void expandKey(BYTE *key, BYTE *roundKey)

1) key: 키 스케줄링을 수행할 16바이트 키

2) roundKey: 키 스케줄링의 결과인 176바이트 라운드 키가 담길 공간

STEP1 : 128비트의 키를 4개의 32비트 워드로 바꾼다.

STEP2 : 첫 4개의 워드 중 마지막 워드는 1바이트 왼쪽 순환 이동된 뒤 S-Box를 이용하여 치환된다. 그 다음에 라운드 상수와 XOR된다.

STEP3 : 첫 워드와 기존 4개의 워드 중 두번째 워드가 XOR되어 두번째 워드가 생성, 이결과를 반복한다.

STEP4 : 2~3을 9번 수행하여 라운트 키를 생성한다.

BYTE* subBytes(BYTE *block, int mode)

1) block: SubBytes 수행할 16바이트 블록, 수행 결과는 해당 배열에 바로 반영

2) mode: SubByte 수행 모드

Step1 : AES state를 Sbox통해서 치환

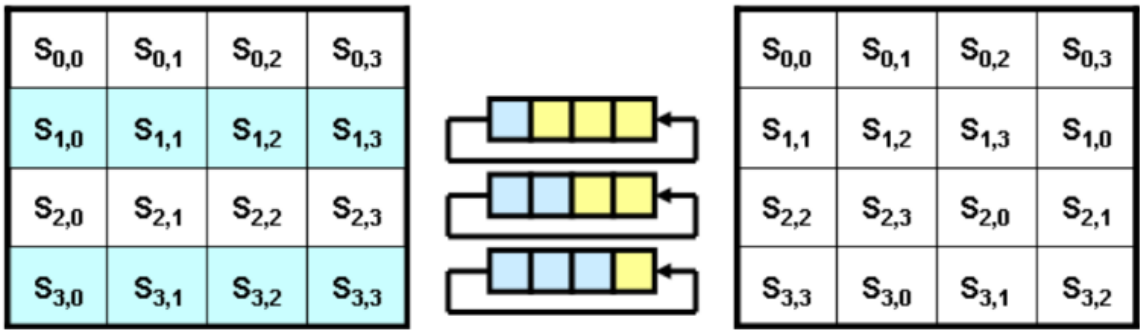
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	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

< Sbox >

BYTE* shiftRows(BYTE *block, int mode)

1) block: ShiftRows 수행할 16바이트 블록, 수행 결과는 해당 배열에 바로 반영

2) mode: ShiftRows수행 모드

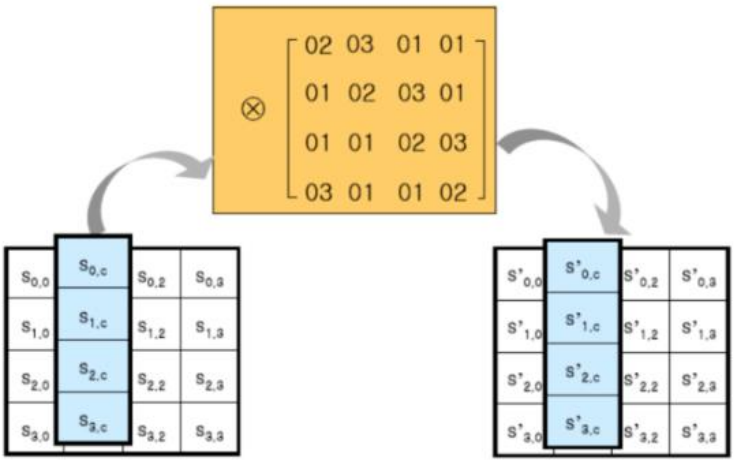


BYTE* mixColumns(BYTE *block, int mode)

1) block: MixColumns을 수행할 16바이트 블록, 수행 결과는 해당 배열에 바로 반영

2) mode: MixColumns의 수행 모드

Step : state에 각 Colume에 대해서 아래와 같이 행렬 곱셈을 수행



< State의 열에 Mixcolumn 변환을 적용한 결과 >

예) 02 • d4 + 03 • bf + 01 • 5d + 01 • 30

$= x(x^7+x^6+x^4+x^2) + (x+1)(x^7+x^5+x^4+x^3+x^2+x+1) + (x^6+x^4+x^3+x^2+1) + (x^5+x^4)$

$= x^2 \mod x^8+x^4+x^3+x+1 = 04$

BYTE* addRoundKey(BYTE *block, BYTE *rKey)

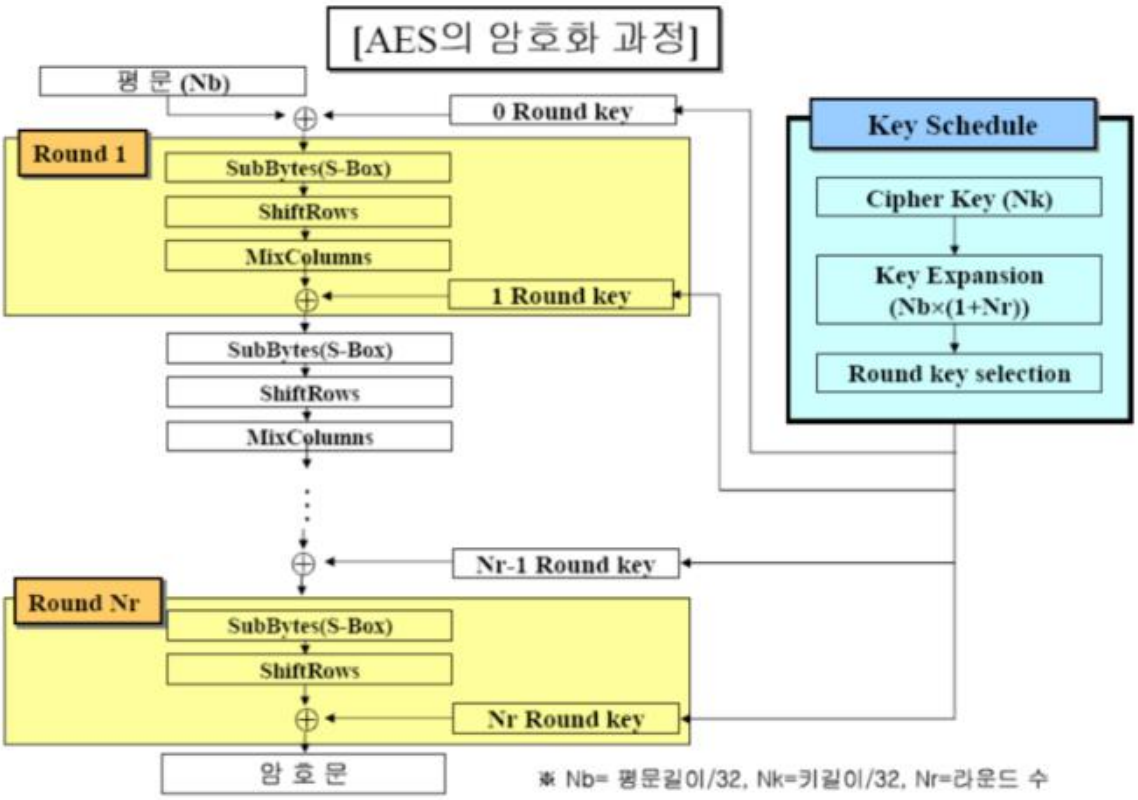
1) block: AddRoundKey를 수행할 16바이트 블록, 수행 결과는 해당 배열에 바로 반영

2) rKey: AddRoundKey를 수행할 16바이트 라운드키

Step : State내의 각각의 byte들이 각 roundkey와 xor연산되어진다

void AES128(BYTE *input, BYTE *output, BYTE *key, int mode)

mode가 ENC일 경우 평문을 암호화하고, DEC일 경우 암호문을 복호화하는 함수



minimal poly m(x) $m(x) = x^8 + x^4 + x^3 + x + 1$ 을 사용한 효율적인 $GF(2^8)$ 내 matrix multiplication

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

state의 단일 column $j(0 \le j \le 3)$ 에 대한 MixColumns Transformation은 다음과 같이 표현할 수 있다.

$$s'_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j}$$
$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$
$$s'_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

x에 곱하는 수	공식
0x01	결과 : x
0x02	1. x가 0x80보다 작을 때 x를 1비트 왼쪽으로 이동. 가장 우측 비트는 0으로 채워짐 2. x가 0x80보다 클때 x를 1비트 왼쪽으로 이동. 가장 우측 비트는 0으로 채워짐 -(1) (1)에서 나온 값과 0x1b를 XOR 연산
0x03	x * 0x03 = x * (0x02 ^ 0x01) = (x * 0x02) ^ (x * 0x01)
0x09	x * 0x09 = x * (0x08 ^ 0x01) = (b * 0x02 * 0x02 * 0x02) ^ (b * 0x01)
0x0b	x * 0x0b = b * (0x08 ^ 0x02 ^ 0x01) = (b * 0x02 * 0x02 * 0x02) ^ (b * 0x02) ^ (b * 0x01)
0x0d	x * 0x0d = b * (0x08 ^ 0x04 ^ 0x01) = (b * 0x08) ^ (b * 0x04) ^ (b * 0x01) = (b * 0x02 * 0x02 * 0x02) ^ (b * 0x02 * 0x02) ^ (b * 0x01)
ox0e	x * 0x0e = b * (0x08 ^ 0x04 ^ 0x02) = (b * 0x02 * 0x02 * 0x02) ^ (b * 0x02 * 0x02) ^ (b * 0x02)