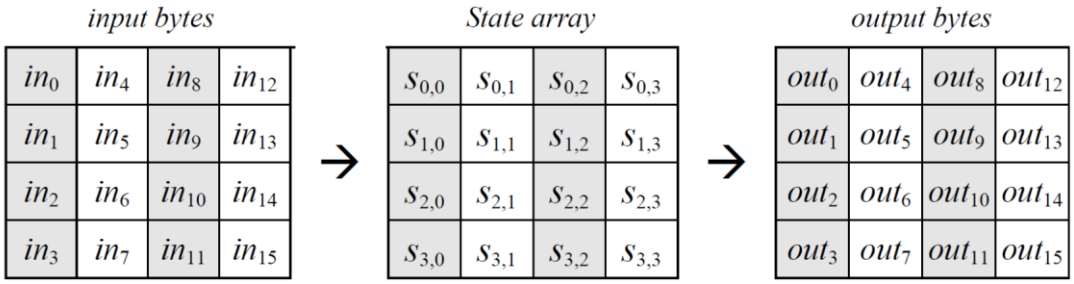
2021 국민대학교 정보보안암호수학과 암호분석경진대회

2번 문제 : 32-bit 자료형을 활용한 AES 구현 (출제: 동아리 CO2)

Advanced Encryption Standard (AES)는 2001년 National Institute of Standards and Technology (NIST)에 의해 제정된 암호화 방식으로, Rijndael 알고리즘에 기반하였다. 기존 블록 암호 알고리즘 Data Encryption Standard (DES)에서 사용하는 Feistel 구조와 다르게, AES 알고리즘은 Substitution, Permutation을 사용하여 연산을 처리한다.



[그림 1] AES state 표기 방안 ($in_k, out_j, S_{i,j} \in \{0,1\}^8$) [1]

AES는 128-bit 블록, 128/192/256-bit의 키를 사용하는 암호 알고리즘이며, 현재 가장 많이 사용되고 있는 대칭 키 암호 알고리즘이다. AES 알고리즘은 8차 기약 다항식 기반으로 연산 처리되며, 코드상으로 표기하는 경우, 8-bit 단위로 연산이 진행된다. 곱셈의 경우, $m(x) = x^8 + x^4 + x^3 + x + 1$ 기반 기약 다항식을 이용하여 곱셈이 정의된다.

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (polynomial notation);

{01010111} ⊕ {10000011} = {11010100} (binary notation);

{57} ⊕ {83} = {d4} (hexadecimal notation).

[그림 2] AES 알고리즘 Addition 정의 [1]

$m(x) = x^8 + x^4 + x^3 + x + 1$, x8=x4+x3+x+1 Irreducible polynomial(기약다항식)

For example, {57} • {83} = {c1}, because

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

and

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1.$$

[그림 3] AES 알고리즘 Multiplication 정의 [1]

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])                  // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                              // See Sec. 5.1.1
      ShiftRows(state)                             // See Sec. 5.1.2
      MixColumns(state)                            // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

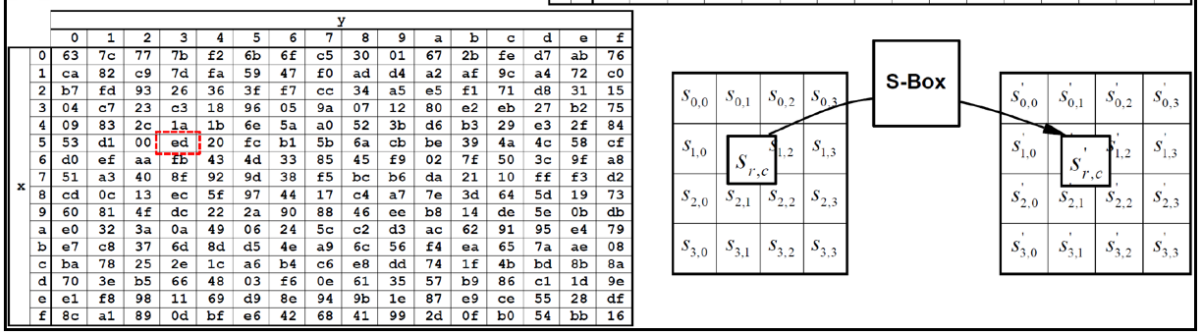
[그림 4] AES 암호화 pseudocode [1]

AES의 암호화 방식은 총 4가지 단계($SubBytes$–> $ShiftRows$–> $MixColumns$–> $AddRoundKey$)가 하나의 라운드로 구성되며 AES-128의 경우 10라운드로 진행된다. 초기 라운드는 $AddRoundKey$ 과정이 진행되며, 마지막 라운드에는 $MixColumns$ 과정이 제외된다.

1. *SubBytes*

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

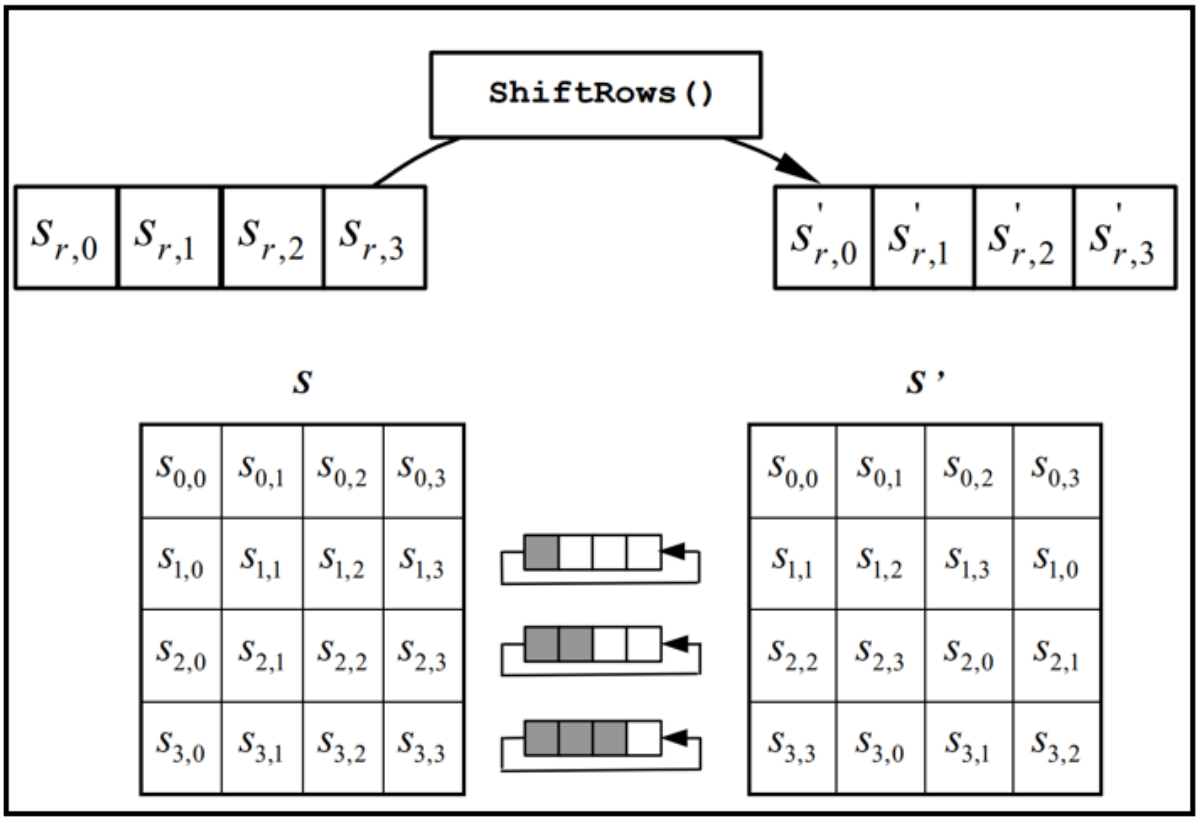


[그림 5] AES *SubBytes* 요약 [1]

$SubBytes$ 는 Non-linear byte substitution으로(Affine Transform), $GF(2^8)$ 서의 역원을 계산한다

(0x00은 0x00으로 대응됨). 마지막 과정으로 $b'_i = b_i + b_{(i+4)mod8} + b_{(i+5)mod8} + b_{(i+6)mod8} + b_{(i+7)mod8} + c_i$ $(0 \le i < 8)$ 연산 처리를 거친다. 이외에도 $SubBytes$ 과정은 사전 계산된 s-box 테이블을 참조하는 방식으로 구현할 수 있다.

2. *ShiftRows*



[그림 6] AES *ShiftRows* 요약 [1]

$ShiftRows$는 $S'_{r,c} = S_{\{r,(c+shift(r,Nb) mod\ Nb)\}}\ for\ 0 < r < 4\ and\ 0 \le c < Nb$ 연산이 수행되며 $state$의 각 행에 대한 순환 시프트 연산을 수행한다. [그림 6]은 AES $ShiftRows$에 대한 요약 그림이다.

3. *MixColumns*

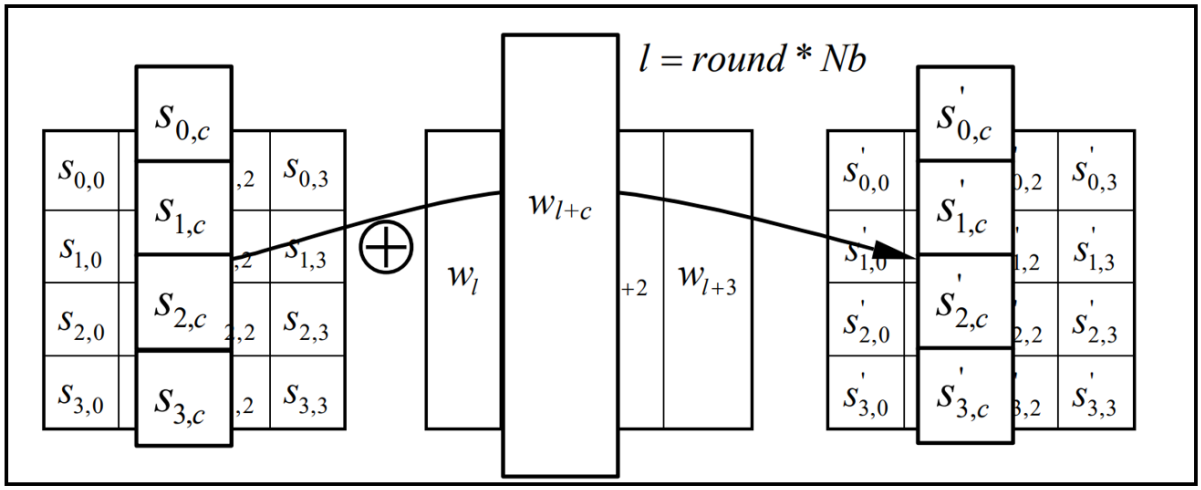
$MixColumns$는 $state$의 $Column\ by\ Column$에 대해 수행되는 연산으로, $Column$을 4개의 term으로 구성된 다항식으로 간주한다. 각 $Column$은 $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$와 $x^4 + 1$ 상에서의 곱셈을 수행한다. 곱셈 연산은 [그림 3]에서 정의된 곱셈 연산 방식을 사용한다.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb$$

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

[그림 7] AES *MixColumns* 요약 [1]

4. *AddRoundKey*



[그림 8] AES *Addroundkey* 요약 [1]

AES의 $AddRoundKey$는 $state$에 $roundkey$를 bitwise $XOR(\oplus)$ 연산을 수행한다. $roundkey$는 초기에 입력된 $masterkey$를 사용하여 확장하는 과정인 $Key\ Expansion$ 과정을 거친다. $Key\ Expansion$ 과정은 다음절에 서술되어 있다. $roundkey$는 사용하는 라운드의 수만큼 생성되며, AES-128의 경우, 총 10개의 $roundkey$가 생성된다. 초기 라운드의 경우 $masterkey$와 연산을 수행하며, 남은 라운드의 경우 $roundkey$를 사용한다.

5. *Key Expansion*

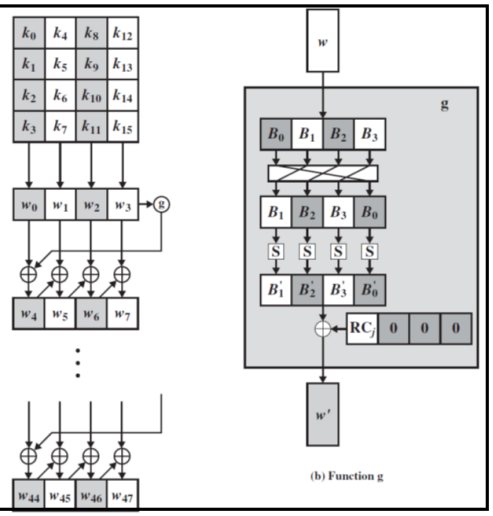
```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```



[그림 9] AES *Key Expansion* 요약 [1]

AES의 $Key\ Expansion$ 과정은 초기 입력 $masterkey$를 통해 각 라운드에 사용할 $roundkey$를 만드는 작업이다. AES-128의 경우, $masterkey$를 사용하여 총 10개의 $roundkey$를 생성한다. [그림 9]는 $roundkey$를 생성하는 과정을 요약한 그림이다. [그림 9]에서 $RotWord(a_0, a_1, a_2, a_3) \rightarrow (a_1, a_2, a_3, a_0)$는 행에 대한 순환 시프트 연산처리 과정이다. $SubWord$ 연산의 경우 하나의 행에 대해 $SubBytes$ 연산하

는 과정이다. 즉, $SubWord(a_0, a_1, a_2, a_3) = (sbox(a_0), sbox(a_1), sbox(a_2), sbox(a_3))$ 과정을 의미한다. 또한 $Rcon = 0x01000000$ 부터 시작한다.

6. 32-bit Table을 활용한 AES 구현 방안

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

$$= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[a_{1,j-1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[a_{2,j-2}] \right)$$

$$\oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

$$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[x] \right) \quad T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[x] \right)$$

$$T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[x] \right) \quad T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[x] \right)$$

[그림 9] AES *SubBytes* & *MixColumns* 결합 방안 요약

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb$$

$Table_0$ $Table_1$ $Table_2$ $Table_3$

[그림 10] AES $32-bit\ Table$ 생성 방안 요약

AES의 라운드 함수 4가지 과정 중, *SubBytes* 와 *ShiftRows* 의 과정은 순서 변경이 가능하며, *SubBytes* 와 *MixColumns* 과정은 결합하여 테이블로 구현하는 방안이 존재한다. *SubBytes* & *MixColumns* 테이블은 8-bit 입력 32-bit 출력 테이블이 총 4개 필요하다. *Table*을 생성하는 방안은 다음과 같다.

- 입력 값이 x 인 경우, $Table_0$ 은 $(\{02\} * sbox(x) \ll 24) + (\{01\} * sbox(x) \ll 16) + (\{01\} * sbox(x) \ll 8) + \{03\} * sbox(x)$ 로 구성할 수 있다. $x \in \{0,1\}^8$
- 입력 값이 x 인 경우, $Table_1$ 은 $(\{03\} * sbox(x) \ll 24) + (\{02\} * sbox(x) \ll 16) + (\{01\} * sbox(x) \ll 8) + \{01\} * sbox(x)$ 로 구성할 수 있다. $x \in \{0,1\}^8$
- 입력 값이 x 인 경우, $Table_2$ 은 $(\{01\} * sbox(x) \ll 24) + (\{03\} * sbox(x) \ll 16) + (\{02\} * sbox(x) \ll 8) + \{01\} * sbox(x)$ 로 구성할 수 있다. $x \in \{0,1\}^8$
- 입력 값이 x 인 경우, $Table_3$ 은 $(\{01\} * sbox(x) \ll 24) + (\{01\} * sbox(x) \ll 16) + (\{03\} * sbox(x) \ll 8) + \{02\} * sbox(x)$ 로 구성할 수 있다. $x \in \{0,1\}^8$
- 실제 연산의 경우, *Subbytes* & *MixColumns* 과정은 32-bit 단위로 연산을 처리할 수 있다.

32-bit 입력($Column$) x 에 대해 $SubBytes$ & $MixColumns$ 의 32-bit 연산 결과 값은 $(Table_0(x \gg 24) \oplus (Table_1((x \gg 16)\&0xff) \oplus (Table_2((x \gg 8)\&0xff) \oplus (Table_3(x\&0xff))$ 이다.

[문제 2-1 (25점)] : 8-bit 자료형(unsigned char)만을 이용한 AES-128 암호화 알고리즘을 구현하시오

- ➔ ***SubBytes*** 과정 구현 (표준에서 주어진 S-box Table 활용)
- ➔ ***ShiftRows*** 과정 구현
- ➔ ***AddRoundKey*** 과정 구현
- ➔ ***MixColumns*** 과정 구현
- ➔ ***Key Expansion*** 과정 구현
- ➔ 전체 암호화 과정 구현

[문제 2-2 (25점)] : 32-bit 자료형(unsigned int)을 활용한 8-bit 입력 32-bit 출력 T-table을 생성하는 설계 방안 코드를 작성하고, T-table 4개에 대한 결과 값을 제시하시오.

[문제 2-3 (25점)] : 8-bit 입력 T-table 4개를 활용하여 ***Subbytes & Mixcolumns*** 연산 처리 방안을 활용한 AES 암호화 함수를 구현하시오.

(*실제로 구현 시, ***SubBytes & MixColumns*** 과정 추가로 ***ShiftRows*** 과정 또한 결합이 가능하다. ***ShiftRows*** & ***SubBytes & MixColumns*** 모두 결합하는 경우 가산점 부여)

[문제 2-4 (20점)] : 주어진 함수를 통해 AES-128 암호화 함수의 성능 측정 결과를 제시하시오

```c
#include <stdio.h>
#include <intrin.h>
__int64 cpucycles() {
    return __rdtsc();
}

void performance_result(unsigned char* pt, unsigned char* key, unsigned char* ct)
{
    unsigned long long cycle1, cycle2;
    cycle1 = cpucycles();
    for (int i = 0; i < 100000; i++) {
        //ENC_AES128(pt, key, ct);
        //암호화 함수 호출
    }
    cycle2 = cpucycles();
    printf("AES Encryption RDTSC = %10lld₩n", (cycle2 - cycle1) / 100000);
}
```

[문제 2-5 (5점)] : AES 암호화 알고리즘에서 8-bit 입력 32-bit 출력 T-table 4개를 사용하는 경우, T-table의 메모리 사용량을 구하고, T-table을 최소로 사용하는 경우의 메모리 사용량과 그 이유를 제시하시오.

[공지사항]

- 해당 문제의 답안 코드는 C-language 기반으로 설계되어야 합니다.
- 답안 제출 시, 사용한 플랫폼, OS를 기술하여야 합니다 (Example: Visual studio code, Window10).
- 성능 측정 함수 사용 시, iOS 기반 운영체제에서는 작동하지 않을 수 있습니다. iOS 사용 시, 성능 측정 결과는 제외하고 함수만 설계해서 제출해주시길 바랍니다.
- 모든 코드는 정상적으로 컴파일이 되어야하며, Test vector를 통한 암호문의 값이 틀린 경우, 감점 요소가 존재합니다.
- 제출 시, 솔루션 파일(.sln)이 아닌 헤더 파일(.h), 코드 파일(.c)만 제출해야 합니다. 2-5번 문제의 경우, main 함수 밑에 주석으로 작성 혹은 word 파일로 제출 부탁드립니다.
- **코드 카피가 적발된 경우, 모든 문제에서 0점처리 됨을 알립니다.**
- **AES 표준문서(참고문헌)을 적극적으로 사용하십시오. 해당 문서에 각 과정에 대한 결과 값이 존재합니다.**
- 문의는 ondoli0312@kookmin.ac.kr 로 메일 부탁드립니다.

[참고문헌]

[1] AES 표준문서: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf