프로젝트 #2 (개인)

소프트웨어학부 암호학

2022년 9월 15일

문제

키의 길이가 128 비트인 AES (Advanced Encryption Standard) 알고리즘을 FIPS-197 문서에 명시된 표준스펙에 따라 구현한다.

정수형 타입

프로그램에 사용하는 모든 정수형 타입은 `uint8_t`, `int16_t`와 같이 부호의 유무와 비트의 크기를 명확하게 알 수 있도록 C언어 표준 정수형 타입을 사용한다.

전역 함수

외부에서 보이는 전역 함수를 아래 열거한 프로토타입을 사용하여 구현한다. 각 함수에 대한 요구사항은 다음과 같다.

- `void KeyExpansion(const uint8_t *key, uint32_t *roundKey)` – 길이가 16 바이트인 사용자 키에서 암복호에 사용할 라운드 키를 생성한다. `roundKey`의 길이는 44 워드이어야 한다.
- `void Cipher(uint8_t *state, const uint32_t *roundKey, int mode)` – 크기가 16 바이트인 `state`를 `roundKey`를 사용하여 암복호화한다. 이 때 `mode`가 `ENCRYPT`이면 암호화를 수행하고 `DECRYPT`이면 복호화를 수행한다.

지역 함수

내부에서만 사용하는 지역 함수는 별도로 지정하지 않고, 각자 필요에 맞게 작성한다. 다음에 열거한 함수와 프로토타입은 참고용이다.

- `static void AddRoundKey(uint8_t *state, const uint32_t *roundKey)` – 라운드 키를 XOR 연산을 사용하여 `state`에 더한다.
- `static void SubBytes(uint8_t *state, int mode)` – `mode`에 따라 순방향 또는 역방향으로 바이트를 치환한다.
- `static void ShiftRows(uint8_t *state, int mode)` – `mode`에 따라 순방향 또는 역방향으로 바이트의 위치를 변경한다.
- `static void MixColumns(uint8_t *state, int mode)` – 기약 다항식 $x^8 + x^4 + x^3 + x + 1$을 사용한 GF(2^8)에서 행렬곱셈을 수행한다. `mode`가 `DECRYPT`이면 역행렬을 곱한다.

골격 파일

구현이 필요한 골격파일 `aes.skeleton.c`와 함께 헤더파일 `aes.h`, 프로그램을 검증할 수 있는 `test.c`, 그리고 `Makefile`을 제공한다. 이 가운데 `test.c`를 제외한 나머지 파일은 용도에 맞게 자유롭게 수정할 수 있다.

테스트 벡터

알고리즘이 올바르게 구현되었다면 주어진 키와 평문에 대한 암호문과 라운드 키가 예시와 같이 일치해야 한다.

<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
```

점검 사항

1. `uint8_t` 정수와 `uint32_t` 정수가 섞여 있는 계산에서 작은 타입에서 큰 타입으로 변환하면 계산상 유리하다. 메모리 접근을 네 번 할 것을 한 번만 하면 되니까 이론적으로는 4배까지 이득을 얻을 수 있다.
2. 타입과 상관없이 메모리를 한꺼번에 복사할 수 있는 `memcpy`와 같은 함수를 잘 활용한다.
3. 빅엔디언, 리틀엔디언과 같은 내부 바이트의 배열을 인지한 상태에서 타입 변환을 시도한다. 내부 바이트 배열에 영향을 받지 않는 연산 방식을 사용하는 것이 코드의 호환성을 높인다.
4. 암호 알고리즘 구현은 스피드가 생명이다. 메모리 접근과 반복문의 횟수를 줄이기 위해 노력한다.

제출물

과제에서 요구하는 함수가 잘 설계되고 구현되었다는 것을 보여주는 자료를 보고서 형식으로 작성한 후 PDF로 변환하여 `이름_학번_PROJ2.pdf`로 제출한다. 여기에는 다음과 같은 것이 반드시 포함되어야 한다.

- 본인이 작성한 함수에 대한 설명
- 컴파일 과정을 보여주는 화면 캡처
- 실행 결과물의 주요 장면과 그에 대한 설명, 소감, 문제점
- 프로그램 소스파일 (`aes.c`, `aes.h`) 별도 제출
- 프로그램 실행 결과 (`aes.txt`) 별도 제출

평가

- Correctness 50%: 프로그램이 올바르게 동작하는 지를 보는 것입니다. 여기에는 컴파일 과정은 물론, 과제가 요구하는 기능이 문제없이 잘 작동한다는 것을 보여주어야 합니다.
- Presentation 50%: 자신의 생각과 작성한 프로그램을 다른 사람이 쉽게 이해할 수 있도록 프로그램 내에 적절한 주석을 다는 행위와 같이 자신의 결과를 잘 표현하는 것입니다. 뿐만 아니라, 프로그램의 가독성, 효율성, 확장성, 일관성, 모듈화 등도 여기에 해당합니다. 이 부분은 상당히 주관적이지만 그러면서도 중요한 부분입니다. 컴퓨터과학에서 중요하게 생각하는 best coding practices를 참조하기 바랍니다.

$\mathcal{HK}$