

## 信息安全作业（二）

计算机 1202 张艺瀚  
学号:20123852

April 9, 2015

程序使用说明：在 Linux 下当前目录执行可执行文件，按照提示操作即可，具体见运行结果截图。

### 1 编程实现求乘法逆元

#### 1.1 代码清单

```
1 int extendedEuclid(int a, int b, int& x, int& y){
2     if(!b){
3         x = 1;
4         y = 0;
5         return a;
6     }
7     int r = extendedEuclid(b, a % b, x, y);
8     int t = x;
9     x = y;
10    y = t - a / b * y;
11    return r;
12 }
13
14 int main(int argc, char** argv){
15     int a, b, x, y;
16     cout << "Input a: ";
17     cin >> a;
18     cout << "Input b: ";
19     cin >> b;
20     extendedEuclid(a, b, x, y);
21     cout << "( a * " << x << " ) mod b = 1" << endl;
22     return 0;
```

23 }

Listing 1: 扩展的 Euclid 算法求乘法逆元的 C++ 实现

## 1.2 说明

从键盘读入 2 个整数  $a$  和  $b$ , `extendedEuclid` 函数返回  $a$  和  $b$  的最大公约数  $\gcd(a, b)$ , 并计算  $a$  模  $b$  的乘法逆元  $x = a^{-1} \bmod b$  和  $b$  模  $a$  的乘法逆元  $y = b^{-1} \bmod a$ , 打印  $x$  和  $y$  到屏幕

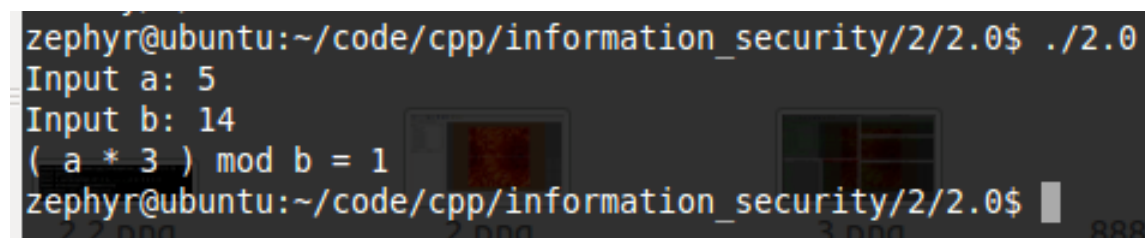
### 思路

1. 执行 Euclid 算法, 若最终余数为 1, 即  $\gcd(a, b) = 1$ , 则存在乘法逆元
2. 逆推, 得到形如  $a \cdot x + k \cdot b = 1$  的式子, 其中  $k \in \mathbb{Z}$ , 则  $x = a^{-1} \bmod b$

### 注意

1. 0 不在群中, 更无逆元
2. 若  $a$  和  $b$  不互素, 则无乘法逆元

## 1.3 运行结果



```
zephyr@ubuntu:~/code/cpp/information_security/2/2.0$ ./2.0
Input a: 5
Input b: 14
( a * 3 ) mod b = 1
zephyr@ubuntu:~/code/cpp/information_security/2/2.0$
```

Figure 1: 扩展的 Euclid 算法求乘法逆元的运行结果

## 2 编程实现换位密码

### 2.1 列换位密码

#### 2.1.1 代码清单

```
1 template<typename T>
2 void display(const vector<T>& v){
3     for_each(v.begin(), v.end(),
4         [](T t){
5             cout << t << " ";
6         });
7     cout << endl;
8 }
9
10 int main(int argc, char** argv){
11     string plaintext, ciphertext;
12     int column;
13     cout << "Input plaintext: ";
14     cin >> plaintext;
15     // plaintext = "datasecurity";
16     cout << "Input column number: ";
17     cin >> column;
18     vector<int> permutation(column);
19     cout << "Input a permutation of { 1, ... , " << column << " }:"
20         << endl;
21     for(size_t i = 0; i < column; ++i){
22         cin >> permutation[i];
23     }
24     int dist = 26;
25     int row = plaintext.size() / column;
26     if(row * column != plaintext.size()){
27         ++row;
28     }
29     int n = row * column;
30     for(size_t i = 0; i < column; ++i){
31         int m = 26, idx;
32         for(size_t j = 0; j < permutation.size(); ++j){
33             if(permutation[j] >= 0 && permutation[j] < m){
34                 idx = j;
35                 m = permutation[j];
36             }
37         }
38         permutation[idx] = -1;
39         for(auto p = plaintext.begin() + idx;;){
40             int d = p - plaintext.begin();
```

```
40     if(d < plaintext.size()){
41         ciphertext.push_back(*p);
42     }else if(d < n){
43         ciphertext.push_back('z');
44     }
45     if(d < n){
46         p += column;
47     }else{
48         break;
49     }
50 }
51 }
52 cout << "Ciphertext is: " << ciphertext << endl;
53 return 0;
54 }
```

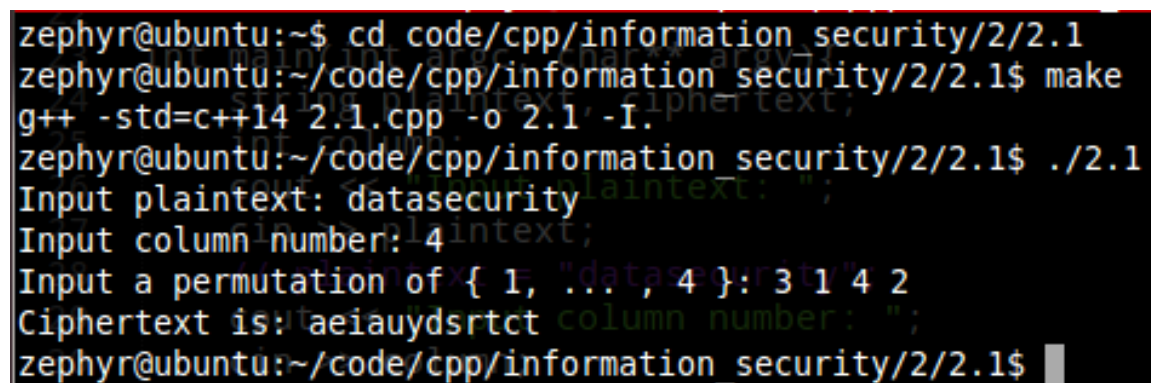
Listing 2: 列换位密码的 C++ 实现

### 2.1.2 说明

从键盘读入矩阵的列数  $column$ 、明文和一个  $(1, 2, \dots, column)$  的置换  $(\pi_1, \pi_2, \dots, \pi_{column})$ ，输出密文，若明文无法刚好填满矩阵，则补  $x$

置换  $(\pi_1, \pi_2, \dots, \pi_{column})$  指明将明文排成  $column$  列的矩阵后，输出密文的列顺序

### 2.1.3 运行结果



```
zephyr@ubuntu:~$ cd code/cpp/information_security/2/2.1
zephyr@ubuntu:~/code/cpp/information_security/2/2.1$ make
g++ -std=c++14 2.1.cpp -o 2.1 -I.
zephyr@ubuntu:~/code/cpp/information_security/2/2.1$ ./2.1
Input plaintext: datasecurity
Input column number: 4
Input a permutation of { 1, ..., 4 }: 3 1 4 2
Ciphertext is: aeiauydsrtct
zephyr@ubuntu:~/code/cpp/information_security/2/2.1$
```

Figure 2: 列换位算法的运行结果

## 2.2 周期换位密码

### 2.2.1 代码清单

```
1 int main(int argc, char** argv){
2     int d;
3     cout << "Input period: ";
4     cin >> d;
5     vector<int> f(d);
6     cout << "Input a permutation of { 1, ... , " << d << " }: ";
7     for(int i = 0; i < d; ++i){
8         cin >> f[i];
9     }
10    string m, c;
11    cout << "Input plaintext: ";
12    cin >> m;
13    c.resize(m.size());
14    for(size_t i = 0; i < m.size(); ++i){
15        c.push_back(m[f[i % d] + floor(i / d) * d - 1]);
16    }
17    cout << "Ciphertext is: " << c << endl;
18    return 0;
19 }
```

Listing 3: 周期换位密码的 C++ 实现

### 2.2.2 说明

从键盘输入明文、周期长度  $d$  和一个  $(1, 2, \dots, d)$  的置换  $(\pi_1, \pi_2, \dots, \pi_d)$ ，将密文打印在屏幕上。函数  $f(i) = \pi_i$  表示明文的第  $i$  个字母是密文中第  $f(i)$  个字母。则我们容易得到如下公式

$$C[i] = M \left[ f(i \bmod d) + \left\lfloor \frac{i}{d} \right\rfloor \cdot d \right]$$

其中  $C$  为密文， $M$  为明文

### 2.2.3 运行结果

见图 3

```
zephyr@ubuntu:~/code/cpp/information_security/2/2.2$ ./2.2
Input period: 4
Input a permutation of { 1, ..., 4 }: 2 4 1 3
Input plaintext: datasecurity
Ciphertext is: aadteusciyrt
zephyr@ubuntu:~/code/cpp/information_security/2/2.2$
```

Figure 3: 周期换位算法的运行结果

### 3 举例说明 Legendre 符号的作用

1. 验证  $a$  是否是模  $p$  的二次剩余。若  $L(a, p) = 1$ ，则  $a$  是模  $p$  的二次剩余，否则不是
2. Jacobi 符号是 Legendre 符号的扩展，它是通过 Legendre 符号来计算的。

(a)  $n$  为素数，则  $J(a, n) = L(a, n)$

(b)  $n$  为合数，则  $J(a, n) = \prod_{i=1}^m L(a, p_i)$ ，其中  $n$  因数分解为  $\prod_{i=1}^m p_i$

3. 在 Solovay-Strassen 素性测试和 Rabin-Miller 素性测试中均用到了 Legendre 符号或 Jacobi 符号