



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	2
2	实验内容	3
2.1	题目：由 sagemath 以及 NTL 完成以下内容	3
2.2	实验过程	3
2.2.1	NTL 部分	3
2.2.2	sagemath 部分	6
3	实验小结	8

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 $(e, (p-1)(q-1))=1$ ，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

2.2 实验过程

本次实验由两部分构成，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

```
1 #include<iostream>
2 #include<NTL/ZZ.h>
3 #include<NTL/RR.h>
4 #include<chrono>
5 using namespace std;
6 using namespace NTL;
7 using namespace std::chrono;
8 int main()
9 {
10 ZZ p = GenPrime_ZZ(1024);
11 ZZ q = GenPrime_ZZ(1024);
12 auto start1 = high_resolution_clock::now();
13 ZZ N = operator*(p, q);
14 auto end1 = high_resolution_clock::now();
15 auto duration1 = duration_cast<microseconds>(end1 - start1);
16 double duration_time1 = duration1.count() / 1000000.0;
17 cout << "N = " << N << endl;
18 cout << "计算乘积所需时间：" << duration_time1 << "s" << endl;
19 auto start2 = high_resolution_clock::now();
20 ZZ e; e = 65537;
21 ZZ gcd = GCD(e, operator*(operator-(p, 1), operator-(q, 1)));
22 while (gcd != 1)
23 {
24 e = operator+
25 (RandomBnd(operator-(operator*(operator-(p, 1), operator-(q, 1)), 2)), 2);
```

```
26 gcd = GCD(e, operator*(operator-(p, 1), operator-(q, 1)));
27 }
28 ZZ d = InvMod(e, operator*(operator-(p, 1), operator-(q, 1)));
29 auto end2 = high_resolution_clock::now();
30 auto duration2 = duration_cast<microseconds>(end2 - start2);
31 double duration_time2 = duration2.count() / 1000000.0;
32 cout << "d = " << d << endl;
33 cout << "选取e和计算d所需时间: " << duration_time2 << "s" << endl;
34 return 0;
35 }
```

(1) 选取两个随机的 1024 比特的素数 p, q ;

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

(2) 计算二者的乘积 $N = pq$, 测量所用的时间;

```
1 auto start1 = high_resolution_clock::now();
2 ZZ N = operator*(p, q);
3 auto end1 = high_resolution_clock::now();
4 auto duration1 = duration_cast<microseconds>(end1 - start1);
5 double duration_time1 = duration1.count() / 1000000.0;
6 cout << "N = " << N << endl;
7 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

```
1 #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

```
1 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

```
1 while (gcd != 1)
2 {
3   e = operator+
4   (RandomBnd(operator-(operator*(operator-(p, 1), operator-(q, 1)), 2)), 2);
5   gcd = GCD(e, operator*(operator-(p, 1), operator-(q, 1)));
6 }
```

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。

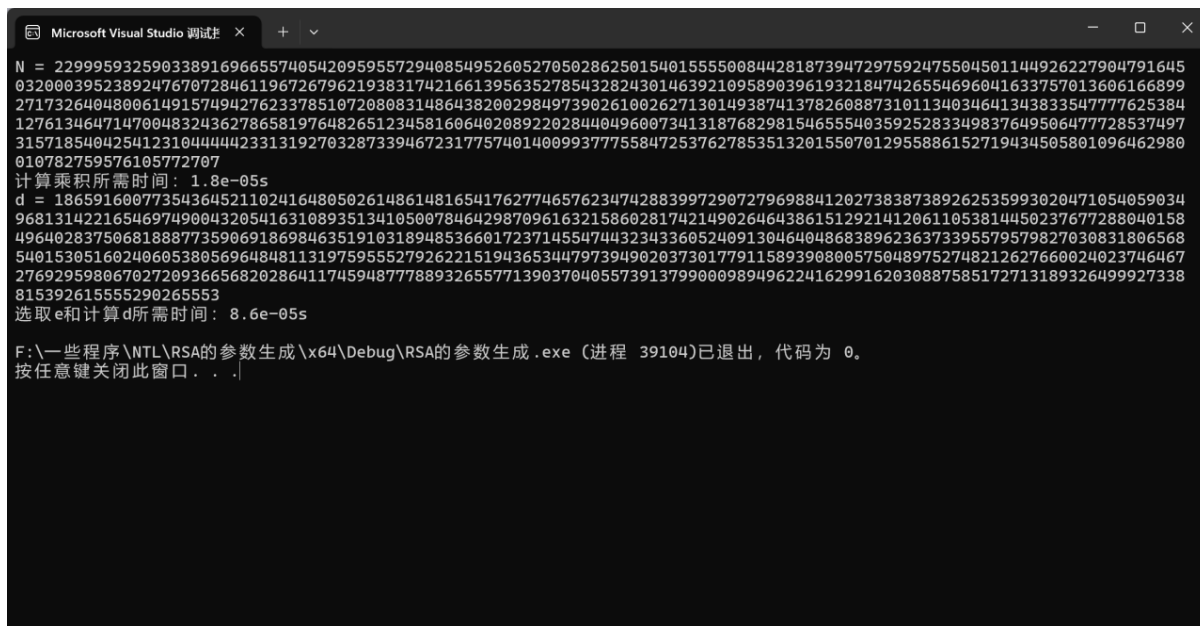
而 $ed + x(p-1)(q-1) = 1$ 表示的意思为 d 对模 $(p-1)(q-1)$ 的逆，即为

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

那么就会用到 NTL 中的取模逆的相关函数

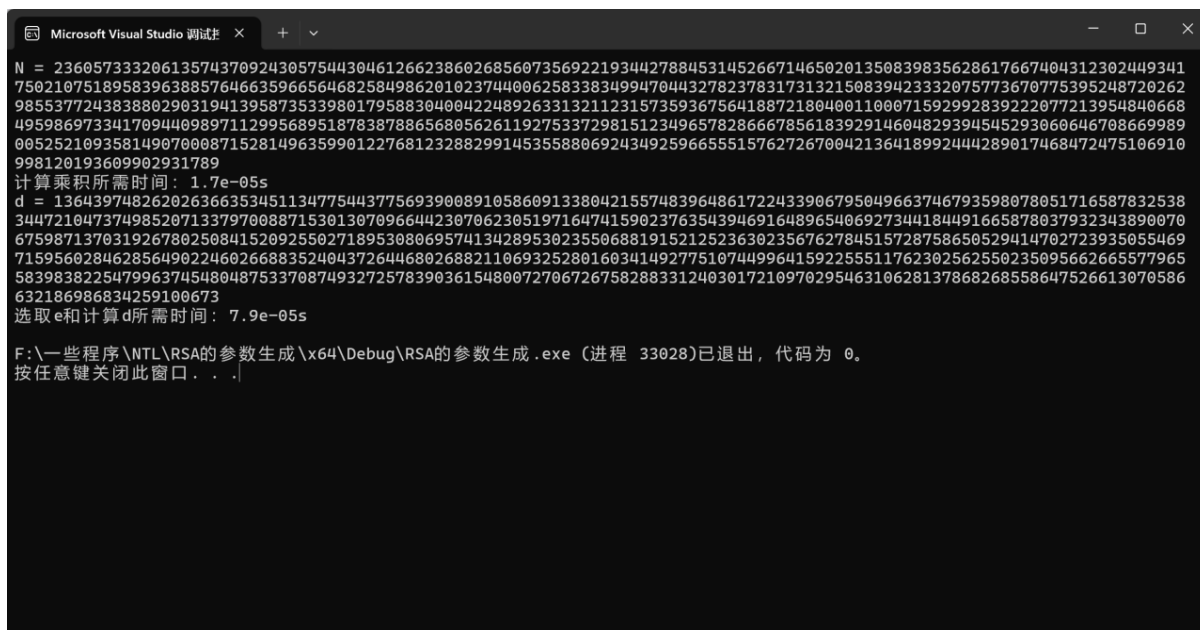
```
1 ZZ d = InvMod(e, operator*(operator-(p, 1), operator-(q, 1)));
```

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



```
Microsoft Visual Studio 调试器
N = 22999593259033891696655740542095955729408549526052705028625015401555500844281873947297592475504501144926227904791645
032000395238924767072846119672679621938317421661395635278543282430146392109589039619321847426554696041633757013606166899
271732640480061491574942762337851072080831486438200298497390261002627130149387413782608873101134034641343833547777625384
127613464714700483243627865819764826512345816064020892202844049600734131876829815465554035925283349837649506477728537497
31571854042541231044442331319270328733946723177574014009937775584725376278535132015507012955886152719434505801096462980
010782759576105772707
计算乘积所需时间: 1.8e-05s
d = 18659160077354364521102416480502614861481654176277465762347428839972907279698841202738387389262535993020471054059034
968131422165469749004320541631089351341050078464298709616321586028174214902646438615129214120611053814450237677288040158
496402837506818887735906918698463519103189485366017237145547443234336052409130464048683896236373395579579827030831806568
540153051602406053805696484811319759555279262215194365344797394902037301779115893908005750489752748212627660024023746467
27692598067027209366568202864117459487778893265577139037040557391379900098949622416299162030887585172713189326499927338
815392615555290265553
选取 e 和计算 d 所需时间: 8.6e-05s
F:\一些程序\NTL\RSA的参数生成\x64\Debug\RSA的参数生成.exe (进程 39104)已退出, 代码为 0。
按任意键关闭此窗口...
```

图 1: 第一次



```
Microsoft Visual Studio 调试器
N = 23605733320613574370924305754430461266238602685607356922193442788453145266714650201350839835628617667404312302449341
750210751895839638857646635966564682584986201023744006258338349947044327823783173132150839423332075773670775395248720262
985537724383880290319413958735339801795883040042248926331321123157359367564188721804001100071592992839222077213954840668
495986973341709440989711299568951878387886568056261192753372981512349657828666785618392914604829394545293060646708669989
005252109358149070008715281496359901227681232882991453558806924349259665551576272670042136418992444289017468472475106910
998120193609902931789
计算乘积所需时间: 1.7e-05s
d = 13643974826202636635345113477544377569390089105860913380421557483964861722433906795049663746793598078051716587832538
34472104737498520713379708871530130709664423070623051971647415902376354394691648965406927344184491665878037932343890070
675987137031926780250841520925502718953080695741342895302355068819152125236302356762784515728758650529414702723935055469
715956028462856490224602668835240437264468026882110693252801603414927751074499641592255511762302562550235095662665577965
583983822547996374548048753370874932725783903615480072706726758288331240301721097029546310628137868268558647526613070586
632186986834259100673
选取 e 和计算 d 所需时间: 7.9e-05s
F:\一些程序\NTL\RSA的参数生成\x64\Debug\RSA的参数生成.exe (进程 33028)已退出, 代码为 0。
按任意键关闭此窗口...
```

图 2: 第二次

```
Microsoft Visual Studio 调试  x + -
N = 12016316925696298092275903071600070698202850050556721725611558257745297279017114644292313669458592168364756350083717
897816700568239857133075639807273342142106371857961317131251212480609704512469664256242160200371894491035008398736825635
148694287004589786396637219184350065331080035214269627153701159383511725537734398868830174197000574440438671827976540016
128540046944051709642328271987266248621117876027387228647466385770091931858381646853696713507159789881414336849651975271
04477671424068204379256001948351457343847797617650113370794042810626704458184974503299790481587562039008588683078006485
360508865041493213397
计算乘积所需时间: 1.6e-05s
d = 19477445519580050144841374846210163881015132839016747011790528002811891449867831738760890506226806934180509743646998
676855335175002212845364072458712591906189784129136681856856680986544530728590756884517214368761254738279886815048186325
615237104360858339699937009535888237791181044918318850306508905828927942693213390209254667984258663850404037956061772604
290942131546109190368817968070284549676049282733005309155400930327829628270495018830876410318059721175937721483398147870
820081202567726920493991053945788652249744806845566742192969601346442205785097730804838878034885168983534617930407884403
64866064500682499957
选取 e 和计算 d 所需时间: 7.4e-05s

F:\一些程序\NTL\RSA的参数生成\x64\Debug\RSA的参数生成.exe (进程 25180)已退出, 代码为 0。
按任意键关闭此窗口. . .
```

图 3: 第三次

2.2.2 sagemath 部分

以下为 sagemath 部分的全部代码

```
1  #!/usr/bin/env python
2  # coding: utf-8
3  import time
4  p = random_prime(2^1024,2^1023)
5  q = random_prime(2^1024,2^1023)
6  start_time = time.time()
7  N = p * q
8  end_time = time.time()
9  end_time - start_time
10 n = (p - 1) * (q - 1)
11 e = 65537
12 while e > 0:
13     if gcd(e,n) == 1:
14         time_1 = time.time()
15         d = inverse_mod(e,n)
16         time_2 = time.time()
17         print(d)
18         print(time_2 - time_1)
19         break
20     else:
21         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))

45703174613483971440272395059610201537548536930098918396260303474759624882400579620703630776040205748662241190022959465583248464435581107271
76389298351111791528899020654076445138814188614007679944857178712615919430931985992837724535405792574112724685109405471817799331051265926589
34288933805460081991227658979973966326431942206595742289795278543304974353913638690626146681819757341657958638467725632715208469266029818349
66675683614980774865678530477932619445502396822404534068781629906752230041620280377538107350025825010459639541588128945322390559856178608664
12207022517748970660689556555786193640105358875892033
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
45232364016353851380517000526318049373973546015864536607215207246721885295309079058790583279103684723739444193
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。