МИНИСТЕРСТВО НАУКИ И ВЫШСЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3
по курсу «Криптография»

Студент:	Семин А. В.
Группа:	М8О-306Б-20
Преподаватель:	А. В. Борисов
Оценка:	
Дата:	

Москва, 2023

Лабораторная работа №3

Тема: Факторизация числа

Вариант: fa

Задание:

1. Получить номер варианта: свое ФИО подать на вход в хеш-функцию, являющуюся стандартом, выход хеш-функции представить в шестнадцатеричном виде и рассматривать младший байт как номер варианта.
2. Разложить число из своего варианта на нетривиальные сомножители.

Описание

Факторизацией натурального числа называется его разложение в произведение простых множителей. Существование и единственность (с точностью до порядка следования множителей) такого разложения следует из основной теоремы арифметики.

В отличие от задачи распознавания простоты числа, факторизация предположительно является вычислительно сложной задачей. В настоящее время неизвестно, существует ли эффективный не квантовый алгоритм факторизации целых чисел. Однако доказательства того, что не существует решения этой задачи за полиномиальное время, также нет.

Предположение о том, что для больших чисел задача факторизации является вычислительно сложной, лежит в основе широко используемых алгоритмов (например, RSA). Многие области математики и информатики находят применение в решении этой задачи. Среди них: эллиптические кривые, алгебраическая теория чисел и квантовые вычисления.

RSA сам по себе не является практически надежным (semantically secured), так как при одних и тех же значениях входных параметров (ключа и сообщения) выдаёт одинаковый результат. Однако, его удобно использовать как вспомогательный алгоритм, для шифровки, например, сеансового ключа (используется в TLS, а также в ранних версиях PGP). Несложно показать, что задача дешифровки сообщения или ключа, зашифрованного с помощью RSA, сводится к проблеме факторизации целых чисел.

Рассмотрим *общий метод решета числового поля* для факторизации целых чисел.

Общий метод решета числового поля ('general number field sieve, GNFS) — метод факторизации целых чисел. Является наиболее эффективным алгоритмом факторизации чисел длиной более 110 десятичных знаков.

Сложность алгоритма оценивается эвристической формулой:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(log\,n)^{\frac{1}{3}}\,(log\,log\,n)^{\frac{2}{3}}\right)=L_n\left[\frac{1}{3},\ \sqrt[3]{\frac{64}{9}}\right]$$

Метод является обобщением специального метода решета числового поля: тогда как последний позволяет факторизовать числа только некоторого специального вида, общий метод работает на множестве целых чисел, за исключением степеней простых чисел (которые факторизуются тривиально извлечением корней).

Метод решета числового поля (как специальный, так и общий) можно представить, как усовершенствование более простого метода — метода рационального решета либо метода квадратичного решета. Подобные им алгоритмы требуют нахождения гладких чисел порядка $\sqrt{n}$. Размер этих чисел экспоненциально растёт с ростом n. Метод решета числового поля, в свою очередь, требует нахождения гладких чисел субэкспоненциального относительно n размера. Благодаря тому, что эти числа меньше, вероятность

того, что число такого размера окажется гладким, выше, что и является причиной эффективности метода решета числового поля. Для достижения ускорения вычислений в рамках метода проводятся в числовых полях, что усложняет алгоритм, по сравнению с более простым рациональным решетом.

Основные принципы

- Метод факторизации Ферма для факторизации натуральных нечетных чисел n, состоящий в поиске таких целых чисел x и y, что $x^2 - y^2 = n$, что ведет к разложению $n = (x - y) * (x + y)$.
- Нахождение подмножества множества целых чисел, произведение которых — квадрат.
- Составление факторной базы: набора $\{-1, p_1, p_2, \dots, p_n\}$, где p_i – простые числа, такие что $p_i \leq B$ для некоторого B.
- Просеивание выполняется подобно решету Эратосфена (откуда метод и получил своё название). Решетом служат простые числа факторной базы и их степени. При просеивании число не «вычёркивается», а делится на число из решета. Если в результате число оказалось единицей, то оно B-гладкое.
- Основная идея состоит в том, чтобы вместо перебора чисел и проверки, делятся ли их квадраты по модулю n на простые числа из факторной базы, перебираются простые числа из базы и сразу для всех чисел вида проверяется, делятся ли они на это простое число или его степень.

Ход выполнения работы:

1. *Вычисление номера варианта.*

Для вычисления номера варианта я использовал стандартную хэш-функцию класса String в Java.

```
public class CRLab2 {
    public static void main (String[] args) {
        String name = "Семин Александр Витальевич";
        String hexString = Integer.toHexString(name.hashCode());
        System.out.println(hexString);
     }
}
```

Сама хэш-функция, переопределенная в классе String, выглядит следующим образом:

```
public int hashCode() {
    int h = hash;
    if (h == 0 && !hashIsZero) {
        h = isLatin1() ? StringLatin1.hashCode(value)
                       : StringUTF16.hashCode(value);
        if (h == 0) {
            hashIsZero = true;
        } else {
            hash = h;
        }
    }
    return h;
}
```

Результат выполнения:
90e24efa
Младший байт: fa

Таким образом, вариант fa. Число, которое нужно факторизовать:
587001250609237122936576780197222991151753736674220109004474293607173917070470680797189229766409054553586774989887452450364068622528079086803614810202827588459315928046299951887979803334204273291981221032237387241947784071943078664650050662310962747619491223879940411727964302362450684574134527511533196962581

2. *Разложение числа на нетривиальные сомножители.*

Код на Python:

```
from math import gcd
import random
```

```
def bin_pow(a, n):
  res = 1
  while n > 0:
    if n % 2 == 1:
      res = res * a
    a = a * a
    n //= 2
  return res

def is_prime(num, test_count):
  for i in range(test_count):
    rnd = random.randint(1, num - 1)
    if gcd(num, rnd) != 1:
      return False
  return True

curVar =
58700125060923712293657678019722299115175373667422010900447429360717391707047068079718922976640905455358677498988745245036406862252807908680361481020282758845931592804629995188797980333420427329198122103223738724194778407194307866465005066231096274761949122387994041172796430236245068457413452751153319696258 1
nextVar
=335761174067164167618543198964247689269563224294181999563165832133259540292187722805137445078814489162047464153361099572168058264117428507939423508728572440101286002782251421459113221509564615169373781381114169967893458821918147723049357233572020159694576243720955242412265730461627230303922059224402752942021
d1 = gcd(curVar, nextVar)
d2 = curVar // d1
print("Проверка на простоту")
print("Первый делитель:", is_prime(d1, 100))
print("Второй делитель:", is_prime(d2, 100))
print("\nВывод делителей")
print("Первый делитель:", d1)
print("Второй делитель:", d2)
```

Вывод в консоль:

Проверка на простоту
Первый делитель: True
Второй делитель: True

Вывод делителей
Первый делитель:
190927791949870532806403720288940560472450600245051099771889880535906270942341
Второй делитель:
307446728742014955035508979831865835587534976830936525621320138940810418897482269597939301803411018182317084702545306986405049662074918237142818023448179366323131691166450721171320938452603535221687189569044071141567114154894064264 1

Таким образом, ответ:
190927791949870532806403720288940560472450600245051099771889880535906270942341 и
307446728742014955035508979831865835587534976830936525621320138940810418897482269597939301803411018182317084702545306986405049662074918237142818023448179366323131691166450721171320938452603535221687189569044071141567114154894064264 1

Выводы

В ходе выполнения данной лабораторной работы я узнал предельные возможности для факторизации числа, изучил метод Ферма для проверки числа на простоту, а также воспользовался средствами языка Python и специальным подходом для нахождения факторизации очень большого числа.

Литература

1. Википедия, свободная энциклопедия, Факторизация целых чисел [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Факторизация_целых_чисел (Дата обращения 02.04.2023)
2. Integer factorization calculator [Электронный ресурс], URL: https://www.alpertron.com.ar/ECM.HTM (Дата обращения: 02.04.2023)