

《信息安全基础综合实验》课程实验报告

实验题目：ElGamal 公钥密码算法

班级：1718039 学号 1：17189110002 姓名 1：祝佳磊

班级：1718039 学号 2：17180210027 姓名 2：李欣

一、实验目的

实验环境：

1. Visual Studio 2017

2. miracl 库

实现目标：实现 ElGamal 加密

二、方案设计

(包括背景、原理、必要的公式、图表、算法步骤等等)

背景：

ElGamal 密码是除了 RSA 密码之外最有代表性的公开密钥密码。

ElGamal 密码建立在离散对数的困难性之上。

ElGamal 算法是由 Taher Elgamal 于 1985 年在 Diffie-Hellman 密钥交换协议基础上改进而来。

原理：

2、离散对数问题：

(1) 设 p 为素数，则模 p 的剩余构成域：

$$F_p = \{0, 1, 2, \dots, p-1\}.$$

F_p 的非零元构成循环群 F_p^* ：

$$F_p^* = \{a^1, a^2, \dots, a^{p-1}\}$$

则称 a 为 F_p^* 的生成元，或模 p 的本原根。

(2) 设 p 是素数， a 为模 p 的本原根， a 的幂乘运算为

$$y = a^x \bmod p, 1 \leq x \leq p-1.$$

则称 x 为模 p 下以 a 为底 y 的对数。

2、离散对数问题：

求解对数 x 的运算为

$$x = \log_a y \bmod p, 1 \leq x \leq p-1$$

上述运算是定义在有限域上的，称为离散对数运算。

(3) 从 x 计算 y 是容易的，但是由 y 计算 x 就困难的多，只要 p 足够大，求解离散对数问题就是相当困难的。

ElGamal的密钥生成

ElGamal公钥密码算法的密钥包括公钥和私钥两部分，假设通信过程中A方负责生成密钥，其他方使用密钥，A方生成密钥的过程如下：

(1) 生成随机大素数 p ，求得 p 的本原根 g 。

(2) 随机选择私钥 x ， $1 < x < p-2$ 。

(3) 计算 $y = g^x \bmod p$ 。

计算得到的公钥为 (p, g, y) ，对外公布。

私钥是 x ，自己私藏。

ElGamal的加密过程

假设B要与A进行通信，B所要加密的明文为 m ，使用ElGamal 密码算法，B方执行的加密过程如下：

- (1)从A方获得加密所需的公钥 (p, g, y) 。
- (2) 选择一随机数 $k, (k, p-1) = 1, 1 \leq k \leq p-2$ 。
- (3)计算

$$y_1 = g^k \bmod p \text{ (随机数 } k \text{ 被加密)}。$$

再用公钥 y ，计算：

$$y_2 = my^k \bmod p \text{ (明文被随机数 } k \text{ 和公钥 } y \text{ 加密)}$$

密文 $c=(y_1, y_2)$ 。

ElGamal的解密过程

当A接收到密文 $c=(y_1, y_2)$ 之后，解密过程如下：
使用自己的私钥 x 计算

$$m = (y_1^{-x})y_2 \bmod p$$

获得明文 m 。

上述解密过程的原理为：

$$(y_1^{-x})y_2 = g^{-xk}my^k = g^{-xk}mg^{xk} \equiv m \bmod p$$

特点：密文由明文和所选随机数 k 来确定，因而是非确定性加密，一般称之为随机化加密，对同一明文由于不同时刻的随机数 k 不同而给出不同的密文。代价是使数据扩展一倍。

三、方案实现

```
#define _CRT_SECURE_NO_WARNINGS
#include <miracl.h>
#include <mirdef.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

void encrypt(big m, big p, big g, big y, big cs[2])
{
    big one, psubone, k, tmp;
    one = mirvar(1);
    psubone = mirvar(0);
    k = mirvar(0);
    tmp = mirvar(0);
    cs[0] = mirvar(0);
    cs[1] = mirvar(0);

    decr(p, 1, psubone);
    while (1)
    {
        bigrand(psubone, k);
        if (mr_compare(k, one) >= 0)
        {
            egcd(k, y, tmp);
            if (mr_compare(tmp, one) == 0)
                break;
        }
    }
    powmod(g, k, p, cs[0]);
    powmod(y, k, p, cs[1]);
    multiply(cs[1], m, cs[1]);
    powmod(cs[1], one, p, cs[1]);

    mirkill(one);
    mirkill(psubone);
    mirkill(k);
    return;
}
```

引入对应的库，定义加密函数，需要传入 m,p,g,y,cs 五个参数

循环生成随机数 k，求 k 和 y 对应的最大公因数，

计算 $y_1 = g^k \bmod p$ ，再计算 $y_2 = m \cdot y^k \bmod p$

```

big decrypt(big cs[2], big p, big x)
{
    big tmp, one, ret;

    tmp = mirvar(0);
    ret = mirvar(0);
    one = mirvar(1);

    powmod(cs[0], x, p, tmp);
    xgcd(tmp, p, tmp, tmp, tmp);
    fft_mult(cs[1], tmp, tmp);
    powmod(tmp, one, p, ret);

    mirkill(tmp);
    mirkill(one);

    return ret;
}

```

定义解密函数

```

59 int main(int argc, char* argv[])
60 {
61     miracl *mip = mirsys(1000, 10);
62     char buf[1001] = { 0 };
63     char msg[151] = { 0 };
64     FILE* fp = NULL;
65     big two, one, p, q, g, x, y, m, c, tmp, psubtwo;
66     big cs[2], ret;
67
68     mip->IOBASE = 10;
69     //读取文件数据
70     if (argc != 2)
71     {
72         fprintf(stderr, "usage: %s <message-file-path>\n", argv[0]);
73         goto exit;
74     }
75     if ((fp = fopen(argv[1], "r")) == NULL)
76     {
77         fprintf(stderr, "file open err: can't open file %s !\n", argv[1]);
78         goto exit;
79     }
80     if (fread(msg, sizeof(char), 150, fp) <= 0)
81     {
82         fprintf(stderr, "file read err: can't read file %s !\n", argv[1]);
83         fclose(fp);
84         goto exit;
85     }
86     fclose(fp);

```

主函数，读取需要加密的文件信息

```

88      //初始化大数
89      one = mirvar(1);
90      two = mirvar(2);
91      p = mirvar(0);
92      q = mirvar(0);
93      g = mirvar(2);
94      x = mirvar(0);
95      y = mirvar(0);
96      m = mirvar(0);
97      c = mirvar(0);
98      tmp = mirvar(0);
99      psubtwo = mirvar(0);
100     irand(time(NULL));
101
102     //
103     cinstr(m, msg);
104
105
106     //选取随机素数q和p
107     //生成强素数p, p=2q+1
108     while (1)
109     {
110         bigbits(499, q);
111         if (isprime(q))
112             break;
113     }

```

初始化大数, 选取随机素数 q 和 p, $p=2q+1$

```

115     while (1)
116     {
117         add(q, two, q);
118         if (!isprime(q))
119         {
120             continue;
121         }
122         add(q, q, p);
123         add(p, one, p);
124         if (isprime(p))
125         {
126             break;
127         }
128     }

```

检验 q 和 p

```

131     do {
132         powmod(g, two, p, tmp);
133         if (mr_compare(tmp, one) > 0)
134         {
135             powmod(g, q, p, tmp);
136             if (mr_compare(tmp, one) > 0) {
137                 break;
138             }
139         }
140         add(g, one, g);
141
142     } while (mr_compare(g, p) < 0);
143
144     //随机取x, 1<x<p-2
145     decr(p, 2, psubtwo);
146     do {
147         bigrand(psubtwo, x);
148     } while (mr_compare(x, one) <= 0);
149
150     //计算y, y=g^x mod p
151     powmod(g, x, p, y);
152

```

生成元 g, 取随机 x

```

153     //输出公钥
154     printf("公钥 : \n");
155     memset(buf, 0, 501);
156     cotstr(p, buf);
157     printf("p = %s\n", buf);
158     memset(buf, 0, 501);
159     cotstr(g, buf);
160     printf("g = %s\n", buf);
161     memset(buf, 0, 501);
162     cotstr(y, buf);
163     printf("y = %s\n", buf);
164     printf("\n");
165
166     //输出私钥
167     printf("私钥 : \n");
168     memset(buf, 0, 501);
169     cotstr(x, buf);
170     printf("x = %s\n", buf);
171     printf("\n");
172
173     //加密
174     encrypt(m, p, g, y, cs);
175

```

输出密钥和公钥, 进行加密

```

176     //输出y1和y2
177     printf("密文 : \n");
178     memset(buf, 0, 501);
179     cotstr(cs[0], buf);
180     printf("y1 = %s\n", buf);
181     memset(buf, 0, 501);
182     cotstr(cs[1], buf);
183     printf("y2 = %s\n", buf);
184     printf("\n");
185
186     //解密
187     ret = decrypt(cs, p, x);
188
189     //输出还原的消息
190     printf("明文 : \n");
191     memset(buf, 0, 501);
192     cotstr(ret, buf);
193     printf("message = %s\n", buf);
194     printf("\n");
195
196     system("pause");
197 exit:
198     mirexit();
199     return 0;
200 }

```

解密并输出还原的信息，程序结束

四、数据分析

使用 secret1.txt 作为待加密信息，使用方法如下 usage: %s <message-file-path>


```
E:\大学课程学习\信安基础实验\实验四\ConsoleApplication2\Debug
λ ConsoleApplication2.exe secret1.txt
公钥:
p = 1461400738365759810352087444038334226291186977543237359872023633894639097379304245305749311984851729736
89639971671649461914470455261538329134955285467
g = 2
y = 1131127176037520921445032847220207380855026123059549909381272276376803776033923920630617882398890455584
51298868481612587773477583168470508248775051950

私钥:
x = 1202911735492777407604694127480305905352355487049156682346115716263713920675142550487359029978140422848
95278044598825991914650062784444439179393632889

密文:
y1 = 110845080822064045100211515241089815368373959900788585909969714204962728413466866580527491843952571836
413619396034209835461008396753242598189136333469
y2 = 341535461798192913979619654771842097200464299031799063232966788595997660314872658142956412288534161548
23734086656219240480323184342804892175288202796

明文:
message = 463043908984145268397876053474118516802738548700188741774731591548492120159413603037149094402303
898784082087634155850402179405220228989097
```

公私钥对如图所示，解密出的明文和之前的信息一致，成功实现了加解密。

五、总结

（完成的心得和其他，主要是自己碰到的问题，以及解决问题的方法等）

整个实验的过程较为顺利，实验主要就是实现 ElGamal 算法的加解密公式。

并对强素数 p 的生成进行优化。