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 |

# 实验目的

熟悉离散对数问题（DLP）及其有关的密码体制。

# 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。

# 实验平台

Microsoft Windows 10 ProWorkstation1803；

SageMath version 8.2, Release Date: 2018-05-05

# 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。一个素数的本原根，满足如下条件：是整数到的一个置换。对于任意一个整数，必然有如下结论：存在一个整数，满足。这里的这个称为的以为底的模离散对数，记为。

算法描述：公开一个素数和其本原根，然后A用户选择一个随机整数，算出，这时候，通过和计算是很困难的，所以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时，可以简单地计算。

### Sage代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | 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，它有很多本原根，这里取的是。

### 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。[1]

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及其一个本原根，然后自己取一个整数作为自己的私钥，本地保留。A用户计算出，公开公钥；B用户收到A用户的公钥，要用此公钥加密一大段消息，那B用户就需要把分成一些小的部分每个部分都满足，然后进行分组加密。

B用户加密过程：选择分组之后的一个部分，针对这个，选一个随机整数，满足，计算一次性密钥；生成密文对，其中，

A用户解密过程：计算出对方使用的一次性密钥，这是可行的，因为，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，这也是可行的，因为有公式存在，所以只需要通过这个公式解出就好了，这个公式等价于存在一个常整数，使得等式，根据扩展的欧几里得算法可以解出而且由于为了尽量减少分组数量——每个分组的大小都取，而最后一组不足可以补位即可——导致不会算出很多个。

### Sage代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | **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

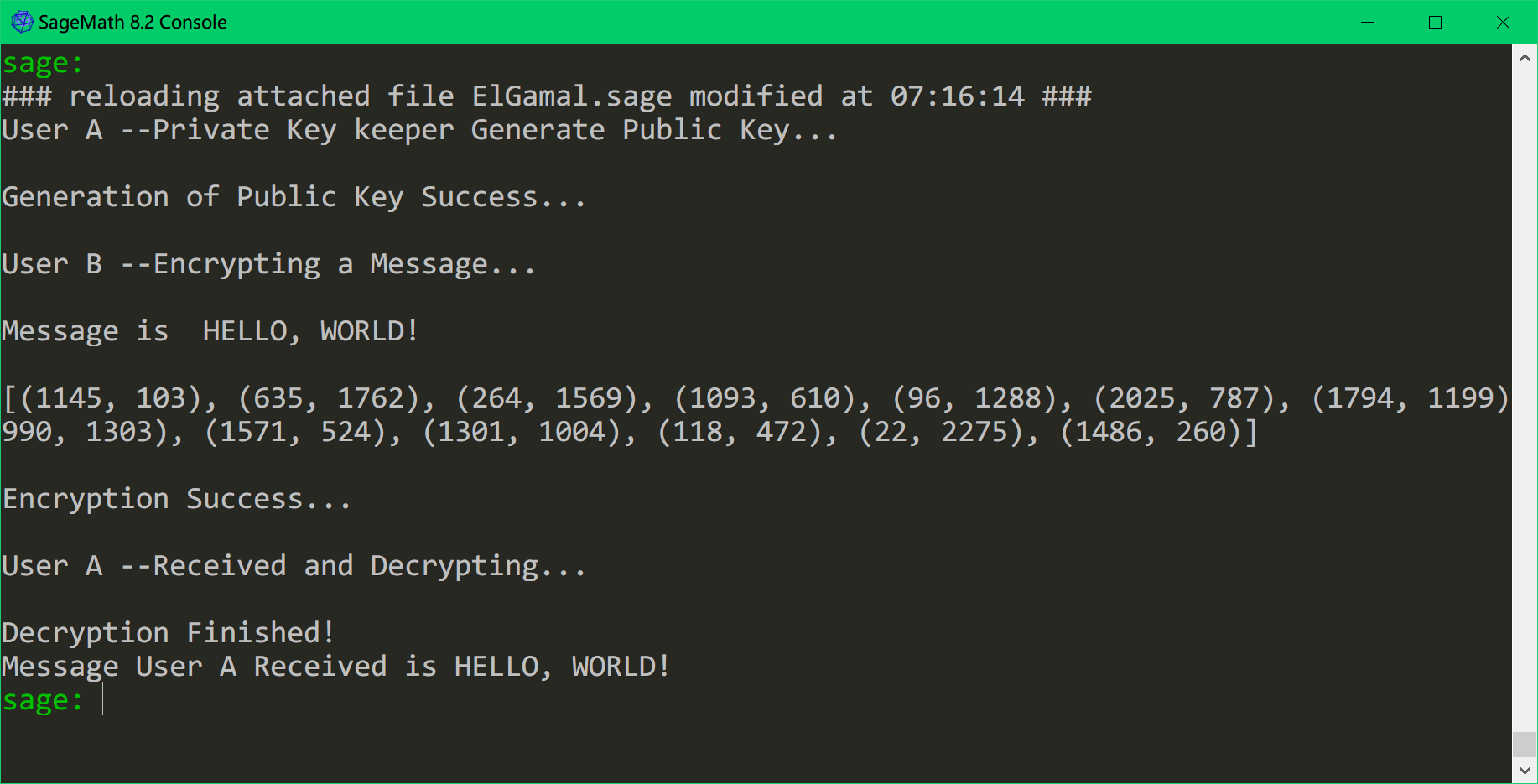


Figure 4 ElGamal运行结果

### 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，比小的的整数幂是，所以只能一个字符分一组。

# 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。

# 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.