

**Московский Авиационный Институт
(Национальный Исследовательский Университет)**

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

**Отчет по лабораторной работе №1
по курсу «Криптография»**

Студент: Полей-Добронравова Амелия Вадимовна

Группа: М8О-307Б, № по списку 16.

Преподаватель: Борисов А.В.

Дата: 08.05.2021

Итоговая оценка:

Подпись преподавателя:

Москва, 2021

Условие

Разложить каждое из чисел n_1 и n_2 на нетривиальные сомножители.

Вариант №16

$n_1=242587413455689311805941697582103544343444025737930609728129303011307601823551$,

$n_2=151093858430251474606868768035913871208482686953174983381615253610702995669437822$

86650144848099932846806364650453365846700

06512692482057168858805251730522412435575537

0476387591838494378611695821743531006167608614420833389111629829780186546090734874556

1834472564647434110644877018611946543743680554031457390231514801060564296939903623927

999086648137755263103834503833267130046044

9150826133047599402952702220438132324240801

480483055996850135609380612773088576264939

Описание

Факторизацией натурального числа называется его разложение в произведение простых множителей. Существование и единственность такого разложения следует из основной теоремы арифметики.

Существует ρ -алгоритм Полларда, но его сложность $O(n^{1/4})$. Число n_1 имеет в своей записи 78 цифр, число n_2 содержит 463 цифры. Таким образом, число n_1 приблизительно равно 10^{77} . Подставляя его в формулу сложности найденного алгоритма, получаем 10^{19} . Язык C++ выполняет приблизительно 10^9 операций в секунду, и C++ считается одним из самых быстрых языков программирования. Следовательно для факторизации числа n_1 данным алгоритмом потребуется приблизительно 10^{10} секунд, а это 2,7 миллиона часов. Остальные похожие алгоритмы имеют приблизительно такую же сложность, на которую не хватит физически времени и машинных мощностей.

На помощь здесь приходит библиотека msieve, которую можно скачать в интернете. Это библиотека на языке Си, реализующая набор алгоритмов для факторизации больших целых чисел и включает в себя реализации общего метода решета числового поля и квадратичного решета.

Используем её через вызов в командной строке. Библиотека msieve смогла факторизовать первое число за **2 минуты 49 секунд**, это отображено в файле msieve.log.

Второе же число имеет 463 цифры. Факторизовать такое большое число за короткое время невозможно. Например, даже библиотека msieve принимает на вход числа максимум из 311 цифр

```
(base) MacBook-Pro-Amelia:msieve-1.53 amelia$ ./msieve 1510938584302514746068687680359138712084826869531749833816152536107029956694378228
66501448480999328468063646504533658467000651269248205716885880525173052241243557553704763875918384943786116958217435310061676086144208333
89111629829780186546090734874556183447256464743411064487701861194654374368055403145739023151480106056429693990362392799908664813775526310
38345038332671300460449150826133047599402952702220438132324240801480483055996850135609380612773088576264939
input integers must be under 311 digits
error -10 converting '1510938584302514746068687680359138712084826869531749833816152536107029956694378228665014484809993284680636465045336
58467000651269248205716885880525173052241243557553704763875918384943786116958217435310061676086144208333891116298297801865460907348745561
83447256464743411064487701861194654374368055403145739023151480106056429693990362392799908664813775526310383450383326713004604491508261330
47599402952702220438132324240801480483055996850135609380612773088576264939'
```

Оказалось, что один из множителей числа n_2 можно найти как НОД с n_2 из 15 вариантов. Второй множитель легко найти, разделив данное число на полученный НОД.

Для выполнения этой части задания напишу небольшой скрипт на Python, использующий библиотеку gcd для поиска НОД.

Ход выполнения работы

- 1) Установить библиотеку msieve.

Для этого, скачав её архив, разархивирую. Затем добавлю в Makefile по рекомендации add 'NO_ZLIB=1' if you don't have zlib. Далее пропишу в терминале make all.

- 2) Запуск программы, передавая ей число n_1 в качестве аргумента

```
(base) MacBook-Pro-Amelia:msieve-1.53 amelia$ ./msieve 242587413455689311805941697582103544343444025737930609728129303011307601823551
sieving in progress (press Ctrl-C to pause)
39587 relations (20295 full + 19292 combined from 213312 partial), need 39272
sieving complete, commencing postprocessing
```

- 3) Результат отображается в файле msieve.log

```
msieve.log
Показать Сейчас Очистить Перезагрузить Поделиться
Sun Jun 6 11:04:47 2021 using large prime bound of 98878300 (26 bits)
Sun Jun 6 11:04:47 2021 using trial factoring cutoff of 27 bits
Sun Jun 6 11:04:47 2021 polynomial 'A' values have 10 factors
Sun Jun 6 11:07:29 2021 39587 relations (20295 full + 19292 combined from 213312 partial),
need 39272
Sun Jun 6 11:07:29 2021 begin with 233607 relations
Sun Jun 6 11:07:29 2021 reduce to 56483 relations in 2 passes
Sun Jun 6 11:07:29 2021 attempting to read 56483 relations
Sun Jun 6 11:07:29 2021 recovered 56483 relations
Sun Jun 6 11:07:29 2021 recovered 45674 polynomials
Sun Jun 6 11:07:29 2021 attempting to build 39587 cycles
Sun Jun 6 11:07:29 2021 found 39587 cycles in 1 passes
Sun Jun 6 11:07:29 2021 distribution of cycle lengths:
Sun Jun 6 11:07:29 2021     length 1 : 20295
Sun Jun 6 11:07:29 2021     length 2 : 19292
Sun Jun 6 11:07:29 2021 largest cycle: 2 relations
Sun Jun 6 11:07:29 2021 matrix is 39176 x 39587 (5.7 MB) with weight 1178858 (29.78/col)
Sun Jun 6 11:07:29 2021 sparse part has weight 1178858 (29.78/col)
Sun Jun 6 11:07:29 2021 filtering completed in 4 passes
Sun Jun 6 11:07:29 2021 matrix is 27839 x 27903 (4.4 MB) with weight 920115 (32.98/col)
Sun Jun 6 11:07:29 2021 sparse part has weight 920115 (32.98/col)
Sun Jun 6 11:07:29 2021 saving the first 48 matrix rows for later
Sun Jun 6 11:07:29 2021 matrix includes 64 packed rows
Sun Jun 6 11:07:29 2021 matrix is 27791 x 27903 (2.6 MB) with weight 645398 (23.13/col)
Sun Jun 6 11:07:29 2021 sparse part has weight 413715 (14.83/col)
Sun Jun 6 11:07:29 2021 commencing Lanczos iteration
Sun Jun 6 11:07:29 2021 memory use: 2.7 MB
Sun Jun 6 11:07:36 2021 lanczos halted after 441 iterations (dim = 27788)
Sun Jun 6 11:07:36 2021 recovered 16 nontrivial dependencies
Sun Jun 6 11:07:36 2021 p39 factor: 331393459585519177520233247108865056209
Sun Jun 6 11:07:36 2021 p39 factor: 732022333087377555663155580882142234639
Sun Jun 6 11:07:36 2021 elapsed time 00:02:49
```

Найдено два нетривиальных сомножителя для n_1 :

- 331393459585519177520233247108865056209
- 732022333087377555663155580882142234639

4) Напишу скрипт для работы со вторым числом

```
from math import gcd

n2 =
151093858430251474606868768035913871208482686953174983381615253610702995669437822866501448480999
328468063646504533658467000651269248205716885880525173052241243557553704763875918384943786116958
217435310061676086144208333891116298297801865460907348745561834472564647434110644877018611946543
743680554031457390231514801060564296939903623927999086648137755263103834503833267130046044915082
6133047599402952702220438132324240801480483055996850135609380612773088576264939

n3 =
141690844477193411432723606433569517503385556872451472327609090923890224945076116311617929837009
763773660959874697853968119080617502373943782494979020311419544728762119216205286391137003028125
331158247702385902798481867910823926760076341189111357818193897834136876367785553468541342743729
023927657307836543731689119550558446364266971611293672837308855338590286435921893375062744052147
0476774178793413097754328106876810009083432628213288672194420754620920548851129

b = gcd(n2, n3)
a = n2//b

print('The first factor:', b)
print('The second factor:', a)
```

5) Результат его работы

```
(base) MacBook-Pro-Amelia:lab1 amelia$ python3 second.py
The first factor: 13947662566596806656822461498390813091641156164225579665309714873467047611085831034684394504688420482122483591144622792
08076762182540253830879952893670507118343712989321426002991737403785142612646952508976462708323071248604591875264878205479161762606274749
93626505280101101880100712184854128131522995503631439
The second factor: 1083291610395748071062330138077843116412670837624588547162666244349294050233280354493184039631633355021401632917126713
7519877859657437533637812444080526501
```

Получаем нетривиальные сомножители для числа n_2 :

- 139476625665968066568224614983908130916411561642
255796653097148734670476110858310346843945046884
204821224835911446227920807676218254025383087995
289367050711834371298932142600299173740378514261
264695250897646270832307124860459187526487820547
916176260627474993626505280101101880100712184854
128131522995503631439
- 108329161039574807106233013807784311641267083762
458854716266624434929405023328035449318403963163
335502140163291712671375198778596574375336378124
44080526501

Затраченное время на факторизацию первого числа: 2 минуты 49 секунд

Затраченное время на факторизацию второго числа: меньше 1 секунды

Для второго числа мы используем только поиск НОД и деление одного числа на другое. Сложность алгоритма Евклида для поиска НОД это $O(\log \min(a, b))$, получаем, что для поиска НОД чисел из 463 цифр потребуется только 1536 операций, что по времени почти мгновенно.

Характеристики компьютера, на котором производился расчёт:

Процессор: 2,3 GHz Intel Core i7

Память: 8 ГБ 1600 MHz DDR3

Операционная система: macOS Mojave 10.14.6

Вывод

Факторизация чисел - первое применение квантового компьютера, которое вспоминают. Длительное время, необходимое для вычисления факторизации больших целых чисел - основа существования RSA алгоритма.

Система RSA используется для защиты программного обеспечения и в схемах цифровой подписи.

Также она используется в открытой системе шифрования PGP и иных системах шифрования (к примеру, DarkCryptTC и формат xdc) в сочетании с симметричными алгоритмами.

Данная лабораторная работа стала поводом для небольшого исследования в данном вопросе, и вывод положительный: на данный момент использование RSA безопасно. Но как только квантовый компьютер будет доработан, нужно будет пересматривать текущую систему и искать новые способы шифрования.