프로젝트 #1 (개인)

컴퓨터학부 암호학

2023년 9월 11일

문제

두 수의 최대공약수를 구하는 유클리드 알고리즘과 확장유클리드 알고리즘, 모듈로 연산에서 곱의 역, 유한체 GF(2^{16})에서 곱셈과 지수 연산을 구현한다.

함수 구현

학생들이 구현할 함수의 프로토타입은 아래에 열거되어 있다. 각 함수에 대한 요구사항을 잘 읽고, 이에 맞춰 올바르게 구현한다.

- `int gcd(int a, int b)` – 유클리드 알고리즘 $\gcd(a, b) = \gcd(b, a \bmod b)$를 사용하여 최대공약수를 계산한다. 만일 a가 0이면 b가 최대공약수가 된다. 그 반대도 마찬가지이다. a, b가 모두 음이 아닌 정수라고 가정한다. 단, 재귀함수 호출을 사용하지 말고 `while` 루프를 사용하여 구현한다.

- `int xgcd(int a, int b, int *x, int *y)` – 확장유클리드 알고리즘을 계산하는 함수이다. 두 수의 최대공약수 $\gcd(a, b) = ax + by$ 식을 만족하는 x와 y를 계산한다. a, b가 모두 음이 아닌 정수라고 가정한다.

- `int mul_inv(int a, int m)` – 모듈로 m에서 a의 곱의 역인 $a^{-1} \bmod m$을 구한다. 만일 역이 존재하지 않는다면 0을 리턴해야 한다. 확장유클리드 알고리즘을 약간 변형하면 쉽게 구할 수 있다.

- `uint64_t umul_inv(uint64_t a, uint64_t m)` – 모듈로 m에서 a의 곱의 역인 $a^{-1} \bmod m$을 구한다. 만일 역이 존재하지 않는다면 0을 리턴해야 한다. 다만 입력과 출력이 모두 unsigned 64 비트로 어떤 특별한 목적을 위해 설계된 함수이다. 확장유클리드 알고리즘을 변형하면 구할 수 있다. 그러나 계산에 사용되는 수가 unsigned 64 비트임에 유의해야 한다.

- `uint16_t gf16_mul(uint16_t a, uint16_t b)` – GF(2^{16}) 상에서 두 수를 곱하는 것으로, $a * b \bmod x^{16} + x^5 + x^3 + x + 1$을 계산하는 함수이다. 여기서 a, b는 GF(2^{16}) 상의 수인 15차 다항식으로 a와 b를 곱하고, 그 결과를 16차 기약다항식 $x^{16} + x^5 + x^3 + x + 1$로 나눈 나머지를 의미한다. $x^{16} \equiv x^5 + x^3 + x + 1 \pmod{x^{16} + x^5 + x^3 + x + 1}$ 특성을 이용한다.

- `uint16_t gf16_pow(uint16_t a, uint16_t b)` – GF(2^{16}) 상에서 거듭제곱을 구하는 것으로, $a^b \bmod x^{16}+x^5+x^3+x+1$을 계산하는 함수이다. 여기서 a, b는 GF(2^{16}) 상의 수인 15차 다항식으로 a^b의 결과를 16차 기약다항식 $x^{16} + x^5 + x^3 + x + 1$로 나눈 나머지를 의미한다. 단, `gf16_mul()`과 square multiplication 알고리즘을 사용하여 구현한다. b가 정수일 때 a^b를 계산하는 일반적인 square multiplication 알고리즘은 다음과 같다.

```
r = 1;
while (b > 0) {
    if (b & 1)
        r = r * a;
    b = b >> 1;
    a = a * a;
}
return r;
```

골격 파일

구현에 필요한 소스파일과 메이크파일인 `Makefile`, `euclid.h`, `euclid.skeleton.c`, `test.c` 파일을 제공한다. 골격파일인 `euclid.skeleton.c`에는 구현이 필요한 함수가 빈 상태로 들어 있다. `test.c`는 빈 함수가 완성된 후에 올바르게 작동하는지 검증하기 위한 것으로 그 내용을 수정해서는 안 된다.

arc4random 함수

`arc4random()`, `arc4random_buf()`, `arc4random_uniform()`, `arc4random_addrandom()` 등의 함수는 암호학적으로 안전한 의사난수를 생성하기 위해 개발되었다. 기존 라이브러리에 있는 `rand()` 함수는 안전을 고려하지 않고 설계되었기 때문에 보안과 관련된 프로그램에서는 사용하지 않는다.

- **Linux** 환경에서 이 함수들을 사용하려면 `bsd` 라이브러리가 설치되어 있어야 한다. 설치법은 다음과 같다.

  ```
  % sudo apt update
  % sudo apt install libbsd-dev
  ```

  설치후 사용하려면 `#include <bsd/stdlib.h>`를 하고, `gcc` 링크시 `-lbsd` 옵션을 사용해야 한다.

  ```
  % gcc -o sample sample.c -lbsd
  ```

- **macOS** 환경에서는 표준라이브러리에 내장되어 있어서 `#include <stdlib.h>` 이외에 별도의 옵션이 필요없다.

리눅스 사용자는 터미널에서 bsd 라이브러리를 먼저 설치하고 `make` 명령어를 통해 실행파일을 생성한다. macOS 사용자는 별도의 설치 없이 바로 실행파일을 생성할 수 있다. 만들기 전에 `euclid.skeleton.c`를 `euclid.c`로 이름을 바꿔야 컴파일러가 불평하지 않는다.

제출물

과제에서 요구하는 함수가 잘 설계되고 구현되었다는 것을 보여주는 자료를 보고서 형식으로 작성한 후 PDF로 변환하여 `이름_학번_PROJ1.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 (`euclid.c`) 별도 제출
- 프로그램 실행 결과 (`euclid.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.

HK