REPORT

암호학 프로젝트1 (GF(2^16))

과목명	암호학
담당교수	오희국 교수님
학생이름	지현도
학과	컴퓨터학부
학번	2021004866
제출일	2023/09/19

HANYANG UNIVERSITY

*본인이작성한함수에대한설명:

1. swap()

```
16  #define swap(a,b,type) do{type tmp=a; a=b; b=tmp;}while(0);
```

코드 간소화를 위해 추후에 쓸 함수에서 사용할 swap함수를 구현해주었다.

2. int gcd(int a, int b)

```
26  int gcd(int a, int b) {
27      while (b != 0) {
28          int temp = a;
29          a = b;
30          b = temp % b;
31      }
32      return a;
```

A와 b의 최대공약수를 구하는 함수이다. B가 0이 되기전까지 While 반복문으로 반복한다.

3. int xgcd(int a, int b, int *x, int *y)

```
42  int xgcd(int a, int b, int *x, int *y)
43  {
44      int d0 = a, d1 = b, q, x0=1, x1=0, y0=0, y1=1;
45
46      while(d1){
47          q = d0/d1;
48          d0 = d0 - q*d1; swap(d0,d1,int); // (d0,d1) = (d1,d0%d1)
49          x0 = x0 - q*x1; swap(x0,x1,int); // xi+1 = xi-1 - q*xi
50          y0 = y0 - q*y1; swap(y0,y1,int); // yi+1 = yi-1 - q*yi
51      }
52
53      *x = x0, *y = y0;
54      return d0; // a와 b의 최대공약수
55  }
```

확장유클리드 알고리즘 함수이다. ax+by = gcd(a,b)인 x,y를 찾는 함수이다. d1=0 일때가 d0이 최대공약수이고, x0, y0가 x,y 가 된다.

d0 = bq0 + r0 (r0 = d0%d1)

d1 = r0q1 + r1

r0 = r1q2 +r2 .....

ri+1 = ri-1 – riqi

4. int mul_inv(int a, int m)

```
int mul_inv(int a, int m)
{
    int x, y;
    if(xgcd(a,m,&x,&y) != 1) return 0; // 서로소가 아니면 역을 구할 수 없음
    return x<0 ? x+m : x;
}
```

Modulo m 에서 a의 역을 구한다.

a*a^-1 = mq +1

a*a^-1 – mq =1

결국 gcd(a,m) 이 1인 경우가 x 가 a의 역원이 된다고 할 수 있다. 기본적으론 과제 설명에는 x가 음수인 경우를 가정하진 않았지만 혹시 음수라면 양수로 변환해준다.

5. uint64_t umul_inv(uint64_t a, uint64_t m)

```
78  uint64_t umul_inv(uint64_t a, uint64_t m)
79  {
80      uint64_t d0 = a, d1 = m, q;
81      long long x0 = 1, x1 = 0; // 음수를 고려하여 Long long 으로 지정
82
83      while(d1 > 1){
84          q = d0/d1;
85          d0 = d0 - q*d1; swap(d0,d1,uint64_t); // (d1,d0%d1)
86          x0 = x0 - (long long)q*x1; swap(x0,x1,long long); // (x1, x0-q*x1)
87      }
88
89      if(d1 == 1) return x1>0 ? (uint64_t)x1 : m-(uint64_t)(-x1); // 만약 음수라면 양수로
            변환해준다
90      else return 0;
91  }
```

우선 숫자가 uint64_t 즉 Unsigned 64bit 이므로 음수를 표현하는 과정에서 큰 변수명이 필요하다. 따라서 long long으로 캐스팅 하였다. 단 반환시에는 다시 Unsigned 64bit 으로 변

환해주는데 이때 만약 음수라면 역원을 양수로 바꾸고 m에서 빼주었다.

6. uint16_t gf16_xtime(uint16_t x)

```
93  uint16_t gf16_xtime(uint16_t x) {
94      return (x << 1) ^ ((x >> 15) & 1 ? 0x2B : 0);  // 0x2B == x^5+x^3+x+1
95  }
```

과제의 skeleton 코드에는 선언되지 않아있던 함수인데, 강의노트에 있던 Gf8_xtime 을 참고하여 15차인것을 감안하여 1칸과 15칸 shift연산으로 바꾸고 x^5+x^3+x+1는 16진수로 0x2B이므로 그 부분또한 수정하여 선언해주었다.

7. uint16_t gf16_mul(uint16_t a, uint16_t b)

```
103 uint16_t gf16_mul(uint16_t a, uint16_t b)
104 {
105     uint16_t r = 0;
106
107     while(b>0){
108         if(b&1) r=r^a;
109         b = b >> 1;
110         a = gf16_xtime(a);
111     }
112
113     return r;
114 }
```

Modulo 가 x^16+x^5+x^3+x+1 인 전제에서 15차 다항식인 a와 b의 곱셈 결과를 구한다.

다항식끼리의 덧셈은 XOR이므로 위에 구현한 xtime()를 통하

여 구현해주었다.

8. uint16_t gf16_pow(uint16_t a, uint16_t b)

```
122 uint16_t gf16_pow(uint16_t a, uint16_t b)
123 {
124     uint16_t ans = 1;
125     while(b>0){
126         if(b&1) ans = gf16_mul(ans,a); // b가 홀수라면 a를 곱해준다.
127         a = gf16_mul(a,a); // a == a^2으로 만들어준다.
128         b >>= 1; // shift 연산으로 2를 나눠준다.
129     }
130     return ans;
131 }
```

a^bmod(x^16+x^5+x^3+x+1) 을 구하는 함수이다.

분할정복 알고리즘을 활용하였다. 곱셈과정은 위에 구현한 gf16_mul()함수를 재사용하였다.

9. uint16_t gf16_inv(uint16_t a)

```
140 uint16_t gf16_inv(uint16_t a)
141 {
142     return gf16_pow(a, 0xfffe);
143 }
144
```

검증함수

*컴파일 과정

```
jihyeondo@192 proj#1-1 % make
gcc -Wall -O3 -c euclid.c
gcc -o test test.o euclid.o
jihyeondo@192 proj#1-1 %
```

오류없이 정상적으로 컴파일되는 모습이다.

*실행 결과물

```
jihyeondo@192 proj#1-1 % ./test
========== 기본 gcd 시험 ==========
gcd(28,0) = 28
gcd(0,32) = 32
gcd(41370,22386) = 42
gcd(22386,41371) = 1
========== 기본 xgcd, mul_inv 시험 ==========
42 = 41370 * -204 + 22386 * 377
41370^-1 mod 22386 = 0, 22386^-1 mod 41370 = 0
1 = 41371 * 4285 + 22386 * -7919
41371^-1 mod 22386 = 4285, 22386^-1 mod 41371 = 33452
========== 무작위 xgcd, mul_inv 시험 ==========
..........................................................................
..........................................................................
..........................................................................
.........................PASSED
========== 기본 GF(2^16) a*b 시험 ==========
3 * 7 = 9
65535 * 12345 = 41504
========== 전체 GF(2^16) a*b 시험 ==========
..........................................................................
..........................................................................
..........................................................................
.........................PASSED
========== 기본 umul_inv 시험 ==========
a = 5, m = 9223372036854775808, a^-1 mod m = 5534023222112865485.....PASSED
a = 17, m = 9223372036854775808, a^-1 mod m = 8138269444283625713.....PASSED
a = 85, m = 9223372036854775808, a^-1 mod m = 9006351518340545789.....PASSED
========== 무작위 umul_inv 시험 ==========
..........................................................................
..........................................................................
..........................................................................
.........................PASSED
jihyeondo@192 proj#1-1 %
```

실행 결과과 예상 출력과 모두 일치하고 오류없이 모두 통과되는 모습을 볼 수 있다. 아마 모든 함수가 정상적으로 작동하는 것 같다.

*프로젝트 소감 및 부족한 점

확장유클리드를 구현하는 것이 처음이라 강의노트를 참고하였음에도 불구하고 어려움이 많았습니다. 특히 계산식을 이해하는 데에 시간이 많이 소요되었고 unsinged64비트의 음수표현에 대해 고민을 많이 한 것 같습니다. 여기선 계산값의 최대값을 간략하게 예측해서 long long 으로 캐스팅 해보았는데, 이것이 맞는 접근법인지도 피드백이 필요할 것 같습니다.

또한 비트를 이용한 계산법의 속도와 편리함을 알 수 있는 시간이였습니다. 여러 문제들에 대해서도 적용 가능한지 생각해보는 계기가 되었습니다.