密码学实验六实验报告

16337183 孟衍璋

1 实验目的

利用合适的算法求解附件中给定参数的模p剩余类域中离散对数挑战问题，并获取DH密钥交换协议中的共同信息。

2 实验内容

设p是一个素数，g是模p剩余类群中的非零元素。已知模数p，g和y，求解x满足同余方程$y \equiv g^x (mod\ p)$的计算数论问题，称为模p剩余类域中离散对数求解问题。DH（Diffie-Hellman）密钥交换协议是 20 世纪 70 年代由 Whitfield Diffie 和 Martin Hellman 共同提出的，在网络安全中有着广泛的应用，其目的是让参与两方能够共享一个秘密信息（密钥）。DH 密钥交换协议的过程如下图：

A 发起方	B 应答方
产生 x_a并计算$y_a \equiv g^{x_a} mod\ p$	产生x_b并计算$y_b \equiv g^{x_b} mod\ p$
A 使用公共信道把 y_a发送给 B	B 使用公共信道把y_b发送给 A
接收到y_b并计算${y_b}^{x_a}(mod\ p)$	接收到y_a并计算${y_a}^{x_b}(mod\ p)$
此时，A 有信息 ${y_b}^{x_a} \equiv g^{x_b x_a}\ (mod\ p)$，B 有信息${y_a}^{x_b} \equiv g^{x_a x_b}\ (mod\ p)$，而 $g^{x_a x_b} \equiv g^{x_b x_a}\ (mod\ p)$。 这样，A 和 B 就得到了共同的信息。	

DH 协议的安全性基于素域（模素数p剩余类域）上离散对数求解的困难性。如果第三方可以在公共信道截获y_a和y_b，通过求取离散对数x_a，就可求出 A 和 B 的共同信息

$$ {y_b}^{x_a} \equiv g^{x_b x_a}\ (mod\ p) $$

给出已知信息：

$p = 568254902274842463133913191337012578621250922758849353787467$
$317363493600872590435893544210146655556112445578284746895502$
$852903766053355394139940833133140337 9$

$g = 241049705597043288134549339784611219899508877136430719518973$
$403120560518695124187509679645906174178166738043707687470530$
$0974836586165283741668656930807264789$

$$\begin{aligned} y_a = &\ 9737682843413262723015537511143226853243408059020696133396 \\ &\ 6714218780152958535240697503092700875257191707971631822107 \\ &\ 775823688434266282940252973400960753 1649 \\ y_b = &\ 4149822765985031146554298777122732051870868431387323913747 \\ &\ 7857916853105088368362837029264468170001148680075557175463 \\ &\ 6242584186517392967015656868274506070831 4 \end{aligned}$$

3 实验及算法原理

使用$Pollard's\,\rho\,algorithm$，解决此离散对数问题。假设$(G,\cdot)$是一个群，$g \in G$是一个n阶元素，我们要计算元素$y_a \in \langle g \rangle$的离散对数。由于$\langle g \rangle$是n阶循环群，可以把$\log_g y_a$看做$\mathbb{Z}_n$中的元素。

与因子分解的ρ算法一样，通过迭代一个貌似随机的函数f，构造一个序列$x_1, x_2, x_3, \cdots$。一旦在序列中得到两个元素x_i和x_j，满足$x_i = x_j$，这里$i < j$，我们就有希望计算出$\log_g y_a$。

设$S_1 \cup S_2 \cup S_3$是群G的一个划分，它们的元素个数大致相同。定义函数$f : \langle g \rangle \times \mathbb{Z}_n \times \mathbb{Z}_n \to \langle g \rangle \times \mathbb{Z}_n \times \mathbb{Z}_n$如下：

$$\begin{cases} (y_a x, a, b+1) & x \in S_1 \\ (x^2, 2a, 2b) & x \in S_2 \\ (gx, a+1, b) & x \in S_3 \end{cases}$$

在这里，将划分S_1, S_2, S_3定义为如下形式：

$$S_1 = \{x \in \mathbb{Z}_p : x \equiv 1 (mod\, 3)\}$$

$$S_2 = \{x \in \mathbb{Z}_p : x \equiv 0 (mod\, 3)\}$$

$$S_3 = \{x \in \mathbb{Z}_p : x \equiv 2 (mod\, 3)\}$$

程序执行的算法如下：

算法 6.2 离散对数算法 Pollard ρ (G, n, α, β)

Procedure $f(x, a, b)$

if $x \in S_1$

 then $f \leftarrow (\beta \cdot x, a, (b+1) \bmod n)$

 else if $x \in S_2$

 then $f \leftarrow (x^2, 2a \bmod n, 2b \bmod n)$

 else $f \leftarrow (\alpha \cdot x, (a+1) \bmod n, b)$

return(f)

main

定义划分 $G = S_1 \cup S_2 \cup S_3$

$(x, a, b) \leftarrow f(1, 0, 0)$

$(x', a', b') \leftarrow f(x, a, b)$

while $x \neq x'$

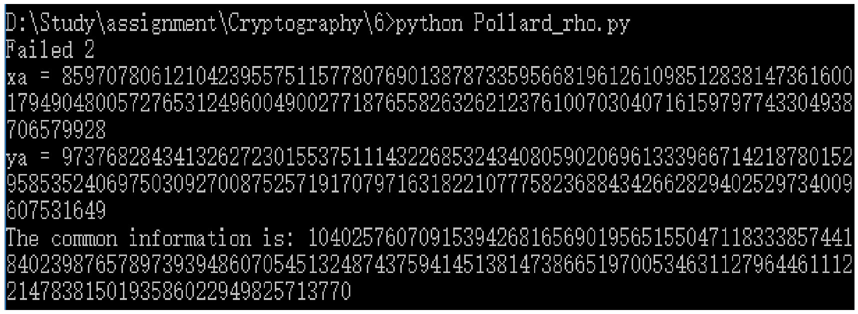
$$\textbf{do}\begin{cases}(x, a, b) \leftarrow f(x, a, b)\\(x', a', b') \leftarrow f(x', a', b')\\(x', a', b') \leftarrow f(x', a', b')\end{cases}$$

if $\gcd(b' - b, n) \neq 1$

 then return(“**failure**”)

 else return$((a - a')(b' - b)^{-1} \bmod n)$

4 运行截图



5 结果分析

经验证，计算出的第一个x_a，用它来计算y_a，得到的结果与之前的y_a相同，这便就证明了，这个x_a的值是正确的。然后再计算共同信息，用式子$y_b^{x_a} mod\, p$得到。

6 总结

本次实验要求求解DLP问题，这本身是一个困难问题，但既然布置成为作业，说明根据题目给出的数据可以找到窍门并攻破它。我在实验中选择了$Pollard's\, \rho\, algorithm$。首先可以验证，$p-1$是元素g的阶。这一点可以根据式子$g^{p-1} mod\, p = 1$得到证明。在找到碰撞之后，发现根据书上的算法程序会得到结果“failure”，这时根据书上的一句话可以找到继续完成下去的线索：“如果$gcd(b'-b, n) = d$，容易证明同余方程$c(b'-b) \equiv a-a'(modn)$，恰有d个解。假如d 不是很大的话，可以直接算出同余方程的d个解并检验哪个解是正确的。”于是输出$gcd(b'-b, n)$发现其值等于2，也就是说一次同余方程$c(b'-b) \equiv a-a'(mod\, n)$有两个解。根据求解一次同余式的方法便可以计算出这些解，最后再验证哪一个是正确的。

这里在求解一次同余方程$c(b'-b) \equiv a-a'(mod\, n)$ 时候遇到一个比较坑的地方，程序中需要用$b'-b$除以$gcd(b'-b, n)$，而$b'-b$在本题目中是一个特别大的数，因此得到的结果便自动显示为了科学计数法，这一转换不要紧，可给后面的计算增加了不小的麻烦，明明感觉算法思路都没有问题，

可是答案总是计算不正确。

后来便思考着用另外的方法求解一次同余方程，再写了一种方法。将$c(b'-b) \equiv a-a' (mod\, n)$式中每一项都除以$gcd(b'-b,n)$之后，就可以用求逆的方法得到结果。这里本该有逆，但得到的结果却显示求逆是求不出来的，花了好大功夫才发现，我计算出的$b'-b$是负数，才导致了程序算不出逆，于是将它加上模数，直到加为正数之后，这才让求逆的运算正常运作。

后来经过同学的提醒才发觉了$b'-b$除以$gcd(b'-b,n)$这里出现的问题。用“/”符号会得到科学计数法，用整除符号“//”才可以得到正确的结果。于是改了这个地方之后，答案终于正确了。