有限体と楕円曲線
Week 0 演習課題

WU Yihsuan

1　演習 1: mod 演算と有限体

1.1　問題 1: 減算と除算の計算方法

減算: 有限体 $\mathbb{F}_p$ において、減算は加法逆元を用いて定義される。$-b$ は b の加法逆元であり、$-b \equiv p - b \pmod{p}$ となる。

除算: 除算は乗法逆元を用いて定義される。b^{-1} は b の乗法逆元で、$b \times b^{-1} \equiv 1 \pmod{p}$ を満たす。乗法逆元は拡張ユークリッド法で計算できる。

1.2　問題 2: 拡張ユークリッド法の実装

説明：

拡張ユークリッド法は、2 つの整数の最大公約数 (gcd) を求めながら、ax + by = gcd(a, b) を満たす整数 x, y を見つける方法である。

```python
def extended_euclidean(a, b):
    #拡張ユークリッドの互除法：
    #ax + by = gcd(a, b) を満たす (gcd, x, y) を返す

    r0, r1 = a, b
    s0, s1 = 1, 0
    t0, t1 = 0, 1

    while r1 != 0:
        q = r0 // r1
        r0, r1 = r1, r0 - q * r1
        s0, s1 = s1, s0 - q * s1
        t0, t1 = t1, t0 - q * t1

    return r0, s0, t0  # gcd, x, y

```

```
# test
a = 5
b = 11
gcd, x, y = extended_euclidean(a, b)
print(f"gcd({a}, {b}) = {gcd}")
print(f"x = {x}, y = {y}")
print(f"{a} * {x} + {b} * {y} = {a*x + b*y}")
```

1.3 問題 3: $3 - 2 \pmod 5$ の計算

$$\begin{aligned} 3 - 2 &\equiv 3 + (-2) \pmod 5 \\ &\equiv 3 + (5 - 2) \pmod 5 \\ &\equiv 3 + 3 \pmod 5 \\ &\equiv 6 \pmod 5 \\ &\equiv 1 \pmod 5 \end{aligned}$$

1.4 問題 4: $3 \div 2 \pmod 5$ の計算

まず 2 の乗法逆元を求める：$2 \times 3 = 6 \equiv 1 \pmod 5$ なので $2^{-1} \equiv 3 \pmod 5$

$$\begin{aligned} 3 \div 2 &\equiv 3 \times 2^{-1} \pmod 5 \\ &\equiv 3 \times 3 \pmod 5 \\ &\equiv 9 \pmod 5 \\ &\equiv 4 \pmod 5 \end{aligned}$$

1.5 問題 5: $(7 \times 5)^{-1} \bmod 13$ の計算

$(7 \times 5)^{-1} \equiv 7^{-1} \times 5^{-1} \bmod 13$ なので、
まず、7 と 5 の乗法逆元を計算：
$7 \times 2 = 14 \equiv 1 \pmod{13}$ なので $7^{-1} \equiv 2 \pmod{13}$
また、$5 \times 8 = 40 \equiv 1 \pmod{13}$ なので $5^{-1} \equiv 8 \pmod{13}$

$$\begin{aligned} (7 \times 5)^{-1} &\equiv 7^{-1} \times 5^{-1} \bmod 13 \\ &\equiv 2 \times 8 \pmod{13} \\ &\equiv 16 \pmod{13} \\ &\equiv 3 \pmod{13} \end{aligned}$$

2　演習 2: 拡大体

2.1　問題 1: $\mathbb{F}_2$ で $10110101 - 0110110$

$\mathbb{F}_2$ では標数が 2 なので、$-a = a$ が成り立つ。つまり減算は加算と同じであり、ビット演算では XOR 演算ができる。

$$\begin{array}{r} 10110101 \\ \oplus \quad 00110110 \\ \hline 10000011 \end{array}$$

Ans: 10000011

2.2　問題 2: $\mathbb{F}_{2^8}$ で $10110101 - 0110110$

$\mathbb{F}_{2^8}$ は $\mathbb{F}_2$ の拡大体であり、標数は同じ 2 である。

$$\begin{array}{r} 10110101 \\ \oplus \quad 00110110 \\ \hline 10000011 \end{array}$$

Ans: 10000011

2.3　問題 3: $\mathbb{F}_{2^3}$ における $x^3 + x + 1$ 以外の既約多項式

モニック 3 次多項式の形：$x^3 + ax^2 + bx + c$（$a, b, c \in \mathbb{F}_2 = \{0, 1\}$）候補の多項式は 8 つある。

$$\begin{aligned}
x^3 &\Rightarrow f(0) = 0 \\
x^3 + x^2 &\Rightarrow f(0) = 0 \\
x^3 + x^2 + x &\Rightarrow f(0) = 0 \\
x^3 + x^2 + x + 1 &\Rightarrow f(1) = 4 \equiv 0 \pmod 2 \\
x^3 + x^2 + 1 &\Rightarrow f(1) = f(0) \equiv 1 \pmod 2 \\
x^3 + x + 1 &\Rightarrow f(1) = f(0) \equiv 1 \pmod 2 \\
x^3 + x &\Rightarrow f(0) = 0 \\
x^3 + 1 &\Rightarrow f(1) = 2 \equiv 0 \pmod 2
\end{aligned}$$

この中に、$x^3 + x^2 + 1$ も既約多項式である。

3　演習 3: 離散対数問題

1．素因数分解問題

大きな合成数 $N = pq$（p, q は大きな素数）を素因数に分解することの困難性。

安全性の理由:

汎用のコンピュータでは効率的な素因数分解アルゴリズムが存在しない。適切なサイズ（2048 ビット以上）では現実的な時間での分解は困難である。

2．格子基底問題（e.g.LWE 問題）

LWE 問題では、誤差付きの線形方程式系から元の秘密ベクトルを推定するのが困難であることを仮定する。

安全性の理由:

1. 現存の最も良い LWE 問題を解くアルゴリズムの実行時間が指数時間である。
2. LWE 問題は LPN 問題の一般化であり、LPN 問題は解くのが困難な問題と予想されている。そのため、LWE 問題が簡単に解けるならば、LPN 問題も簡単に解けるようになるため、安全性が保証されている。

参考: https://tex2e.github.io/blog/crypto/lwe-key-exchange

3.1　問題 2: 離散対数問題を効率的に解くアルゴリズム

- G を位数 n の有限巡回群とする
- 要素（h ∈G）
- 離散対数問題: $G^x = h$ を満たす整数 x(0≤x < n) を求める

Baby-step Giant-step 法

- 計算量：$O(\sqrt{n})$ (←まだよく理解していない)
- 全探索 $O(n)$ より高速

アルゴリズム: $g^x = h \pmod p$ を解く場合：

1. $m = \lceil \sqrt{n} \rceil$ とする (m: 分割サイズ)
2. Baby steps: $\{(j, g^j) : j = 0, 1, \ldots, m-1\}$ を計算・保存
3. Giant steps: $i = 0, 1, \ldots, m-1$ について $h \cdot (g^{-m})^i$ を計算
4. 一致する値を見つけたら $x = im + j$

効率化できる理由:

n 個の要素を m ×m の 2 次元グリッドに分割し、行と列を別々に探索することで、$O(n)$

$\rightarrow O(\sqrt{n})$ に計算量を削減できる。

参考: https://zenn.dev/t_shunsuke/articles/note-baby-step-giant-step

3.2 問題 3: $5^x \equiv 8 \pmod{23}$ の離散対数 x

Baby-step Giant-step 法を適用する。

パラメータ設定:

探索範囲は $x \in \{0, 1, 2, \ldots, 22\}$ なので：

$$n = 23$$
$$m = \lceil \sqrt{n} \rceil = \lceil \sqrt{23} \rceil = 5$$

Baby Steps:

$\{(j, 5^j) : j = 0, 1, 2, 3, 4\}$

$$5^0 = 1$$
$$5^1 = 5$$
$$5^2 = 25 \equiv 2 \pmod{23}$$
$$5^3 = 5 \times 2 = 10 \pmod{23}$$
$$5^4 = 5 \times 10 = 50 \equiv 4 \pmod{23}$$

Giant Steps: $i = 0, 1, 2, \ldots$ について $8 \cdot (5^{-5})^i = 8 \cdot 15^i \bmod 23$ を計算：
($5^{-5} \equiv 15 \pmod{23}$ のため)

$i = 0$：

$$8 \times 15^0 = 8 \times 1 = 8$$

$i = 1$：

$$8 \times 15^1 = 120 \equiv 5 \pmod{23}$$

$i = 2$：

$$15^2 = 225 \equiv 18 \pmod{23}$$
$$8 \times 15^2 = 8 \times 18 = 144 \equiv 6 \pmod{23}$$

Giant step の $i = 1$ で値 5 は、Baby step の $j = 1$ の値 5 と一致する。

解の計算:

$$x = im + j = 1 \times 5 + 1 = 6$$

Ans: $x = 6$

4　演習 4: 楕円曲線

4.1　問題 1: ECDLP を効率的に計算するアルゴリズム

Pollard's rho 法 (まだよく理解していない)

効率性:

計算量：$O(\sqrt{n})$

アルゴリズムの概要:

楕円曲線上の点 P, Q について $Q = kP$ の k を求める場合、ランダムウォークを用いて衝突を見つける手法。

参考：

https：//zenn.dev/anko/articles/ctf − crypto − elliptic

4.2　問題 2: スカラー倍算を効率的に計算するアルゴリズム

楕円曲線上の点 P に対して、kP を効率的に計算する。

Binary 法 (まだよく理解していない)

効率性:

- 時間計算量：$O(\log k)$ 回の点加算
- 一般な方法との比較: P+P+ ⋯ +P + P (K-1 回加算すると) →$O(k)$

アルゴリズムの概要:

k を二進表現し、各ビットに対応する計算を行う。

kP を計算する場合

1. k を二進展開：$k = \sum_{i=0}^{n-1} b_i 2^i$
2. $Q = O$（無限遠点）で初期化
3. $i = n-1$ から 0 まで：
   - $Q = 2Q$（倍算）
   - $b_i = 1$ なら $Q = Q + P$（加算）

4.3　問題 3: Circom のデフォルト楕円曲線

Circom ライブラリは **BN254** 楕円曲線をデフォルトで使用している。

参考: https://github.com/trailofbits/circomspect/blob/main/doc/analysis_passes.md