山东大学
SHANDONG UNIVERSITY

SHANDONG UNIVERSITY

密码学引论作业: 离散对数

网络空间安全学院 (研究院)
2021 级网安三班
谢子洋 202100460116

2023 年 4 月 24 日

1　离散对数

1.1　Pohlig-Hellman 算法

1.1.1　离散对数问题

已知 n 阶循环群 $G =< \alpha >$, 生成元 α 阶为 n.

求出 $x \in Z_n$ 满足

$$\alpha^x = \beta \in G$$

1.1.2　算法

1. 因子分解 $n = \prod_{i=1}^{k} p_i^{c_i}$.

2. 逐个计算 $y_i = x\ mod\ p_i^{c_i}\ \ (1 \le i \le k)$, 对任一 y_i 计算过程如下:

   2.1. 令 $\beta_0 = \beta,\ \ j = 0$.

   2.2. 遍历 $s\ (0 \le s \le p_i - 1)$, 直到找到 s 满足 $(\alpha^{\frac{n}{q}})^s = \beta_j^{\frac{n}{q^{j+1}}}$, 则令 $a_j = s$.

   2.3. 令 $\beta_{j+1} = \beta_j \alpha^{-a_j q^j}$.

   2.3. 若 $j = c_i$, 则可计算 $y_i = \sum_{e=0}^{c_i - 1} a_e q^e$. 否则令 j=j+1, 并返回步骤 2.2.

3. 中国剩余定理计算同余方程组 $x \equiv y_i\ (mod\ p_i^{c_i})\ \ (1 \le i \le k)$, 得到最终解 x.

1.2　Pohlig-Hellman 算法实现

Pohlig-Hellman 算法解离散对数问题要求 G 为有限 n 阶循环群.

当 p 为素数时,$G = Z_p^*$ 为 n 阶乘法循环群, 且群的阶 n=p-1.

在该循环群内, 乘法运算为 $a \cdot b\ mod\ p$.

本文针对在 p-1 阶循环群 Z_p^* 上的情况进行了实现, 群上运算均为乘法模运算.

使用 Python 实现, 代码如下:

```
import gmpy2
import libnum
from functools import reduce
#中国剩余定理解同余方程组
def CRT(aList,mList,mMul=-1):
    #简单筛选特殊情况
    if(len(aList)!=len(mList) or len(aList)==0 or len(mList)==0):return -1
    size=len(mList)
    if size==1:return aList[0]#当方程组仅有一个方程时

    #计算多个向量
    MList=[reduce(lambda x,y:x*y, mList[0:i]+mList[i+1:]) for i in range(size)]
    MinvertList=[gmpy2.invert(MList[i],mList[i]) for i in range(size)]
    ToAddList=[(MList[i]*MinvertList[i]*aList[i])%mMul for i in range(size)]

    #累加得到CRT最终结果
    if(mMul==-1):
        mMul=reduce(lambda x,y:x*y,mList)
    result=reduce(lambda x,y:(x+y)%mMul,ToAddList)
    return result

#计算x mod p^c
def calModExpon(q,c,alpha,beta,n):
    aList=[]
    betaList=[beta]
    N=n+1
    for j in range(c):
        #计算第j次项的系数
        temp1=gmpy2.powmod(betaList[j], (n//(q**(j+1))), N)
        temp2=gmpy2.powmod(alpha,       (n//q),           N)
        i=0
        temp3=1
        while(temp1!=temp3):
            temp3=gmpy2.mod((temp3*temp2),N)
            i+=1
            if(i==q):
                print("error")
        aList.append(i)
        betaList.append((betaList[-1]*gmpy2.invert(alpha,N)**(aList[-1]*q**j))%N)
    return sum([aList[i]*q**i for i in range(c)])

#主函数
def Log_PohligHellman(alpha,beta,p):
    #1. 进行因子分解
    n=p-1
    factorList=libnum.factorize(n)
    print("Facators: \n",end="")
    print(factorList)

    #2.1 计算模幂的值
    aList=[calModExpon(q,factorList[q],alpha,beta,n) for q in factorList]
    #2.2 计算各方程的模数
    mList=[key**factorList[key] for key in factorList]
    print("Equations:\n",end="")
    for i in range(len(aList)):
        print(f"x={aList[i]} mod {mList[i]}")
```

```
    #3. CRT解同余方程组
    result=CRT(aList,mList,n)
    print(f"So x={result}")
    return result
Log_PohligHellman(6,29,41)
Log_PohligHellman(2,29,37)
Log_PohligHellman(2,18,29)
```

1.3 Pohlig-Hellman 算法求解

根据上文提出的算法给定测试数据进行求解, 得到结果如下:

表 1: 离散对数结果

p	a	b	log_a^b
41	6	29	7
37	2	29	21

原始输出如下:

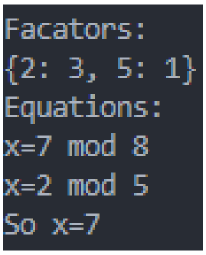

图 1: 例 1 输出

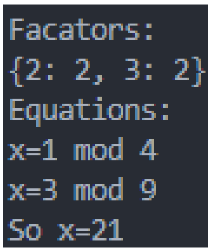

图 2: 例 2 输出

参考文献

[1] Dougals R.Stinson. 密码学原理与实践: 第三版. 北京: 电子工业出版社,2009.7.