2019007329 이지현

KeyExpansion 함수

roundkey를 생성하기 위한 함수이다.

roundkey[0], roundkey[1], roundkey[2], roundkey[3]는 그대로 사용하기 때문에 key를 그대로 사용한다. roundkey[4]~roundkey[43]는 g 함수를 거쳐야 한다.

먼저 g_rotword은 왼쪽으로 8비트씩 움직여야 된다. 제일 왼쪽에 있던 8비트는 맨 뒤로 가야 하므로 roundKey[index] << 24, roundKey[index] >> 8를 각각 계산한 후 xor 연산을 한다.

g_subword는 8비트씩 s-box를 이용해 새로운 값으로 대체한다. sub_index에는 교체할 8비트를 넣어두고 sbox를 이용해 값을 대체한 후, xor 연산으로 합쳐준다.

g_rcon은 rcon 리스트를 이용해 xor 연산해준다.

g 함수를 거친 값은 총 4개의 roundkey를 생성한다. 이 과정을 10번 반복한다.

Cipher 함수

mode 값에 따라 암호화, 복호화를 진행한다. mode == 1이면 암호화를 진행한다. 암호화는 먼저 AddRoundKey를 진행하고(1 round), 8 round까지 SubByte, ShiftRows, MixColumns, AddRoundKey를 반복한다. 이 때 AddRoundKey에 사용되는 roundkey는 전체 roundkey에서 4개씩 나누어 사용하기 위해 make_roundkey함수를 사용하였다. 맨 마지막 라운드인 10 round에서는 SubBytes, ShiftRows, AddRoundKey를 순서대로 실행한다.

복호화의 경우 암호화의 반대로 과정이 이루어지기 때문에 make_roundkey 함수를 이용해 roundkey[40]~[43]을 roundkey로 사용하여 AddRoundKey를 진행한다. 이후, SubByte, ShiftRows, MixColumns을 진행하고, 4*(10-i)부터 4*(10-i)+4까지 roundkey를 이용해 MixColumns, AddRoundKey를 실행한다. 맨 마지막 라운드인 10 round에서는 SubByte, ShiftRows, AddRoundKey를 진행한다.

AddRoundKey 함수

roundkey를 state와 xor 연산하는 함수이다. 먼저 8비트 state를 32비트로 바꾸어 준 뒤, temp 변수에 저장한다. 총 4번 xor 연산을 해 준 뒤, memcpy를 이용해 temp 값을 state 값으로 넘겨준다.

SubBytes 함수

mode == 1일 경우 sbox를 이용해 바이트를 치환해주고, mode == 0일 경우 isbox를 이용해 바이트를 치환해준다.

ShiftRows 함수

첫 row는 바뀌지 않으므로 i는 1부터 시작한다. mode == 1인 경우 둘째 row는 현재 위치 index+4, 셋째 row는 index+8, 셋째 row는 index+12이므로 state[index] = temp[(index + 4*i)%16] 이다. % 16은 값이 16이 넘어갈 경우, 16을 나눈 나머지 값으로 지정되게 하기 위해 사용하였다.

복호화의 경우 암호화와 반대로 둘째 row는 현재 위치 index+12 셋째 row는 index+8, 셋째 row는 index+4이므로 state[index] = temp[(index + 4*(4-i))%16] 이다.

MixColumns 함수

GF(28) 곱셈을 위해 gf8_mul 함수를 이용하였다. memcpy를 이용해 temp에 state를 복사해준다. 암호화의 경우,

state[index] = gf8_mul(M[m_index],temp[t_index]) ^ gf8_mul(M[m_index+1],temp[t_index+1]) ^

gf8_mul(M[m_index+2],temp[t_index+2]) ^ gf8_mul(M[m_index+3],temp[t_index+3]);

위와 같이 temp와 M를 gf8_mul 함수로 곱해준다.

복호화의 경우 M 대신 IM을 이용한다.

컴파일 & 실행 결과

```
lejinhy@DESKTOP-IPANOF9:/mnt/c/Users/82103/Downloads/proj#2$ make
gcc -Wall -O3   -c -o aes.o aes.c
gcc -o test test.o aes.o  -lbsd
lejinhy@DESKTOP-IPANOF9:/mnt/c/Users/82103/Downloads/proj#2$ ./te
<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 .....PASSED
---
AES 성능시험.....................PASSED
CPU 사용시간 = 49.6756초
```

설명&소감

AES의 암호화, 복호화 과정을 구현하였다.

과제를 하며 전체적인 AES 과정을 알 수 있었다. 이론적인 내용 외에도 비트 연산에서 많은 시간을 할애하였다. 이번 과제에서 다른 비트간의 연산을 다루며 비트 연산에 대해 더 자세히 알게 된 것 같아 뿌듯했다. 이전 과제에서 구현했던 함수를 이번 과제에 사용하며 전에 배웠던 내용을 다시 복습할 수 있었고, 어렵더라도 과제를 수행해야 다음 과제를 수행하는데 큰 어려움이 없다는 것을 느꼈다.

문제점

고쳐야 할 점은 과제에서 사용되는 round, column 수 등이 aes.h 파일에 정의되어 있었는데, 사용하지 않고 직접 int로 정의하여 사용하였다는 점이다. 이후 과제에서는 헤더파일에 정의되어 있는 변수를 사용할 예정이다. 또한 AES 과정은 제대로 진행되었지만, CPU 사용시간이 40초대로 만족스럽지 않은 결과가 나왔다. 제대로 된 정답이 나오는 것도 중요하지만, 알고리즘의 성능 또한 개선해보도록 노력할 것이다.