

Московский авиационный институт  
(национальный исследовательский университет)

Факультет информационных технологий и прикладной  
математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Криптография»

Студент: И. Д. Черненко  
Преподаватель: А. В. Борисов  
Группа: М8О-306Б-18  
Дата:  
Оценка:  
Подпись:

Москва, 2021

## Лабораторная работа №1

**Задача:** Разложить каждое из чисел  $n_1$  и  $n_2$  на нетривиальные сомножители.

6)  $n_1=108762353292448487441247663685513658893167646930627178946128889967643172154127$ ,  
 $n_2=1611765569148804856242867384258680719850010286298191204635154152942043219729044752688$   
6147483136114545465725205417369977940016871273001825655775233013745768986374654630793295  
4424777478728351215498316173711656264574423456572770974636411400558323154796702302541456  
94131224473280404169708453094322175307224333415061668790581352676527375610862399155982339  
31006566824074208096468336520404693863268533117447729991162579236036416014409092228354404  
809885779998800076550137

# 1 Метод решения

Изначально была написана реализация р-алгоритма Полларда, но из-за медленной скорости работы было использовано готовое решение `msieve[1]`, из которого была использована реализация общего метода решета числового поля. Для факторизации второго числа данный инструмент уже не подошел и я так и не смог дойти до ответа таким путем, поэтому методом перебора различных вариантов решения подобной задачи было решено найти НОД между моим числом и числами из остальных вариантов. Была использована библиотека `math` из `Python`.

## 2 Факторизация 1-ого числа

zebr@DESKTOP-T2MUUB0:~\$Math/ msieve/ msieve -v 108762353292448487441247663685513658893

```
Msieve v. 1.54 (SVN 1038)
Fri Feb 26 21:00:34 2021
random seeds: 5f8afe67 95704422
factoring 108762353292448487441247663685513658893167646930627178946128889967643172154
(78 digits)
searching for 15-digit factors
commencing quadratic sieve (78-digit input)
using multiplier of 23
using generic 32kb sieve core
sieve interval: 12 blocks of size 32768
processing polynomials in batches of 17
using a sieve bound of 921203 (36471 primes)
using large prime bound of 92120300 (26 bits)
using trial factoring cutoff of 26 bits
polynomial 'A' values have 10 factors
restarting with 19770 full and 189095 partial relations

36852 relations (19770 full + 17082 combined from 189095 partial), need 36567
sieving complete, commencing postprocessing
begin with 208865 relations
reduce to 51799 relations in 2 passes
attempting to read 51799 relations
recovered 51799 relations
recovered 39348 polynomials
attempting to build 36852 cycles
found 36852 cycles in 1 passes
distribution of cycle lengths:
length 1 : 19770
length 2 : 17082
largest cycle: 2 relations
matrix is 36471 x 36852 (5.3 MB) with weight 1104534 (29.97/col)
sparse part has weight 1104534 (29.97/col)
filtering completed in 3 passes
matrix is 24834 x 24893 (3.9 MB) with weight 830943 (33.38/col)
sparse part has weight 830943 (33.38/col)
saving the first 48 matrix rows for later
```

```

matrix includes 64 packed rows
matrix is 24786 x 24893 (2.4 MB) with weight 575039 (23.10/col)
sparse part has weight 370786 (14.90/col)
commencing Lanczos iteration
memory use: 3.6 MB
lanczos halted after 393 iterations (dim = 24784)
recovered 17 nontrivial dependencies
p39 factor: 260951289862485772644727258162652873363
p39 factor: 416791782672403295662841737728685758229
elapsed time 00:00:07

```

### 3 Факторизация 2-ого числа

```

1 | from math import gcd
2 |
3 | n2 =
   | 1611765569148804856242867384258680719850010286298191204635154152942043219729044752688614748313611
4 |
5 | ns = list()
6 | ns.append
   | (284994967805859272853477327862245466978346919806585432133556769959269315271111)
7 | ns.append
   | (134212447269268081486469603983165720134193017053749088894818590919340492696122353647947404066646
8 |
9 | ns.append
   | (352358118079150493187099355141629527101749106167997255509619020528333722352217)
10 | ns.append
   | (169512848540208376377324702550860778129688385180093459660532447790298998967239009844131423368703
11 |
12 | ns.append
   | (119760639583941053725652803731328419697649739176243841021915621242807618608591)
13 | ns.append
   | (191624208718068015686171299450972805253515909112884480565867902529671655940443466481172561918665
14 |
15 | ns.append
   | (344845228130159226488163571070417679235025139015802019152516926202711846660141)
16 | ns.append
   | (159875654421086081200268325250466663128403853515497934091096482467392357863922639791813442919273
17 |
18 | ns.append
   | (160769357899975610828199539114109518167531134514190990785144666932076614717841)

```

19 ns.append  
(125017149737222798202655599967517010894791895137836734347092348310415859721663206658630092156681

20

21 ns.append  
(274114822339589629024026495441557479713813228028980117869052278950681241194819)

22 ns.append  
(159875654421086081200268325250466663128403853515497934091096482467392357863922639791813442919273

23

24 ns.append  
(108762353292448487441247663685513658893167646930627178946128889967643172154127)

25 ns.append  
(268887320029090028117214498253204095765884136483366193842361283776500643966781)

26

27 ns.append  
(141774678697875076503878344320169483769305800714713500792858319214425694670422365904947589804271

28 ns.append  
(123248268911937923199906141216645363665087045422689358104089185316148911496103)

29

30 ns.append  
(158968690785896053229304195025980740908911607577490592481192836972930851627507291449244730388234

31 ns.append  
(284994967805859272853477327862245466978346919806585432133556769959269315271111)

32

33 ns.append  
(144705635774304031878986296122750910474479908149467861238329198698492351931644628770804907791822

34 ns.append  
(472379552736871494058143239162622860896965275113543450580272489891667080207763)

35

36 ns.append  
(126248550402016873100084225758153795732832649752247840500246535964887535681028029224454761807072

37 ns.append  
(361996727456784871855604181056605672088622666207578160811291060873997151708887)

38

39 ns.append  
(191624208718068015686171299450972805253515909112884480565867902529671655940443466481172561918665

40 ns.append  
(313230894596513941163065516500542159481861849753982064716706926040955753912601)

41

42 ns.append  
(196034400067344801010996612379825913878831222300011028544413898468704368209191843772656487365265

43 ns.append

```

44         (374456902508739435218273258671224457341348406488533188195528827819627513233269)
45     ns.append
46         (168843226853565253697616154422540493335291734846688074164655523608094046836939053377756690137486
47         ns.append
48         (61121970174911146319545193754425119520875945215282784640177276523929376501913)
49     ns.append
50         (166981202821111487603574159347402180221234004474088470134427127019583208585679731493672560996991
51     ns.append
52         (383456614884902466726252731294544234658015390619372835826246625499154384118189)
53     ns.append
54         (141690844477193411432723606433569517503385556872451472327609090923890224945076116311617929837009
55     ns.append
56         (242587413455689311805941697582103544343444025737930609728129303011307601823551)
57     ns.append
58         (151093858430251474606868768035913871208482686953174983381615253610702995669437822866501448480999
59     ns.append
60         (181552877565998943910618543225528579935321447209736978912489118450818545230489)
61     ns.append
62         (150334999063135051279428968431307824504008023479374928828438810281152931865133418631450924765400
63     ns.append
64         (319373613270896663765954115654922624879359841665992852658124487372881123570003)
65     ns.append
66         (154062250949081794905352464916598198236271013859014568010836766059230094210745557212887498531331
67     ns.append
68         (374456902508739435218273258671224457341348406488533188195528827819627513233269)
69     ns.append
70         (162657059238403440123105985940845525481005091143114558077381732038544567859777669506831279614525
71
72     for n in ns:
73         tmp = gcd(n2, n)
74         if tmp > 1:
75             print("factor1: ", tmp)
76             print("factor2: ", '%.0f' % (n2 / tmp))
77             break

```

Результат:

```
factor1: 163397696065821074680902655996825570159706795236045906521559460962578519078
8561057256489685565690727114066165297231829395018127947226623668148836316196400727929
2058185071950349333064642775523089637311981469057198581127811557725160954236258017514
8578313739080898244696381665260084479643389434792645421908712913
factor2: 986406545475121984616867352713876213077101742105242959537427249917500252748
2210596174648387419880628453282971116194905653753945702097018977536037164679168
```

Process finished with exit code 0

## 4 Выводы

Выполнив 1-ю лабораторную работу по курсу «Криптография», я понял, что факторизация больших целых чисел является сложной вычислительной задачей, что объясняет её использование в криптографии в качестве основы для некоторых алгоритмов шифрования. К сожалению, самостоятельно реализовать решение данной задачи я не смог, но покопавшись в исходном коде применяемого готового решения стало понятно, что на данном этапе реализовать данный алгоритм эффективно я не смогу.



## Список литературы

- [1] *Факторизация\_целых\_чисел* — *Википедия*.  
URL: [https://ru.wikipedia.org/wiki/Факторизация\\_целых\\_чисел](https://ru.wikipedia.org/wiki/Факторизация_целых_чисел) (дата обращения: 25.02.2021).
- [2] *msieve*  
URL: <https://github.com/radii/msieve> (дата обращения: 25.02.2021).