公钥密码学数学基础第一次实验报告

周家熠、王煜涵、潘子豪、刘一童

2024 年 10 月 1 日

目录

1 摘要

本次实验为公钥密码学数学基础实验课第一次实验，由周家熠、王煜涵、潘子豪、刘一童小组完成。实验内容为用 NTL 以及 sagemath 完成较为基础的数学问题。于 9 月 28 日完成实验，29 日撰写实验报告。小组成员任务分别为：周家熠、王煜涵分别负责完成 NTL 部分和 sagemath 部分内容；潘子豪负责代码整理以及优化；刘一童负责撰写实验报告。本文作者刘一童学号：202300460117，所属班级为 2023 级密码二班。本实验报告为用 overleaf 所含的 LaTeX 在线编译工具完成。项目链接：https://www.overleaf.com/read/srzckvhckghg6f989d

2 实验内容

2.1 题目：由 sagemath 以及 NTL 完成以下内容

- 选取两个随机的 1024 比特的素数 p, q
- 计算二者乘积 $N = pq$，测量所用时间
- 选取参数 e=65537，测试是否满足 $(e,(p\text{-}1)(q\text{-}1))$=1，不满足重新选取 e，如满足则计算

$$ed + x(p-1)(q-1) = 1$$

并测量所用时间

2.2 实验过程

本次实验由两部本构成，下面会将实验具体内容以及相关代码附上。

2.2.1 NTL 部分

下面是完成 NTL 部分所需的全部代码

```
#include<iostream>
#include<NTL/ZZ.h>
#include<NTL/RR.h>
#include<chrono>
using namespace std;
using namespace NTL;
using namespace std::chrono;
int main()
{
ZZ p = GenPrime_ZZ(1024);
ZZ q = GenPrime_ZZ(1024);
auto start1 = high_resolution_clock::now();
ZZ N = operator*(p, q);
auto end1 = high_resolution_clock::now();
auto duration1 = duration_cast<microseconds>(end1 - start1);
double duration_time1 = duration1.count() / 1000000.0;
cout << "N = " << N << endl;
cout << "计算乘积所需时间：" << duration_time1 << "s" << endl;
auto start2 = high_resolution_clock::now();
ZZ e;e = 65537;
ZZ gcd = GCD(e, operator*(operator-(p,1),operator-(q,1)));
while (gcd != 1)
{
e = operator+
(RandomBnd(operator-(operator*(operator-(p, 1), operator-(q, 1)), 2)),2);
```

```
gcd = GCD(e, operator*(operator-(p, 1), operator-(q, 1)));
}
ZZ d = InvMod(e, operator*(operator-(p, 1), operator-(q, 1)));
auto end2 = high_resolution_clock::now();
auto duration2 = duration_cast<microseconds>(end2 - start2);
double duration_time2 = duration2.count() / 1000000.0;
cout << "d = " << d << endl;
cout << "选取e和计算d所需时间: " << duration_time2 << "s" << endl;
return 0;
}
```

(1) 选取两个随机的 1024 比特的素数 p, q;

```
ZZ p = GenPrime_ZZ(1024);
ZZ q = GenPrime_ZZ(1024);
```

(2) 计算二者的乘积 $N = pq$, 测量所用的时间;

```
auto start1 = high_resolution_clock::now();
ZZ N = operator*(p, q);
auto end1 = high_resolution_clock::now();
auto duration1 = duration_cast<microseconds>(end1 - start1);
double duration_time1 = duration1.count() / 1000000.0;
cout << "N = " << N << endl;
cout << "计算乘积所需时间: " << duration_time1 << "s" << endl;
```

这里用到了标准 C++11 版本中的计时器函数，对 $N = pq$ 这一运算过程进行微秒级别计时操作并输出结果，其头文件为

```
#include<chrono>
```

(3) 选取参数 e=65537，测试是否满足 (e,(p-1)(q-1))=1，不满足重新选取 e，如满足则计算

$$ed + x(p-1)(q-1) = 1$$

并测量所用时间

首先对给出的条件进行分析 $(e,(p-1)(q-1)) = 1$ 即为要保证 e 与 $(p-1)(q-1)$ 互素，这要用到 NTL 中关于求最大公因子的函数

```
ZZ gcd = GCD(e, operator*(operator-(p,1),operator-(q,1)));
```

并根据第一个限制条件，用 **while** 循环对 e 进行讨论取值

```
while (gcd != 1)
{
e = operator+
(RandomBnd(operator-(operator*(operator-(p, 1), operator-(q, 1)), 2)),2);
gcd = GCD(e, operator*(operator-(p, 1), operator-(q, 1)));
}
```

对于 e 不满足的情况则再循环进行取值直到满足限制条件为止。

而 $ed + x(p-1)(q-1) = 1$ 表示的意思为 d 对模 $(p-1)(q-1)$ 的逆，即为

$$ed \equiv 1(mod(p-1)(q-1))$$

那么就会用到 NTL 中的取模逆的相关函数

```
ZZ d = InvMod(e, operator*(operator-(p, 1), operator-(q, 1)));
```

下面是用 NTL 完成实验的实验结果，为了确保真实性，我们进行了三次

图 1: 第一次

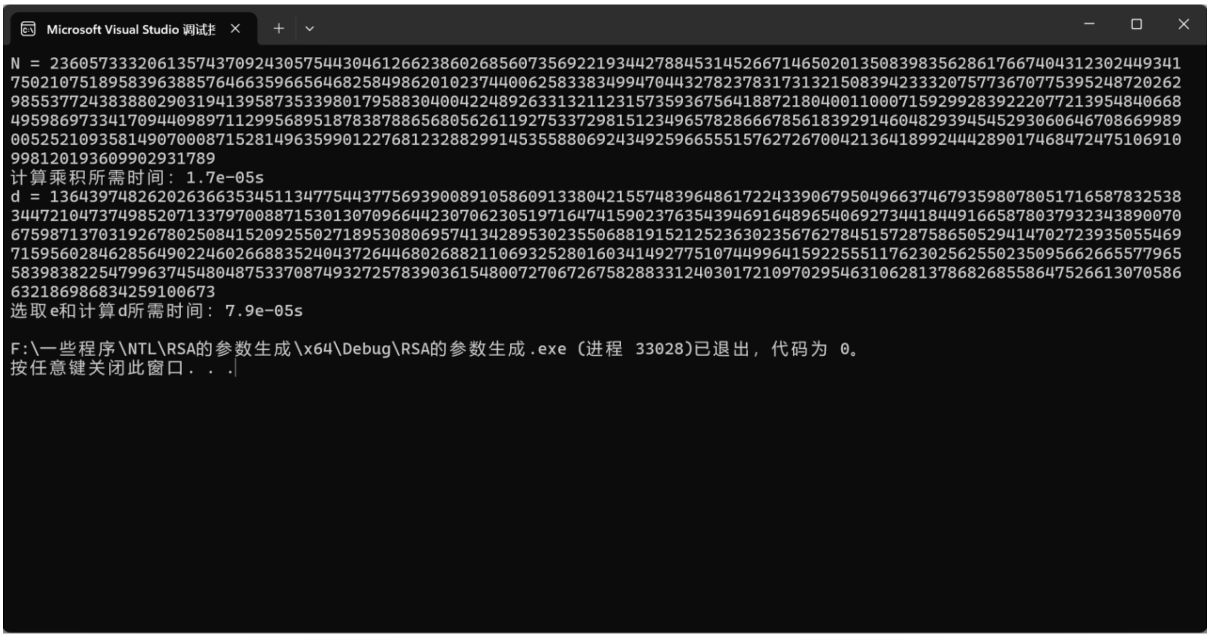

图 2: 第二次

图 3: 第三次

2.2.2 sagemath 部分

以下为 sagemath 部分的全部代码

```
#!/usr/bin/env python
# coding: utf-8
import time
p = random_prime(2^1024,2^1023)
q = random_prime(2^1024,2^1023)
start_time = time.time()
N = p * q
end_time = time.time()
end_time - start_time
n = (p - 1) * (q - 1)
e = 65537
while e > 0:
    if gcd(e,n) == 1:
        time_1 = time.time()
        d = inverse_mod(e,n)
        time_2 = time.time()
        print(d)
        print(time_2 - time_1)
        break
    else:
        e = randint(2,(p-1)(q-1))
```

与 NTL 同样，用 sagemath 完成三个问题也是寄托于 sagemath 内嵌的三个函数，下面是用 sagemath 所完成的实验结果

```
In [10]: import time

In [11]: p = random_prime(2^1024,2^1023)
         q = random_prime(2^1024,2^1023)

In [12]: start_time = time.time()

In [13]: N = p * q

In [14]: end_time = time.time()

In [15]: end_time - start_time

Out[15]: 0.027230024337768555

In [16]: n = (p - 1) * (q - 1)

In [17]: e = 65537

In [18]: while e > 0:
             if gcd(e,n) == 1:
                 time_1 = time.time()
                 d = inverse_mod(e,n)
                 time_2 = time.time()
                 print(d)
                 print(time_2 - time_1)
                 break
             else:
                 e = randint(2, (p-1)(q-1))

         4570317461348397144027239505961020153754853693009891839626030347475962488240057962070363077604020574866224119002295946558324846443558110727l
         763892983511117915288990206540764451388141886140076799448571787126159194309319859928377245354057925741127246851094054718177993310512659265 89
         342889338054600819912276589799739663264319422065957422897952785433049743539136386906261466818197573416579586384677256327152084692660298183 49
         666756836149807748656785304779326194455023968224045340687816299067522300416202803775381073500258250104596395415881289453223905598561786086 64
         1220702251774897066068955655557861936401053588758892033
         1.1920928955078125e-05
```

图 4: sagemath 实验结果 1

```
In [1]: import time

In [2]: p = random_prime(2^1024,2^1023)
        q = random_prime(2^1024,2^1023)

In [3]: start_time = time.time()

In [4]: N = p * q

In [5]: end_time = time.time()

In [6]: end_time - start_time

Out[6]: 10.11940860748291

In [7]: n = (p - 1) * (q - 1)

In [8]: e = 65537

In [9]: while e > 0:
            if gcd(e,n) == 1:
                time_1 = time.time()
                d = inverse_mod(e,n)
                time_2 = time.time()
                print(d)
                print(time_2 - time_1)
                break
            else:
                e = randint(2, (p-1)(q-1))

        754194905060579572363100104458508758233597564391030505070753455600846836159679831975160141309919812269703790151850964708360475
        705639260703835486372696691832945728455799508234688441516643281768358578847994484341043927860080467479360732796720355393393919
        752706808226692425456212441969882593210665622634459663675848171863088900302502445023206496887817149222707825938592203210558413
        345487997494709806299825966820771866435125364733916104996492840538638377262226307501298117444183891049616231625007712769451725
        4523236401635385138051700052631804937397354601586453660721520724672188529530907905879058327910368472373944419 3
        1.1682510375976562e-05

In [ ]: |
```

图 5: sagemath 实验结果 2

```
In [1]: import time

In [2]: p = random_prime(2^1024,2^1023)
        q = random_prime(2^1024,2^1023)

In [3]: start_time = time.time()

In [4]: N = p * q

In [5]: end_time = time.time()

In [6]: end_time - start_time

Out[6]: 7.339306592941284

In [7]: n = (p - 1) * (q - 1)

In [8]: e = 65537

In [9]: while e > 0:
            if gcd(e,n) == 1:
                time_1 = time.time()
                d = inverse_mod(e,n)
                time_2 = time.time()
                print(d)
                print(time_2 - time_1)
                break
            else:
                e = randint(2,(p-1)(q-1))

        322298154182836718166939523707830730199058617950023531772773372248215536784485532859555808608356280952971745019859600809196454
        633284101190978124784584782153175928253633934842610463628466290410422076917358176618365588366703888410233127016955828232457059
        029308615971241613488403046304568476149271404636644880132052867156998202678427433193880751973523435122774373532351560802419727
        905214526282737383135120531863610909174579370217825335007420369830169376115946531628886465628092230552274266404000822889897690
        70358430587946617812734982609653046140570047516733666905326384288705886693618951935938047880258320553741243005
        1.4066696166992188e-05
```

图 6: sagemath 实验结果 3

3 实验小结

本次实验主要还是为了了解并熟练运用 NTL 以及 sagemath 这两个数学工具，而实验所选题目也与课堂内容相关联。实验难度不大，但是因为第一次使用这两个数学工具，所以实验的完成速度并不快。