云南大学数学与统计学院
上机实践报告

课程名称： 近代密码学实验	**年级：** 2015 级	**上机实践成绩：**
指导教师： 陆正福	**姓名：** 刘鹏	
上机实践名称： 离散对数问题实验	**学号：** 20151910042	**上机实践日期：** 2018-05-29
上机实践编号： No.03	**组号：**	**上机实践时间：** 15:24

一、实验目的

熟悉离散对数问题（DLP）及其有关的密码体制。

二、实验内容

1. 熟悉离散对数问题（DLP），了解离散对数求解的困难性；
2. 编程实现 Diffie-Hellman 体制；
3. 编程实现 EIGamal 体制。

三、实验平台

Windows 10 ProWorkstation1803；
SageMath version 8.2, Release Date: 2018-05-05

四、实验记录与实验结果分析

1 题

编程实现与离散对数问题（DLP）有关的算法。求解离散对数问题常见的算法有：Shanks 的大步小步算法（baby-step giant-step algorithm）、Pollard rho 算法、Pohlig-Hellman 算法、Index Calculus 算法等。对于三十位以上的素数，已知最优的模p剩余类域中离散对数求解算法是应用了数域筛法技术的 Index Calculus 算法。

2 题

编程实现 Diffie-Hellman 密钥交换体制。

Solution:

Diffie 和 Hellman 在一篇具有独创意义的论文中首次提出了公钥算法，给出了公钥密码学的定义，该算法通常称为 Diffie-Hellman 密钥交换。该算法的目的是使两个用户能安全地交换密钥，以便在后续的通信中用该密钥对消息加密。该算法本身只限于进行密钥交换。[1]

Diffie-Hellman 算法的有效性是建立在计算离散对数是很困难的这一基础上的。一个素数p的本原根a，满足如下条件：$a \bmod p$，$a^2 \bmod p$，$\cdots$，$a^{p-1} \bmod p$是整数1到$p-1$的一个置换。对于任意一个整数b，必然有如下结论：存在一个整数i，满足$b \equiv a^i \bmod p$。这里的这个i称为b的以a为底的模p离散对数，记为$\mathrm{dlog}_{a,p}(b)$。

算法描述：公开一个素数p和其本原根a，然后 A 用户选择一个随机整数$X_A < p$，算出$Y_A = a^{X_A} \bmod p$，这时候，通过Y_A和p计算X_A是很困难的，所以Y_A可以在公共信道进行传输。当 A 用户接收到来自 B 用户的采用相同方法生成的$Y_B = a^{X_B} \bmod p$时，可以简单地计算$M = (Y_B)^{X_A} = (a^{(X_B)})^{X_A} = (a^{X_A})^{X_B} = (Y_A)^{X_B}$。

Sage 代码

```
R = IntegerModRing(2341)
Key_Liu_Peng = 19961019
```

```
Y_Liu_Peng = R(7^(Key_Liu_Peng))

Key_Zheng_Mao_Sen = 19970323
Y_Zheng_Mao_Sen = R(7^(Key_Zheng_Mao_Sen))

Key1 = R(Y_Liu_Peng^(Key_Zheng_Mao_Sen))
Key2 = R(Y_Zheng_Mao_Sen^(Key_Liu_Peng))

print Key2 == Key1
```

Figure 1 Diffie-Hellman 密钥交换过程模拟

可以看到，这个程序里面需要对私钥进行以本原根为底取指数，然后取模p的结果。这里面要用到快速幂取模算法，幸运的是 SageMath 里面内置了。这里的素数$p = 2341$，它有很多本原根，这里取的是7。

程序截图

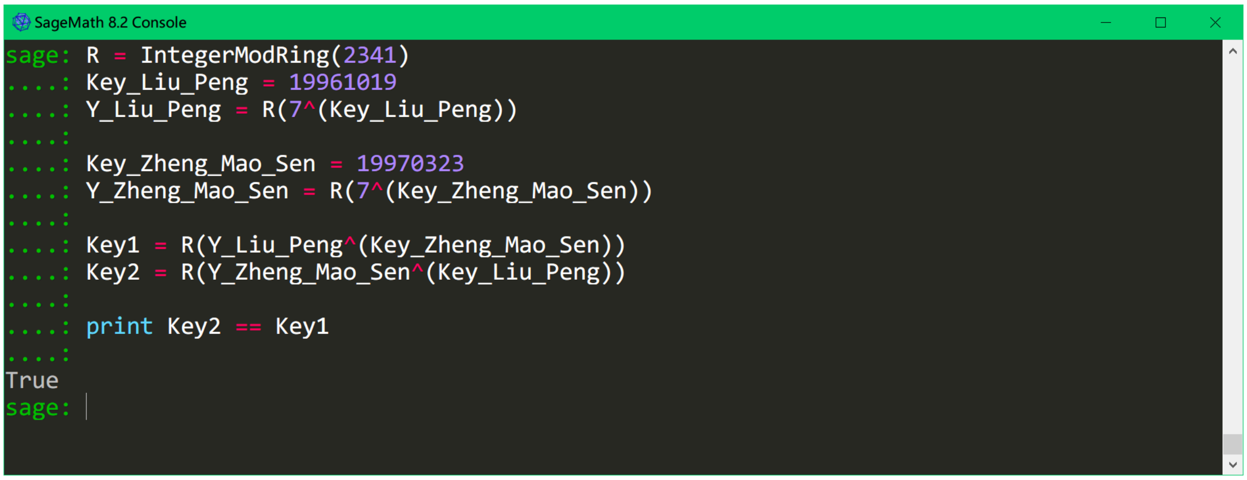

Figure 2 Diffie-Hellman 密钥交换体制过程模拟截图

过程分析

Diffie-Hellman 体制非常简单，核心的原理就是素数本原根的性质与离散对数的反推困难性。可以看到最后的 Key2 与 Key1 是相等的。而中间值都是可以在公共信道传输并避免被攻击者进行分析的。

3 题

编程实现 ElGamal 体制。

Solution:

1984 年，T. Elgamal 提出了一种基于离散对数的公开密钥体制，一种与 Diffie-Hellman 密钥分配体制密切相关的体制。ElGamal 密码体系应用于一些技术标准中，如数字签名标准（DSS）和 S/MIME 电子邮件标准。ElGamal 是一种公钥密码体制，与 RSA 类似，只不过 RSA 是基于大整数分解困难性，ElGamal 是基于离散对数求解困难性。

私钥持有用户生成公钥过程： A 用户想要公开自己的公钥，就首先选择一个大质数p及其一个本原根a，然后自己取一个整数X_A $(1 < X_A < p-1)$作为自己的私钥，本地保留。A 用户计算出$Y_A = a^{X_A} \bmod p$，公开公钥$PU = \{p,\ a,\ Y_A\}$；B 用户收到 A 用户的公钥，要用此公钥加密一大段消息M，那 B 用户就需要把M分成一些小的部分每个部分m_i都满足$1 \le m_i \le q-1$，然后进行分组加密。

B **用户加密过程：**选择M分组之后的一个部分m，针对这个m，选一个随机整数k，满足$1 \leq k \leq p-1$，计算一次性密钥$K_m = (Y_A)^k \bmod p$；生成密文对(C_1, C_2)，其中$C_1 = a^k \bmod q$，$C_2 = K_m M \bmod q$

A **用户解密过程：**计算出对方使用的一次性密钥$K_m = (C_1)^{X_A} \bmod q$，这是可行的，因为$(C_1)^{X_A} = (a^k)^{X_A} = (a^{X_A})^k = (Y_A)^k = K_m$，而离散对数的求解困难性保证过程不会被其他人看到；接着计算明文$M = (C_2 K^{-1}) \bmod q$，这也是可行的，因为有公式$C_2 = K_m M \bmod q$存在，所以只需要通过这个公式解出M就好了，这个公式等价于存在一个常整数s，使得等式$M \cdot K_m + (-s) \cdot q = C_2$，根据扩展的欧几里得算法可以解出$M$。而且由于为了尽量减少分组数量——每个分组的大小都取$q-1$，而最后一组不足$q-1$可以补位即可——导致不会算出很多个M。

Sage 代码

```
print "User A --Private Key keeper Generate Public Key...\n"
p = 2341
Primitive_root = 7

R = IntegerModRing(2341)

Private_Key = ZZ(1996)
if Private_Key not in range(2, p-2):
    raise ValueError("Bad Private Key")

Public_Key  = R(Primitive_root^Private_Key)

print "Generation of Public Key Success...\n"
print "User B --Encrypting a Message...\n"

m = "HELLO, WORLD!"
print "Message is ", m, "\n"
m = [ord(x) for x in m]
Cipher_Text = []
for i in m:
    k = randint(1, p)
    K = R(Public_Key^k)

    C_1 = R(Primitive_root^k)
    C_2 = R(K*i)
    Cipher_Text.append((C_1, C_2))

print Cipher_Text, "\n"

print "Encryption Success...\n"
print "User A --Received and Decrypting...\n"

Plain_Text = []
for i in Cipher_Text:
    K = R(i[0]^Private_Key)
    M = R(i[1] * K^(-1))
    Plain_Text.append(M)

tmp = [chr(i) for i in Plain_Text]
Received_Message = ""
```

```
for i in tmp:
    Received_Message += i
print "Decryption Finished!"

print "Message User A Received is", Received_Message
```

Figure 3 ElGamal 公钥密码体制

程序截图

Figure 4 ElGamal 运行结果

过程分析

由于这是个严格的分组密码，所以要考虑如何分组。我使用了一个不算大的质数2341来模拟实验，可以考虑 ASCII 编码下的字符串加密。每个 ASCII 字符占用 7 个字节，比2341小的2的整数幂是$2^{11} = 2048$，所以只能一个字符分一组。

六、实验体会

SageMath 的文档在国内比较少，要读官方的数论篇才能有所应用。总体来看，SageMath 在使用上还是比较方便的。

七、参考文献

[1] STALLINGS W. 密码编码学与网络安全：原理与实践 [M]. 6th ed. 北京: 机械工业出版社, 2015.