

《信息安全基础综合实验》课程实验报告

实验题目：基于中国剩余定理的秘密共享方案

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：使用中国剩余定理完成秘密共享

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：秘密共享是将秘密以适当的方式拆分，拆分后的每一个子秘密由不同的参与者管理，单个参与者无法恢复秘密信息，只有若干个参与者一同协作才能恢复秘密消息。并且，当其中某些参与者出问题，秘密仍可以恢复

(t, n) 门限秘密共享方案

将秘密 k 分成 n 个子秘密 k_1, k_2, \dots, k_n ，满足下面两个条件：

- (1) 如果已知任意 t 个 k_i 值，易于恢复出 k ；
- (2) 如果已知任意 $t - 1$ 个或更少个 k_i 值，不能恢复出 k 。

将一个密钥分成 n 份，那么 n 个人中至少 t 人在场才能获得密钥。

1983 年, Asmuth 和 Bloom 提出基于中国剩余定理的(t,n)门限秘密共享方案

原理:

高度概括原理就是使用同余方程进行加密, 使用中国剩余定理来解密

(t,n)门限, 一个秘密k, 被分割成n个子秘密(d_i, k_i)

对于某个秘密 k, 计算
$$\begin{cases} k_1 \equiv k \pmod{d_1} \\ k_2 \equiv k \pmod{d_2} \\ \vdots \\ k_n \equiv k \pmod{d_n} \end{cases}$$
 则子秘密为(d_i, k_i)。

要求一: 选择n个整数d₁, d₂, ..., d_n, 满足

(1) d₁ < d₂ < ... < d_n; d_i严格递增

(2) (d_i, d_j) = 1, i ≠ j; d_i两两互素

(3) N = d₁ × d₂ × ... × d_t M = d_{n-t+2} × d_{n-t+3} × ... × d_n, 有N > M

t个最小的d_i的乘积严格大于t-1个最大的d_i的乘积

要求二: N > k > M

n 个子秘密中任意选择 t 个, (k_{i₁}, d_{i₁}), (k_{i₂}, d_{i₂}), ..., (k_{i_t}, d_{i_t}), 恢复出秘密 k

计算
$$\begin{cases} x \equiv k_{i_1} \pmod{d_{i_1}} \\ x \equiv k_{i_2} \pmod{d_{i_2}} \\ \vdots \\ x \equiv k_{i_t} \pmod{d_{i_t}} \end{cases}$$
 恢复出秘密 $x \equiv k \pmod{N_1}$, N₁ = d_{i₁} d_{i₂} ... d_{i_t}。

t 个子秘密能恢复出秘密

t-1 个子秘密不能



没有足够的信息去确定k

任意选择 t-1 个子秘密:

(k_{j₁}, d_{j₁}), (k_{j₂}, d_{j₂}), ..., (k_{j_{t-1}}, d_{j_{t-1}})

$x \equiv k \pmod{M_1}$, M₁ = d_{j₁} d_{j₂} ... d_{j_{t-1}}

$N_1 > N > k > M > M_1$

三、方案实现

```

1  #include"miracl.h"
2  #include<stdio.h>
3  #include<stdlib.h>
4  #include<time.h>
5
6  #define T 3
7  #define S 5
8  #define RCV 3
9  #define OFFSET 7
10 #define BITS 500
11
12 int main(void)
13 {
14     miracl *mip = mirsys(1000, 16);
15
16     int i = 0, j = 0, seed[RCV]; // 初始化
17     big K, k[S], d[S], N, M, v;
18     big D[S], D_1[S], n, x;
19
20     K = mirvar(0); N = mirvar(1); M = mirvar(1);
21     v = mirvar(1); n = mirvar(1); x = mirvar(0);
22     for (i = 0; i < S; i++) {
23         k[i] = mirvar(0); d[i] = mirvar(0);
24         D_1[i] = mirvar(0); D[i] = mirvar(0);
25     }
26

```

这部分代码的主要作用是导入 miracl 库，初始化变量。这里我们将秘密分成 5 份，t 设置成 3。

```

27     irand(time(NULL));
28     puts("secret K="); // generate K
29     bigbits(BITS, K);
30     cotnum(K, stdout);
31
32     expb2(BITS / T + OFFSET, v); // = 2^(BITS / T + OFFSET)
33     incr(v, 1, v); // = v+1 // find d[i]
34     for (i = 0; i < S; i++) {
35         while (1) {
36             incr(v, 2, v); // 保证 di 严格递增
37             if (isprime(v)) {
38                 printf("d[%d]= ", i);
39                 cotnum(v, stdout);
40                 copy(v, d[i]);
41                 break;
42             }
43         }
44     }
45

```

这里我们使用 bigbits 函数随机生成需要加密的秘密，然后我们开始生成 d。

这里我们采用了取巧的方法，我们先随机生成一个素数为 d1，然后在 d1 的基础上不停加 2，然后判断这个数是否为素数，如果是素数，就为 d2，依次往后。

这样可以保证 d 之间是严格递增，且互素的。但是后来我们也发现了一些问

题，采用这种方法，会减少 d 的取值范围。

```

46 puts("N= ");
47 for (i = 0; i < T; i++) {
48     fft_mult(d[i], N, N); //N = d1 * d2 * d3 * d4 ... dt
49 } cotnum(N, stdout);
50 puts("M= ");
51 for (i = 1; i < T; i++) {
52     fft_mult(d[S - i], M, M);
53 } cotnum(M, stdout);
54
55 for (i = 0; i < S; i++) { //compute k[i]
56     copy(K, k[i]);
57     divide(k[i], d[i], d[i]); //ki = ki mod di
58     printf("k[%d]= ", i);
59     cotnum(k[i], stdout);
60 }
61
62 srand(time(NULL)); //generate RCV random differente numbers
63 for (i = 0; i < RCV; i++) { //choose num(RCV) random
64     seed[i] = rand() % S;
65     do {
66         for (j = 0; j < i; j++) {
67             if (seed[i] == seed[j]) break;
68         }
69         if (i != j) {
70             seed[i] = rand() % S;
71         }
72         else {
73             break;
74         }
75     } while (1);
76 }
77
78 // chinese remainder theory recover K
79 for (i = 0; i < RCV; i++) {
80     fft_mult(n, d[seed[i]], n);
81 }
82 printf("\nchoose %d k:\n", RCV);

```

加密过程，先计算 N ，然后计算 $k[i]$ ，最后开始加密

```

82 printf("\nchoose %d k:\n", RCV);
83 for (i = 0; i < RCV; i++) {
84     printf("k[%d]= ", seed[i]);
85     cotnum(k[seed[i]], stdout);
86     copy(n, v);
87     divide(v, d[seed[i]], D[seed[i]]);
88     invmodp(D[seed[i]], d[seed[i]], D_1[seed[i]]);
89     fft_mult(k[seed[i]], D[seed[i]], d[seed[i]]); //k[seed[i]]*D[seed[i]]=d[seed[i]]
90     fft_mult(d[seed[i]], D_1[seed[i]], d[seed[i]]);
91     divide(d[seed[i]], n, n);
92     add(d[seed[i]], x, x);
93 }
94 divide(x, n, n);
95
96 putchar('\n');
97 printf("x= "); cotnum(x, stdout);
98 printf("K= "); cotnum(K, stdout);
99 !mr_compare(x, K) ? puts("x == K") : puts("x != K");
100
101 mirexit();
102 system("pause");
103 return 0;
104 }

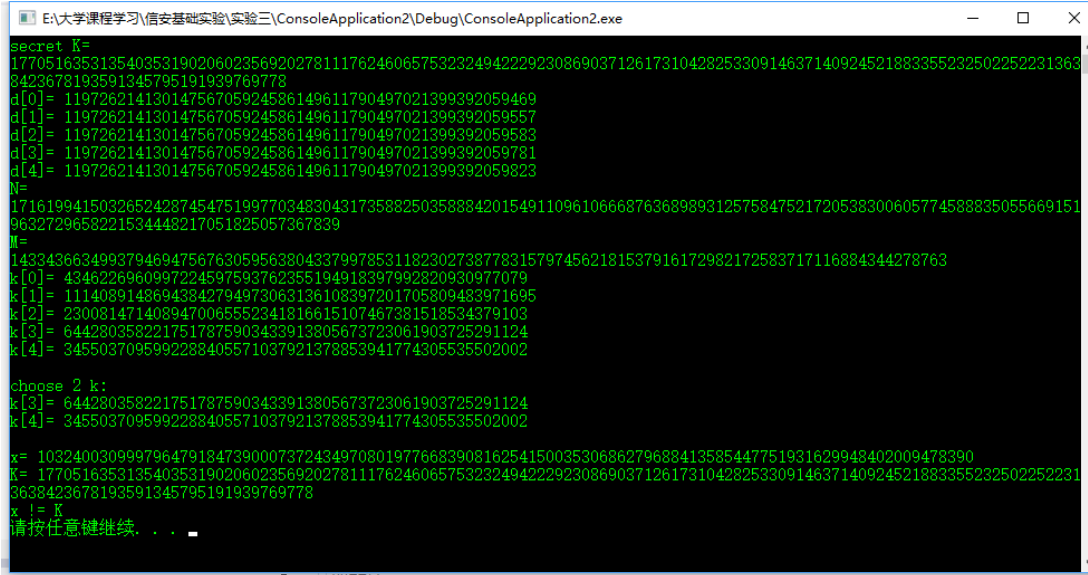
```

解密过程

四、数据分析

(包括算法测试数据的分析等等)

1.当 RCV 的值设置为 2 时 (小于门限 3):



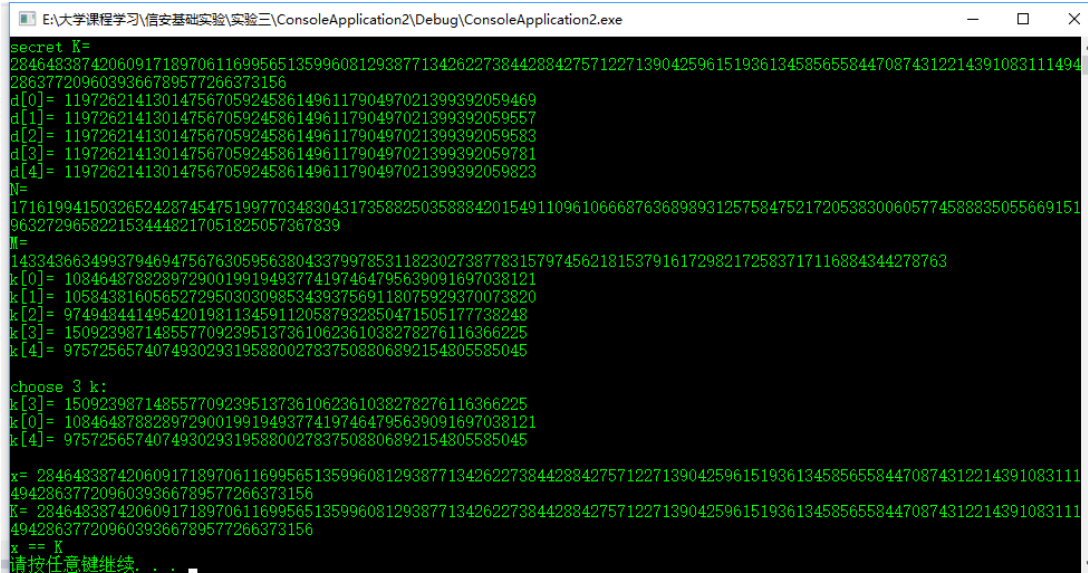
```
secret K=
177051635313540353190206023569202781117624606575323249422292308690371261731042825330914637140924521883355232502252231363
8423678193591345795191939769778
d[0]= 11972621413014756705924586149611790497021399392059469
d[1]= 11972621413014756705924586149611790497021399392059557
d[2]= 11972621413014756705924586149611790497021399392059583
d[3]= 11972621413014756705924586149611790497021399392059781
d[4]= 11972621413014756705924586149611790497021399392059823
N=
171619941503265242874547519977034830431735882503588842015491109610666876368989312575847521720538300605774588835055669151
9632729658221534448217051825057367839
M=
143343663499379469475676305956380433799785311823027387783157974562181537916172982172583717116884344278763
k[0]= 4346226960997224597593762355194918397992820930977079
k[1]= 11140891486943842794973063136108397201705809483971695
k[2]= 2300814714089470065552341816615107467381518534379103
k[3]= 6442803582217517875903433913805673723061903725291124
k[4]= 3455037095992288405571037921378853941774305535502002

choose 2 k:
k[3]= 6442803582217517875903433913805673723061903725291124
k[4]= 3455037095992288405571037921378853941774305535502002

x= 103240030999796479184739000737243497080197766839081625415003530686279688413585447751931629948402009478390
K= 177051635313540353190206023569202781117624606575323249422292308690371261731042825330914637140924521883355232502252231
3638423678193591345795191939769778
x != K
请按任意键继续. . .
```

发现不能成功解密出正确的消息。

2.当 RCV 的值设置为 3 时:



```
secret K=
284648387420609171897061169956513599608129387713426227384428842757122713904259615193613458565584470874312214391083111494
2863772096039366789577266373156
d[0]= 11972621413014756705924586149611790497021399392059469
d[1]= 11972621413014756705924586149611790497021399392059557
d[2]= 11972621413014756705924586149611790497021399392059583
d[3]= 11972621413014756705924586149611790497021399392059781
d[4]= 11972621413014756705924586149611790497021399392059823
N=
171619941503265242874547519977034830431735882503588842015491109610666876368989312575847521720538300605774588835055669151
9632729658221534448217051825057367839
M=
143343663499379469475676305956380433799785311823027387783157974562181537916172982172583717116884344278763
k[0]= 10846487882897290019919493774197464795639091697038121
k[1]= 10584381605652729503030985343937569118075929370073820
k[2]= 9749484414954201981134591120587932850471505177738248
k[3]= 1509239871485577092395137361062361038278276116366225
k[4]= 9757256574074930293195880027837508806892154805585045

choose 3 k:
k[3]= 1509239871485577092395137361062361038278276116366225
k[0]= 10846487882897290019919493774197464795639091697038121
k[4]= 9757256574074930293195880027837508806892154805585045

x= 284648387420609171897061169956513599608129387713426227384428842757122713904259615193613458565584470874312214391083111
4942863772096039366789577266373156
K= 284648387420609171897061169956513599608129387713426227384428842757122713904259615193613458565584470874312214391083111
4942863772096039366789577266373156
x == K
请按任意键继续. . .
```

正确解密出了消息

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)

整个实验过程较为顺利,并没有出现较多的问题,主要是有上次中国剩余定理实验的积累,解密的过程并没有花费太多的时间。通过这个实验,我们也了解到了中国剩余定理在加密领域的应用