

# DiffHellman

## 程序

```
1  #include<cstdio>
2  #include<vector>
3  using namespace std;
4
5  struct Point
6  {
7      int x, y;
8      Point() {}
9      Point(int X, int Y)
10     {
11         x = X;
12         y = Y;
13     }
14 };
15
16 int Inv[10001];
17 int k;
18
19 int Mod(int a, int p)
20 {
21     return (a % p >= 0 ? a % p : (a % p) + p);
22 }
23
24 void Inverse(int p)
25 {
26     for (int i = 1; i < p; i++)
27         for (int j = 1; j < p; j++)
28             if ((i * j) % p == 1)
29                 Inv[i] = j;
30 }
31
32 Point Plus(Point p1, Point p2, int p, int a)
33 {
34     int Lambda;
35     Point TmpPoint;
36     if (p1.x == p2.x && p1.y == p2.y)
37         Lambda = Mod((3 * p1.x * p1.x + a) * Inv[Mod(2 * p1.y, p)], p);
38     else
39         Lambda = Mod((p2.y - p1.y) * Inv[Mod(p2.x - p1.x, p)], p);
40     TmpPoint.x = Mod(Lambda * Lambda - p1.x - p2.x, p);
41     TmpPoint.y = Mod(Lambda * (p1.x - TmpPoint.x) - p1.y, p);
42     //printf("lambda=%d,x=%d,y=%d\n", Lambda, TmpPoint.x, TmpPoint.y);
43     return TmpPoint;
44 }
45
46 Point Minus(Point p1, Point p2, int p, int a)
47 {
48     Point Minus_p2(p2.x, Mod(p - p2.y, p));
```

```

49     return Plus(p1, Minus_p2, p, a);
50 }
51
52 Point Multiple(int k, Point p1, int p, int a)
53 {
54     Point TmpPoint(p1.x, p1.y);
55     for (int i = 1; i < k; i++)
56     {
57         TmpPoint = Plus(TmpPoint, p1, p, a);
58         //printf("(%d,%d)\n", TmpPoint.x, TmpPoint.y);
59     }
60     return TmpPoint;
61 }
62
63 Point Choose_G(int p, int a, int b)
64 {
65     printf("\nChoose a generator from below (input in the format of x y):\n");
66     vector<Point> G_list;
67     for (int i = 0; i < p; i++)
68         for (int j = 0; j < p; j++)
69             if (((i * i * i + a * i + b) % p) == ((j * j) % p))
70                 G_list.push_back(Point(i, j));
71     for (int i = 0; i < G_list.size(); i++)
72         printf("(%d,%d) ", G_list[i].x, G_list[i].y);
73     int X, Y;
74     printf("\nYou choose ");
75     scanf("%d%d", &X, &Y);
76     bool flag;
77     do
78     {
79         flag = false;
80         for (int i = 0; i < G_list.size(); i++)
81         {
82             if (G_list[i].x == X && G_list[i].y == Y)
83             {
84                 flag = true;
85             }
86         }
87         if (!flag)
88         {
89             printf("\nWrong! You should choose among the list above!");
90             printf("\nYou choose ");
91             scanf("%d%d", &X, &Y);
92         }
93     } while (!flag);
94     //printf("Success!\n");
95     return Point(X, Y);
96 }
97
98 void Diff(int &p, int &a, int &b)
99 {
100     printf("\n-----\n");
101     printf("ENCODE BEGIN:");
102     printf("\n-----\n");
103     printf("\nInput parameters of ECC( $y^2 = x^3 + ax + b \pmod p$ ):\n");
104     printf("a=");
105     scanf("%d", &a);
106     printf("b=");

```

```

107     scanf("%d", &b);
108     printf("p=");
109     scanf("%d", &p);
110
111     printf("\n");
112     Inverse(p);
113
114     Point G = Choose_G(p, a, b);
115
116     printf("\nYou have choosen ECC E_%d(%d,%d) and generator (%d,%d)\n", p, a,
b, G.x, G.y);
117
118     int n_A,n_B;
119
120     printf("\nInput Alice's private key n_A: ");
121     scanf("%d", &n_A);
122
123     Point P_A = Multiple(n_A, G, p, a);
124     printf("\nAlice's public key P_A is (%d,%d)\n", P_A.x, P_A.y);
125
126     printf("\nInput Bob's private key n_B: ");
127     scanf("%d", &n_B);
128
129     Point P_B = Multiple(n_B, G, p, a);
130     printf("\nBob's public key P_B is (%d,%d)\n", P_B.x, P_B.y);
131
132     Point K_A=Multiple(n_A,P_B,p,a);
133     printf("\nAlice get the key K=(%d,%d)",K_A.x,K_A.y);
134
135     Point K_B=Multiple(n_B,P_A,p,a);
136     printf("\nBob get the key K=(%d,%d)",K_B.x,K_B.y);
137
138     printf("\n\n");
139 }
140
141 int main()
142 {
143     int p,a,b;
144     Diff(p,a,b);
145 }

```

## 程序解释

算法描述：

- 取素数  $p \approx 2180$  和参数  $a, b$ ，则得椭圆曲线上的点及无穷远点构成Abel群  $E_p(a, b)$
- 取  $E_p(a, b)$  的某个生成元  $G = (x_1, y_1)$ ， $G$  的阶，即满足  $nG = O$  的最小正整数  $n$  很大。 $E_p(a, b)$  和  $G$  作为公开参数
- Alice 和 Bob 之间的密钥交换如下进行
  - Alice 随机选取保密的整数  $n_A < n$ ，计算  $P_A = n_A G$  并发给 Bob
  - Bob 随机选取秘密的  $n_B$  并计算  $P_B = n_B G$  发给 Alice
  - Alice 和 Bob 分别由  $K = n_A P_B$  和  $K = n_B P_A$  生成共享的密钥
$$K = n_A P_B = n_A (n_B G) = n_B (n_A G) = n_B P_A$$

简单模拟即可。

## 运行截图

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

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

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

Choose a generator from 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 chosen ECC  $E_{11}(1,6)$  and generator (2,7)

Input Alice's private key n_A: 7

Alice's public key P_A is (7,2)

Input Bob's private key n_B: 5

Bob's public key P_B is (3,6)

Alice get the key K=(10,9)
Bob get the key K=(10,9)
```

样例选取了椭圆曲线  $E_{11}(1,6)$ ，生成元  $g = (2,7)$

Alice 选取私钥为 7，Bob 选取私钥为 5

得到 Alice 公钥为 (7,2)，Bob 公钥为 (3,6)

两人经过密钥交换后都得到了，密钥 (10,9)

## AES

## 程序

```
1  #include <iomanip>
2  #include <iostream>
3  using namespace std;
4  unsigned char SBox[] = {
5      0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b,
6      0xfe, 0xd7, 0xab, 0x76,
7      0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf,
8      0x9c, 0xa4, 0x72, 0xc0,
9      0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1,
10     0x71, 0xd8, 0x31, 0x15,
11     0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2,
12     0xeb, 0x27, 0xb2, 0x75,
13     0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3,
14     0x29, 0xe3, 0x2f, 0x84,
15     0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39,
16     0x4a, 0x4c, 0x58, 0xcf,
```

```

11     0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f,
    0x50, 0x3c, 0x9f, 0xa8,
12     0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21,
    0x10, 0xff, 0xf3, 0xd2,
13     0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d,
    0x64, 0x5d, 0x19, 0x73,
14     0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14,
    0xde, 0x5e, 0x0b, 0xdb,
15     0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62,
    0x91, 0x95, 0xe4, 0x79,
16     0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea,
    0x65, 0x7a, 0xae, 0x08,
17     0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f,
    0x4b, 0xbd, 0x8b, 0x8a,
18     0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9,
    0x86, 0xc1, 0x1d, 0x9e,
19     0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9,
    0xce, 0x55, 0x28, 0xdf,
20     0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f,
    0xb0, 0x54, 0xbb, 0x16};
21 unsigned char InvSBox[256] = {
22     0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e,
    0x81, 0xf3, 0xd7, 0xfb,
23     0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44,
    0xc4, 0xde, 0xe9, 0xcb,
24     0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b,
    0x42, 0xfa, 0xc3, 0x4e,
25     0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49,
    0x6d, 0x8b, 0xd1, 0x25,
26     0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc,
    0x5d, 0x65, 0xb6, 0x92,
27     0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57,
    0xa7, 0x8d, 0x9d, 0x84,
28     0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05,
    0xb8, 0xb3, 0x45, 0x06,
29     0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03,
    0x01, 0x13, 0x8a, 0x6b,
30     0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce,
    0xf0, 0xb4, 0xe6, 0x73,
31     0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8,
    0x1c, 0x75, 0xdf, 0x6e,
32     0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e,
    0xaa, 0x18, 0xbe, 0x1b,
33     0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe,
    0x78, 0xcd, 0x5a, 0xf4,
34     0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59,
    0x27, 0x80, 0xec, 0x5f,
35     0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f,
    0x93, 0xc9, 0x9c, 0xef,
36     0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c,
    0x83, 0x53, 0x99, 0x61,
37     0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63,
    0x55, 0x21, 0x0c, 0x7d};
38
39 unsigned char Key[11][4][4];
40
41 void SubBytes(unsigned char Text[][4])
42 {

```

```

43     for (int r = 0; r < 4; r++)
44         for (int c = 0; c < 4; c++)
45             Text[r][c] = SBox[Text[r][c]];
46 }
47
48 void ShiftRows(unsigned char Text[][4])
49 {
50     unsigned char t[4];
51     for (int r = 1; r < 4; r++)
52     {
53         for (int c = 0; c < 4; c++)
54             t[c] = Text[r][(c + r) % 4];
55         for (int c = 0; c < 4; c++)
56             Text[r][c] = t[c];
57     }
58 }
59
60 unsigned char Multiply(unsigned char a, unsigned char b)
61 {
62     unsigned char bKey[4];
63     unsigned char res = 0;
64     bKey[0] = b;
65     for (int i = 1; i < 4; i++)
66     {
67         bKey[i] = bKey[i - 1] << 1;
68         if (bKey[i - 1] & 0x80)
69             bKey[i] ^= 0x1b;
70     }
71     for (int i = 0; i < 4; i++)
72         if ((a >> i) & 0x01)
73             res ^= bKey[i];
74     return res;
75 }
76
77 void MixColumns(unsigned char Text[][4])
78 {
79     unsigned char t[4];
80     for (int c = 0; c < 4; c++)
81     {
82         for (int r = 0; r < 4; r++)
83             t[r] = Text[r][c];
84         for (int r = 0; r < 4; r++)
85             Text[r][c] = Multiply(0x02, t[r]) ^ Multiply(0x03, t[(r + 1) % 4]) ^
Multiply(0x01, t[(r + 2) % 4]) ^ Multiply(0x01, t[(r + 3) % 4]);
86     }
87 }
88
89 void AddRoundKey(unsigned char Text[][4], unsigned char Key[][4])
90 {
91     for (int c = 0; c < 4; c++)
92         for (int r = 0; r < 4; r++)
93             Text[r][c] ^= Key[r][c];
94 }
95
96 void KeyExpansion(unsigned char *key, unsigned char Key[][4][4])
97 {
98     unsigned char rc[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
0x36};

```

```

99     for (int r = 0; r < 4; r++)
100         for (int c = 0; c < 4; c++)
101             Key[0][r][c] = key[r + c * 4];
102     for (int i = 1; i <= 10; i++)
103         for (int j = 0; j < 4; j++)
104             {
105                 unsigned char t[4];
106                 for (int r = 0; r < 4; r++)
107                     t[r] = j ? Key[i][r][j - 1] : Key[i - 1][r][3];
108                 if (j == 0)
109                     {
110                         unsigned char temp = t[0];
111                         for (int r = 0; r < 3; r++)
112                             t[r] = SBox[t[(r + 1) % 4]];
113                         t[3] = SBox[temp];
114                         t[0] ^= rc[i - 1];
115                     }
116                 for (int r = 0; r < 4; r++)
117                     Key[i][r][j] = Key[i - 1][r][j] ^ t[r];
118             }
119     }
120
121 void InvSubBytes(unsigned char Text[][4])
122 {
123     for (int r = 0; r < 4; r++)
124         for (int c = 0; c < 4; c++)
125             Text[r][c] = InvSBox[Text[r][c]];
126 }
127
128 void InvShiftRows(unsigned char Text[][4])
129 {
130     unsigned char t[4];
131     for (int r = 1; r < 4; r++)
132     {
133         for (int c = 0; c < 4; c++)
134             t[c] = Text[r][(c - r + 4) % 4];
135         for (int c = 0; c < 4; c++)
136             Text[r][c] = t[c];
137     }
138 }
139
140 void InvMixColumns(unsigned char Text[][4])
141 {
142     unsigned char t[4];
143     for (int c = 0; c < 4; c++)
144     {
145         for (int r = 0; r < 4; r++)
146             t[r] = Text[r][c];
147         for (int r = 0; r < 4; r++)
148             Text[r][c] = Multiply(0x0e, t[r]) ^ Multiply(0x0b, t[(r + 1) % 4]) ^
Multiply(0x0d, t[(r + 2) % 4]) ^ Multiply(0x09, t[(r + 3) % 4]);
149     }
150 }
151
152 unsigned char *Cipher(unsigned char *Input)
153 {
154     unsigned char Text[4][4];
155

```

```

156     for (int r = 0; r < 4; r++)
157         for (int c = 0; c < 4; c++)
158             Text[r][c] = Input[c * 4 + r];
159
160     AddRoundKey(Text, Key[0]);
161
162     for (int i = 1; i <= 10; i++)
163     {
164         SubBytes(Text);
165         ShiftRows(Text);
166         if (i != 10)
167             MixColumns(Text);
168         AddRoundKey(Text, Key[i]);
169     }
170
171     for (int r = 0; r < 4; r++)
172         for (int c = 0; c < 4; c++)
173             Input[c * 4 + r] = Text[r][c];
174
175     return Input;
176 }
177
178 unsigned char *InvCipher(unsigned char *Input)
179 {
180     unsigned char Text[4][4];
181
182     for (int r = 0; r < 4; r++)
183         for (int c = 0; c < 4; c++)
184             Text[r][c] = Input[c * 4 + r];
185
186     AddRoundKey(Text, Key[10]);
187     for (int i = 9; i >= 0; i--)
188     {
189         InvShiftRows(Text);
190         InvSubBytes(Text);
191         AddRoundKey(Text, Key[i]);
192         if (i)
193             InvMixColumns(Text);
194     }
195
196     for (int r = 0; r < 4; r++)
197         for (int c = 0; c < 4; c++)
198             Input[c * 4 + r] = Text[r][c];
199     return Input;
200 }
201
202 int main()
203 {
204     unsigned char Data[100001][4][4];
205     string str;
206
207     cout << "Input the initial key: ";
208     cin >> str;
209     if (str.length() != 16)
210     {
211         cout << "\nWrong! Length of the key should be 16!\n\nInput the initial
key: ";
212         cin >> str;

```



```

213     }
214
215     for (int r = 0; r < 4; r++)
216         for (int c = 0; c < 4; c++)
217             Key[0][r][c] = str[r * 4 + c];
218
219     cout << "\nInput your text: ";
220     cin >> str;
221
222     cout << "\nYour plain text in HEX is \n";
223     int tmp = (str.length() % 16 == 0 ? 0 : 16 - str.length() % 16);
224     for (int i = 1; i <= tmp; i++)
225         str += ((char)(0));
226
227     tmp = 0;
228     for (int k = 1; k <= str.length() / 16; k++)
229     {
230         for (int r = 0; r < 4; r++)
231             for (int c = 0; c < 4; c++)
232             {
233                 Data[k][r][c] = str[tmp];
234                 tmp++;
235             }
236         for (int r = 0; r < 4; r++)
237         {
238             for (int c = 0; c < 4; c++)
239             {
240                 cout << hex << setw(2) << setfill('0') << (int)(Data[k][r][c])
241                 << ' ';
242             }
243             cout << endl;
244         }
245         cout << "\nThe cipher text in HEX is \n";
246         for (int k = 1; k <= str.length() / 16; k++)
247         {
248             Cipher(*Data[k]);
249             for (int r = 0; r < 4; r++)
250             {
251                 for (int c = 0; c < 4; c++)
252                 {
253                     cout << hex << setw(2) << setfill('0') << (int)(Data[k][r][c])
254                     << ' ';
255                 }
256                 cout << endl;
257             }
258             cout << "\nThe plain text in HEX is \n";
259             for (int k = 1; k <= str.length() / 16; k++)
260             {
261                 InvCipher(*Data[k]);
262                 for (int r = 0; r < 4; r++)
263                 {
264                     for (int c = 0; c < 4; c++)
265                     {
266                         cout << hex << setw(2) << setfill('0') << (int)(Data[k][r][c])
267                         << ' ';

```

```
268         cout << endl;
269     }
270 }
271 }
```

## 程序解释

```
1 unsigned char SBox[] = {
2     0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, ...};
3 unsigned char InvSBox[256] = {
4     0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, ...};
```

S 盒和逆 S 盒是给定的

```
1 unsigned char Multiply(unsigned char, unsigned char)
```

用来计算 GF 域下的乘法

对应过程	加密函数	解密函数
SubBytes	void SubBytes(unsigned char)	void InvSubBytes(unsigned char)
ShiftRows	void ShiftRows(unsigned char)	void InvShiftRows(unsigned char)
MixColumns	void MixColumns(unsigned char)	void InvMixColumns(unsigned char)
AddRoundKey	void AddRoundKey(unsigned char, unsigned char)	无
Encode&Decode	unsigned char *Cipher(unsigned char *)	unsigned char *InvCipher(unsigned char *)

## 运行截图

```

PS E:\20-21-2\密码学\实验\AES> gcc AES.cpp -o AES -lstdc++
PS E:\20-21-2\密码学\实验\AES> ./AES
Input the initial key: s;dkf;k

Wrong! Length of the key should be 16!

Input the initial key: s;dkf;k,>e9==~w`

Input your text: '56ijh0-5o-rp2[qkg'N0%VGPrTLB;67[jle

Your plain text in HEX is
27 35 36 69
6a 68 30 2d
35 6f 2d 72
70 32 5b 71
6b 67 27 4e
4f 25 59 47
50 72 54 4c
42 3b 36 37
5b 6a 6c 65
00 00 00 00
00 00 00 00
00 00 00 00

The cipher text in HEX is
c0 40 8f 73
fa c3 94 03
36 ea 0b 55
15 7c 0b 8a
81 bb da 18
27 98 ae 0a
2a c6 59 7f
4e 2d 8d 3e
e1 15 61 d0
9f ea 4c ab
16 64 3b bb
67 b4 e6 97

The plain text in HEX is
27 35 36 69
6a 68 30 2d
35 6f 2d 72
70 32 5b 71
6b 67 27 4e
4f 25 59 47
50 72 54 4c
42 3b 36 37
5b 6a 6c 65
00 00 00 00
00 00 00 00
00 00 00 00

```

- 密钥为 16 字节（128 位）
- 要加密的数据理论上可以无限长，文本可以为任意类型
- 16 字节一组进行加密，采用 ECB 模式，不足位补 0x00
- 输出采用 16 进制

## exgcd

### 程序

```

1  #include <cstdio>
2  using namespace std;
3
4  void ex_gcd(long long a, long long b, long long &x, long long &y)
5  {
6      if (b == 0)
7      {
8          x = 1;

```

```

9         y = 0;
10        return;
11    }
12    ex_gcd(b, a % b, x, y);
13    long long t = x;
14    x = y;
15    y = t - a / b * y;
16    return;
17 }
18
19 long long gcd(long long a, long long b)
20 {
21     if (a % b == 0)
22         return b;
23     return gcd(b, a % b);
24 }
25
26 long long Inverse(long long a, long long p)
27 {
28     long long x, y;
29     ex_gcd(a, p, x, y);
30     x = x + (x < 0 ? p : 0);
31     return x;
32 }
33
34 int main()
35 {
36     printf("Input 2 integers: ");
37     long long x, y, ans;
38     scanf("%lld%lld", &x, &y);
39     ans = gcd(x, y);
40     printf("GCD is %lld\n\n", ans);
41
42     printf("Input an integer and mod: ");
43     scanf("%lld%lld", &x, &y);
44     ans = Inverse(x, y);
45     printf("Inverse is %lld\n", ans);
46 }

```

## 程序解释

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

是扩展欧几里得算法，计算结果得到  $ax + by = 1$

```
1 long long gcd(long long a, long long b)
```

为求最大公约数

```
1 long long Inverse(long long a, long long p)
```

为利用扩展欧几里得算法求模运算的逆

## 运行截图

---

```
PS E:\20-21-2\密码学\实验\exgcd> gcc exgcd.cpp -o exgcd -lstdc++
PS E:\20-21-2\密码学\实验\exgcd> ./exgcd
Input 2 integers: 5436721698 3284638
GCD is 2

Input an integer and mod: 5435432 436465437547
Inverse is 143006431539
```

- 随机选取 5436721698 和 3284638，得到最大公约数为 2
- 随机选取  $a = 5435432, p = 436465437547$ ，计算得到  $a^{-1} \equiv 143006431539 \pmod{p}$