

Walchand College Of Engineering, Sangli
Department of Computer Science and Engineering
Subject: C&NS Lab

Batch: B4

Name: Gayatri Sopan Gade

PRN:2020BTECS00210

Assignment 7

Title: Advanced Encryption Standard

Aim: To Demonstrate Advanced Encryption Standard

Theory:

AES algorithm (Rijndael algorithm) is a symmetric block cipher algorithm. The length of the data packet must be 128 bits, and the length of the key used should be 128, 192 or 256 bits. For three AES algorithms with different key lengths, they are called "AES-128", "AES-192", "AES-256".

Code:

decoding.h

```
/*  
this header file implements the algorithm for 128-bit decryption  
*/  
#include<iostream>  
#include "lookup_table_decoding.h"  
#include "key_expand.h"  
using namespace std;  
void decryption(unsigned char * temp,unsigned char * extendedkeys)  
{
```

```

int kp=10;
while(kp>0)
{
    //subtract round key
    for(int i=0;i<16;i++)
    {
        temp[i]^=extendedkeys[(kp*16)+i];
    }

    //inverse mix column step
    if(kp<10){
        unsigned char temp2[16];
        for (int i = 0; i < 16; i++)
        {
            temp2[i] = temp[i];
        }

        temp[0] = (unsigned char)lookup14[temp2[0]] ^
lookup11[temp2[1]] ^ lookup13[temp2[2]] ^ lookup9[temp2[3]];
        temp[1] = (unsigned char)lookup9[temp2[0]] ^
lookup14[temp2[1]] ^ lookup11[temp2[2]] ^ lookup13[temp2[3]];
        temp[2] = (unsigned char)lookup13[temp2[0]] ^
lookup9[temp2[1]] ^ lookup14[temp2[2]] ^ lookup11[temp2[3]];
        temp[3] = (unsigned char)lookup11[temp2[0]] ^
lookup13[temp2[1]] ^ lookup9[temp2[2]] ^ lookup14[temp2[3]];

        temp[4] = (unsigned char)lookup14[temp2[4]] ^
lookup11[temp2[5]] ^ lookup13[temp2[6]] ^ lookup9[temp2[7]];

```

```

    temp[5] = (unsigned char)lookup9[temp2[4]] ^
lookup14[temp2[5]] ^ lookup11[temp2[6]] ^ lookup13[temp2[7]];
    temp[6] = (unsigned char)lookup13[temp2[4]] ^
lookup9[temp2[5]] ^ lookup14[temp2[6]] ^ lookup11[temp2[7]];
    temp[7] = (unsigned char)lookup11[temp2[4]] ^
lookup13[temp2[5]] ^ lookup9[temp2[6]] ^ lookup14[temp2[7]];

    temp[8] = (unsigned char)lookup14[temp2[8]] ^
lookup11[temp2[9]] ^ lookup13[temp2[10]] ^ lookup9[temp2[11]];
    temp[9] = (unsigned char)lookup9[temp2[8]] ^
lookup14[temp2[9]] ^ lookup11[temp2[10]] ^ lookup13[temp2[11]];
    temp[10] = (unsigned char)lookup13[temp2[8]] ^
lookup9[temp2[9]] ^ lookup14[temp2[10]] ^ lookup11[temp2[11]];
    temp[11] = (unsigned char)lookup11[temp2[8]] ^
lookup13[temp2[9]] ^ lookup9[temp2[10]] ^ lookup14[temp2[11]];

    temp[12] = (unsigned char)lookup14[temp2[12]] ^
lookup11[temp2[13]] ^ lookup13[temp2[14]] ^ lookup9[temp2[15]];
    temp[13] = (unsigned char)lookup9[temp2[12]] ^
lookup14[temp2[13]] ^ lookup11[temp2[14]] ^ lookup13[temp2[15]];
    temp[14] = (unsigned char)lookup13[temp2[12]] ^
lookup9[temp2[13]] ^ lookup14[temp2[14]] ^ lookup11[temp2[15]];
    temp[15] = (unsigned char)lookup11[temp2[12]] ^
lookup13[temp2[13]] ^ lookup9[temp2[14]] ^ lookup14[temp2[15]];
}

// Shifts rows right
unsigned char temp2[16];
for (int i = 0; i < 16; i++)
{

```

```
temp2[i] = temp[i];
}

//column one
temp [0] = temp2[0];
temp [4] = temp2[4];
temp [8] = temp2[8];
temp [12] = temp2[12];

//column two
temp [1] = temp2[13];
temp [5] = temp2[1];
temp [9] = temp2[5];
temp [13] = temp2[9];

//column three
temp [2] = temp2[10];
temp [6] = temp2[14];
temp [10] = temp2[2];
temp [14] = temp2[6];

//column four
temp [3] = temp2[7];
temp [7] = temp2[11];
temp [11] = temp2[15];
temp [15] = temp2[3];

//substitution bits
for(int i=0;i<16;i++)
{
    temp[i]=in_sbox[temp[i]];
}

kp--;
```

```

    }

    //subtract round key
    for(int i=0;i<16;i++)
    {
        temp[i]^=extendedkeys[i];
    }
}

```

encoding.h

```

/*
this header file implements the algorithm for 128-bit encryption
*/
#include<iostream>
#include "lookup_table_encoding.h"
#include "key_expand.h"
using namespace std;
void encryption(unsigned char * temp,unsigned char * extendedkeys )
{
    int kp=0;
    for(int i=0;i<16;i++)
    {
        temp[i]^=extendedkeys[i];
    }
    kp++;
    while(kp<11)
    {

```

```
//substitution bits
for(int i=0;i<16;i++)
{
    temp[i]=sbox[temp[i]];
}
//shift row
unsigned char * temp2 = new unsigned char[16];
for(int i=0;i<16;i++)
temp2[i]=temp[i];
//1st column
temp[0]=temp2[0];
temp[4]=temp2[4];
temp[8]=temp2[8];
temp[12]=temp2[12];
//2nd column
temp[1]=temp2[5];
temp[5]=temp2[9];
temp[9]=temp2[13];
temp[13]=temp2[1];
//3rd column
temp[2]=temp2[10];
temp[6]=temp2[14];
temp[10]=temp2[2];
temp[14]=temp2[6];
//4th column
temp[3]=temp2[15];
temp[7]=temp2[3];
temp[11]=temp2[7];
temp[15]=temp2[11];
```

```

//MIX column
if(kp<10)
{
    for (int i = 0; i < 16; i++) {
        temp2[i] = temp[i];
    }
    //1st row
    temp[0] = (unsigned char) lookup2[temp2[0]] ^
lookup3[temp2[1]] ^ temp2[2] ^ temp2[3];
    temp[1] = (unsigned char) temp2[0] ^
lookup2[temp2[1]] ^ lookup3[temp2[2]] ^ temp2[3];
    temp[2] = (unsigned char) temp2[0] ^ temp2[1] ^
lookup2[temp2[2]] ^ lookup3[temp2[3]];
    temp[3] = (unsigned char) lookup3[temp2[0]] ^
temp2[1] ^ temp2[2] ^ lookup2[temp2[3]];
    //2nd row
    temp[4] = (unsigned char)lookup2[temp2[4]] ^
lookup3[temp2[5]] ^ temp2[6] ^ temp2[7];
    temp[5] = (unsigned char)temp2[4] ^
lookup2[temp2[5]] ^ lookup3[temp2[6]] ^ temp2[7];
    temp[6] = (unsigned char)temp2[4] ^ temp2[5] ^
lookup2[temp2[6]] ^ lookup3[temp2[7]];
    temp[7] = (unsigned char)lookup3[temp2[4]] ^
temp2[5] ^ temp2[6] ^ lookup2[temp2[7]];
    //3rd row
    temp[8] = (unsigned char)lookup2[temp2[8]] ^
lookup3[temp2[9]] ^ temp2[10] ^ temp2[11];

```

```

        temp[9] = (unsigned char)temp2[8] ^
lookup2[temp2[9]] ^ lookup3[temp2[10]] ^ temp2[11];
        temp[10] = (unsigned char)temp2[8] ^ temp2[9] ^
lookup2[temp2[10]] ^ lookup3[temp2[11]];
        temp[11] = (unsigned char)lookup3[temp2[8]] ^
temp2[9] ^ temp2[10] ^ lookup2[temp2[11]];
        //4th row
        temp[12] = (unsigned char)lookup2[temp2[12]] ^
lookup3[temp2[13]] ^ temp2[14] ^ temp2[15];
        temp[13] = (unsigned char)temp2[12] ^
lookup2[temp2[13]] ^ lookup3[temp2[14]] ^ temp2[15];
        temp[14] = (unsigned char)temp2[12] ^ temp2[13] ^
lookup2[temp2[14]] ^ lookup3[temp2[15]];
        temp[15] = (unsigned char)lookup3[temp2[12]] ^
temp2[13] ^ temp2[14] ^ lookup2[temp2[15]];
    }

    //Add Round Key
    for(int i=0;i<16;i++)
    {
        temp[i]^=extendedkeys[kp*16+i];
    }
    kp++;
}
}

```


key_expand.h

```
/*
this header file includes algorithm for expanding our key
so that we can use our key for 10 rounds
*/
#ifndef KEY_EXPAND_H_INCLUDED
#define KEY_EXPAND_H_INCLUDED

// s-box table
unsigned char sbox[256] =
{
    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01,
    0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
    0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4,
    0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
    0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5,
    0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
    0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12,
    0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
    0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B,
    0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
    0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB,
    0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9,
    0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
    0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6,
    0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
```

```
    0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7,
0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
    0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE,
0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
    0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3,
0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
    0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56,
0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
    0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD,
0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
    0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35,
0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
    0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E,
0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
    0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99,
0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
};
```

```
// s-box table for decryption
```

```
unsigned char in_sbox[256] =
{
    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40,
0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
    0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E,
0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
    0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C,
0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B,
0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
```

```

    0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4,
0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
    0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15,
0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4,
0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
    0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF,
0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
    0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2,
0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9,
0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
    0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7,
0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
    0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB,
0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12,
0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
    0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5,
0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
    0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB,
0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69,
0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
};

// r-con table used in expansion
unsigned char r[256] = {

```

0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,
0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d,
0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f,
0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72,
0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74,
0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab,
0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,

```
    0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5,  
0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,  
    0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83,  
0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d  
};
```

```
//left shift row by one value
```

```
void leftshift(unsigned char * input)
```

```
{  
    unsigned char temp = input[0];  
    input[0] = input[1];  
    input[1] = input[2];  
    input[2] = input[3];  
    input[3] = temp;  
}
```

```
//function to substitute corresponding values in s-box
```

```
void sboxreplace(unsigned char * input)
```

```
{  
    input[0] = sbox[input[0]];  
    input[1] = sbox[input[1]];  
    input[2] = sbox[input[2]];  
    input[3] = sbox[input[3]];  
}
```

```
//generating 11 pairs of 128-bits keys
```

```
void Key_extenxion(unsigned char originalkey[16], unsigned char  
extended[176]) {
```

```
    // first key remains same same as original key
```

```

    for (int i = 0; i < 16; i++)
        extended[i] = originalkey[i];
    // variables to keep record of keys generated
    int nb = 16;
    int keysgenerated= 1;
    unsigned char tmp[4];

    while (nb < 176) {
        //initially start 4 bits will be same as last 4 generated
bits
        for (int i = 0; i < 4; i++)
            tmp[i] = extended[i + nb - 4];

        // main process for generating keys
        if (nb % 16 == 0)
        {
            leftshift(tmp);
            sboxreplace(tmp);
            tmp[0] ^= r[keysgenerated++];
        }

        for (int i = 0; i < 4; i++)
        {
            extended[nb]= extended[nb - 16] ^ tmp[i];
            nb++;
        }
    }
}

```

```
#endif // KEY_EXPAND_H_INCLUDED
```

lookup_table_decoding.h

```
//Galois Multiplication Lookup tables for decryption  
unsigned char lookup9[256] =  
{  
  
0x00,0x09,0x12,0x1b,0x24,0x2d,0x36,0x3f,0x48,0x41,0x5a,0x53,0x6c,0x6  
5,0x7e,0x77,  
  
0x90,0x99,0x82,0x8b,0xb4,0xbd,0xa6,0xaf,0xd8,0xd1,0xca,0xc3,0xfc,0xf  
5,0xee,0xe7,  
  
0x3b,0x32,0x29,0x20,0x1f,0x16,0x0d,0x04,0x73,0x7a,0x61,0x68,0x57,0x5  
e,0x45,0x4c,  
  
0xab,0xa2,0xb9,0xb0,0x8f,0x86,0x9d,0x94,0xe3,0xea,0xf1,0xf8,0xc7,0xc  
e,0xd5,0xdc,  
  
0x76,0x7f,0x64,0x6d,0x52,0x5b,0x40,0x49,0x3e,0x37,0x2c,0x25,0x1a,0x1  
3,0x08,0x01,  
  
0xe6,0xef,0xf4,0xfd,0xc2,0xcb,0xd0,0xd9,0xae,0xa7,0xbc,0xb5,0x8a,0x8  
3,0x98,0x91,  
  
0x4d,0x44,0x5f,0x56,0x69,0x60,0x7b,0x72,0x05,0x0c,0x17,0x1e,0x21,0x2  
8,0x33,0x3a,
```

```
0xdd,0xd4,0xcf,0xc6,0xf9,0xf0,0xeb,0xe2,0x95,0x9c,0x87,0x8e,0xb1,0xb
8,0xa3,0xaa,

0xec,0xe5,0xfe,0xf7,0xc8,0xc1,0xda,0xd3,0xa4,0xad,0xb6,0xbf,0x80,0x8
9,0x92,0x9b,

0x7c,0x75,0x6e,0x67,0x58,0x51,0x4a,0x43,0x34,0x3d,0x26,0x2f,0x10,0x1
9,0x02,0x0b,

0xd7,0xde,0xc5,0xcc,0xf3,0xfa,0xe1,0xe8,0x9f,0x96,0x8d,0x84,0xbb,0xb
2,0xa9,0xa0,

0x47,0x4e,0x55,0x5c,0x63,0x6a,0x71,0x78,0x0f,0x06,0x1d,0x14,0x2b,0x2
2,0x39,0x30,

0x9a,0x93,0x88,0x81,0xbe,0xb7,0xac,0xa5,0xd2,0xdb,0xc0,0xc9,0xf6,0xf
f,0xe4,0xed,

0x0a,0x03,0x18,0x11,0x2e,0x27,0x3c,0x35,0x42,0x4b,0x50,0x59,0x66,0x6
f,0x74,0x7d,

0xa1,0xa8,0xb3,0xba,0x85,0x8c,0x97,0x9e,0xe9,0xe0,0xfb,0xf2,0xcd,0xc
4,0xdf,0xd6,

0x31,0x38,0x23,0x2a,0x15,0x1c,0x07,0x0e,0x79,0x70,0x6b,0x62,0x5d,0x5
4,0x4f,0x46
};
unsigned char lookup11[256] =
```



```
{  
  
0x00,0x0b,0x16,0x1d,0x2c,0x27,0x3a,0x31,0x58,0x53,0x4e,0x45,0x74,0x7  
f,0x62,0x69,  
  
0xb0,0xbb,0xa6,0xad,0x9c,0x97,0x8a,0x81,0xe8,0xe3,0xfe,0xf5,0xc4,0xc  
f,0xd2,0xd9,  
  
0x7b,0x70,0x6d,0x66,0x57,0x5c,0x41,0x4a,0x23,0x28,0x35,0x3e,0x0f,0x0  
4,0x19,0x12,  
  
0xcb,0xc0,0xdd,0xd6,0xe7,0xec,0xf1,0xfa,0x93,0x98,0x85,0x8e,0xbf,0xb  
4,0xa9,0xa2,  
  
0xf6,0xfd,0xe0,0xeb,0xda,0xd1,0xcc,0xc7,0xae,0xa5,0xb8,0xb3,0x82,0x8  
9,0x94,0x9f,  
  
0x46,0x4d,0x50,0x5b,0x6a,0x61,0x7c,0x77,0x1e,0x15,0x08,0x03,0x32,0x3  
9,0x24,0x2f,  
  
0x8d,0x86,0x9b,0x90,0xa1,0xaa,0xb7,0xbc,0xd5,0xde,0xc3,0xc8,0xf9,0xf  
2,0xef,0xe4,  
  
0x3d,0x36,0x2b,0x20,0x11,0x1a,0x07,0x0c,0x65,0x6e,0x73,0x78,0x49,0x4  
2,0x5f,0x54,  
  
0xf7,0xfc,0xe1,0xea,0xdb,0xd0,0xcd,0xc6,0xaf,0xa4,0xb9,0xb2,0x83,0x8  
8,0x95,0x9e,
```

```
0x47,0x4c,0x51,0x5a,0x6b,0x60,0x7d,0x76,0x1f,0x14,0x09,0x02,0x33,0x3
8,0x25,0x2e,

0x8c,0x87,0x9a,0x91,0xa0,0xab,0xb6,0xbd,0xd4,0xdf,0xc2,0xc9,0xf8,0xf
3,0xee,0xe5,

0x3c,0x37,0x2a,0x21,0x10,0x1b,0x06,0x0d,0x64,0x6f,0x72,0x79,0x48,0x4
3,0x5e,0x55,

0x01,0x0a,0x17,0x1c,0x2d,0x26,0x3b,0x30,0x59,0x52,0x4f,0x44,0x75,0x7
e,0x63,0x68,

0xb1,0xba,0xa7,0xac,0x9d,0x96,0x8b,0x80,0xe9,0xe2,0xff,0xf4,0xc5,0xc
e,0xd3,0xd8,

0x7a,0x71,0x6c,0x67,0x56,0x5d,0x40,0x4b,0x22,0x29,0x34,0x3f,0x0e,0x0
5,0x18,0x13,

0xca,0xc1,0xdc,0xd7,0xe6,0xed,0xf0,0xfb,0x92,0x99,0x84,0x8f,0xbe,0xb
5,0xa8,0xa3
};
unsigned char lookup13[256] =
{

0x00,0x0d,0x1a,0x17,0x34,0x39,0x2e,0x23,0x68,0x65,0x72,0x7f,0x5c,0x5
1,0x46,0x4b,
```

0xd0,0xdd,0xca,0xc7,0xe4,0xe9,0xfe,0xf3,0xb8,0xb5,0xa2,0xaf,0x8c,0x81,0x96,0x9b,

0xbb,0xb6,0xa1,0xac,0x8f,0x82,0x95,0x98,0xd3,0xde,0xc9,0xc4,0xe7,0xea,0xfd,0xf0,

0x6b,0x66,0x71,0x7c,0x5f,0x52,0x45,0x48,0x03,0x0e,0x19,0x14,0x37,0x3a,0x2d,0x20,

0x6d,0x60,0x77,0x7a,0x59,0x54,0x43,0x4e,0x05,0x08,0x1f,0x12,0x31,0x3c,0x2b,0x26,

0xbd,0xb0,0xa7,0xaa,0x89,0x84,0x93,0x9e,0xd5,0xd8,0xcf,0xc2,0xe1,0xec,0xfb,0xf6,

0xd6,0xdb,0xcc,0xc1,0xe2,0xef,0xf8,0xf5,0xbe,0xb3,0xa4,0xa9,0x8a,0x87,0x90,0x9d,

0x06,0x0b,0x1c,0x11,0x32,0x3f,0x28,0x25,0x6e,0x63,0x74,0x79,0x5a,0x57,0x40,0x4d,

0xda,0xd7,0xc0,0xcd,0xee,0xe3,0xf4,0xf9,0xb2,0xbf,0xa8,0xa5,0x86,0x8b,0x9c,0x91,

0x0a,0x07,0x10,0x1d,0x3e,0x33,0x24,0x29,0x62,0x6f,0x78,0x75,0x56,0x5b,0x4c,0x41,

```
0x61,0x6c,0x7b,0x76,0x55,0x58,0x4f,0x42,0x09,0x04,0x13,0x1e,0x3d,0x30,0x27,0x2a,
```

```
0xb1,0xbc,0xab,0xa6,0x85,0x88,0x9f,0x92,0xd9,0xd4,0xc3,0xce,0xed,0xe0,0xf7,0xfa,
```

```
0xb7,0xba,0xad,0xa0,0x83,0x8e,0x99,0x94,0xdf,0xd2,0xc5,0xc8,0xeb,0xe6,0xf1,0xfc,
```

```
0x67,0x6a,0x7d,0x70,0x53,0x5e,0x49,0x44,0x0f,0x02,0x15,0x18,0x3b,0x36,0x21,0x2c,
```

```
0x0c,0x01,0x16,0x1b,0x38,0x35,0x22,0x2f,0x64,0x69,0x7e,0x73,0x50,0x5d,0x4a,0x47,
```

```
0xdc,0xd1,0xc6,0xcb,0xe8,0xe5,0xf2,0xff,0xb4,0xb9,0xae,0xa3,0x80,0x8d,0x9a,0x97
```

```
};
```

```
unsigned char lookup14[256] =
```

```
{
```

```
0x00,0x0e,0x1c,0x12,0x38,0x36,0x24,0x2a,0x70,0x7e,0x6c,0x62,0x48,0x46,0x54,0x5a,
```

```
0xe0,0xee,0xfc,0xf2,0xd8,0xd6,0xc4,0xca,0x90,0x9e,0x8c,0x82,0xa8,0xa6,0xb4,0xba,
```

0xdb,0xd5,0xc7,0xc9,0xe3,0xed,0xff,0xf1,0xab,0xa5,0xb7,0xb9,0x93,0x9d,0x8f,0x81,

0x3b,0x35,0x27,0x29,0x03,0x0d,0x1f,0x11,0x4b,0x45,0x57,0x59,0x73,0x7d,0x6f,0x61,

0xad,0xa3,0xb1,0xbf,0x95,0x9b,0x89,0x87,0xdd,0xd3,0xc1,0xcf,0xe5,0xeb,0xf9,0xf7,

0x4d,0x43,0x51,0x5f,0x75,0x7b,0x69,0x67,0x3d,0x33,0x21,0x2f,0x05,0x0b,0x19,0x17,

0x76,0x78,0x6a,0x64,0x4e,0x40,0x52,0x5c,0x06,0x08,0x1a,0x14,0x3e,0x30,0x22,0x2c,

0x96,0x98,0x8a,0x84,0xae,0xa0,0xb2,0xbc,0xe6,0xe8,0xfa,0xf4,0xde,0xd0,0xc2,0xcc,

0x41,0x4f,0x5d,0x53,0x79,0x77,0x65,0x6b,0x31,0x3f,0x2d,0x23,0x09,0x07,0x15,0x1b,

0xa1,0xaf,0xbd,0xb3,0x99,0x97,0x85,0x8b,0xd1,0xdf,0xcd,0xc3,0xe9,0xe7,0xf5,0xfb,

0x9a,0x94,0x86,0x88,0xa2,0xac,0xbe,0xb0,0xea,0xe4,0xf6,0xf8,0xd2,0xdc,0xce,0xc0,

```

0x7a,0x74,0x66,0x68,0x42,0x4c,0x5e,0x50,0x0a,0x04,0x16,0x18,0x32,0x3
c,0x2e,0x20,

0xec,0xe2,0xf0,0xfe,0xd4,0xda,0xc8,0xc6,0x9c,0x92,0x80,0x8e,0xa4,0xa
a,0xb8,0xb6,

0x0c,0x02,0x10,0x1e,0x34,0x3a,0x28,0x26,0x7c,0x72,0x60,0x6e,0x44,0x4
a,0x58,0x56,

0x37,0x39,0x2b,0x25,0x0f,0x01,0x13,0x1d,0x47,0x49,0x5b,0x55,0x7f,0x7
1,0x63,0x6d,

0xd7,0xd9,0xcb,0xc5,0xef,0xe1,0xf3,0xfd,0xa7,0xa9,0xbb,0xb5,0x9f,0x9
1,0x83,0x8d
};

```

lookup_table_encoding.h

```

//Galois Multiplication lookup tables for encryption
unsigned char lookup2[] =
{

0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1
a,0x1c,0x1e,

0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3
a,0x3c,0x3e,

```

0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,

0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,

0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,

0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,

0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,

0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,

0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,

0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,

0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,

```
0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
```

```
0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
```

```
0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
```

```
0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
```

```
0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5
```

```
};
```

```
unsigned char lookup3[] =
```

```
{
```

```
0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
```

```
0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
```

```
0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
```


0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,

0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,

0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,

0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,

0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,

0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,

0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,

0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,

0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,

```
0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4
c,0x49,0x4a,

0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7
c,0x79,0x7a,

0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2
c,0x29,0x2a,

0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1
c,0x19,0x1a
};
```

aes.cpp

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <sstream>
#include "key_expand.h"
#include "encoding.h"
#include "decoding.h"
#include <typeinfo>
#include <unistd.h>
using namespace std;
int main()
{
    // we will read from file input.txt
```

```

    int extendedlength = 0;
    int choice;
    string myText;
label:
    cout << "Welcome to 128 bits AES encryption" << endl;
    cout << endl;
    cout << "Enter you choice " << endl;
    cout << "1- Encoding" << endl;
    cout << "2- Decoding" << endl;
    cin >> choice;

    switch (choice)
    {
    case 1:
    {
        // encryption of text data
        ifstream File;
        string filepath = "encryption.aes";
        // clearing encryption.aes before editing
        File.open(filepath.c_str(), std::ofstream::out |
std::ofstream::trunc);
        if (!File.is_open() || File.fail())
        {
            File.close();
            printf("\nError : failed to erase file content !");
        }
        File.close();
        // reading plain text from input.txt
        fstream newfile;

```

```

newfile.open("input.txt", ios::in); // open a file to
perform read operation using file object
if (newfile.is_open())
{ // checking whether the file is open
    cout << "Reading plain text from input.txt .....\\n";
    usleep(1000);
    string tp;
    cout << "Reading KEY from key.txt .....\\n";
    usleep(1000);
    cout << "Now encrypting ....\\n";
    usleep(1000);
    cout << "writing encrypted data in encryption.aes ..\\n";
    usleep(1000);
    cout << endl;
    while (getline(newfile, tp))
    {
        // read data from file object and put it into
string.

        int messlength = tp.length();
        int extendedlength;
        if ((messlength % 16) != 0)
        {
            extendedlength = messlength + (16 - (messlength
% 16));
        }
        else
        {
            extendedlength = messlength;
        }
    }
}

```

```

        unsigned char *encryptedtext = new unsigned
char[extendedlength];
        for (int i = 0; i < extendedlength; i++)
        {
            if (i < messlength)
                encryptedtext[i] = tp[i];
            else
                encryptedtext[i] = 0;
        }
        // getting key from key.txt
        string k;
        ifstream infile;
        infile.open("key.txt");
        if (infile.is_open())
        {
            getline(infile, k); // The first line of file
should be the key
            infile.close();
        }

        else
            cout << "Unable to open file";

        istringstream tempkey(k);
        unsigned char key[16];
        unsigned int x;
        for (int i = 0; i < 16; i++)
        {
            tempkey >> hex >> x;

```

```

        key[i] = x;
    }
    // extending key
    unsigned char extendedkeys[176];
    Key_extenxion(key, extendedkeys);

    // encrypting our plain text
    for (int i = 0; i < extendedlength; i += 16)
    {
        unsigned char *temp = new unsigned char[16];
        for (int j = 0; j < 16; j++)
        {
            temp[j] = encryptedtext[i + j];
        }
        encryption(temp, extendedkeys);
        for (int j = 0; j < 16; j++)
        {
            encryptedtext[i + j] = temp[j];
        }
    }
    // storing our encrypted data in encryption.aes
    ofstream fout; // Create Object of Ofstream
    ifstream fin;
    fin.open("encryption.aes");
    fout.open("encryption.aes", ios::app); // Append
mode

    if (fin.is_open())
        fout << encryptedtext << "\n"; // Writing data
to file

```

```

        fin.close();
        fout.close();
    }
    cout << "128-bit AES encryption is done sucessfully\n";
    cout << "Data has been appended to file encryption.aes";
    newfile.close(); // close the file object.
}
break;
}

case 2:
{
    cout << "Reading encrypted data from encryption.txt
.....\n";
    usleep(1000);
    string tp;
    cout << "Reading KEY from key.txt ..... \n";
    usleep(1000);
    cout << "Now Decrypting ....\n";
    usleep(1000);
    cout << "writing decrypted data in outputtext.txt ..\n";
    usleep(1000);
    cout << endl;
    cout << "Following is our decrypted text:- \n";
    // clearing outputtext file
    ifstream File;
    string filepath = "outputtext.txt";
    File.open(filepath.c_str(), std::ifstream::out |
std::ifstream::trunc);

```

```

if (!File.is_open() || File.fail())
{
    File.close();
    printf("\nError : failed to erase file content !");
}
File.close();

ifstream MyReadFile;
MyReadFile.open("encryption.aes", ios::in | ios::binary);
if (MyReadFile.is_open())
{
    while (getline(MyReadFile, myText))
    {
        cout.flush();
        char *x;
        x = &myText[0];
        int messlength = strlen(x);
        char *msg = new char[myText.size() + 1];

        strcpy(msg, myText.c_str());

        int n = strlen((const char *)msg);
        unsigned char *decryptedtext = new unsigned char[n];
        // decrypting our encrypted data
        for (int i = 0; i < n; i++)
        {
            decryptedtext[i] = (unsigned char)msg[i];
        }
        // reading key from key.txt file
    }
}

```



```

        string k;
        ifstream infile;
        infile.open("key.txt");
        if (infile.is_open())
        {
            getline(infile, k); // The first line of file
should be the key

            infile.close();
        }

        else
            cout << "Unable to open file";

        istringstream tempkey(k);
        unsigned char key[16];
        unsigned int x1;
        for (int i = 0; i < 16; i++)
        {
            tempkey >> hex >> x1;
            key[i] = x1;
        }
        // extending key
        unsigned char extendedkeys[176];
        Key_extenxion(key, extendedkeys);
        // decrypting our data
        for (int i = 0; i < messlength; i += 16)
        {
            unsigned char *temp = new unsigned char[16];
            for (int j = 0; j < 16; j++)
                temp[j] = decryptedtext[i + j];

```

```

        decryption(temp, extendedkeys);
        for (int j = 0; j < 16; j++)
            decryptedtext[i + j] = temp[j];
    }
    // printing our plain text
    for (int i = 0; i < messlength; i++)
    {
        cout << decryptedtext[i];
        if (decryptedtext[i] == 0 && decryptedtext[i -
1] == 0)

            break;
    }
    // storing plain text in outputtext.txt file
    cout << endl;
    ofstream fout; // Create Object of Ofstream
    ifstream fin;
    fin.open("outputtext.txt");
    fout.open("outputtext.txt", ios::app); // Append
mode

    if (fin.is_open())
        fout << decryptedtext << "\n"; // Writing data
to file

    fin.close();
    fout.close(); // Closing the file
    usleep(500);
}
}
else

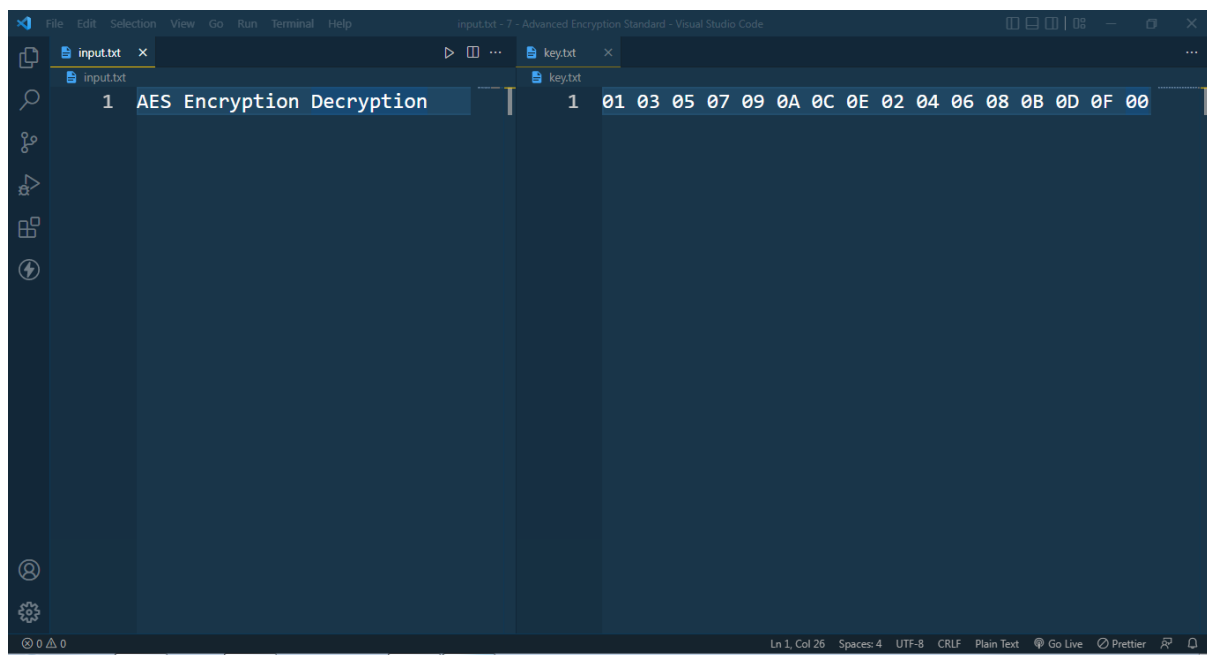
```

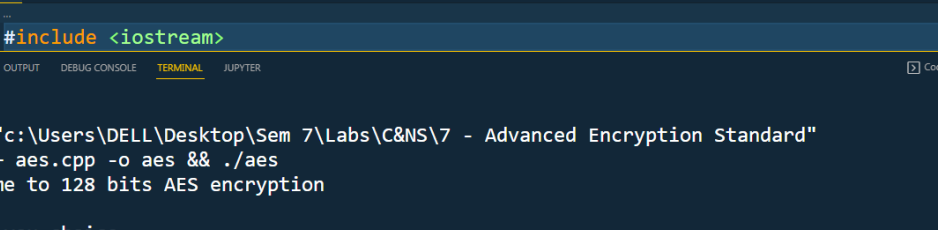
```

    {
        cout << "Can not open input file\n ";
    }
    cout << "\n Data has been appended to file outputtext.txt";
    MyReadFile.close();
    break;
}
}
}

```

Output:





The screenshot shows a Visual Studio Code editor with a C++ file named `aes.cpp` open. The code includes the `<iostream>` header. Below the code editor, the `TERMINAL` tab is active, displaying the output of a C++ program. The program prompts the user to enter a choice (1 for Encoding, 2 for Decoding). The user has entered '1', and the program proceeds to read plain text from `input.txt`, read a key from `key.txt`, and perform encryption. The output indicates that the 128-bit AES encryption was successful and the data has been appended to `encryption.aes`. The terminal prompt shows the user is `DELL@DELL-PC` using `MINGW64` in the directory `~/Desktop/Sem 7/Labs/C&NS/7 - Advanced Encryption Standard`.

```

1 #include <iostream>

$ cd "c:\Users\DELL\Desktop\Sem 7\Labs\C&NS\7 - Advanced Encryption Standard"
&& g++ aes.cpp -o aes && ./aes
Welcome to 128 bits AES encryption

Enter you choice
1- Encoding
2- Decoding
1
Reading plain text from input.txt .....
Reading KEY from key.txt .....
Now encrypting ....
writing encrypted data in encryption.aes ..

128-bit AES encryption is done sucessfully
Data has been appended to file encryption.aes
DELL@DELL-PC MINGW64 ~/Desktop/Sem 7/Labs/C&NS/7 - Advanced Encryption Standard
$
```

The screenshot displays the Visual Studio Code interface during an AES encryption and decryption task. The top toolbar includes standard menu options like File, Edit, Selection, View, Go, Run, Terminal, and Help. Two source files are open: 'encryption.aes' and 'outputtext.txt'. In 'encryption.aes', three lines of code are visible: Line 1 contains 'I=8DC1'; Line 2 contains a complex string of hexadecimal values and symbols: '62W\$0LDC1\$h0iSUB[>000DC16000000BC410Y'; Line 3 is empty. The 'outputtext.txt' file shows two lines: Line 1 reads 'AES Encryption Decryption' and Line 2 is empty. Below the editor, the 'TERMINAL' tab is active, displaying the execution output. It starts with 'Welcome to 128 bits AES encryption', followed by prompts for encoding or decoding choice. The user has chosen decoding, leading to messages about reading encrypted data from 'encryption.txt', reading the key from 'key.txt', performing decryption, and writing the result to 'outputtext.txt'. A final message states 'Following is our decrypted text:- AES Encryption Decryption'. At the bottom of the terminal, it says 'Data has been appended to file outputtext.txt' and shows the shell prompt 'DELL@DELL-PC MINGW64 ~/Desktop/Sem 7/Labs/C&NS/7 - Advanced Encryption Standard'. The status bar at the very bottom indicates the current cursor position as 'Ln 1, Col 6' and lists various settings like 'Spaces: 4', 'UTF-8', 'CRLF', 'Plain Text', 'Go Live', 'Prettier', and 'R'.

Conclusion:

AES instruction set is now integrated into the CPU (offers throughput of several GB/s) to improve the speed and security of applications that use AES for encryption and decryption. Even though it's been 20 years since its introduction we have failed to break the AES algorithm as it is infeasible even with the current technology.