bits, 192 bits and 256 bits) and in their number of rounds (10, 12, and 14, respectively) required to open the vault that is wrapped around the data.[1] In this algorithm, encryption is done by interchanging some of the characters with key and data in it. The encrypted files are set to be read only, so that the data in the files cannot be tampered. The main feature of the system is disabling the delete option in the right click menu for the encrypted files. This provides more security for the files on the disk.**

Keywords— Encryption, Decryption, Multiple Files, Encrypted Files, Decrypted Files, Advanced Encryption Standards (AES), Security, Secret Key.

INTRODUCTION

Organizations, Corporate Companies, Government agencies, Schools, Hospitals, and enterprises - all of these organizations have the confidential data on their desktop or in the drive. To provide security to all these confidential data on the desktop by converting the plain text to cipher text when the file is encrypted, which is set to be read only and delete option is disabled for all the encrypted files. The idea of using this algorithm is, AES is unbreakable when compared to DES and IDEA algorithm. Unlike DES can be attacked by a high-order differential attack requiring 2^{64}-2^{52} chosen plain texts breaks 6 rounds with a complexity of $2^{126.8}$ encryptions. DES is insecure because, a brute force attack is possible. Other than this, there are few other attacks which proves DES algorithm insecure are one round attack, full 16-round attack and Meet-in-the-middle attack

EXISTING SYSTEM

The necessity of providing security to files on the desktop has been essential since many of them implemented different algorithms and techniques to provide security. At present there are many algorithms such as DES, IDEA, and RSA. The main disadvantage is the above algorithms are breakable at certain point. There exists the problem for decrypting the file unless the secret key entered for encryption and decryption are similar. After encryption of files, there is no security of file deletion using right click menu. In this case the encrypted files can be easily deleted. To avoid these drawbacks, the proposed system has few enhancements in securing the information in the disk.

ADVANCE ENCRYPTION STANDARD

AES is the Advanced Encryption Standard, a United States government standard algorithm for changing the plain text to cipher text i.e. encrypting and decrypting the data. The National Institute of Standards and Technology (NIST) published a request for comments for the "Development of a Federal Information Processing Standard for Advanced Encryption Standard" on January 2, 1997. NIST searched for alternatives that have higher level of security than that offered by the other algorithms such as DES, IDEA and RSA. Data Encryption Standard (DES) which grew vulnerable to brute-force attacks due to its 56-bit effective key length. AES candidates were required to support a symmetric block cipher that supported multiple key lengths. The algorithm had to be publicly defined, free to use, and able to run efficiently in both hardware and software. The central design principle of the AES algorithm is the adoption of symmetry at different platforms and the efficiency of processing. After a 5-year standardization process, the NIST adopted the Rijndael algorithm as the Advance Encryption Standard (AES).
AES is a symmetric block cipher with a block size of 128 bits. Key lengths can be 128bits, 192 bits, or 256 bits;8 called AES-128, AES-192 and AES-256 respectively.

Version	Key Length (Nk Words)	Block Size (Nb Words)	Number of Rounds (Nr)
AES - 128	4	4	10
AES - 192	6	4	12
AES - 256	8	4	14

Table 1: AES Versions

The main loop of AES performs the following methods:

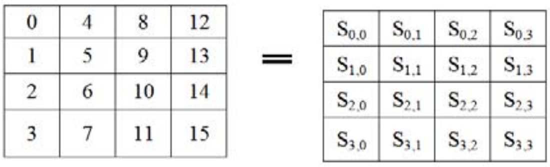
1. Convert to State Array
2. Transformations (and Their Inverse)
    i. AddRoundKey
    ii. SubBytes
    iii. ShiftRows
    iv. MixColumns
3. Key Expansion

1. CONVERT TO STATE ARRAY

A term associated with AES is "the State," an 'intermediate cipher,' or the ciphertext before the final round has been applied. AES formats plaintext into 16 byte (128-bit) blocks, and treats each block as a 4x4 State array. It then performs four operations in each round. The arrays contains row and column information used in the operations, especially MixColumns() and Shiftrows().

2. TRANSFORMATIONS (AND THEIR INVERSE)

The round transformation modifies the 128-bit State. The initial State is the input plaintext and the final State is the output cipher text. The State is organized as a 4 X 4 matrix of bytes. The round transformation scrambles the bytes of the State either individually, rowwise, or columnwise by applying the functions SubBytes, ShiftRows, MixColumns, and AddRoundKey sequentially.



0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

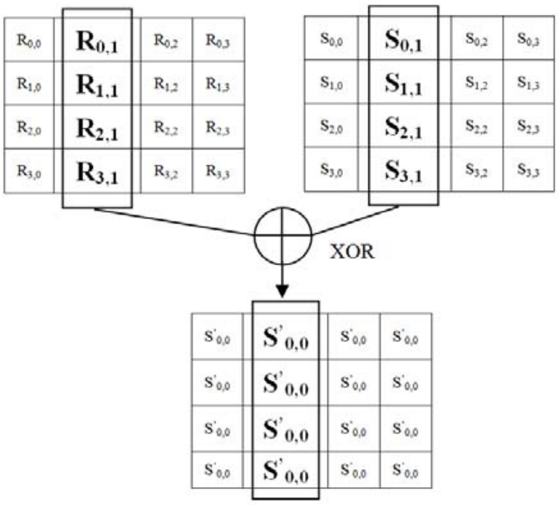
=

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Fig: Transformation of Matrix

I. ADDROUNDKEY

Each of the 16 bytes of the state is XORed against each of the 16 bytes of a portion of the expanded key for the current round. The Expanded Key bytes are never reused. So once the first 16 bytes are XORed against the first 16 bytes of the expanded key then the expanded key bytes 1-16 are never used again. The next time the Add Round Key function is called bytes 17-32 are XORed against the state. [3]



$R_{0,0}$	$R_{0,1}$	$R_{0,2}$	$R_{0,3}$
$R_{1,0}$	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$
$R_{2,0}$	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$
$R_{3,0}$	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

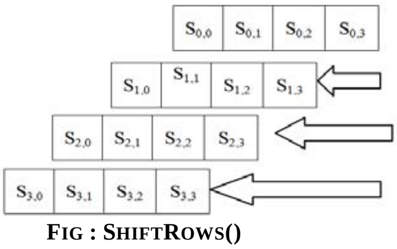
$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$
$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$
$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$
$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$	$S'_{0,0}$

II. SUBBYTES

Replace each byte in the state array with its corresponding value from the S-Box. An S-Box is asubstitution table, where one byte is substituted for another, based on a substitution algorithm.

III. SHIFTROWS

ShiftRows() provides diffusion by mixing data within rows. Row zero of the State is not shifted, row 1 is shifted 1 byte, row 2 is shifted 2 bytes, and row 3 is shifted 3 bytes,



$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

FIG : SHIFTROWS()

MIXCOLUMNS

MixColumns()also provides diffusion by mixing data within columns. The 4 bytes of each column in the State are treated as a 4-byte number and transformed to another 4-byte number via finite field mathematics,

- ❖ $S'_{0,C} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,C}$
- ❖ $S'_{1,C} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$
- ❖ $S'_{2,C} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$
- ❖ $S'_{3,C} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c}$

KEY EXPANSION

AES uses a key schedule to expand a short key into a number of separate round keys. A second block of material in understanding the AES algorithm is the routine for expansion of a 128 bit key into a collection of 44 round keys of 8 bits each.The procedure for expanding a key requires the use of the S-Box used for AES. If that is already loaded the first section can be skipped. [2]

```
KeyExpansion( Byte key( 4*Nk ),  word w ( Nb*( Nr+1) ), Nk )
Begin
        word temp
        i=0
        While (I <Nk)
        w[i] = word ( key [ 4*I ], key [ 4*i+1 ], key [4*i+2], key [
        4*i+3 ] )
        i=i+1
        end while
        i=Nk
        while ( i<Nb * (Nr+1) )
        temp  = w[ i – 1 ]
        if  ( i mod Nk = 0)
        temmp = Subword(RotWord ( temp )
      ) xorRcon[i/Nk]
        else if ( Nk> 6 and I mod Nk = 4 )
                temp = SubWord ( temp )
        end if
w[i] = w[ i-Nk ] xor temp
i  =  i + 1
end while
end
```

PROPOSED SYSTEM

Information security (sometimes shortened to InfoSec) is the practice of defending information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. It is a general term that can be used regardless of the form the data may take (electronic, physical, etc...).[4] Since the early days of writing, politicians, diplomats and military commanders understood that it was necessary to provide some mechanism to protect the confidentiality of correspondence and to have some means of detecting tampering.

The proposed system uses AES to encrypt the files with a secret key to secure the confidential data on a desktop. Consider a user has multiple bank account details stored on the desktop. To provide security to the desktop files, the proposed system is used. Initially user will browse the text file for encrypting it. The user has to give 16 bytes of SecretKey twice to confirm the SecretKey. If the SecretKey entered by user matches then the file is encrypted successfully else, a pop up message will be displayed as File cannot be Encrypted. After encryption process, the file is stored on a disk with FilenameEncrypted.txt. In this case, there is chance of manipulating the data in the encrypted text file. In order to overcome, the encrypted file is set as read only such that the file is not modified. At a time three files can be encrypted and store it in a disk.

Once the file is encrypted, during the process of decryption, the user has to enter the same SecretKey which is used for the encryption of file. If the SecretKey entered is same while encryption process, then file will be decrypted successfully else, file cannot be decrypted message is displayed.

In all the existing system, the encrypted files can be deleted where there is no security for the files where the data is lost. In order to overcome this drawback, the delete option in the right click menu is disabled in the right click menu for all the encrypted files. By this the encrypted files cannot be deleted from the disk. Hence the important information in the drive after encryption is secure.

IMPLEMENTATION

The following pseudo code shows the implementation of the proposed system.

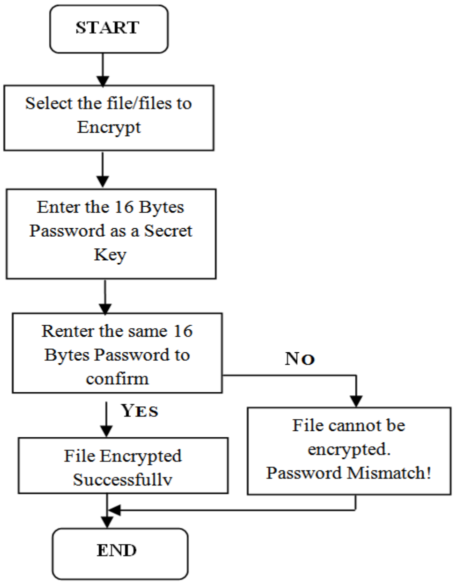

START
Select the file/files to Encrypt
Enter the 16 Bytes Password as a Secret Key
Renter the same 16 Bytes Password to confirm
NO
YES
File Encrypted Successfullv
File cannot be encrypted. Password Mismatch!
END

FIG: ENCRYPTION

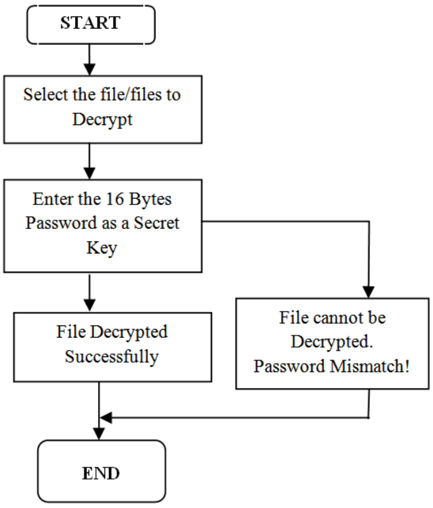

START
Select the file/files to Decrypt
Enter the 16 Bytes Password as a Secret Key
File Decrypted Successfully
File cannot be Decrypted. Password Mismatch!
END

FIG: DECRYPTION

ALGORITHM

```
Algorithm Start (FileName, SecretKey) {
        Browse files
        Enter 16 byte SecretKey
}
Algorithm Encrypt() {
        Convert to State Array
        AddRoundKey()
        SubBytes()
        ShiftRows()
        MixColumns()
        Key Expansion
}
Algorithm Decrypt {
        Convert to State Array
        AddRoundKey()
        InvSubBytes()
        InvShiftRows()
        InvMixColumns()
        Key Expansion
}
```

WORKING OF PROPOSED SYSTEM

1. Initially the user selects the file from the disk for the encryption process.
2. On selecting the files, the user must enter 16 bytes SecretKey as a password.
3. After entering the 16 bytes SecretKey, the user must re-enter to confirm password and Click on encrypt.
4. The encrypted file is created with FilenameEncryption.txt in a disk. The encrypted files are set as read only.
5. To decrypt the encrypted text files, go to the decrypt screen. Select the encrypted files and enter the same SecretKey which is used for encryption process.
6. In this system, the encrypted files cannot be deleted and delete option in right click menu is disabled for all encrypted files. This feature provides security.

ANALYSIS AND CONCLUSION

This proposed system can be used for securing files on the desktop which has important files such as details of multiple accounts, multiple numbers of files which has the confidential data. This system provides security to those files. The encrypted files cannot be tampered because these are set to read only. This provides security for tampering the data.

REFERENCES

[1] Key sizes of 128, 160, 192, 224, and 256 bits are supported by the Rijndael algorithm, but only the 128, 192, and 256-bit key sizes are specified in the AES standard.

[2] csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[3] "Efficient software implementation of AES on 32-bit platforms". Lecture Notes in Computer Science: 2523. 2003

[4] en.wikipedia.org/wiki/Information_security#cite_note-1.

[5] Nikolić, Ivica (2009). "Distinguisher and Related-Key Attack on the Full AES-256". *Advances in Cryptology – CRYPTO 2009*. Springer Berlin / Heidelberg. pp. 231–249. doi:10.1007/978-3-642-03356-8_14. ISBN 978-3-642-3355-1.