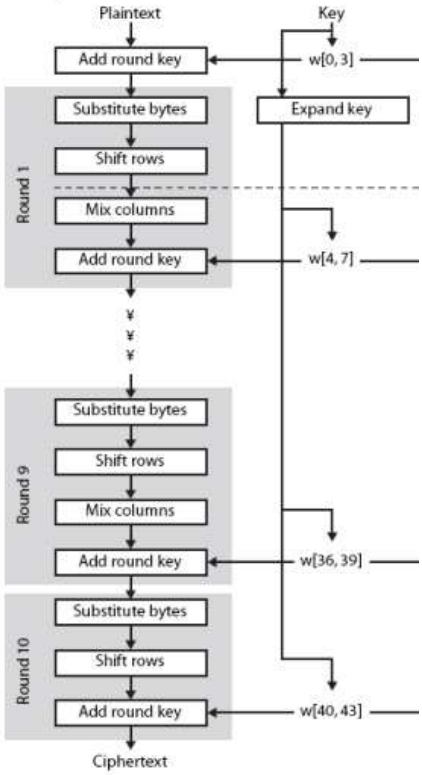
2019102652 유동민

Cipher

```
void Cipher(uint8_t *state, const uint32_t *roundKey, int mode)
{
  int round = 0;
  if(mode == ENCRYPT) {
    addRoundKey(state, &roundKey[0]);
    for(round = 1; round < 10; round++){
      subBytes(state, mode);
      shiftRows(state, mode);
      mixColumns(state, mode);
      addRoundKey(state, &roundKey[(round)*(KEYLEN/4)]);
    }
    subBytes(state, mode);
    shiftRows(state, mode);
    addRoundKey(state, &roundKey[40]);
  }
  else if(mode == DECRYPT) {
      uint8_t *tmp;
      addRoundKey(state, &roundKey[40]);
      for(round = 1; round < 10; round++) {
          shiftRows(state, mode);
          subBytes(state, mode);
          tmp = &roundKey[RNDKEYSIZE-(4*round)-4];
          addRoundKey(state,tmp);
          mixColumns(state, mode);
      }
     subBytes(state, mode);
     shiftRows(state, mode);
     addRoundKey(state, &roundKey[0]);
  }
  else{
    fprintf(stderr, "Invalid mode!\n");
    exit(1);
  }
}
```



코드설명:

ENC: 최초 Plaintext를 키와 xor합니다. 그 다음 Sub bytes, shift rows, mix columns, add round key를 실행합니다. 이를 10회 반복합니다. (마지막 라운드는 mix columns를 하지 않는다.)

DEC: 최초 Ciphertext를 마지막 라운드 키와 xor합니다. 그 다음 inverse sub bytes, invers shift rows, inverse mix columns, add round key를 실행합니다. 이를 10회 반복합니다. (마지막 라운드는 min columns를 하지 않는다.)

결과:

```
<키>
0f 15 71 c9 47 d9 e8 59 0c b7 ad d6 af 7f 67 98
<라운드 키>
0f 15 71 c9 47 d9 e8 59 0c b7 ad d6 af 7f 67 98
dc 90 37 b0 9b 49 df e9 97 fe 72 3f 38 81 15 a7
d2 c9 6b b7 49 80 b4 5e de 7e c6 61 e6 ff d3 c6
c0 af df 39 89 2f 6b 67 57 51 ad 06 b1 ae 7e c0
2c 5c 65 f1 a5 73 0e 96 f2 22 a3 90 43 8c dd 50
58 9d 36 eb fd ee 38 7d 0f cc 9b ed 4c 40 46 bd
71 c7 4c c2 8c 29 74 bf 83 e5 ef 52 cf a5 a9 ef
37 14 93 48 bb 3d e7 f7 38 d8 08 a5 f7 7d a1 4a
48 26 45 20 f3 1b a2 d7 cb c3 aa 72 3c be 0b 38
fd 0d 42 cb 0e 16 e0 1c c5 d5 4a 6e f9 6b 41 56
b4 8e f3 52 ba 98 13 4e 7f 4d 59 20 86 26 18 76
---
<평문>
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
<암호문>
ff 0b 84 4a 08 53 bf 7c 69 34 ab 43 64 14 8f b9
<복호문>
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
<역암호문>
1f e0 22 1f 19 67 12 c4 be cd 5c 1c 60 71 ba a6
<복호문>
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
Random testing.....No error found
```

KeyExpansion

```
void KeyExpansion(const uint8_t *key, uint32_t *roundKey)
{
  int round = 0;
  int count = 0;
  uint8_t tmp[4];
  uint8_t rcon = 0x01;
  uint8_t word[KEYLEN/4];
  for(round = 0; round < 11; round ++) {
    for(count = 0 ; count < KEYLEN/4 ; count++) {
    if(round == 0) {
      roundKey[count] = *((uint32_t *)(key+(count*4)));
      continue;
    }
    else if(count == 0){
      tmp[0] = sbox[*((uint8_t *) (roundKey)+(round*KEYLEN-3))] ^ Rcon[round];
      tmp[1] = sbox[*((uint8_t *) (roundKey)+(round*KEYLEN-2))];
      tmp[2] = sbox[*((uint8_t *) (roundKey)+(round*KEYLEN-1))];
      tmp[3] = sbox[*((uint8_t *) (roundKey)+(round*KEYLEN-4))];
      roundKey[(round*4)+count] = *((uint32_t *)tmp) ^ roundKey[(round-1)*4];
    }
    else {
      tmp[0] = *((uint8_t *) (roundKey)+(round*KEYLEN+(count-1)*4));
      tmp[1] = *((uint8_t *) (roundKey)+(round*KEYLEN+(count-1)*4+1));
      tmp[2] = *((uint8_t *) (roundKey)+(round*KEYLEN+(count-1)*4+2));
      tmp[3] = *((uint8_t *) (roundKey)+(round*KEYLEN+(count-1)*4+3));
      roundKey[(round*4)+count] = *((uint32_t *)tmp) ^ roundKey[(round-1)*4 + count ];
    }
  }//for2
 }//for1
}
```

코드설명: 4word의 키를 44word로 확장합니다.

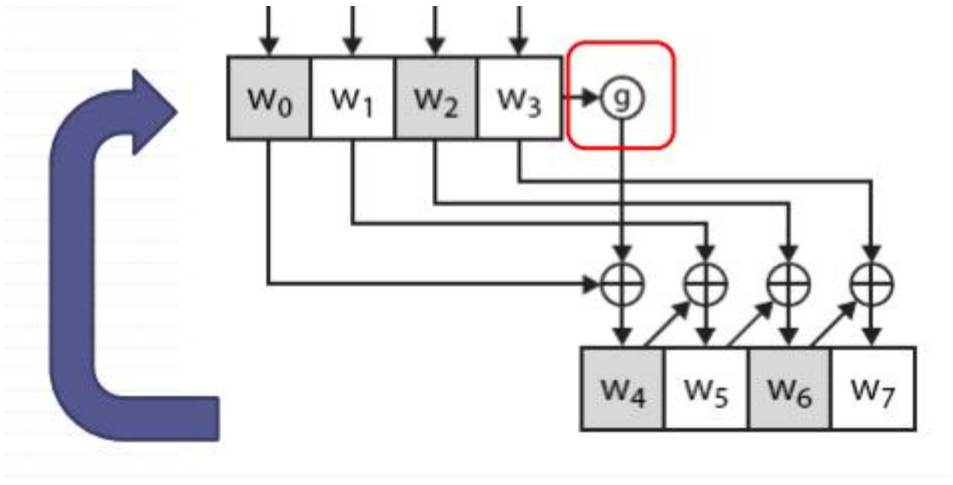
마지막 word를 다음의 알고리즘을 이용하여 다음 확장될 키의 첫 word를 생성합니다.

1) 마지막word의 바이트를 왼쪽으로 한칸씩 shift합니다.

2) sbox를 이용해 바이트를 치환합니다.

3) 라운드에 대응하는 Rcon과 xor합니다.

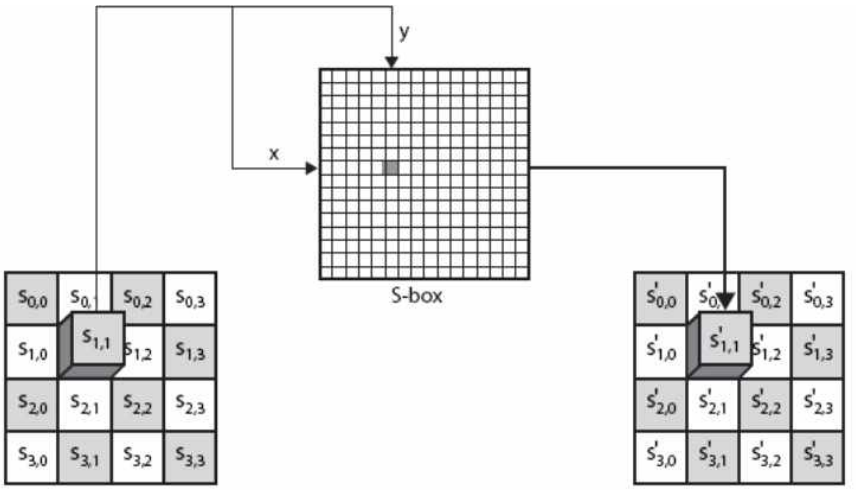
두 번째부터 이전 word와 xor하여 word를 생성합니다.



subBytes

```
void subBytes(uint8_t *block, int mode){
    int i;
    switch(mode){
        case ENCRYPT:
            for(i = 0; i < KEYLEN; i++) block[i] = sbox[block[i]];
            break;
        case DECRYPT:
            for(i= 0; i < KEYLEN; i++) block[i] = isbox[block[i]];
            break;
        default:
            fprintf(stderr, "Invalid mode!\n");
            exit(1);
    }
}
```

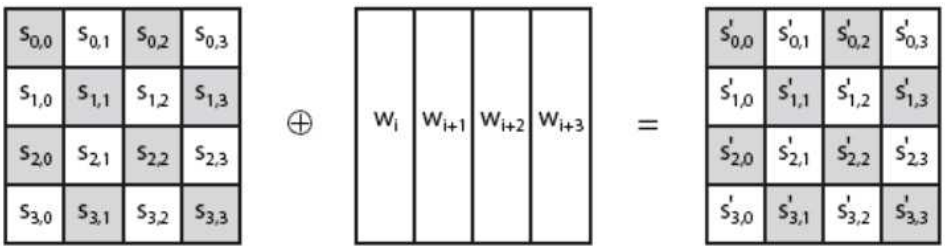
코드 설명: 4 word의 block을 sbox를 이용하여 바꾸어 줍니다.



addRoundKey

```
void addRoundKey(uint32_t *block, uint32_t *rKey){
  int i;
  for(i=0;i<(KEYLEN/4);i++){
    block[i] = block[i] ^ rKey[i];
  }
}
```

코드 설명: 4 word의 block과 들어온 라운드키를 xor합니다.

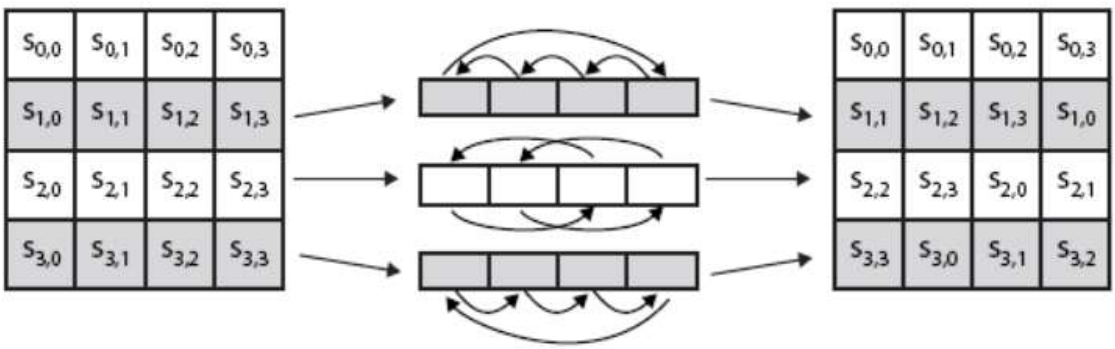


shiftRows

```
void shiftRows(uint8_t *block, int mode){
    int i;
    uint8_t tmp[KEYLEN];
    for(i = 0; i < KEYLEN; i++) tmp[i] = block[i];
    switch(mode){
        case ENCRYPT:
            for(i = 0; i < KEYLEN; i++) {
                if(i % 4 == 0) block[i] = tmp[i];
                else if(i % 4 == 1) block[i] = tmp[(i + 4) % 16];
                else if(i % 4 == 2) block[i] = tmp[(i + 8) % 16];
                else block[i] = tmp[(i + 12) % 16];
            }
            break;
        case DECRYPT:
            for(i = 0; i < KEYLEN; i++) {
                if(i % 4 == 0) block[i] = tmp[i];
                else if(i % 4 == 1) block[i] = tmp[(i + 12) % 16];
                else if(i % 4 == 2) block[i] = tmp[(i + 8) % 16];
                else block[i] = tmp[(i + 4) % 16];
            }
            break;
        default:
            fprintf(stderr, "Invalid mode!\n");
            exit(1);
    }
}
```

코드 설명: 4 word의 block을 ShiftRow합니다.

1) 각 word의 첫 번째 항목은 변경시키지 않습니다.
2) 각 word의 두 번째 항목은 왼쪽으로 1칸식 이동합니다.
3) 각 word의 세 번째 항목은 왼쪽으로 2칸식 이동합니다.
4) 각 word의 네 번째 항목은 왼쪽으로 3칸식 이동합니다.



mixColumns

```c
void mixColumns(uint8_t *block, int mode){
    int i = 0;
    uint8_t tmp[KEYLEN];
    for(i = 0; i < KEYLEN; i++) tmp[i] = block[i];
    switch(mode){
        case ENCRYPT:
            for(i = 0; i < KEYLEN; i++) {
                if(i % 4 == 0) block[i] = multipleGF(tmp[i], 2) ^ multipleGF(tmp[i+1], 3) ^ tmp[i+2] ^ tmp[i+3];
                else if(i % 4 == 1) block[i] = tmp[i-1] ^ multipleGF(tmp[i], 2) ^ multipleGF(tmp[i+1], 3) ^ tmp[i+2];
                else if(i % 4 == 2) block[i] = tmp[i-2] ^ tmp[i-1] ^ multipleGF(tmp[i], 2) ^ multipleGF(tmp[i+1], 3);
                else block[i] = multipleGF(tmp[i-3], 3) ^ tmp[i-2] ^ tmp[i-1] ^ multipleGF(tmp[i], 2);
            }
            break;
        case DECRYPT:
            for(i = 0; i < KEYLEN; i++) {
                if(i % 4 == 0) block[i] = multipleGF(tmp[i], 14) ^ multipleGF(tmp[i+1], 11) ^ multipleGF(tmp[i+2], 13) ^ multipleGF(tmp[i+3], 9);
                else if(i % 4 == 1) block[i] = multipleGF(tmp[i-1], 9) ^ multipleGF(tmp[i], 14) ^ multipleGF(tmp[i+1], 11) ^ multipleGF(tmp[i+2], 13);
                else if(i % 4 == 2) block[i] = multipleGF(tmp[i-2], 13) ^ multipleGF(tmp[i-1], 9) ^ multipleGF(tmp[i], 14) ^ multipleGF(tmp[i+1], 11);
                else block[i] = multipleGF(tmp[i-3], 11) ^ multipleGF(tmp[i-2], 13) ^ multipleGF(tmp[i-1], 9) ^ multipleGF(tmp[i], 14);
            }
            break;
        default:
            fprintf(stderr, "Invalid mode!\n");
            exit(1);
    }
}
```

코드 설명: 4 word의 block을 4x4 행렬과 GF(2^8)을 이용하여 곱셈합니다. DEC의 경우 역행렬을 이용하여 곱셈합니다.