# 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**

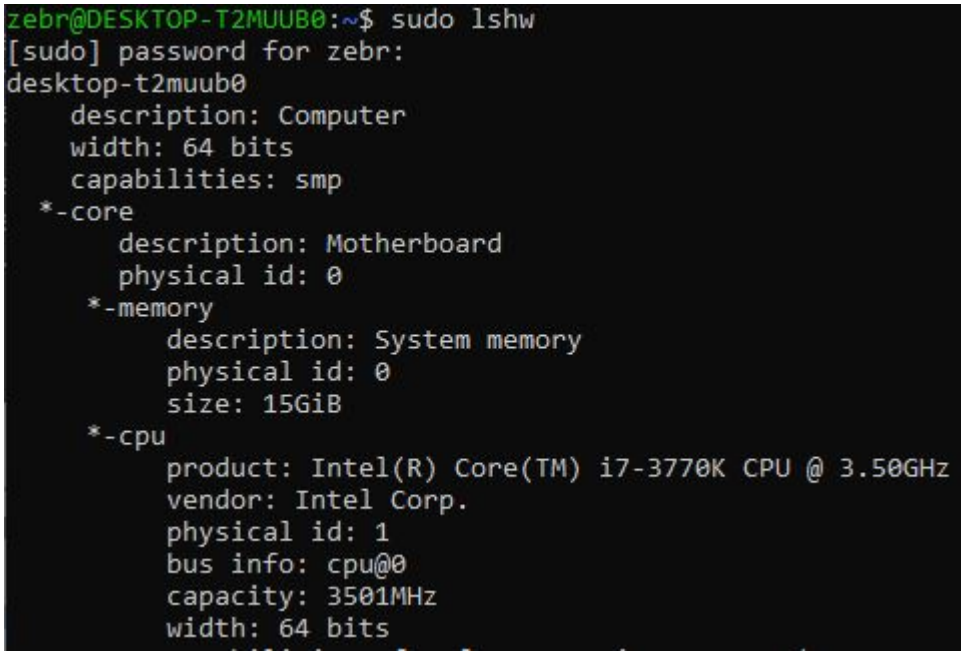
# Лабораторная работа №1

**Задача:** Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.
6) n1=10876235329244848744124766368551365889316764693062717894612888996764317215412 7,
n2=16117655691488048562428673842586807198500102862981912046351541529420432197290447526886147483136114545465725205417369977940016871273001825655775233013745768986374654630793295442477747872835121549831617371165626457442345657277097463641140055832315479670230254145694131224473280404169708453094322175307224333415061668790581352676527375610862399155982339310065668240742080964683365204046938632685331174477299911625792360364160144090922283544048098857799988000765501 37

# 1 Метод решения

Изначально была написана реализация p-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 Характеристики ПК




## 3 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
```

# 4 Факторизация 2-ого числа

```
1 from math import gcd
2
3 n2 =
      16117655691488048562428673842586807198500102862981912046351541529420432197290447526886147483136
4
5 ns = list()
```

```
ns.append
    (284994967805859272853477327862245466978346919806585432133556769959269315271111)
ns.append
    (134212447269268081486469603983165720134193017053749088894818590919340492696122353647947404066646


ns.append
    (352358118079150493187099355141629527101749106167997255509619020528333722352217)
ns.append
    (16951284854020837637732470255086077812968838518009345966053244779029899896723900984413142336870


ns.append
    (11976063958394105372565280373132841969764973917624384102191562124280761860859 1)
ns.append
    (19162420871806801568617129945097280525351590911288448056586790252967165594044346648117256191866


ns.append
    (34484522813015922648816357107041767923502513901580201915251692620271184666014 1)
ns.append
    (15987565442108608120026832525046666312840385351549793409109648246739235786392263979181344291927


ns.append
    (16076935789997561082819953911410951816753113451419099078514466693207661471784 1)
ns.append
    (12501714973722279820265559996751701089479189513783673434709234831041585972166320665863009215668


ns.append
    (27411482233958962902402649544155747971381322802898011786905227895068124119481 9)
ns.append
    (15987565442108608120026832525046666312840385351549793409109648246739235786392263979181344291927


ns.append
    (10876235329244848744124766368551365889316764693062717894612888996764317215412 7)
ns.append
    (26888732002909002811721449825320409576588413648336619384236128377650064396678 1)


ns.append
    (14177467869787507650387834432016948376930580071471350079285831921442569467042236590494758980427


ns.append
    (12324826891193792319990614121664536366508704542268935810408918531614891149610 3)


ns.append
    (15896869078589605322930419502598074090891160757749059248119283697293085162750729144924473038823
```

```
ns.append
    (284994967805859272853477327862245466978346919806585432133556769959269315271111)

ns.append
    (144705635774304031878986296122750910474479908149467861238329198698492351931644628770804907791822

ns.append
    (472379552736871494058143239162622860896965275113543450580272489891667080207763)

ns.append
    (126248550402016873100084225758153795732832649752247840500246535964887535681028029224454761807072

ns.append
    (361996727456784871855604181056605672088622666207578160811291060873997151708887)

ns.append
    (191624208718068015686171299450972805253515909112884480565867902529671655940443466481172561918665

ns.append
    (313230894596513941163065516500542159481861849753982064716706926040955753912601)

ns.append
    (196034400067344801010996612379825913878831222300011028544413898468704368209191843772656487365265

ns.append
    (374456902508739435218273258671224457341348406488533188195528827819627513233269)

ns.append
    (168843226853565253697616154422540493335291734846688074164655523608094046836939053377756690137486

ns.append
    (611219701749111463195451937544251195208759452152827846401772765239293765019 13)

ns.append
    (166981202821111487603574159347402180221234004474088470134427127019583208585679731493672560996991

ns.append
    (383456614884902466726252731294544234658015390619372835826246625499154384118189)

ns.append
    (141690844477193411432723606433569517503385556872451472327609090923890224945076116311617929837009

ns.append
    (242587413455689311805941697582103544343444025737930609728129303011307601823551)

ns.append
    (151093858430251474606868768035913871208482686953174983381615253610702995669437822866501448480999
```

```
55 ns.append
       (181552877565998943910618543225528579935321447209736978912489118450818545230489)
56
57 ns.append
       (150334999063135051279428968431307824504008023479374928828438810281152931865133418631450924765400
58 ns.append
       (319373613270896663765954115654922624879359841665992852658124487372881123570003)
59
60 ns.append
       (154062250949081794905352464916598198236271013859014568010836766059230094210745557212887498531331
61 ns.append
       (374456902508739435218273258671224457341348406488533188195528827819627513233269)
62
63 ns.append
       (162657059238403440123105985940845525481005091143114558077381732038544567859777669506831279614525
64
65 for n in ns:
66     tmp = gcd(n2, n)
67     if tmp > 1:
68         print("factor1: ", tmp)
69         print("factor2: ", '%.0f' % (n2 / tmp))
70         break
```

```
Результат:

factor1:  163397696065821074680902655996825570159706795236045906521559460962578519078
856105725648968555656907271140661652972318293950181279472266236681488363161964007279292058185071950349333064642775523089637311981469057198581127811557725160954236258017514857831373908089824469638166526008447964338943479264542190871291 3
factor2:  986406545475121984616867352713876213077101742105242959537427249917500252748221059617464838741988062845328297111619490565375394570209701897753603716467916 8

Process finished with exit code 0
```

# 5 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` (дата обращения: 25.02.2021).

[2] *msieve*
URL: `https://github.com/radii/msieve` (дата обращения: 25.02.2021).