08660-1: Data Security and Cryptography

Project 2: Software Implementation of Advanced Encryption Standard (AES)

Group Members:

1. Saurav Subham 104871034

Group Code:

- Adding up the last digit in all group members' IDs to obtain S1. Adding up the second-to-last digit in all group members' IDs to obtain S2.
  S1 = 4
  S2 = 3
- Let A = S1 mod 5, B = (S2 mod 5) + 5
  A = 4 mod 5 = 4
  B = 3 mod 5 + 5 = 3 + 5 = 8
- Group Code (A, B) = **(4, 8)**

Group Code (A, B) = (4, 8)

• 0000 0000 0000 0000 0000 0000 0000 abd7 (plaintext)

• 1a0c 24f2 8754 95bc b708 0e43 920f 5687 (key)

The program is written in C/C++ for operating system Windows.

Key Schedule Results for Each Round with the AES 128 bits:

(Chaining ECB)

```
Round 0:

        Start:00 00 00 00 00 00 00 00 00 00 00 00 00 00 ab d7

        Output:1a 0c 24 f2 87 54 95 bc b7 08 0e 43 92 0f fd 50

Round 1:

        Key:6d bd 33 bd ea e9 a6 01 5d e1 a8 42 cf ee fe c5

        Output:aa ea 09 6d 49 eb a9 55 dd 9b b4 28 78 22 35 f4

Round 2:

        Key:47 06 95 37 ad ef 33 36 f0 0e 9b 74 3f e0 65 b1
        Output:16 50 0b d9 4d 61 6f 81 3a 0d d4 5d 29 13 bf 52
Round 3:
        Key:a2 4b 5d 42 0f a4 6e 74 ff aa f5 00 c0 4a 90 b1
        Output:4e 11 65 2c 8d df 15 ff 44 a1 4a d5 80 e6 f9 26
Round 4:
        Key:7c 2b 95 f8 73 8f fb 8c 8c 25 0e 8c 4c 6f 9e 3d
        Output:ba b5 91 34 77 77 2e 22 68 f8 3b 4e 0a 55 66 ac
Round 5:
        Key:c4 20 b2 d1 b7 af 49 5d 3b 8a 47 d1 77 e5 d9 ec
        Output:40 89 c4 f8 ae 95 b3 1b bc 27 49 5e c3 4f 78 ff
Round 6:
        Key:3d 15 7c 24 8a ba 35 79 b1 30 72 a8 c6 d5 ab 44
        Output:7c 13 13 02 eb 43 bd bc 5f cd 41 da 5d 41 10 4c
Round 7:
        Key:7e 77 67 90 f4 cd 52 e9 45 fd 20 41 83 28 8b 05
        Output:da e4 0b 6b 5c 77 10 50 46 cd 39 3f b1 47 b7 78
Round 8:
        Key:ca 4a 0c 7c 3e 87 5e 95 7b 7a 7e d4 f8 52 f5 d1
        Output:ca 4a 0c 7c 3e 87 5e 95 7b 7a 7e d4 f8 52 f5 d1
Round 9:
        Key:d1 ac 32 3d ef 2b 6c a8 94 51 12 7c 6c 03 e7 ad
        Output:f7 4c 29 dd 4c 79 c2 8c d7 36 32 23 d7 4a ef 8e
Round 10:
        Key:9c 38 a7 6d 73 13 cb c5 e7 42 d9 b9 8b 41 3e 14
        Output:f4 8e 84 74 5a 16 14 04 e9 94 7c dd 85 68 1b 32
```

Modified AES

```
unsigned char Sbox[256] =
 {
 0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,    //0
 0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,   //1
 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,   //2
 0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,    //3
 0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,   // 8
 0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,   //5
 0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,   //6
 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,   //7
 0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,   //4
 0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,   //9
 0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,   //10
 0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,  //11
 0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,  //12
 0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,    //13
 0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,    //14
 0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16    //15
};
```

Round 0:

Start:00 00 00 00 00 00 00 00 00 00 00 00 00 00 ab d7

Output:1a 0c 24 f2 87 54 95 bc b7 08 0e 43 92 0f fd 50

Round 1:

```
        Key:6d bd 84 bd ea e9 11 01 5d e1 1f 42 cf ee 49 c5
        Output:aa ea be 6d 3c 5c a9 97 dd 9b 03 28 8e d4 83 03
Round 2:
        Key:47 1a 22 37 ad f3 33 36 f0 12 2c 74 3f fc 65 b1
        Output:da d4 bf 1f 7a 22 bc 66 97 22 61 c1 14 9c 2c fc
Round 3:
        Key:f3 57 ea 42 5e a4 d9 74 ae b6 f5 00 91 4a 90 b1
        Output:ac a7 20 bc ee 10 29 78 e7 b2 e6 33 bb f7 60 79
Round 4:
        Key:85 37 22 c3 db 93 fb b7 75 25 0e b7 e4 6f 9e 06
        Output:c1 16 bf 58 67 db f0 e2 f5 1f 58 ac a9 0a 27 47
Round 5:
        Key:3d 3c 4d aa e6 af b6 1d 93 8a b8 aa 77 e5 26 ac
        Output:60 84 61 df 0a 94 ce 51 7d 22 9a df 1e a6 19 30
Round 6:
        Key:c4 cb dc 5f 22 64 6a 42 b1 ee d2 e8 c6 0b f4 44
        Output:a5 88 49 a6 08 c7 94 8b 06 a2 a4 80 1a 57 29 65
Round 7:
        Key:af 74 83 eb 8d 10 e9 a9 3c fe 3b 41 fa f5 cf 05
        Output:f6 73 06 f4 22 84 de 2e 1b 66 94 71 78 9c 60 9a
Round 8:
        Key:c9 fe e8 c6 44 ee 01 6f 78 10 3a 2e 82 e5 f5 2b
        Output:fa 54 26 52 43 cf c0 47 1a 98 c6 17 d5 d8 02 3b
Round 9:
        Key:0b 18 19 ea 4f f6 18 85 37 e6 22 ab b5 03 d7 80
        Output:82 1f f0 7c 31 0f 5c 9b 2b 93 2d ee 99 65 90 e4
Round 10:
        Key:46 16 10 3f 09 e0 08 ba 3e 06 2a 11 8b 05 fd 91
        Output:6a 60 c8 56 ce 3c 68 aa cf 4b a6 05 65 c5 b7 b9
```

Code in C/C++

```
-----------------------------------main.cpp-----------------------------------
                                        -
#include <iostream>
using namespace std;
#include "source.h"
#include "function.h"


int main ()
{
  unsigned char plainText[] =
    { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0xab, 0xd7
  };
  unsigned char key[16] =
    { 0x1a, 0x0c, 0x24, 0xf2, 0x87, 0x54, 0x95, 0xbc, 0xb7, 0x08, 0x0e, 0x43,
    0x92,
    0x0f, 0x56, 0x87
  };
  unsigned char expandedkey[176];
  KeyExpansion (key, expandedkey);
  AesEncrypt (plainText, key, expandedkey);
}

-----------------------------------source.h-----------------------------------

unsigned char s_box[256] = {
  0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE,
0xD7, 0xAB, 0x76, //0
  0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C,
0xA4, 0x72, 0xC0, //1
  0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71,
0xD8, 0x31, 0x15, //2
  0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB,
0x27, 0xB2, 0x75, //3
  0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64,
0x5D, 0x19, 0x73, //8
  0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A,
0x4C, 0x58, 0xCF, //5
  0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50,
0x3C, 0x9F, 0xA8, //6
  0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10,
0xFF, 0xF3, 0xD2, //7
```

```
  0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29,
0xE3, 0x2F, 0x84, //4
  0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE,
0x5E, 0x0B, 0xDB, //9
  0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91,
0x95, 0xE4, 0x79, //10
  0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65,
0x7A, 0xAE, 0x08, //11
  0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B,
0xBD, 0x8B, 0x8A, //12
  0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86,
0xC1, 0x1D, 0x9E, //13
  0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE,
0x55, 0x28, 0xDF, //14
  0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0,
0x54, 0xBB, 0x16  //15
};

unsigned char mu12[256] =
  { 0x00, 0x02, 0x04, 0x06, 0x08, 0x0a, 0x0c, 0x0e, 0x10, 0x12, 0x14, 0x16,
  0x18, 0x1a, 0x1c, 0x1e,
  0x20, 0x22, 0x24, 0x26, 0x28, 0x2a, 0x2c, 0x2e, 0x30, 0x32, 0x34, 0x36,
  0x38, 0x3a, 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,
```

```c
  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 mul3[256] =
  { 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, 0x4c, 0x49, 0x4a,
  0x6b, 0x68, 0x6d, 0x6e, 0x67, 0x64, 0x61, 0x62, 0x73, 0x70, 0x75, 0x76,
  0x7f, 0x7c, 0x79, 0x7a,
  0x3b, 0x38, 0x3d, 0x3e, 0x37, 0x34, 0x31, 0x32, 0x23, 0x20, 0x25, 0x26,
  0x2f, 0x2c, 0x29, 0x2a,
  0x0b, 0x08, 0x0d, 0x0e, 0x07, 0x04, 0x01, 0x02, 0x13, 0x10, 0x15, 0x16,
  0x1f, 0x1c, 0x19, 0x1a
};

unsigned char rcon[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
};

-----------------------------------function.h-----------------------------------

void KeyExpansionCore (unsigned char temp[], unsigned char i)
{
  unsigned char t = temp[0];
  temp[0] = temp[1];
  temp[1] = temp[2];
  temp[2] = temp[3];
  temp[3] = t;
  for (int i = 0; i < 4; i++)
    {
      temp[i] = s_box[temp[i]];
    }
  temp[0] = temp[0] ^ rcon[i];
}

void KeyExpansion (unsigned char key[], unsigned char expandedkey[176])
{
  for (int i = 0; i < 16; i++)
    {
      expandedkey[i] = key[i];
    }
  int numberofbytesgen = 16;
  int reconindx = 1;
  unsigned char temp[4];
  while (numberofbytesgen < 176)
    {
      for (int i = 0; i < 4; i++)
  {
    temp[i] = expandedkey[numberofbytesgen - 4 + i];
  }
      if (numberofbytesgen % 16 == 0)
  {
    KeyExpansionCore (temp, reconindx);

    reconindx++;
  }
      for (int a = 0; a < 4; a++)
  {
    expandedkey[numberofbytesgen] =
      expandedkey[numberofbytesgen - 16] ^ temp[a];
    numberofbytesgen++;
  }
    }
}

void AddRoundKey (unsigned char state[], unsigned char key[])
{
  cout << endl << "ROUNDKEY" << '\t' << endl;
  for (int i = 0; i < 16; i++)
    {
      state[i] = state[i] ^ key[i];
      cout << hex << (int) key[i] << '\t';
    }
  cout << endl;
};
```

```
void SubBytes (unsigned char state[])
{
  for (int i = 0; i < 16; i++)
    {
      state[i] = s_box[state[i]];
    }
};

void ShiftRows (unsigned char state[])
{
  unsigned char temp[16];

  for (int i = 0; i < 16; i++)
    {
      temp[i] = state[i];
    }
  state[0] = temp[0];
  state[1] = temp[5];
  state[2] = temp[10];
  state[3] = temp[15];
  state[4] = temp[4];
  state[5] = temp[9];
  state[6] = temp[14];
  state[7] = temp[3];
  state[8] = temp[8];
  state[9] = temp[13];
  state[10] = temp[2];
  state[11] = temp[7];
  state[12] = temp[12];
  state[13] = temp[1];
  state[14] = temp[6];
  state[15] = temp[11];
};

void MixedColumns (unsigned char state[])
{
  unsigned char temp[16];
  temp[0] =
    (unsigned char) (mu12[state[0]] ^ mul3[state[1]] ^ state[2] ^ state[3]);
  temp[4] =
    (unsigned char) (mu12[state[4]] ^ mul3[state[5]] ^ state[6] ^ state[7]);
  temp[8] =
    (unsigned char) (mu12[state[8]] ^ mul3[state[9]] ^ state[10] ^ state[11]);
  temp[12] =
    (unsigned char) (mu12[state[12]] ^ mul3[state[13]] ^ state[14] ^
         state[15]);
  temp[1] =
    (unsigned char) (state[0] ^ mu12[state[1]] ^ mul3[state[2]] ^ state[3]);
  temp[5] =
    (unsigned char) (state[4] ^ mu12[state[5]] ^ mul3[state[6]] ^ state[7]);
  temp[9] =
    (unsigned char) (state[8] ^ mu12[state[9]] ^ mul3[state[10]] ^ state[11]);
  temp[13] =
    (unsigned char) (state[12] ^ mu12[state[13]] ^ mul3[state[14]] ^
         state[15]);
  temp[2] =
    (unsigned char) (state[0] ^ state[1] ^ mu12[state[2]] ^ mul3[state[3]]);
  temp[6] =
    (unsigned char) (state[4] ^ state[5] ^ mu12[state[6]] ^ mul3[state[7]]);
  temp[10] =
    (unsigned char) (state[8] ^ state[9] ^ mu12[state[10]] ^ mul3[state[11]]);
  temp[14] =
    (unsigned char) (state[12] ^ state[13] ^ mu12[state[14]] ^
         mul3[state[15]]);
  temp[3] =
    (unsigned char) (mul3[state[0]] ^ state[1] ^ state[2] ^ mu12[state[3]]);
  temp[7] =
    (unsigned char) (mul3[state[4]] ^ state[5] ^ state[6] ^ mu12[state[7]]);
  temp[11] =
    (unsigned char) (mul3[state[8]] ^ state[9] ^ state[10] ^ mu12[state[11]]);
  temp[15] =
    (unsigned char) (mul3[state[12]] ^ state[13] ^ state[14] ^
         mu12[state[15]]);
  for (int i = 0; i < 16; i++)
    {
      state[i] = temp[i];
    }
};

void AesEncrypt (unsigned char message[], unsigned char key[],
      unsigned char expandedkey[])
{
  unsigned char state[16];
  for (int i = 0; i < 16; i++)
    {
      state[i] = message[i];
    };
  AddRoundKey (state, key);
  cout << "Encrypted message for 0 round" << endl;
  for (int i = 0; i < 16; i++)
```

```
      {

        cout << '\t' << hex << (int) state[i];
      }
    cout << endl << endl;
    int numberofrounds = 9;
    for (int i = 0; i < numberofrounds; i++)
      {
        SubBytes (state);
        ShiftRows (state);
        MixedColumns (state);
        AddRoundKey (state, expandedkey + (16 * (i + 1)));
        cout << endl << "Encrypted message after" << " " << i +
    1 << "th" << " " << "round" << endl;
        for (int i = 0; i < 16; i++)
    {
      cout << hex << (int) state[i] << '\t';
    }
        cout << endl << endl << endl;
      }
    SubBytes (state);
    ShiftRows (state);
    AddRoundKey (state, expandedkey + 160);
    cout << endl << "Encrypted message after 10th round" << endl;
    for (int i = 0; i < 16; i++)
      {
        cout << hex << (int) state[i] << '\t';

      }
  }
```

_______________END OF CODE_______________