ijecs open access www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 6 June, 2014 Page No. 6405-6408

Advance Encryption Standard implementation

Sher jung , Shikha Kumari, Gagan Thakral
Department of Computer Science
Al-Falah School of Eng. & Tech.
Faridabad, India
Sherjung2005@gmail.com

Department of Computer Science
JB Knowledge park
Faridabad, India
shikhadivakar@gmail.com

Department of Computer Science
JB Knowledge Park
Faridabad, India
gagan15.thakral@gmail.com

ABSTRACT

In today's world most of the communication is done electronic media. Data security plays vital role in such communication. Hence there is need to protect data from malicious attack. This can be achieved by cryptography. The earlier encryption algorithm was DES which has several loopholes such as small key size, sensible to brute force attack etc. These loopholes were overcome by AES algorithm. AES was announced by NIST (National institute of standard and technology) in November 2001. AES became the successor of DES algorithm because of its security, its ease of implementation, and low memory requirements.

There are three different versions of AES. Each of them has block length 128 bits and key length is 128, 192 and 256 bits

Keywords:

Encryption, decryption, ciphers, S-box, State, keys.

1. AES algorithm basic concept

AES is a symmetric key cryptography. It contains no. of rounds, which are fixed. It is a block cipher technique in which each cipher has variable block length and key length. AES supports block size of 128 bits and key size of 128,192 and 256 bits. To encrypt messages longer than the block size, a mode of operation is chosen. 128 bit key contains 10 rounds, 192 bit contains 12 rounds and 256 bit contains 14 rounds. In each round transformation is done using corresponding cipher key to ensure security of encryption. Different transformation operates on immediate results, called **state**. Let us consider key length of 128 bits. State is rectangular array of bytes and since key size is 128 bits (16 bytes) so rectangular array is of 4X4matrx. First 128 bit is expanded into 11 round key, each of 128 bits. Each round key is of length 128 bits. Initial round is XORed to plain text. Each round key consists following operation:-

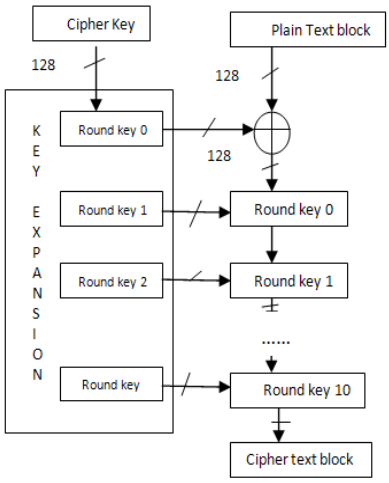
Substitute bytes: In this byte to byte transformation is performed using a table.

Shift rows: rows are shifted cyclically to left.

Mix column: Matrix multiplication on column is performed.

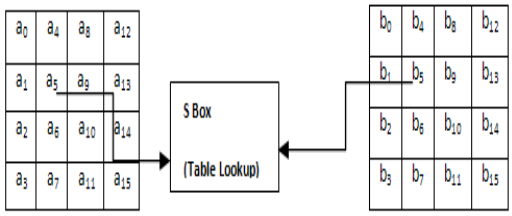
Add Round Key: state and expanded key is XORed.

In tenth round mix column round is omitted. Rest is same as that of round one to nine.



2. Sub Bytes

AES uses S-Box substitution, which is also called sub byte transformation. It is non linear substitution technique. It provides reversible transformation on each segment of plain text during encryption which reverses during decryption. In this simple single function is applied over again and again to each byte during stages of encryption returning a single byte. Each byte is transformed to other byte and called as full transformation.



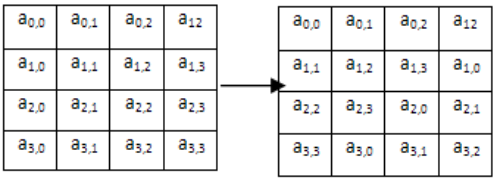

```
void subBytes(unsigned char *state)
{
int j;
for (j= 0; j < 16; j++)

state[j] = getSBoxValue(state[j]);
}
```

3. Shift Rows

In this left circular shift rows is performed. Each row is cyclically shifted to left.

The 1st row is shifted 0 positions to the left, 2nd row is shifted 1 position to the left, 3rd row is shifted 2 positions to the left, 4th row is shifted 3 positions to the left.



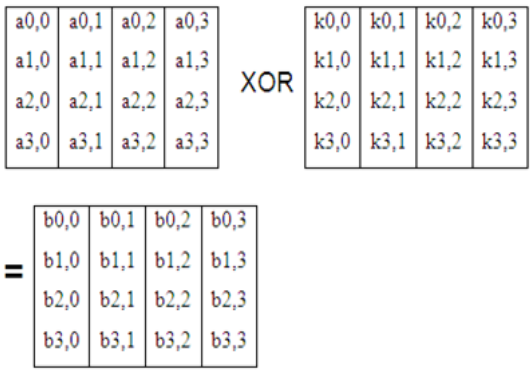
```
void shiftRows(unsigned char *state)
{
int i;
/* iterate over the 4 rows and call shiftRow() with that
row */
for (i = 0; i < 4; i++)
shiftRow(state+i*4, i);
}
void shiftRow(unsigned char *state, unsigned char nbr)
{
int i, j;
unsigned char tmp;
/* each iteration shifts the row to the left by 1 */
for (i = 0; i < nbr; i++)
{
tmp = state[0];
for (j = 0; j < 3; j++)
state[j] = state[j+1];
state[3] = tmp;
}
}
```

4. Mix Column

In this columns are processed. A inear transformation on columns of the state is performed. In this principle only a matrix multiplication needs to be executed.

5. Add Round Key

In this, each byte of input data and expanded key is XORed. One AddRoundKey is applied before first round. Also the third transformation i.e MixColumn is missing in last round.



where b(i,j)=a(i,j) XOR k(i,j)

```
void addRoundKey(unsigned char *state, unsigned char
*roundKey)
{
int i;
for (i = 0; i < 16; i++)
state[i] = state[i] ^ roundKey[i] ;
}
```

6. Implementation of AES algorithm

Each round in AES algorithm contains all the four operations on the state.

```
void aes_round(unsigned char *state, unsigned char
*roundKey)
{
subBytes(state);
shiftRows(state);
mixColumns(state);
addRoundKey(state, roundKey);
}
```

7. AES Encryption

Input plain text,the key size and the output are the parameters. Number of rounds based on key size is calculated and then calculate the expanded key size based on no of rounds.

```
char aes_encrypt(unsigned char *input, unsigned char
*output, unsigned char *key,
enum keySize size)
{
/* the expanded keySize */
int expandedKeySize;
/* the number of rounds */
int nbrRounds;
/* the expanded key */
unsigned char *expandedKey;
/* the 128 bit block to encode */
unsigned char block[16];
int i,j;
/* set the number of rounds */
switch (size)
{
case SIZE_16:
nbrRounds = 10;
break;
case SIZE_24:
nbrRounds = 12;
break;
case SIZE_32:
nbrRounds = 14;
break;
default:
return UNKNOWN_KEYSIZE;
break;
}
```

8. AES Decryption

The whole encryption process is reversed. The subbytes, mix column and shift rows column operation is reversed. But add roundkey operation remains same.

```
void invSubBytes(unsigned char *state)
{
int i;
/* substitute all the values from the state with the value
in the SBox
* using the state value as index for the SBox
*/
for (i = 0; i < 16; i++)
state[i] = getSBoxInvert(state[i]);
}
void invShiftRows(unsigned char *state)
{
int i;
/* iterate over the 4 rows and call invShiftRow() with
that row */
for (i = 0; i < 4; i++)
invShiftRow(state+i*4, i);
}
void invShiftRow(unsigned char *state, unsigned char
nbr)
{
int i, j;
unsigned char tmp;
/* each iteration shifts the row to the right by 1 */
for (i = 0; i < nbr; i++)
{
tmp = state[3];
for (j = 3; j > 0; j--)
state[j] = state[j-1];
state[0] = tmp;
}
}


void invMixColumns(unsigned char *state)
{
int i, j;
unsigned char column[4];
/* iterate over the 4 columns */
for (i = 0; i < 4; i++)
```

```
{
/* construct one column by iterating over the 4 rows */
for (j = 0; j < 4; j++)
{
column[j] = state[(j*4)+i];
}
/* apply the invMixColumn on one column */
invMixColumn(column);
/* put the values back into the state */
for (j = 0; j < 4; j++)
{
state[(j*4)+i] = column[j];
}
}
}
void invMixColumn(unsigned char *column)
{
unsigned char cpy[4];
int i;
for(i = 0; i < 4; i++)
{
cpy[i] = column[i];
}
column[0] = galois_multiplication(cpy[0],14) ^
galois_multiplication(cpy[3],9) ^
galois_multiplication(cpy[2],13) ^
galois_multiplication(cpy[1],11);
column[1] = galois_multiplication(cpy[1],14) ^
galois_multiplication(cpy[0],9) ^
galois_multiplication(cpy[3],13) ^
galois_multiplication(cpy[2],11);
column[2] = galois_multiplication(cpy[2],14) ^
galois_multiplication(cpy[1],9) ^
galois_multiplication(cpy[0],13) ^
galois_multiplication(cpy[3],11);
column[3] = galois_multiplication(cpy[3],14) ^
galois_multiplication(cpy[2],9) ^
galois_multiplication(cpy[1],13) ^
galois_multiplication(cpy[0],11);
}
```

9. CONCLUSION

In this paper I discussed AES algorithm and implementation of each round. I have also discussed implementation of encryption and decryption.

REFERENCES

[1] AES page available via http://www.nist.gov/CryptoToolkit.

[2] Computer Security Objects Register (CSOR): http://csrc.nist.gov/csor/.

[3] Cryptography and network security, 2nd edition by Atul Kahate**.**

[4] J. Daemen and V. Rijmen, *AES Proposal: Rijndael*, AES Algorithm Submission, September 3, 1999, available at [1].

[5] J. Daemen and V. Rijmen, *The block cipher Rijndael*, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

[6] B. Gladman's AES related home page http://fp.gladman.plus.com/cryptography_technology/.

[7] A. Lee, NIST Special Publication 800-21, *Guideline for Implementing Cryptography in the Federal Government*, National Institute of Standards and Technology, November 1999.