



INSTITUTO POLITÉCNICO NACIONAL
Escuela Superior de Cómputo



Introduction to Cryptography

AES key expansion

Profesora: Sandra Díaz Santiago

Grupo: 6CM1

Alumno: Rojas Vazquez Diego

Sesión de Laboratorio 5

Fecha: 16 de mayo 2024

Para el desarrollo de esta práctica se nos pidió desarrollar un programa que implementará el algoritmo de Key Expansion, que recibiera 32 dígitos hexadecimales y escribiera los valores de w en hexadecimal en un archivo. El código generado fue el siguiente:

```
#include <stdio.h>
#include <stdint.h>

//uint32_t es un tipo de dato entero sin signo de 32 bits para el par de hexadecimales
uint8_t rotword(uint8_t temp[4]){ //usamos como puntero a temp
    uint8_t rotated_temp = temp[0];
    temp[1] = temp[1];
    temp[2] = temp[2];
    temp[3] = temp[3];
    temp[0] = rotated_temp;
}

uint8_t buscar(uint8_t x){
    uint8_t sbox[16][16] = {
        {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}
    };
    uint8_t fila = (x >> 4) & 0x0F; //solo mantenemos los bits más significativos
    uint8_t columna = x & 0x0F; //solo mantenemos los bits menos significativos
    return sbox[fila][columna];
}

uint8_t subword(uint8_t temp[4]){ //usamos como puntero a temp
    temp[0] = buscar(temp[0]);
    temp[1] = buscar(temp[1]);
    temp[2] = buscar(temp[2]);
    temp[3] = buscar(temp[3]);
}
```

```

}

uint8_t keyexpansion(uint8_t *key, uint8_t w[44][4]){
    uint32_t rcon[10] = {
        0x01000000, 0x02000000, 0x04000000, 0x08000000, 0x10000000,
        0x20000000, 0x40000000, 0x80000000, 0x1B000000, 0x36000000
    };
    uint8_t temp[4];
    int i;
    FILE* fichero = fopen("keyschedule.txt", "wt");
    for(i = 0; i < 4; i++){
        w[i][0] = key[4*i];
        w[i][1] = key[(4*i)+1];
        w[i][2] = key[(4*i)+2];
        w[i][3] = key[(4*i)+3];
        fprintf(fichero,"%02X%02X%02X%02X",w[i][0],w[i][1],w[i][2],w[i][3]); //escribo los 4 bytes
        //de w[i] individual para evitar errores
    }
    for(i = 4; i < 44; i++){
        temp[0] = w[i-1][0]; temp[1] = w[i-1][1]; temp[2] = w[i-1][2]; temp[3] = w[i-1][3];
        if((i % 4) == 0){
            fprintf(fichero,"\n");//Cada que termina con una "matriz" cada 128 bits da un salto
            rotword(temp);
            subword(temp);
            temp[0] = temp[0] ^ (rcon[(i/4)-1] >> 24);
            temp[1] = temp[1] ^ ((rcon[(i/4)-1] >> 16) & 0x000000ff);
            temp[2] = temp[2] ^ ((rcon[(i/4)-1] >> 8) & 0x000000ff);
            temp[3] = temp[3] ^ (rcon[(i/4)-1] & 0x000000ff);
        }
        w[i][0] = (w[i-4][0]) ^ temp[0]; w[i][1] = (w[i-4][1]) ^ temp[1]; w[i][2] = (w[i-4][2]) ^
        temp[2]; w[i][3] = (w[i-4][3]) ^ temp[3];
        fprintf(fichero,"%02X%02X%02X%02X",w[i][0],w[i][1],w[i][2],w[i][3]); //escribo los
        //siguientes 128 bits
    }
    fclose(fichero);
}

void main(){
    char key_ch[33]; //un espacio extra para el caracter nulo al recibir en consola
    //uint8_t es un tipo de dato que se utiliza comúnmente para representar valores enteros sin
    //signo de 8 bits
    uint8_t key[16];
    uint8_t key_exp[44][4];
    int i;

    printf("Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): ");
    scanf("%32s", key_ch);

    for (i = 0; i < 16; i++) { //recorremos de dos en dos hexadecimales
        sscanf(key_ch + 2 * i, "%2hhx", &key[i]); //sscanf() interpreta %2hhx como un especificador
        //de formato que indica que debe leer
        //dos caracteres hexadecimales (%2h) y convertirlos en un byte (hhx), almacenando el resultado
        //en &key[i].
    }

    keyexpansion(key, key_exp);
}

```

El código comienza con la inclusión de las bibliotecas estándar `stdio.h` y `stdint.h` para el manejo de entrada/salida y tipos de datos enteros de tamaño específico, respectivamente. Luego, se definen varias funciones necesarias para la expansión de clave.

La función rotword debería rotar los bytes de entrada una posición hacia la izquierda, pero en realidad parece que no realiza esta operación correctamente. En cambio, simplemente asigna el primer byte al mismo valor que ya tenía.

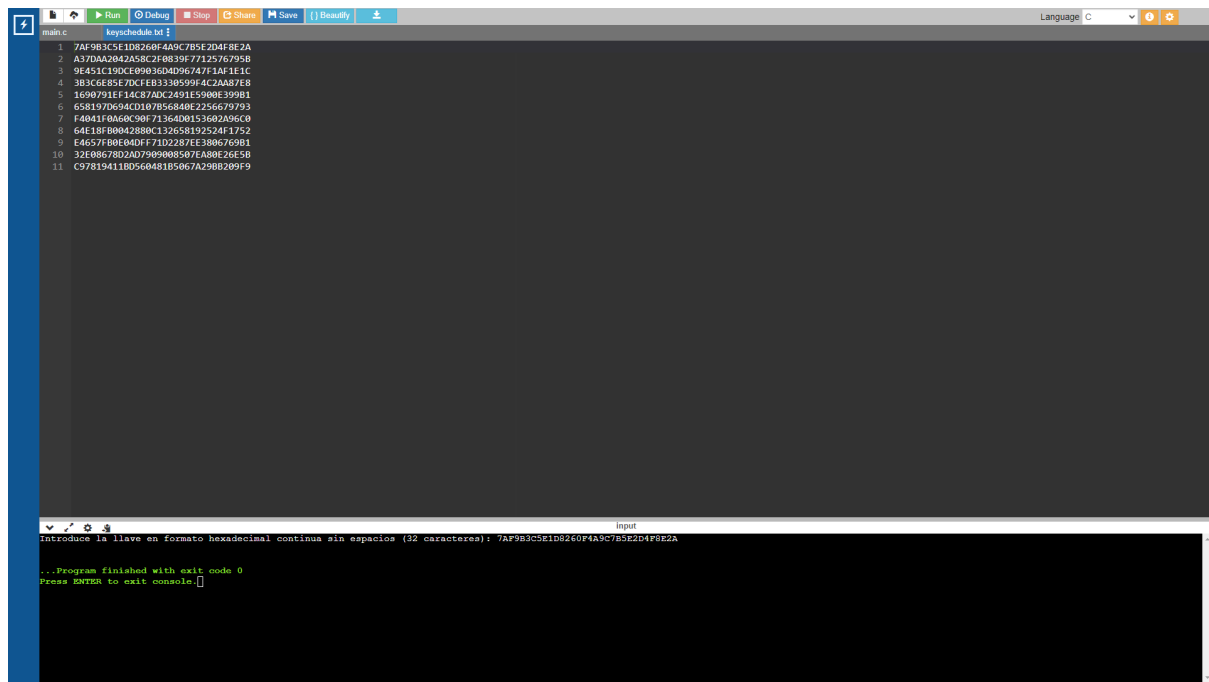
La función buscar implementa una tabla de sustitución conocida como S-Box, utilizada en AES para realizar una sustitución byte a byte. Dado un byte de entrada, esta función separa el byte en dos partes (fila y columna) y busca en la tabla S-Box para obtener un nuevo valor.

La función subword aplica la función buscar a cada byte del array de entrada, lo que significa que sustituye cada byte utilizando la tabla S-Box.

La función principal, keyexpansion, toma una clave de 16 bytes y genera una matriz de subclaves de tamaño 44x4. Primero, copia los primeros 16 bytes de la clave original en las primeras 4 subclaves. Luego, genera las siguientes 40 subclaves en bloques de 4 bytes utilizando operaciones de manipulación de bytes como rotaciones, sustituciones y XORs.

En la función principal main, se solicita al usuario que ingrese una clave de 32 caracteres hexadecimales (16 bytes) sin espacios. La clave ingresada se convierte de su representación hexadecimal a bytes. Luego, se llama a la función keyexpansion para generar las subclaves.

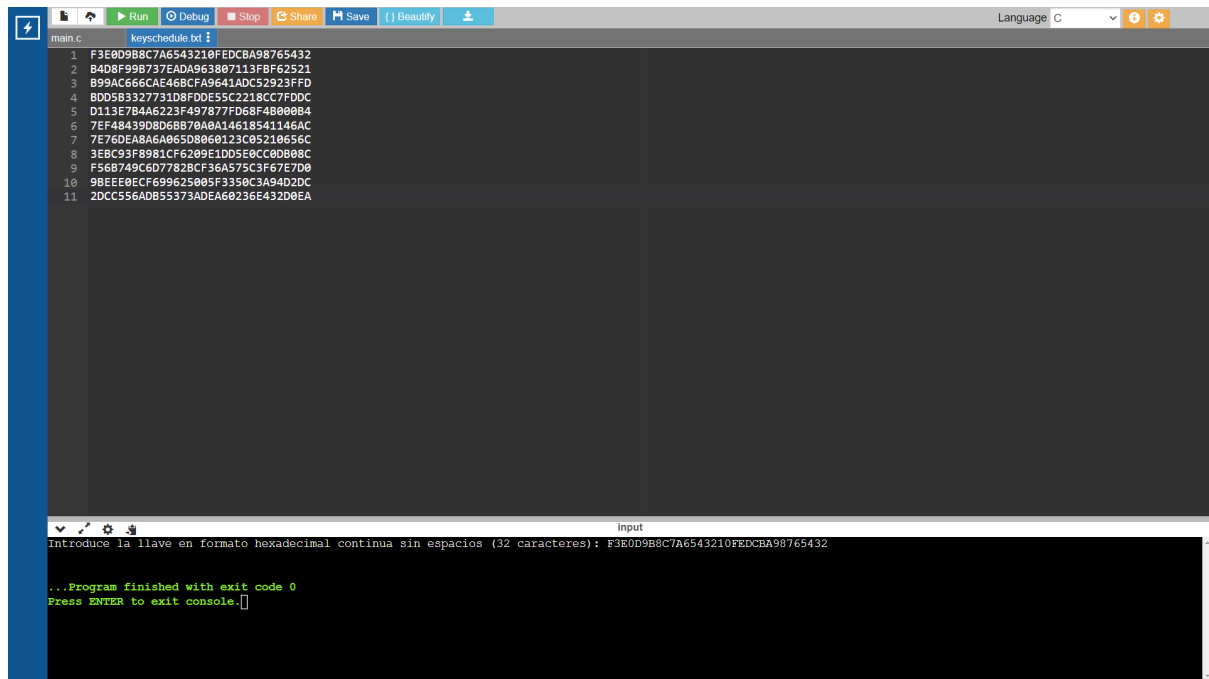
Pruebas del programa con distintas llaves



```
main.c | keySchedule.txt |
1  7AF983C5E1DB260F4A9C7B5E2D4F8E2A
2  A37DAA2842A58C2F8B39F7712576795B
3  9E451C19DCE09036D4D96747F1AF1E1C
4  3B3C6E85E7DCFE8338599F4C2AA87E8
5  1690791E1AC87A2C2491E5908E39981
6  658197D694CD107B56848E225667973
7  F4841F0A60C90F71364D0153602A96C0
8  64E18F80042880C132638192524F1752
9  E4657F84E0ADF771D2287EE380676981
10 32E08678D2AD7909008507EA80E26E5B
11 C97819411BD560481B5067A298B209F9

Input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 7AF983C5E1DB260F4A9C7B5E2D4F8E2A

...Program finished with exit code 0
Press ENTER to exit console.
```



```
main.c | keySchedule.txt |
1  F3E0D9B8C7A6543210FEDCBA98765432
2  B4D8F99B737EADA963807113FBF62521
3  B99AC666CAE468CFA9641ADC52923FFD
4  BDD583327731D8FDE55C2218CC7FDDC
5  D113E7B4A6223F497877FD68F4800084
6  7EF48439D8D6B878A0A14618541146AC
7  7E76D8A8A965D8060123C05210656C
8  3E8C93F8981CF6209E1DD5E8CC00808C
9  F568749C6D7782BCF36A575C3F67E7D0
10 9BEEE0ECF699625005F3350C3A94D2DC
11 2DCC556ADB55373ADEA60236E432D0EA

Input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): F3E0D9B8C7A6543210FEDCBA98765432

...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows a C program interface with a menu bar (Run, Debug, Stop, Share, Save, Beautify) and a language dropdown set to 'C'. The main window displays a file named 'keyschedule.txt' with 11 lines of hexadecimal data:

```
1 1A2B3C4D5E6F7A8B9C0D1E2F3A4B5C6
2 90987622C5F79CA959FA128663B14E80
3 625059EFA7A75546FE5D47C09DEC0940
4 389F58F69F390DA061644A60FC884320
5 805A42511F634FF17E070591828F46B1
6 832918999C4A5768E24D52F960C21448
7 730CE2CBEF46B5A30D08E75A6DC9F312
8 0FD1EF02E0975AA1ED9C8DFB80554EE9
9 422DC01CA28A9ABD4F262746CF7369AF
10 D3A239657118A3083E3E849EF14DED31
11 44416CA23559CF7A08674BE4FA2AA6D5
```

Below the code editor is an 'Input' window with the following text:

Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 1A2B3C4D5E6F7A8B9C0D1E2F3A4B5C6

...Program finished with exit code 0
Press ENTER to exit console.

The screenshot shows the same C program interface as above, but with a different set of 11 hexadecimal keys in the 'keyschedule.txt' file:

```
1 ABCDEF0123456789ABCDEF0123456789
2 8CA36A31AFE6008042BE2B9276E85B1
3 423CDF9EDDAF041E9F112F8CE9F9749
4 CDE775C2203D8583C9CC977B07530032
5 000A16E120379362E9FB0419EEA8042B
6 38C8E41018FF7772F104736B1FAC7740
7 D8591119C0A6666B31A215002E0E6240
8 A9F28B106954D07B58F6C87B76F8AA3B
9 11B317F278E7CA89201102F256E9A8C9
10 BB4D052FC34A1FA6E3B1D54B52B59D
11 589A007198D01FD778880283CD39871E
```

The 'Input' window now shows:

Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): ABCDEF0123456789ABCDEF012345678

...Program finished with exit code 0
Press ENTER to exit console.

```
main.c | keySchedule.txt |
1 123456789ABCDEF0123456789ABCDEF
2 AB514B0E31ED95FE23D9C386B8651D89
3 FF1CEFA9CEf17A57ED2889D154AD4A58
4 D8FFAEC3150EDC94F8266545AC68C11D
5 4280DE67578E02F3AFA867B603C3A6A8
6 29AEFA057E20F8F6D1889F40D24839EB
7 BC1DE8ECC23D101A13B58F5AC1FE66B1
8 84A6A6244698B63E552E396494D08FD5
9 26D6D527604D631935635A7DA183D5A8
10 0FB8D6E56FF6B5FC5A95EF81FB263A29
11 364C5640598AE38C032F0C3DF8093614

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 123456789ABCDEF0123456789ABCDEF

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 E2F3A4B5C6D7E8F9ABCDEF0123456708
2 C59D2185034AC97CA887267D88C24175
3 FAB8A218F9F26B6451754D19DA876C6C
4 A9115C4850E3372C01967A35D08217659
5 18EC6483480F53AF4999299A92B85FC3
6 4780ABAD0F8FF8024616D198D4AE8E5B
7 2F64B29420EB4A9666FD900EB2531555
8 5889EB687862A1FE1E9F3AF0ACC2FA5
9 49C2FE6E31A05F902F3F656083F34AC5
10 BECF28C88F6F7758A050123823A358FD
11 AEC5429C21AA35C481FA27FCA2597F01

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): E2F3A4B5C6D7E8F9ABCDEF0123456708

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 0F1E2D3C4B5A69788796A5B4C3D2E10F
2 20ABD54A68F1BC32EC6719862F85F889
3 377E94E05C8F280F80E831599F5DC900
4 E832499D848D61428455961B8B0899CB
5 F482A782488FC6C844EA96D8D0FE28F10
6 7A9AD1483A2517887ECF8153A12D8E43
7 6842C8525267DFA2CA85E898D85D0CA
8 75D5882627B267FC0B1A3975869FE9BF
9 B10EA62E968CC1D29DA6F8A71B391118
10 051C248393A0E551E061DF6153F8CCE
11 6A69DAABF9C93FFAF7CF220CE2F02EE2

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 0F1E2D3C4B5A69788796A5B4C3D2E10F

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 9876543210ABCDEFEDCBA09876543201
2 A156774EB1FDBAA15C361A392A622838
3 46FC4349F701F9E8AB37E3D18153CB89
4 4E005457B001A5BF1236466E9363B007
5 9AFB014023FAA4FF31CCE291A2AF6F16
6 B082A90793780DF8A2B4EF690018807F
7 F32D64D56055692DC2E18644C2FA063B
8 96000B37F655621A34B4E45EF64EE265
9 542F937AA27AF16096CE153E6080F75B
10 9FE2F8433D988A23AB561F1DCB06E846
11 B6146819888C6A3A20DA7527E88C9D61

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 9876543210ABCDEFEDCBA0987654321

...Program finished with exit code 0
Press ENTER to exit console.
```



```
main.c | keySchedule.txt |
1 5E6F7A8B9C0D1E2F3A4B5C6D7E8F9A00
2 AC1CC2E83011DC70A5A80AA74D51AAA
3 3C1F60449C0E8C396543C2972012603
4 701397A8741D20207249170200C83181
5 13F850A467E6788F15AF6C8D15675D0C
6 5A7E1C5A3D9867D5283708583D505654
7 5D2DAD7A6085CAAF4882C1F775D297A3
8 80982570E02DEFDA8AF2E28DD7D898B
9 C167734D214A9C9289E5B28A54980B31
10 FA21588AD86BC418528E76A206167D93
11 A366A7567800634E2A8315EC2C95687F

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 5E6F7A8B9C0D1E2F3A4B5C6D7E8F9A

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 DEADBEEFDEADBEEFDEADBEEFDEADBEEF
2 C23810301C95AEDFC23810301C95AEDF
3 5C12FAAE40875A71828F4A419E2AE49E
4 53F79DA5137067D491CF8D990FE5690B
5 2D2E648E3E5EA35AAF912ECFA07447C4
6 D08CC492E3E267C84C734907EC070EC3
7 33796F8CD09080749CE8417370EF4FB0
8 22A6EB5BF23DE32F6ED5A25C1E3AEDEC
9 D026BE95221B5D0A4CCEFFE652F4120A
10 CB9977F2E9822A48A54CDAEF788C7A4
11 95F581BB7C779BF3D93B4E5D2E8389F9

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): DEADBEEFDEADBEEFDEADBEEFDEADBEEF

...Program finished with exit code 0
Press ENTER to exit console.
```



INSTITUTO POLITÉCNICO NACIONAL
Escuela Superior de Cómputo



Introduction to Cryptography

AES key expansion

Profesora: Sandra Díaz Santiago

Grupo: 6CM1

Alumno: Rojas Vazquez Diego

Sesión de Laboratorio 5

Fecha: 16 de mayo 2024

Para el desarrollo de esta práctica se nos pidió desarrollar un programa que implementará el algoritmo de Key Expansion, que recibiera 32 dígitos hexadecimales y escribiera los valores de w en hexadecimal en un archivo. El código generado fue el siguiente:

```
#include <stdio.h>
#include <stdint.h>

//uint32_t es un tipo de dato entero sin signo de 32 bits para el par de hexadecimales
uint8_t rotword(uint8_t temp[4]){ //usamos como puntero a temp
    uint8_t rotated_temp = temp[0];
    temp[1] = temp[1];
    temp[2] = temp[2];
    temp[3] = temp[3];
    temp[0] = rotated_temp;
}

uint8_t buscar(uint8_t x){
    uint8_t sbox[16][16] = {
        {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}
    };
    uint8_t fila = (x >> 4) & 0x0F; //solo mantenemos los bits más significativos
    uint8_t columna = x & 0x0F; //solo mantenemos los bits menos significativos
    return sbox[fila][columna];
}

uint8_t subword(uint8_t temp[4]){ //usamos como puntero a temp
    temp[0] = buscar(temp[0]);
    temp[1] = buscar(temp[1]);
    temp[2] = buscar(temp[2]);
    temp[3] = buscar(temp[3]);
}
```

```

}

uint8_t keyexpansion(uint8_t *key, uint8_t w[44][4]){
    uint32_t rcon[10] = {
        0x01000000, 0x02000000, 0x04000000, 0x08000000, 0x10000000,
        0x20000000, 0x40000000, 0x80000000, 0x1B000000, 0x36000000
    };
    uint8_t temp[4];
    int i;
    FILE* fichero = fopen("keyschedule.txt", "wt");
    for(i = 0; i < 4; i++){
        w[i][0] = key[4*i];
        w[i][1] = key[(4*i)+1];
        w[i][2] = key[(4*i)+2];
        w[i][3] = key[(4*i)+3];
        fprintf(fichero,"%02X%02X%02X%02X",w[i][0],w[i][1],w[i][2],w[i][3]); //escribo los 4 bytes
        //de w[i] individual para evitar errores
    }
    for(i = 4; i < 44; i++){
        temp[0] = w[i-1][0]; temp[1] = w[i-1][1]; temp[2] = w[i-1][2]; temp[3] = w[i-1][3];
        if((i % 4) == 0){
            fprintf(fichero,"\n");//Cada que termina con una "matriz" cada 128 bits da un salto
            rotword(temp);
            subword(temp);
            temp[0] = temp[0] ^ (rcon[(i/4)-1] >> 24);
            temp[1] = temp[1] ^ ((rcon[(i/4)-1] >> 16) & 0x000000ff);
            temp[2] = temp[2] ^ ((rcon[(i/4)-1] >> 8) & 0x000000ff);
            temp[3] = temp[3] ^ (rcon[(i/4)-1] & 0x000000ff);
        }
        w[i][0] = (w[i-4][0]) ^ temp[0]; w[i][1] = (w[i-4][1]) ^ temp[1]; w[i][2] = (w[i-4][2]) ^
        temp[2]; w[i][3] = (w[i-4][3]) ^ temp[3];
        fprintf(fichero,"%02X%02X%02X%02X",w[i][0],w[i][1],w[i][2],w[i][3]); //escribo los
        //siguientes 128 bits
    }
    fclose(fichero);
}

void main(){
    char key_ch[33]; //un espacio extra para el caracter nulo al recibir en consola
    //uint8_t es un tipo de dato que se utiliza comúnmente para representar valores enteros sin
    //signo de 8 bits
    uint8_t key[16];
    uint8_t key_exp[44][4];
    int i;

    printf("Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): ");
    scanf("%32s", key_ch);

    for (i = 0; i < 16; i++) { //recorremos de dos en dos hexadecimales
        sscanf(key_ch + 2 * i, "%2hhx", &key[i]); //sscanf() interpreta %2hhx como un especificador
        //de formato que indica que debe leer
        //dos caracteres hexadecimales (%2h) y convertirlos en un byte (hhx), almacenando el resultado
        //en &key[i].
    }

    keyexpansion(key, key_exp);
}

```

El código comienza con la inclusión de las bibliotecas estándar `stdio.h` y `stdint.h` para el manejo de entrada/salida y tipos de datos enteros de tamaño específico, respectivamente. Luego, se definen varias funciones necesarias para la expansión de clave.

La función rotword debería rotar los bytes de entrada una posición hacia la izquierda, pero en realidad parece que no realiza esta operación correctamente. En cambio, simplemente asigna el primer byte al mismo valor que ya tenía.

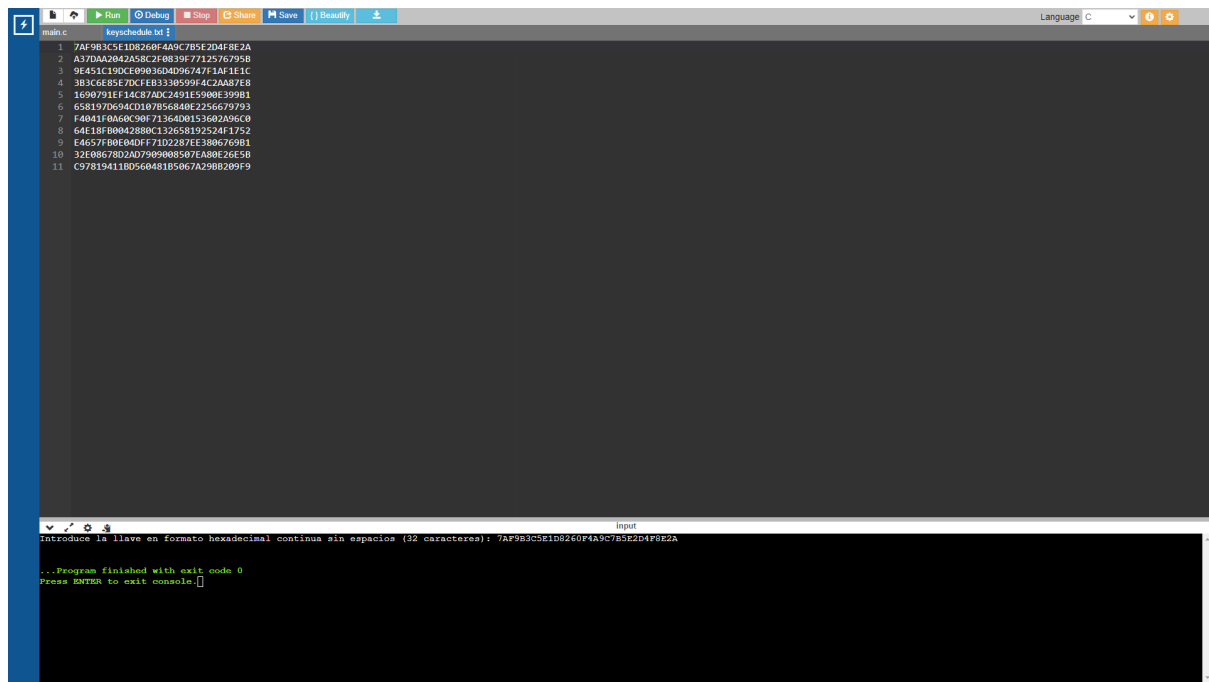
La función buscar implementa una tabla de sustitución conocida como S-Box, utilizada en AES para realizar una sustitución byte a byte. Dado un byte de entrada, esta función separa el byte en dos partes (fila y columna) y busca en la tabla S-Box para obtener un nuevo valor.

La función subword aplica la función buscar a cada byte del array de entrada, lo que significa que sustituye cada byte utilizando la tabla S-Box.

La función principal, keyexpansion, toma una clave de 16 bytes y genera una matriz de subclaves de tamaño 44x4. Primero, copia los primeros 16 bytes de la clave original en las primeras 4 subclaves. Luego, genera las siguientes 40 subclaves en bloques de 4 bytes utilizando operaciones de manipulación de bytes como rotaciones, sustituciones y XORs.

En la función principal main, se solicita al usuario que ingrese una clave de 32 caracteres hexadecimales (16 bytes) sin espacios. La clave ingresada se convierte de su representación hexadecimal a bytes. Luego, se llama a la función keyexpansion para generar las subclaves.

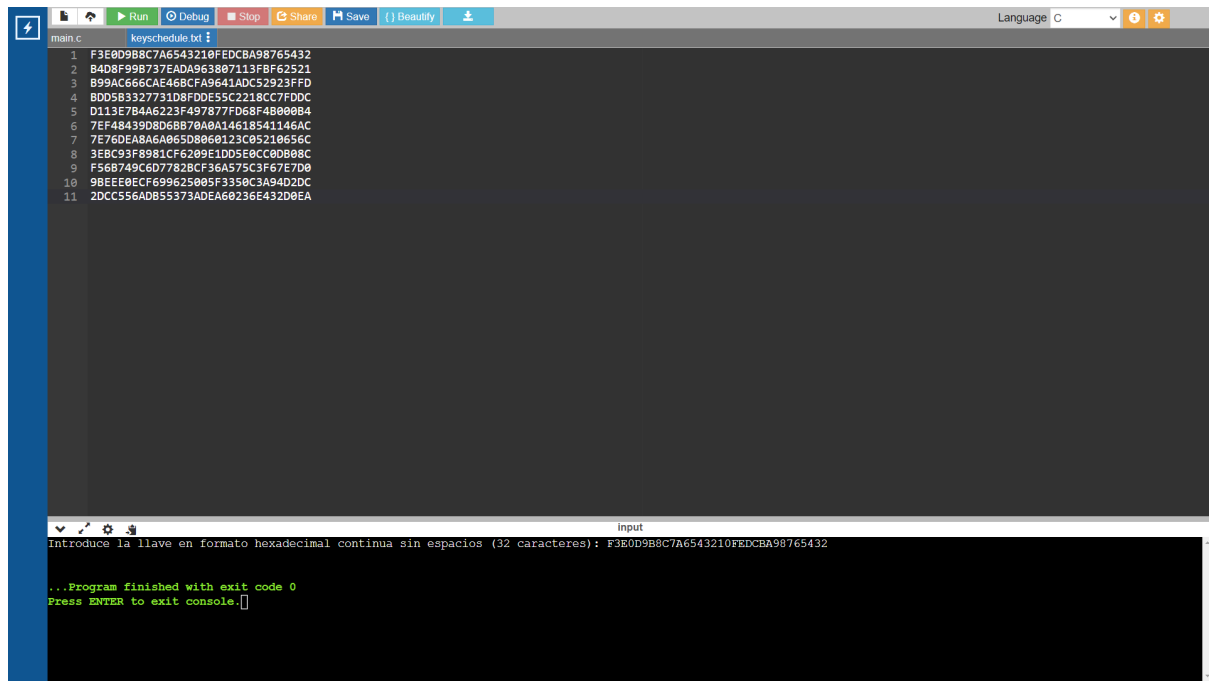
Pruebas del programa con distintas llaves



```
main.c | keySchedule.txt |
1  7AF983C5E1D8260F4A9C7B5E2D4F8E2A
2  A37DAA2842A58C2F8839F7712576795B
3  9E451C19DCE09036D4D96747F1AF1E1C
4  3B3C6E85E7DCFEB338599F4C2AA87E8
5  1690791E1AC87A2C2491E5908E39981
6  658197D694CD107B56848E225667973
7  F4841F0A60C90F71364D0153602A96C0
8  64E18F80042880C132638192524F1752
9  E4657F84E0ADF771D2287EE380676981
10 32E08678D2AD7909008507EA80E26E5B
11 C97819411B0560481B5067A298B209F9

Input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 7AF983C5E1D8260F4A9C7B5E2D4F8E2A

...Program finished with exit code 0
Press ENTER to exit console.
```



```
main.c | keySchedule.txt |
1  F3E0D9B8C7A6543210FEDCBA98765432
2  B4D8F99B737EADA963807113FBF62521
3  B99AC666CAE468CFA9641ADC52923FFD
4  BDD583327731D8FDE55C2218CC7FDDC
5  D113E7B4A6223F497877FD68F4800084
6  7EF48439D8D6B878A0A14618541146AC
7  7E76D8A8A965D8060123C05210656C
8  3E8C93F8981CF6209E1DD5E8CC00808C
9  F568749C6D7782BCF36A575C3F67E7D0
10 9BEEE0ECF699625005F3350C3A94D2DC
11 2DCC556ADB55373ADEA60236E432D08EA

Input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): F3E0D9B8C7A6543210FEDCBA98765432

...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows a C program interface with a menu bar (Run, Debug, Stop, Share, Save, Beautify) and a language dropdown set to C. The main window displays a file named 'keyschedule.txt' with 11 lines of hexadecimal data:

```
1 1A2B3C4D5E6F7A8B9C0D1E2F3A4B5C6
2 90987622C5F79CA959FA128663B14E80
3 625059EFA7A75546FE5D47C09DEC0940
4 389F58F69F390D0061644A60FC884320
5 805A42511F634FF17E070591828F46B1
6 832918999C4A5768E24D52F960C21448
7 730CE2CB EF46B5A30D0BE75A6DC9F312
8 0FD1EF02E0975AA1ED9C8DFB80554EE9
9 422DC01CA28A9ABD4F262746CF7369AF
10 D3A239657118A3083E3E849EF14DED31
11 44416CA23559CF7A08674BE4FA2AA6D5
```

Below the code editor is an 'Input' window with the following text:

Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 1A2B3C4D5E6F7A8B9C0D1E2F3A4B5C6

...Program finished with exit code 0
Press ENTER to exit console.

The screenshot shows the same C program interface as above, but with a different set of 11 hexadecimal keys in the 'keyschedule.txt' file:

```
1 ABCDEF0123456789ABCDEF0123456789
2 8CA36A31AFE6008042BE2B9276E85B1
3 423CDF9EDDAF041E9F112F8CE9F9749
4 CDE775C2203D8583C9CC977B07530032
5 000A16E120379362E9FB0419EEA8042B
6 38C8E41018FF7772F104736B1FAC7740
7 D8591119C0A6666B31A215002E0E6240
8 A9F28B106954D07B58F6C87B76F8AA3B
9 11B317F278E7CA89201102F256E9A8C9
10 BB4D052FC34A1FA6E3B1D54B52B59D
11 589A007198D01FD778880283CD39871E
```

The 'Input' window now shows:

Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): ABCDEF0123456789ABCDEF012345678

...Program finished with exit code 0
Press ENTER to exit console.

```
main.c | keySchedule.txt |
1 123456789ABCDEF0123456789ABCDEF
2 AB514B0E31ED95FE23D9C386B8651D89
3 FF1CEFA9CEf17A57ED2889D154AD4A58
4 D8FFAEC3150EDC94F8266545AC68C11D
5 4280DE67578E02F3AFA867B603C3A6A8
6 29AEFA057E20F8F6D1889F40D24839EB
7 BC1DE8ECC23D101A13858F5AC1FE66B1
8 84A6A6244698B63E552E396494D08FD5
9 26D6D527604D631935635A7DA183D5A8
10 0FB8D6E56FF685FC5A95EF81F8263A29
11 364C5640598AE38C032F0C3DF8093614

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 123456789ABCDEF0123456789ABCDEF

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 E2F3A4B5C6D7E8F9ABCDEF0123456708
2 C59D2185034AC97CA887267D88C24175
3 FAB8A218F9F26B6451754D19DA876C6C
4 A9115C4850E3372C01967A35D08217659
5 18EC6483480F53AF4999299A92885FC3
6 4780ABAD0F8FF8024616D198D4AE8E5B
7 2F64B29420EB4A9666FD900EB2531555
8 5889EB687862A1FE1E9F3AF0ACC2FA5
9 49C2FE6E31A05F902F3F656083F34AC5
10 BECF28C886F7758A050123823A358FD
11 AEC5429C21AA35C481FA27FCA2597F01

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): E2F3A4B5C6D7E8F9ABCDEF0123456708

...Program finished with exit code 0
Press ENTER to exit console.
```



```
main.c | keySchedule.txt |
1 0F1E2D3C4B5A69788796A5B4C3D2E10F
2 20ABD54A68F1BC32EC6719862F85F889
3 377E94E05C8F280F80E831599F5DC900
4 E832499D848D61428455591B8B0899CB
5 F482A782488FC6C844EA96D8D0FE28F10
6 7A9AD1483A2517887ECF8153A12D8E43
7 6842C8525267DFA2CA85E898D85D0CA
8 75D5882627B267FC0B1A3975869FE9BF
9 B10EA62E968CC1D29DA6F8A71B391118
10 051C248393A0E551E061DF6153F8CCE
11 6A69DAABF9C93FFAF7CF220CE2F02EE2

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 0F1E2D3C4B5A69788796A5B4C3D2E10F

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 9876543210ABCDEFEDCBA09876543201
2 A156774EB1FDBAA15C361A392A622838
3 46FC4349F701F9E8AB37E3D18153CBE9
4 4E005457B001A5BF1236466E9363B007
5 9AFB014023FAA4FF31CCE291A2AF6F16
6 B082A90793780DF8A2B4EF690018807F
7 F32D64D56055692DC2E18644C2FA063B
8 96000B37F655621A34B4E45EF64EE265
9 542F937AA27AF16096CE153E6080F75B
10 9FE2F8433D988A23AB561F1DCB06E846
11 B6146819888C6A3A20DA7527E88C9D61

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 9876543210ABCDEFEDCBA0987654321

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 5E6F7A8B9C0D1E2F3A4B5C6D7E8F9A00
2 AC1CC2E83011DC70A5A80AA74D51AAA
3 3C1F60449C0E8C396543C2972012603
4 701397A8741D20207249170200C83181
5 13F850A467E6788F15AF6C8D15675D0C
6 5A7E1C5A3D9867D5283708583D505654
7 5D2DAD7A6085CAAF4882C1F775D297A3
8 80982570E02DEFDA8AF2E28DD7D898B
9 C167734D214A9C9289E5B28A54980B31
10 FA21588AD86BC418528E76A206167D93
11 A366A7567800634E2A8315EC2C95687F

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): 5E6F7A8B9C0D1E2F3A4B5C6D7E8F9A

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c | keySchedule.txt |
1 DEADBEEFDEADBEEFDEADBEEFDEADBEEF
2 C23810301C95AEDFC23810301C95AEDF
3 5C12FAAE40875A71828F4A419E2AE49E
4 53F79DA5137067D491CF8D950FE5690B
5 2D2E648E3E5EA35AAF912ECFA07447C4
6 D08CC492E3E267C84C734907EC070EC3
7 33796F8CD09080749CE8417370EF4FB0
8 22A6E85BF23DE32F6ED5A25C1E3AEDEC
9 D026E95221B5D0A4CCEFFE652F4120A
10 CB9977F2E9822A48A54CDAEF788C7A4
11 95F581B87C779BF3D93B4E5D2E8389F9

input
Introduce la llave en formato hexadecimal continua sin espacios (32 caracteres): DEADBEEFDEADBEEFDEADBEEFDEADBEEF

...Program finished with exit code 0
Press ENTER to exit console.
```