单表代换

程序

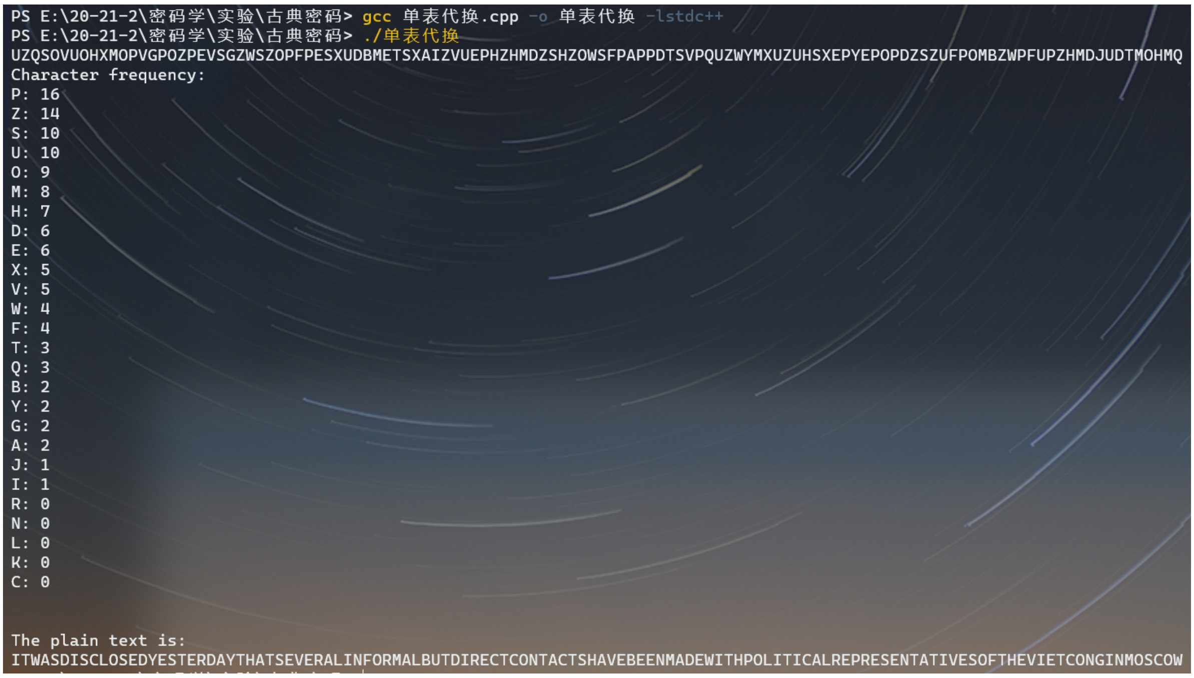
```cpp
#include <algorithm>
#include <cstdio>
#include <iostream>
#include <map>
using namespace std;

const int numberOfChara = 120; //number of characters

struct prob
{
    char chara; //character
    int freq;   //frequency
} p[26];

char set_freq[] = {'E', 'T', 'A', 'I', 'S', 'O', 'C', 'N', 'R', 'L', 'D', 'H', 'V',
                   'M', 'W', 'F', 'P', 'Y', 'B', 'G', 'U', 'K', 'J', 'X', 'Q', 'Z'}; //sort by the frequency

bool cmp(prob p1, prob p2) //for func sort
{
    return p1.freq > p2.freq;
}

int main()
{
    //initialization
    char c[numberOfChara + 1]; //store the input string
    for (int i = 0; i < 26; i++)
    {
        p[i].chara = i + 'A';
        p[i].freq = 0;
        }

    //input
    for (int i = 1; i <= numberOfChara; i++)
    {
        c[i] = getchar();
        p[c[i] - 'A'].freq++;
    }

    //sort and match
    sort(p, p + 26, cmp);
    printf("Character frequency:\n");
    for (int i=0;i<26;i++)
        printf("%c: %d\n",p[i].chara,p[i].freq);
    printf("\n\nThe plain text is: \n");
    map<char, char> m;
    for (int i = 0; i < 26; i++)
        m[p[i].chara] = set_freq[i];

    //output
    for (int i = 1; i <= numberOfChara; i++)
        cout << m[c[i]];
}
```

程序解释

```
char set_freq[] = {'E', 'T', 'A', 'I', 'S', 'O', 'C', 'N', 'R', 'L', 'D', 'H', 'V',
                   'M', 'W', 'F', 'P', 'Y', 'B', 'G', 'U', 'K', 'J', 'X', 'Q', 'Z'};
```

为根据科学统计结果得出的各个字母平均出现概率的排序。

运行截图



```
PS E:\20-21-2\密码学\实验\古典密码> gcc 单表代换.cpp -o 单表代换 -lstdc++
PS E:\20-21-2\密码学\实验\古典密码> ./单表代换
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZVUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSXEPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
Character frequency:
P: 16
Z: 14
S: 10
U: 10
O: 9
M: 8
H: 7
D: 6
E: 6
X: 5
V: 5
W: 4
F: 4
T: 3
Q: 3
B: 2
Y: 2
G: 2
A: 2
J: 1
I: 1
R: 0
N: 0
L: 0
K: 0
C: 0


The plain text is:
ITWASDISCLOSEDYESTERDAYTHATSEVERALINFORMALBUTDIRECTCONTACTSHAVEBEENMADEWITHPOLITICALREPRESENTATIVESOFTHEVIETCONGINMOSCOW
```

结果为 IT WAS DISCLOSED YESTERDAY THAT SEVERAL INFORMAL BUT DIRECT CONTACT SHAVE BEEN MADE WITH POLITICAL REPRES ENTATIVES OF THE VIET CONGIN MOSCOW

RSA

程序

```cpp
#include <cstdio>
#include <ctime>
#include <iostream>
using namespace std;

long long prime[10]={2,3,5,7,11,13,17,19,23,29};
long long p,q;
//const long long p = 10191161;
//const long long q = 10191133;
long long n;

void ex_gcd(long long a, long long b, long long &x, long long &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return;
    }
    ex_gcd(b, a % b, x, y);
    long long t = x;
    x = y;
    y = t - a / b * y;
    return;
}

long long qMultiply(long long x, long long y, long long c)
{
    return (x * y - (long long)((long double)x / c * y) * c + c) % c;
}

long long qPower(long long base, long long power, long long c)
{
    long long ans = 1, res = base;
    while (power)
    {
        if (power & 1) //power is even
            ans = qMultiply(ans, res, c) % c;
        res = qMultiply(res, res, c) % c;
        power >>= 1; //power/2
    }
    return ans;
}

bool is_coprime(long long a, long long b)
{
    if (a == 1 || b == 1)
```

```
        return true;
    while (true)
    {
        long long t = a % b;
        if (t == 0)
            break;
        else
        {
            a = b;
            b = t;
        }
    }
    return (b == 1);
}

bool mr(long long number)//Miller-Rabin
{
    long long a,b,s=0,t=number-1;

    while(!(t&1))
    {
        s++;
        t>>=1;
    }

    for(long long i=0;i<10;i++)
    {
        a=qPower(prime[i],t,number);
        for(long long j=1;j<=s;j++)
        {
            b=(a*a)%number;
            if(b==1 && a!=1 && a!=number-1)
              return false;
            a=b;
        }
        if(a!=1)
            return false;
    }
    return true;
}

long long RSA_encode(long long m, long long &d, long long &e, long long &phi_n)
{
    long long temp;
    n = p * q;
    phi_n = n - p - q + 1;
    e = rand() % phi_n + 1;
    while (!is_coprime(e, phi_n))
        e = rand() % (phi_n - 2) + 2;
    ex_gcd(e, phi_n, d, temp);
    while (d <= 0)
        d += phi_n;
    return (qPower(m, e, n));
}

long long RSA_decode(long long n, long long d, long long c)
{
    return (qPower(c, d, n));
```

```
}

int main()
{
    srand(time(0));
    printf("\n------------------------------------------------------------\n");
    printf("ENCODE BEGIN:");
    printf("\n------------------------------------------------------------\n");
    long long m, d, e, phi_n, c, Message;
    printf("\nInput your plain text: ");
    scanf("%lld", &m);
    p=rand();
    long long prand=rand()%20+1;
    while (p<(prand*m)) p+=rand();
    q=rand();
    long long qrand=rand()%20+1;
    while (q<(qrand*m)) q+=rand();
    while (!mr(p)) p++;
    while (!mr(q)) q++;
    printf("\nPrimes are p=%lld, q=%lld\n",p,q);
    c = RSA_encode(m, d, e, phi_n);
    printf("\nPK is (%lld,%lld)\nSK is (%lld,%lld)\n", e, n, d, n);
    printf("\nCipher text is %lld\n", c);

    printf("\n------------------------------------------------------------\n");
    printf("DECODE BEGIN:");
    printf("\n------------------------------------------------------------\n");
    Message = RSA_decode(n, d, c);
    printf("\nPlain text is %lld\n\n", Message);
}
```

程序解释

```
void ex_gcd(long long a, long long b, long long &x, long long &y)
```

为扩展欧几里得算法

```
long long qMultiply(long long x, long long y, long long c)
```

为快速乘算法，主要为了防止乘法溢出

```
long long qPower(long long base, long long power, long long c)
```

为快速幂算法，加快幂运算，将时间复杂度从平凡算法的 $O(n)$ 提高到 $O(\log n)$

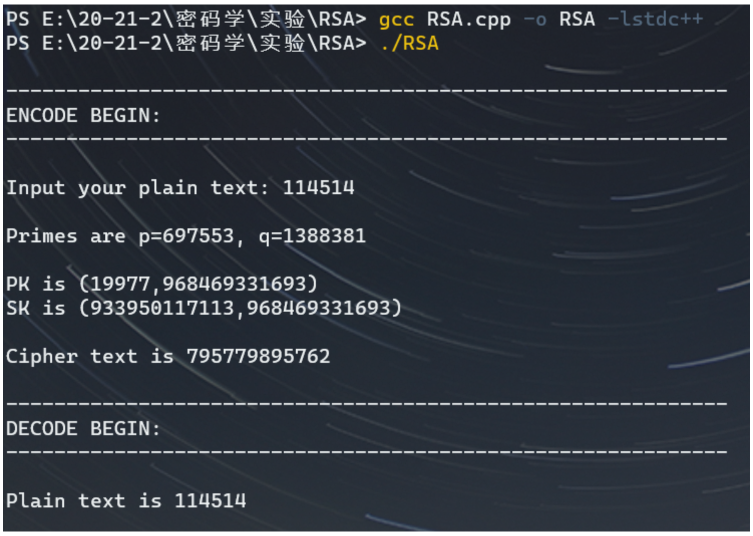
```
bool is_coprime(long long a, long long b)
```

用于验证两个数是否互质

```
bool mr(long long number)
```

为 Miller-Rabin 素数筛，用于生成两个大素数，同时速度比平凡算法快得多。为保证其准确性，使用 10 位以内的素数作示例，素数集使用 `prime[10]={2,3,5,7,11,13,17,19,23,29}` 即可保证无缺漏或错判。

运行截图



```
PS E:\20-21-2\密码学\实验\RSA> gcc RSA.cpp -o RSA -lstdc++
PS E:\20-21-2\密码学\实验\RSA> ./RSA

------------------------------------------------------------
ENCODE BEGIN:
------------------------------------------------------------

Input your plain text: 114514

Primes are p=697553, q=1388381

PK is (19977,968469331693)
SK is (933950117113,968469331693)

Cipher text is 795779895762

------------------------------------------------------------
DECODE BEGIN:
------------------------------------------------------------

Plain text is 114514
```

公钥为 (19977, 968469331693)

私钥为 (933950117113, 968469331693)

密文为 795779895762

解密正确

ElGamal

程序

```
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <vector>
using namespace std;

struct Point
{
    int x, y;
    Point() {}
    Point(int X, int Y)
    {
        x = X;
        y = Y;
    }
};

int Inv[10001];
int k;

int Mod(int a, int p)
{
    return (a % p >= 0 ? a % p : (a % p) + p);
}

```

```
void Inverse(int p)
{
    for (int i = 1; i < p; i++)
        for (int j = 1; j < p; j++)
            if ((i * j) % p == 1)
                Inv[i] = j;
}

Point Plus(Point p1, Point p2, int p, int a)
{
    int Lambda;
    Point TmpPoint;
    if (p1.x == p2.x && p1.y == p2.y)
        Lambda = Mod((3 * p1.x * p1.x + a) * Inv[Mod(2 * p1.y, p)], p);
    else
        Lambda = Mod((p2.y - p1.y) * Inv[Mod(p2.x - p1.x, p)], p);
    TmpPoint.x = Mod(Lambda * Lambda - p1.x - p2.x, p);
    TmpPoint.y = Mod(Lambda * (p1.x - TmpPoint.x) - p1.y, p);
    //printf("lambda=%d,x=%d,y=%d\n",Lambda,TmpPoint.x,TmpPoint.y);
    return TmpPoint;
}

Point Minus(Point p1, Point p2, int p, int a)
{
    Point Minus_p2(p2.x, Mod(p - p2.y, p));
    return Plus(p1, Minus_p2, p, a);
}

Point Multiple(int k, Point p1, int p, int a)
{
    Point TmpPoint(p1.x, p1.y);
    for (int i = 1; i < k; i++)
    {
        TmpPoint = Plus(TmpPoint, p1, p, a);
        //printf("(%d,%d)\n",TmpPoint.x,TmpPoint.y);
    }
    return TmpPoint;
}

Point Choose_G(int p, int a, int b)
{
    printf("\nChoose a generator frome below (input in the format of x y):\n");
    vector<Point> G_list;
    for (int i = 0; i < p; i++)
        for (int j = 0; j < p; j++)
            if (((i * i * i + a * i + b) % p) == ((j * j) % p))
                G_list.push_back(Point(i, j));
    for (int i = 0; i < G_list.size(); i++)
        printf("(%d,%d) ", G_list[i].x, G_list[i].y);
    int X, Y;
    printf("\nYou choose ");
    scanf("%d%d", &X, &Y);
    bool flag;
    do
    {
        flag = false;
        for (int i = 0; i < G_list.size(); i++)
        {
            if (G_list[i].x == X && G_list[i].y == Y)
            {
                flag = true;
            }
        }
        if (!flag)
        {
            printf("\nWrong! You should choose among the list above!");
            printf("\nYou choose ");
            scanf("%d%d", &X, &Y);
        }
    } while (!flag);
    //printf("Success1\n");
    return Point(X, Y);
}

Point Implant(int m, int p, int a, int b)
{
    k = rand() % 20 + 30;
    for (int i = 0; i < p; i++)
    {
        int Tmp = Mod(k * m + i, p);
        for (int j = 0; j < p; j++)
            if ((((((Tmp * Tmp) % p) * Tmp) % p + (a * Tmp) % p + b) % p) == ((j * j) % p))
            {
                //printf("Tmp=%d k=%d i=%d j=%d\n",Tmp,k,i,j);
                return Point(Mod(Tmp, p), j);
            }
    }
    return Point(-1, -1);
}

void ElGamal_encode(int &p, int &a, int &b, int &n_A, Point &Cm1, Point &Cm2)
{
    printf("\n------------------------------------------------------------\n");
    printf("ENCODE BEGIN:");
    printf("\n------------------------------------------------------------\n");
    printf("\nInput parameters of ECC(y^2=x^3+ax+b mod p):\n");
    printf("a=");
    scanf("%d", &a);
    printf("b=");
    scanf("%d", &b);
    printf("p=");
    scanf("%d", &p);

    printf("\n");
    Inverse(p);

    //for (int i=1;i<p;i++)
    //printf("Inv[%d]=%d\n",i,Inv[i]);

    Point G = Choose_G(p, a, b);
    //printf("Success2\n");

    printf("\nYou have choosen ECC E_%d(%d,%d) and generator (%d,%d)\n", p, a, b, G.x, G.y);

    printf("\nInput your private key n_A: ");
    scanf("%d", &n_A);

    Point P_A = Multiple(n_A, G, p, a);
    printf("\nYour public key P_A is (%d,%d)\n", P_A.x, P_A.y);

    printf("\nInput your plain text m: ");
    int m;
    scanf("%d", &m);
    Point P_m = Implant(m, p, a, b);
    printf("\nThe message P_m you want to send is (%d,%d)\n", P_m.x, P_m.y);

    int k = rand() % (p - 1) + 1;
    Cm1 = Multiple(k, G, p, a);
    Cm2 = Plus(P_m, Multiple(k, P_A, p, a), p, a);
    printf("Your cipher text is {(%d,%d),(%d,%d)}\n\n", Cm1.x, Cm1.y, Cm2.x, Cm2.y);
}

void ElGamal_decode(int p, int a, int n_A, Point Cm1, Point Cm2)
{
    printf("\n------------------------------------------------------------\n");
    printf("DECODE BEGIN:");
    printf("\n------------------------------------------------------------\n");
    Point P_m = Minus(Cm2, Multiple(n_A, Cm1, p, a), p, a);
    printf("\nThe message P_m is (%d,%d)\n\n", P_m.x, P_m.y);
    //printf("\nYour origin plain text m mod p is %d\n\n",P_m.x/k);
}

int main()
{
    srand(time(0));
    int p, a, b, n_A;
    Point Cm1, Cm2;
    ElGamal_encode(p, a, b, n_A, Cm1, Cm2);
    ElGamal_decode(p, a, n_A, Cm1, Cm2);
}
```

程序解释

```
int Mod(int a, int p)
```

是为了取模时将负数变为正数

```
Point Plus(Point p1, Point p2, int p, int a)
Point Minus(Point p1, Point p2, int p, int a)
Point Multiple(int k, Point p1, int p, int a)
```

为椭圆曲线的加、减、乘法

```
void Inverse(int p)
```

为将 p 以内的数求逆并存储

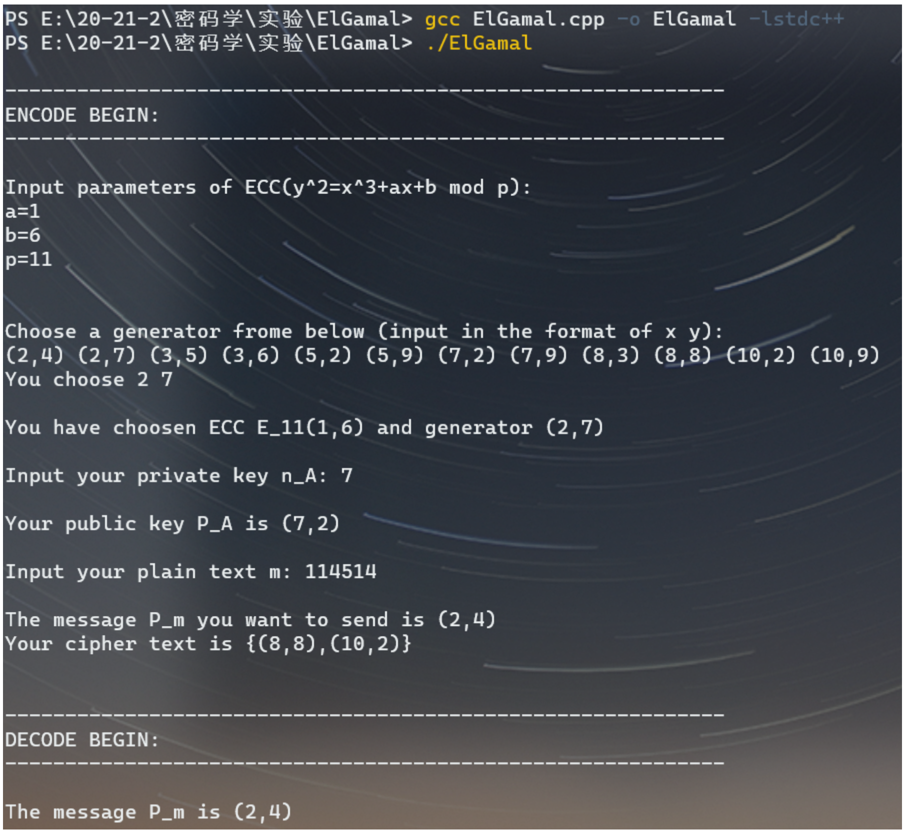
```
Point Choose_G(int p, int a, int b)
```

为计算生成元

```
Point Implant(int m, int p, int a, int b)
```

将明文嵌入到椭圆曲线上

运行截图



```
PS E:\20-21-2\密码学\实验\ElGamal> gcc ElGamal.cpp -o ElGamal -lstdc++
PS E:\20-21-2\密码学\实验\ElGamal> ./ElGamal

------------------------------------------------------------
ENCODE BEGIN:
------------------------------------------------------------

Input parameters of ECC(y^2=x^3+ax+b mod p):
a=1
b=6
p=11


Choose a generator frome below (input in the format of x y):
(2,4) (2,7) (3,5) (3,6) (5,2) (5,9) (7,2) (7,9) (8,3) (8,8) (10,2) (10,9)
You choose 2 7

You have choosen ECC E_11(1,6) and generator (2,7)

Input your private key n_A: 7

Your public key P_A is (7,2)

Input your plain text m: 114514

The message P_m you want to send is (2,4)
Your cipher text is {(8,8),(10,2)}


------------------------------------------------------------
DECODE BEGIN:
------------------------------------------------------------

The message P_m is (2,4)
```

选取了椭圆曲线 $E_{11}(1,6)$，生成元选取了 $(2,7)$，Alice 密钥选取了 7，要传递的消息为 $(2,4)$

得到 Alice 的公钥为 $(7,2)$，密文为 $\{(8,8),(10,2)\}$