Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Криптография»

Студент:	Я. С. Поскряков
Преподаватель:	А. В. Борисов
Группа:	М8О-306Б
Дата:	
Оценка:	
Подпись:	

Москва, 2020

Лабораторная №1. Факторизация.

Задача: Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.
Вариант №17:
n1=181552877565998943910618543225528579935321447209736978912489118450818545230489
n2=150334999063135051279428968431307824504008023479374928828438810281152931865133
418631450924765400917258000645743935615213286021088135604627169932920125305843303
616232167430518821188511162822374965949771668681638403317861379756861892717375280
069279523162223543493433550049659931535778659520881621348942909061872472941613174
69653368470808158015990205733111051137495109727760731079959209757862236842344071
8164595720091899427036135539095740807639167195995008580910433
Для каждого числа необходимо выполнить факторизацию.

1 Описание

Факторизация - разложение числа на простые множители. Для выполнения данной задачи существуют множество алгоритмов, и в начале, я хотел написать и проверить работу одного из этих алгоритмов Метод Полларда p-1, Метод Полларда Ро, Метод Бента (модификация метода Полларда "Ро"), Метод Полларда Монте-Карло. В начале реализовал Метод Полларда Ро.

Однако, в связи с тем, что задача, поставленная передо мной, требовала лишь конечного результата и не регламентировала следование по какому-либо определенному пути, я решил воспользоваться уже готовым продуктом - msieve. Данная программа позволила разложить мне первое число всего за 8 секунд, разложение же второго представляло собой более сложную ситуацию, как минимум по причине того, что хоть и возможно факторизовать это число без хаков, но это заняло бы огромное количество времени. Поэтому, после того, как мне стало известно о том, что мое второе число имеет общий НОД с другим из чисел в вариантах, я обрадовался. Собственно, после того, как мне стало известно об этом, я перебирал все числа, искал их НОД с моим, а затем разделил свое число, на найденный НОД.

2 Попытка выполнить задание через алгоритм Полларда Ро.

```python
from random import randint
from time import perf_counter

MY_NUM = 181552877565998943910618543225528579935321447209736978912489118450818545230489


def gcd(a, b):
    while b:
        a, b = b, a % b
    return a


def miller_rabin_test(a, n, s, d):
    x = pow(a, d, n)
    if x in (1, n - 1):
        return True
    for r in range(s - 1):
        x = (x * x) % n
        if x == 1:
            return False
        elif x == n - 1:
            return True
    return False


def miller_rabin(n):
    s, d = 0, n - 1
    while d % 2 == 0:
        s, d = s + 1, d // 2
    for _ in range(n.bit_length()):
        a = randint(2, n - 2)
        if not miller_rabin_test(a, n, s, d):
            return False
    return True


def pollards_rho_iter(n):
    """      """
    x = y = 2
    a = 1
    while True:
        d = 1
        while d == 1:
            x = (x * x + a) % n
            y = (y * y + a) % n
            y = (y * y + a) % n
            d = gcd(n, abs(x - y))
        if d < n:
```

```
            return d
        x = y = randint(1, n - 1)
        a = randint(-100, 100) % n


def factor(n):
    """     -     -"""
    #    ,
    ans = []
    for x in (2, 3, 5, 7, 11, 13):
        while n % x == 0:
            ans.append(x)
            n //= x
    if n == 1:
        pass
    elif miller_rabin(n): #
        ans.append(n)
    else: #
        d = pollards_rho_iter(n)
        ans.extend(factor(d))
        ans.extend(factor(n // d))
    return sorted(ans)


nums = [10, 437, 3127, 23707, 1752967, 6682189, 12659363, 494370889, 1435186847]
for num in nums:
    st = perf_counter()
    fct = factor(num)
    en = perf_counter()
    print(' {}    {}  {:02f}c'.format(num, fct, en-st))
```

3 Консоль. Разбор первого числа.

(base) yar@yarmachine: /Math msieve/msieve 1815528775659989439106185432255285799353214472097369789124891184508185452304 89

sieving complete, commencing postprocessing

(base) yar@yarmachine: /Math tail -n 3 msieve.log

Sat Feb 29 01:33:23 2020 p39 factor: 398579455296534097174088228586442482949

Sat Feb 29 01:33:23 2020 p39 factor: 455499838622960911273301695661963459461

Sat Feb 29 01:33:23 2020 elapsed time 00:00:08

4 Разбор второго числа.

n1 = 131186186802338870843656988010086375259459973814083801678676
32590301780458333878357672754959267146332613208470005543362294113
67110638183361259555995952592066358028224811325614156674303320145
81087426233799928510815309548958100238498530152882072491356249234
775478456501396916030147830189673542345190817454789651
n2 = 1145966680849166836380585149133844172746680983710378982667941
454563175053894262153311839273655534383836399994632270716841094179
5680248292142414665913159483

5 Выводы

Факторизация целых чисел используется для решения проблем сохранности информации.

В процессе моего ознакомления с данным понятием очень часто я встречал упоминание того факта, что алгоритм шифрования RSA использует факторизацию челых чисел. Идея RSA состоит в следующем:

1) Генерируются случайно два простых числа большой размерности. (p, q)

2) Вычисляется их произведение. (n = p*q)

3) Вычисляется функция Эйлера $\phi(n) = (p-1)*(q-1)$

4) Выбирается число e, такое, что оно простое, оно меньше, а также взаимно простое с $\phi(n)$

5) Таким образом пара чисел (e, n) является открытым ключом.

6) Вычисляется d, такое что (d*e) % $\phi = 1$

7) Теперь пара чисел (d, n) - закрытый ключ.

Сам процесс шифрования организован следующим образом.

Возводится сообщение в степень e по модулю n. Затем эти данные отправляются.

Расшифровка же заключается в том, что полученное сообщение возводится в степень d, вычисляется остаток от деления полученного числа на n.

Таким образом факторизация может помочь, зная n, вычислить p и q, однако в связи с тем, что такое вычисление крайне сложно, в плане вычислительных процессов и затраченного времени, алгоритм RSA при достаточно больших выбранных простых чисел обеспечивает надежную сохранность информации.

В процессе выполнения лабораторной работы, я узнал достаточно много новой и интересной информации, связанной как непосредственно с факторизацией, алгоритмом RSA, так и с криптографией. Надеюсь, что и дальше будет также.