Assignment #2

Subject	Information Security
Professor	David Choi
Major	Computer Science & Engineering
Student No	20172655
Name	Kangsan Lee
Submission Date	13 October, 2021

Q1. (1 point) Secure Hash Algorithm (SHA) is one kind of popular hash function, where SHA-256, SHA-384, and SHA-512 algorithms can produce the hash values with 256, 384, and 512 bits in length, respectively. Please explain why we usually say SHA-256, SHA-384, and SHA-512 algorithms are designed to match the security of AES with 128, 192, and 256 bits, respectively.

Answer) AES is a block cipher using the 128, 192, 256 bits of keys and SHA is a hash function that makes the input into 256, 384, 512 bits of output. AES and SHA needs the same size of test to break, so it is called that SHA-256, 384, 512 are designed to match the security of AES with 128, 192, and 256 bits.

Q2. (1 point) Using the Euclid's gcd theorem, determine the following:

(a) gcd (24140,16762) = 34

(b) gcd (4655,12075) = 35

Q2.

(a) gcd (24140, 16762)

= gcd (16762, 7378)

= gcd (7378, 2006)

= gcd (2006, 1360)

= gcd (1360, 646)

= gcd (646, 68)

= gcd (68, 34)

= gcd (34, 0)

= $\underline{34}$ //

(b) gcd (12075, 4655)

= gcd (4655, 2765)

= gcd (2765, 1890)

= gcd (1890, 875)

= gcd (875, 140)

= gcd (140, 35)

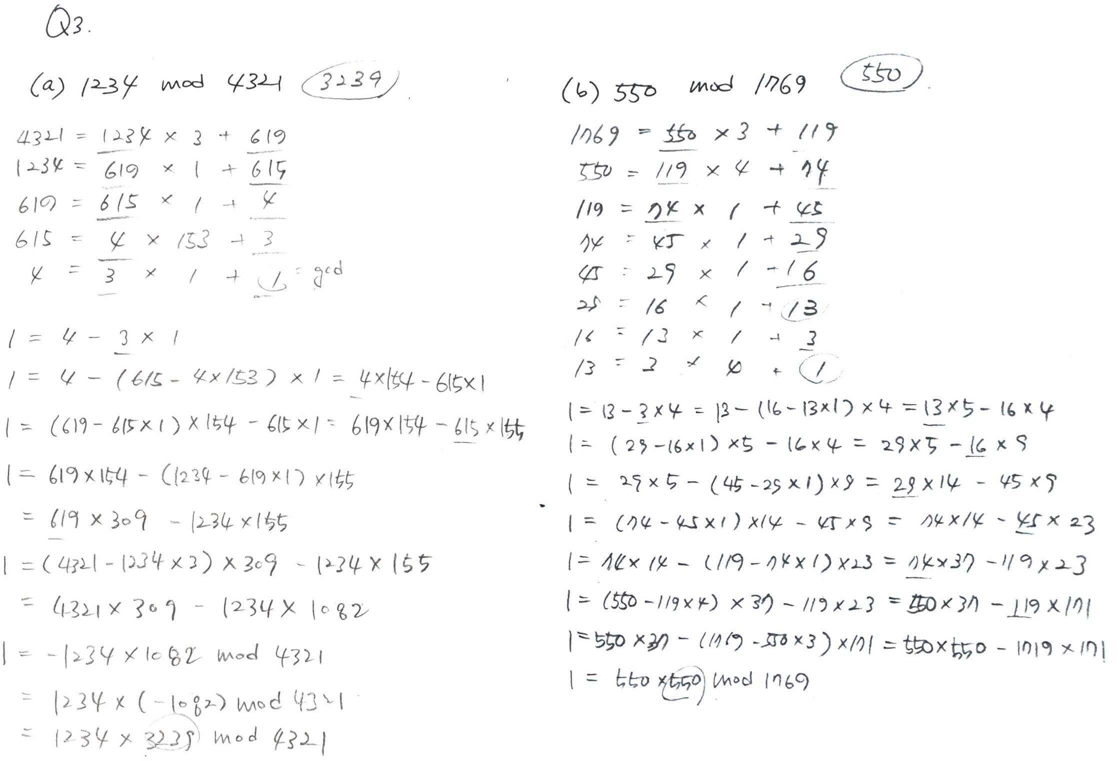
= gcd (35, 0)

= $\underline{35}$ //

Q3. (2 point) Using the "extended" Euclidean algorithm, find the multiplicative inverse of

(a) 1234 mod 4321 Answer) 3239

(b) 550 mod 1769 Answer) 550



Q3.

(a) 1234 mod 4321 $\boxed{3239}$.

$4321 = 1234 \times 3 + 619$
$1234 = 619 \times 1 + 615$
$619 = 615 \times 1 + 4$
$615 = 4 \times 153 + 3$
$4 = 3 \times 1 + 1 = \gcd$

$1 = 4 - 3 \times 1$
$1 = 4 - (615 - 4 \times 153) \times 1 = 4 \times 154 - 615 \times 1$
$1 = (619 - 615 \times 1) \times 154 - 615 \times 1 = 619 \times 154 - 615 \times 155$
$1 = 619 \times 154 - (1234 - 619 \times 1) \times 155$
$= 619 \times 309 - 1234 \times 155$
$1 = (4321 - 1234 \times 3) \times 309 - 1234 \times 155$
$= 4321 \times 309 - 1234 \times 1082$
$1 = -1234 \times 1082 \bmod 4321$
$= 1234 \times (-1082) \bmod 4321$
$= 1234 \times \boxed{3239} \bmod 4321$

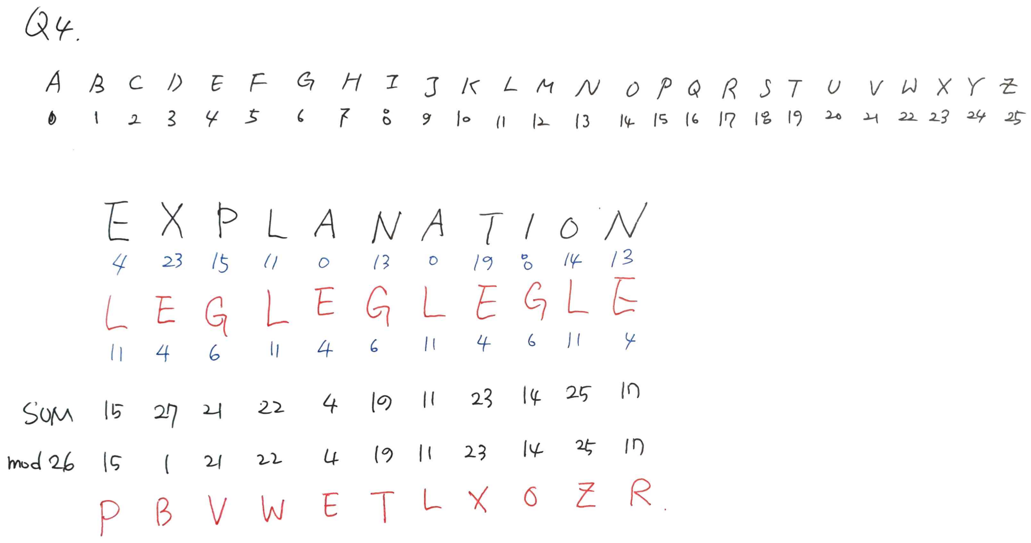
(b) 550 mod 1769 $\boxed{550}$.

$1769 = 550 \times 3 + 119$
$550 = 119 \times 4 + 74$
$119 = 74 \times 1 + 45$
$74 = 45 \times 1 + 29$
$45 = 29 \times 1 + 16$
$29 = 16 \times 1 + 13$
$16 = 13 \times 1 + 3$
$13 = 3 \times 4 + 1$

$1 = 13 - 3 \times 4 = 13 - (16 - 13 \times 1) \times 4 = 13 \times 5 - 16 \times 4$
$1 = (29 - 16 \times 1) \times 5 - 16 \times 4 = 29 \times 5 - 16 \times 9$
$1 = 29 \times 5 - (45 - 29 \times 1) \times 9 = 29 \times 14 - 45 \times 9$
$1 = (74 - 45 \times 1) \times 14 - 45 \times 9 = 74 \times 14 - 45 \times 23$
$1 = 74 \times 14 - (119 - 74 \times 1) \times 23 = 74 \times 37 - 119 \times 23$
$1 = (550 - 119 \times 4) \times 37 - 119 \times 23 = 550 \times 37 - 119 \times 171$
$1 = 550 \times 37 - (1769 - 550 \times 3) \times 171 = 550 \times 550 - 1769 \times 171$
$1 = 550 \times \boxed{550} \bmod 1769$

Q4. (1 point) Using the Vigenère cipher, encrypt the word "explanation" using the key "leg".

Answer) pbvwetlxozr



Q4.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

	E	X	P	L	A	N	A	T	I	O	N
	4	23	15	11	0	13	0	19	8	14	13
	L	E	G	L	E	G	L	E	G	L	E
	11	4	6	11	4	6	11	4	6	11	4
SUM	15	27	21	22	4	19	11	23	14	25	17
mod 26	15	1	21	22	4	19	11	23	14	25	17
	P	B	V	W	E	T	L	X	O	Z	R.

Q5. (2 points) This problem explores using a one-time pad version of the Vigenère cipher. In this scheme, the key is a stream of random numbers between 0 and 26. For example, if the key is 3 19 5..., then the first letter of the plaintext is encrypted with a shift of 3 letters, the second with a shift of 19 letters, the third with a shift of 5 letters, and so on.

(a) Encrypt the plaintext "send more money" with the keystream
9 0 1 7 23 15 21 14 11 11 2 8 9
Answer) beokjdmsxzpmh

Q5.

(a)	S	E	N	D	M	O	R	E	M	O	N	E	Y.
	18	4	13	3	12	14	17	4	12	14	13	4	24
	9	0	1	7	23	15	21	14	11	11	2	8	9
SUM	27	4	14	10	35	29	38	18	23	25	15	12	33
mod 26	1	4	14	10	9	3	12	18	23	25	15	12	7
	B	E	O	K	J	D	M	S	X	Z	P	M	H

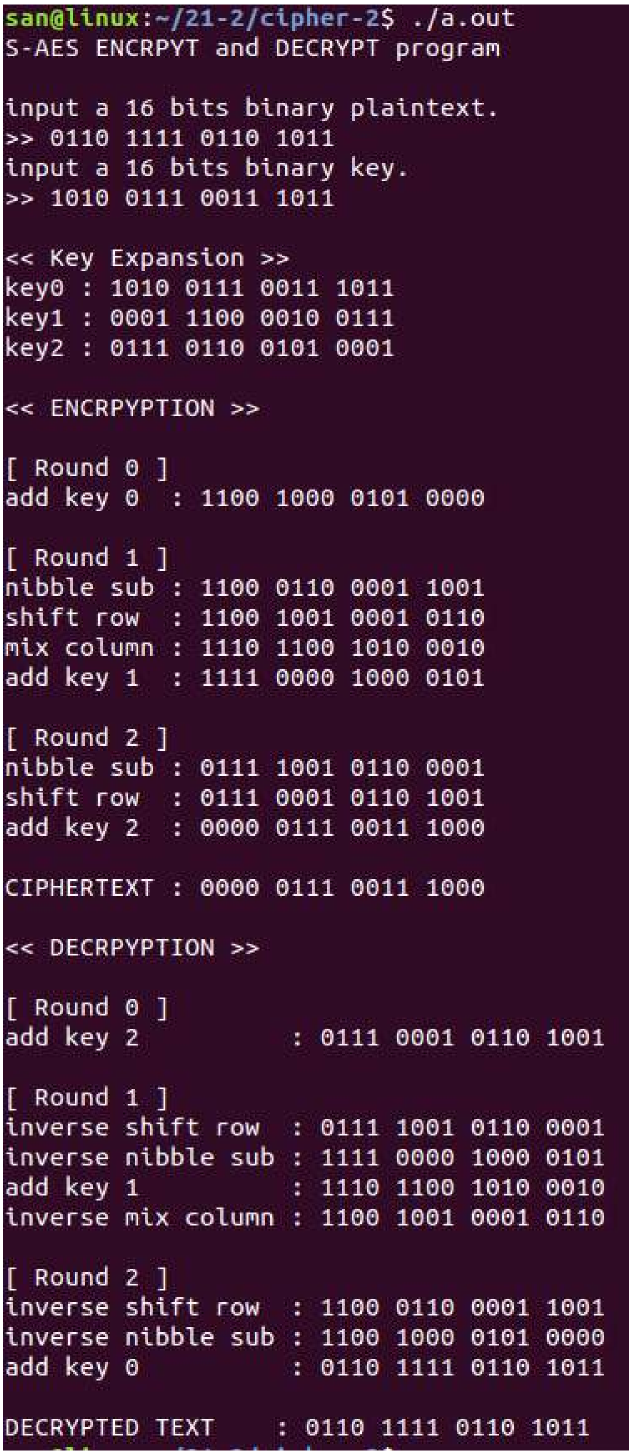
(b) Using the ciphertext produced in part (a), find a key so that the ciphertext decrypts to the plaintext "cashnotneeded."
Answer) zewdwptftvmie

(b)	B	E	O	K	J	D	M	S	X	Z	P	M	H
	1	4	14	10	9	3	12	18	23	25	15	12	7
plain	C	A	S	H	N	O	T	N	E	E	D	E	D
	2	0	18	7	13	14	19	13	4	4	3	4	3
Sub	-1	4	-4	3	-4	-11	-7	5	19	21	12	8	4.
mod 26	25	4	22	3	22	15	19	5	19	21	12	8	4
	Z	E	W	D	W	P	T	F	T	V	M	I	E.

Q6. (8 points: 4 points for encryption and 4 points for decryption)

Write a programming code that can encrypt and decrypt using S-AES (Simplified AES). Decryption should work correspondingly.

test case) plaintext 0110 1111 0110 1011 (“ok” in ASCII)
key 1010 0111 0011 1011
ciphertext 0000 0111 0011 1000



```
san@linux:~/21-2/cipher-2$ ./a.out
S-AES ENCRPYT and DECRYPT program

input a 16 bits binary plaintext.
>> 0110 1111 0110 1011
input a 16 bits binary key.
>> 1010 0111 0011 1011

<< Key Expansion >>
key0 : 1010 0111 0011 1011
key1 : 0001 1100 0010 0111
key2 : 0111 0110 0101 0001

<< ENCRPYPTION >>

[ Round 0 ]
add key 0   : 1100 1000 0101 0000

[ Round 1 ]
nibble sub : 1100 0110 0001 1001
shift row   : 1100 1001 0001 0110
mix column : 1110 1100 1010 0010
add key 1   : 1111 0000 1000 0101

[ Round 2 ]
nibble sub : 0111 1001 0110 0001
shift row   : 0111 0001 0110 1001
add key 2   : 0000 0111 0011 1000

CIPHERTEXT : 0000 0111 0011 1000

<< DECRPYPTION >>

[ Round 0 ]
add key 2           : 0111 0001 0110 1001

[ Round 1 ]
inverse shift row   : 0111 1001 0110 0001
inverse nibble sub : 1111 0000 1000 0101
add key 1           : 1110 1100 1010 0010
inverse mix column : 1100 1001 0001 0110

[ Round 2 ]
inverse shift row   : 1100 0110 0001 1001
inverse nibble sub : 1100 1000 0101 0000
add key 0           : 0110 1111 0110 1011

DECRYPTED TEXT    : 0110 1111 0110 1011
```

I removed most of the printarr functions because of the code length.
The file “ase.c” has the original code that shows the process of s-aes.

saes.c

```
//20172655 LEE KANG SAN
#include <stdio.h>
#include <string.h>

void key_expansion(int *key0, int *key1, int *key2);
//make key1 and key2 by expanding key0.
void input(int (*arr)[8], int *key);
void addkey(int (*arr)[8], int *key);
//xor with key, same when encrypt and decrypt.
void nibble_substitution(int (*arr)[8], int inverse);
//change the value of nibble from the s-box, when decrypt, use the another
s-box called inverse s-box.
void shift_row(int (*arr)[8]);
//shift two bottom nibbles, same when encrypt and decrypt.
void mix_column(int (*arr)[8], int inverse);
//change nibble values, then do matrix multiplication with [1,4,4,1]
//when decrypt, do matrix multiplication with [2,9,9,2]
void printarr(int (*arr)[8]);

int s1[4][4]={9, 4, 10, 11, 13, 1, 8, 5, 6, 2, 0, 3, 12, 14, 15, 7}; //s-box
int s2[4][4]={10, 5, 9, 11, 1, 7, 8, 15, 6, 0, 2, 3, 12, 4, 13, 14}; //ivnerse s-box

int main() {
        int plain[2][8]; //plain = n0 n1 n2 n3
        int key0[16], key1[16], key2[16]; //key0=w0w1, key1=w2w3, key2=w4w5
        //using int type array, assume that each cell represents 1 bit.

        printf("S-AES ENCRPYT and DECRYPT program\n");
        /* input a 16 bits length binary plaintext and key. */
        input(plain, key0); //input plain text, key
        /* key expansion */
        key_expansion(key0, key1, key2); //key expansion

        printf("\n<< ENCRPYPTION >>\n");

        printf("\n[ Round 0 ]\n");
        addkey(plain, key0); //xor with key 0
        printf("\n[ Round 1 ]\n");
        nibble_substitution(plain, 0); //nibble substitution using s-box
        shift_row(plain); //shift row(2 bottom nibbles, n1 and n3)
        mix_column(plain, 0); //mix column
        addkey(plain, key1); //xor with key 1

        printf("\n[ Round 2 ]\n");
        nibble_substitution(plain, 0); //nibble substitution using s-box
        shift_row(plain); //shift row(2 bottom nibbles, n1 and n3)
        addkey(plain, key2); //xor with key 2

        printf("\nCIPHERTEXT : ");
        printarr(plain);
        printf("\n");

        printf("<< DECRPYPTION >>\n");

        printf("\n[ Round 0 ]\n");
        addkey(plain, key2); //xor with key 2

        printf("\n[ Round 1 ]\n");
        shift_row(plain); //inverse shift row(2 bottom nibbles, n1 and n3)
        nibble_substitution(plain, 1); //inverse nibble substitution using inverse s-box
        addkey(plain, key1); //xor with key 1
        mix_column(plain, 1); //inverse mix column

        printf("\n[ Round 2 ]\n");
        shift_row(plain); //inverse shift row(2 bottom nibbles, n1 and n3)
        nibble_substitution(plain, 1); //inverse nibble substitution using inverse s-box
        addkey(plain, key0); //xor with key 0

        printf("\nDECRYPTED TEXT     : ");
        printarr(plain);
        return 0;
}

void key_expansion(int *key0, int *key1, int *key2) {
//make key1 and key2 by calculating key0
```

```
int rcon1[8]={1,0,0,0,0,0,0,0};
int rcon2[8]={0,0,1,1,0,0,0,0};
int w0[8], w1[8], w2[8], w3[8], w4[8], w5[8], temp, a, b, n1, n2;
int x1[8], x3[8]; //for rotate and substitute w1, w3

//get w0, w1 from key0
for(int i=0; i<8; i++) {
        w0[i]=key0[i];
        x1[i]=w1[i]=key0[i+8];
}

/* key 1 */
//x1 = rot(w1)
for(int i=0; i<4; i++) {
        temp=x1[i];
        x1[i]=x1[i+4];
        x1[i+4]=temp;
}
//x1 = sub(rot(w1))
a=x1[0]; a=a<<1; a=a|x1[1];
b=x1[2]; b=b<<1; b=b|x1[3];
n1=s1[a][b];
a=x1[4]; a=a<<1; a=a|x1[5];
b=x1[6]; b=b<<1; b=b|x1[7];
n2=s1[a][b]; //get nible value from s-box

//write binary format in x1 array
for(int i=3; i>=0; i--) {
        x1[i]=n1;
        x1[i]=x1[i]&1;
        n1=n1>>1;
}
for(int i=7; i>=4; i--) {
        x1[i]=n2;
        x1[i]=x1[i]&1;
        n2=n2>>1;
}

//get w2 = w1 ^ rcon(1) ^ sub(rot(w1))
for(int i=0; i<8; i++)
```

```
                w2[i]=w0[i]^rcon1[i]^x1[i];

        //get w3 = w2 ^ w1
        for(int i=0; i<8; i++)
                w3[i]=w2[i]^w1[i];

        //get key 1
        for(int i=0; i<8; i++) {
                key1[i]=w2[i];
                key1[i+8]=w3[i];
        }
//////////////////////////////////////////////
        /* key 2 */

        //get w3
        for(int i=0; i<8; i++)
                x3[i]=w3[i];

        //x3 = rot(w3)
        for(int i=0; i<4; i++) {
                temp=x3[i];
                x3[i]=x3[i+4];
                x3[i+4]=temp;
        }

        //x3 = sub(rot(w3))
        a=x3[0]; a=a<<1; a=a|x3[1];
        b=x3[2]; b=b<<1; b=b|x3[3];
        n1=s1[a][b];
        a=x3[4]; a=a<<1; a=a|x3[5];
        b=x3[6]; b=b<<1; b=b|x3[7];
        n2=s1[a][b]; //get nible value from s-box

        //write binary format in x3 array
        for(int i=3; i>=0; i--) {
                x3[i]=n1;
                x3[i]=x3[i]&1;
                n1=n1>>1;
        }
        for(int i=7; i>=4; i--) {
            x3[i]=n2;
            x3[i]=x3[i]&1;
            n2=n2>>1;
        }

        //get w4 = w2 ^ rcon(2) ^ sub(rot(w3))
        for(int i=0; i<8; i++)
            w4[i]=w2[i]^rcon2[i]^x3[i];

        //get w5 = w4 ^ w3
        for(int i=0; i<8; i++)
            w5[i]=w4[i]^w3[i];

        //get key 2
        for(int i=0; i<8; i++) {
            key2[i]=w4[i];
            key2[i+8]=w5[i];
        }
}
void input(int (*arr)[8], int *key) {
//input function
        char input[50];
        int trim[16], n=0;
        printf("\ninput a 16 bits binary plaintext.\n");
        printf(">> ");
        fgets(input, 49, stdin);

        //remove white space
        for(int i=0, j=0; j<16; i++)
            if(input[i]=='0' || input[i]=='1')
                trim[j++]=input[i]-'0';

        //set plain text = N0 N1 N2 N3
        for(int i=0; i<4; i++) //N0
            arr[0][i]=trim[n++];
        for(int i=0; i<4; i++) //N1
            arr[1][i]=trim[n++];
        for(int i=4; i<8; i++) //N2
            arr[0][i]=trim[n++];
        for(int i=4; i<8; i++) //N3
```

```
                arr[1][i]=trim[n++];

        printf("input a 16 bits binary key.\n");
        printf(">> ");
        fgets(input, 49, stdin);

        //remove white space
        for(int i=0, j=0; j<16; i++)
                if(input[i]=='0' || input[i]=='1')
                        trim[j++]=input[i]-'0';

        //set key0
        for(int i=0, n=0; i<16; i++)
                        key[i]=trim[n++];
}

void addkey(int (*arr)[8], int *key) {
//xor with key 16 bit key
        int n=0;
        //N0
        for(int i=0; i<4; i++)
                arr[0][i]=arr[0][i]^key[n++];
        //N1
        for(int i=0; i<4; i++)
                arr[1][i]=arr[1][i]^key[n++];
        //N2
        for(int i=4; i<8; i++)
                arr[0][i]=arr[0][i]^key[n++];
        //N3
        for(int i=4; i<8; i++)
                arr[1][i]=arr[1][i]^key[n++];
}
void nibble_substitution(int (*arr)[8], int inverse) {
//using nibble bits as a index of s-box, substitute with the s-box value.
        int (*sbox)[4];
        int nib[4];
        int i, j, temp;

        //select sbox
        if(inverse==0) sbox=s1;
        else if(inverse==1) sbox=s2;

        //get nibble value
        for(int k=0, m=0, n=0; k<4; k++)
        {
                if(k==2) {n=0; m++;}
                temp=0;
                temp=temp|arr[m][n++];
                temp=temp<<1;
                temp=temp|arr[m][n++];
                i=temp;

                temp=0;
                temp=temp|arr[m][n++];
                temp=temp<<1;
                temp=temp|arr[m][n++];
                j=temp;
                nib[k]=sbox[i][j];
        }

        //substitution(calc bit by bit)
        for(int i=3; i>=0; i--) {
                arr[0][i]=nib[0];
                arr[0][i]=arr[0][i]&1;
                nib[0]=nib[0]>>1;
        }
        for(int i=7; i>=4; i--) {
                arr[0][i]=nib[1];
                arr[0][i]=arr[0][i]&1;
                nib[1]=nib[1]>>1;
        }
        for(int i=3; i>=0; i--) {
                arr[1][i]=nib[2];
                arr[1][i]=arr[1][i]&1;
                nib[2]=nib[2]>>1;
        }
        for(int i=7; i>=4; i--) {
                arr[1][i]=nib[3];
                arr[1][i]=arr[1][i]&1;
                nib[3]=nib[3]>>1;
        }
}
void shift_row(int (*arr)[8]) {
//shift the bottom nibbles, n1 and n3
        int temp;
        for(int i=0; i<4; i++) {
                temp=arr[1][i];
                arr[1][i]=arr[1][i+4];
                arr[1][i+4]=temp;
        }
}
void mix_column(int (*arr)[8], int inverse) {
        int s00=0, s01=0, s10=0, s11=0;
        int mul2[16]={0,2,4,6,8,10,12,14,3,1,7,5,11,9,15,13};
        int mul4[16]={0,4,8,12,3,7,11,15,6,2,14,10,5,1,13,9};
        int mul9[16]={0,9,1,8,2,11,3,10,4,13,5,12,6,15,7,14};

        for(int i=0; i<4; i++) { //get s00, s10
                s00=s00<<1;
                s00=s00|arr[0][i];
                s10=s10<<1;
                s10=s10|arr[1][i];
        }
        for(int i=4; i<8; i++) { //get s01, s11
                s01=s01<<1;
                s01=s01|arr[0][i];
                s11=s11<<1;
                s11=s11|arr[1][i];
        }
        //hear
        int x00, x01, x10, x11;
        if(inverse==0) { //calculate s'00, s'01, s'10, s'11
                x00=s00^mul4[s10];
                x10=mul4[s00]^s10;
                x01=s01^mul4[s11];
                x11=mul4[s01]^s11;
        }
        else if(inverse==1) { //calculate when inverse
                x00=mul9[s00]^mul2[s10];
                x10=mul2[s00]^mul9[s10];
        x01=mul9[s01]^mul2[s11];
        x11=mul2[s01]^mul9[s11];
    }
    //
    for(int i=3; i>=0; i--) { //put s'00 into arr
        arr[0][i]=x00;
        arr[0][i]=arr[0][i]&1;
        x00=x00>>1;
    }
    for(int i=7; i>=4; i--) { //put s'01 into arr
        arr[0][i]=x01;
        arr[0][i]=arr[0][i]&1;
        x01=x01>>1;
    }
    for(int i=3; i>=0; i--) { //put s'10 into arr
        arr[1][i]=x10;
        arr[1][i]=arr[1][i]&1;
        x10=x10>>1;
    }
    for(int i=7; i>=4; i--) { //put s'11 into arr
        arr[1][i]=x11;
        arr[1][i]=arr[1][i]&1;
        x11=x11>>1;
    }
}
void printarr(int (*arr)[8]) {
//print current state to see the process of s-aes.
    for(int i=0; i<4; i++) //N0
        printf("%d", arr[0][i]);
    printf(" ");
    for(int i=0; i<4; i++) //N1
        printf("%d", arr[1][i]);
    printf(" ");
    for(int i=4; i<8; i++) //N2
        printf("%d", arr[0][i]);
    printf(" ");
    for(int i=4; i<8; i++) //N3
        printf("%d", arr[1][i]);
    printf("\n");
}
```