(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу

«Криптография»

Студент: Кузьмичев Александр Николаевич
Группа: М80 – 306Б-18
Вариант: 12
Преподаватель: Борисов А.В.
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2020

Вариант 12

Задача: Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.

Первое число:

313230894596513941163065516500542159481861849753982064716706926040955753912601

Второе число:

196034400067344801010996612379825913878831222300011028544413898468704368209191843772656487365265595933792721394282928384361525292628178919637247173089242245223053111826538592314858736495639204502526776240411959783887447103901725323630830637454127437535567150099119639452450919227848747342902206784846015011491899683841540164482032449394186206120858468684059402522378692407944426271409549030177207712639579023599983600397129061698889472537300204217414852744899172l

автоматизирующую этот процесс. Второй множитель я нашел путем деления второго числа на НОД.

Реализация алгоритма Полларда:

```
from math import gcd, pi
from functools import reduce
from datetime import datetime
From random import randint

CHECK_NUM =
31323089459651394116306551650054215948186184975398206471670692604095575391260 1



def f(x, a, b):
    return a * x ** 2 + b


def is_prime(N):
    if N in (0, 1):
        return False
    if N == 2:
        return True
    if N % 2 == 0:
        return False
    s = N - 1
    while s % 2 == 0:
        s //= 2
    for i in range(50):
        a = randint(1, N - 1)
        exp = s
        mod = pow(a, exp, N)
        while exp != N - 1 and mod != 1 and mod != N - 1:
            mod = mod * mod % N
            exp *= 2
        if mod != N - 1 and exp % 2 == 0:
            return False
    return True
```

```
def find_factor(n):
    maxiterssq = pi / 4 * n
    x = randint(1, n - 1)
    y = x
    d = 1
    iters = 0
    a = randint(1, n - 1)
    b = randint(1, n - 1)
    while d in (1, n):
        if iters ** 2 > maxiterssq:
            a = randint(1, n - 1)
            b = randint(1, n - 1)
            x = randint(1, n - 1)
            y = x
            iters = 0
        x = f(x, a, b) % n
        y = f(f(y, a, b), a, b) % n
        d = gcd(abs(x - y), n)
        iters += 1
    return d


def find_prime_factor(n, factors):
    if is_prime(n):
        factors.append(n)
    else:
        tmp = n // find_factor(n)
        find_prime_factor(tmp, factors)


def factor(n, factors):
    while n % 2 == 0:
        factors.append(2)
        n //= 2
    while n % 3 == 0:
        factors.append(3)
        n //= 3
    while n > 1:
        find_prime_factor(n, factors)
        n //= factors[-1]
```

```
def find_all_factors(prime_factors, all_factors):
    all_factors.append(1)
    all_factors.append(prime_factors[0])
    for i in range(1, len(prime_factors)):
        tmp = []
        for f in all_factors:
            if f * prime_factors[i] not in all_factors:
                tmp.append(f * prime_factors[i])
        all_factors += tmp
    all_factors.sort()


def pollard_rho(n):
    factors = []
    factor(n, factors)
    factors.sort()
    all_factors = []
    find_all_factors(factors, all_factors)
    return all_factors


def get_info(num, factors, start_time):
    print("Original number: {0}".format(num))
    print("Factors:")
    print(*factors, sep='\n')
    print("Time: {0}".format(datetime.now() - start_time))


if __name__ == '__main__':
    n = 0
    with open('test1', 'r') as file:
        n = int(file.read())
    start_time = datetime.now()
    factors = pollard_rho(n)
    factors = factors[1:len(factors) - 2]
    get_info(n, factors, start_time)

    factors.clear()
    with open('test2', 'r') as file:
        n = int(file.read())
    start_time = datetime.now()
    factors.append(gcd(n, CHECK_NUM))
```

```
factors.append(n / factors[0])
get_info(n, factors, start_time)
```

Первое число

Math/msieve -v 313230894596513941163065516500542159481861849753982064716706926040955753912601

Msieve v. 1.54 (SVN 1038)

Wed Mar 3 11:45:40 2021

random seeds: b1b6e29f 5923d566

factoring 313230894596513941163065516500542159481861849753982064716706926040955753912601 (78 digits)

searching for 15-digit factors

commencing quadratic sieve (78-digit input)

using multiplier of 1

using generic 32kb sieve core

sieve interval: 12 blocks of size 32768

processing polynomials in batches of 17

using a sieve bound of 999269 (39162 primes)

using large prime bound of 99926900 (26 bits)

using trial factoring cutoff of 27 bits

polynomial 'A' values have 10 factors

sieving in progress (press Ctrl-C to pause)

39466 relations (20704 full + 18762 combined from 211699 partial), need 39258

39466 relations (20704 full + 18762 combined from 211699 partial), need 39258

sieving complete, commencing postprocessing

begin with 232403 relations

reduce to 55904 relations in 2 passes

attempting to read 55904 relations

recovered 55904 relations

recovered 42055 polynomials

attempting to build 39466 cycles

found 39466 cycles in 1 passes

distribution of cycle lengths:

  length 1 : 20704

  length 2 : 18762

largest cycle: 2 relations

matrix is 39162 x 39466 (5.7 MB) with weight 1173705 (29.74/col)

sparse part has weight 1173705 (29.74/col)

filtering completed in 4 passes

matrix is 26858 x 26922 (4.2 MB) with weight 893440 (33.19/col)

sparse part has weight 893440 (33.19/col)

saving the first 48 matrix rows for later

matrix includes 64 packed rows

matrix is 26810 x 26922 (2.8 MB) with weight 656264 (24.38/col)

sparse part has weight 473134 (17.57/col)

commencing Lanczos iteration

memory use: 4.1 MB

lanczos halted after 425 iterations (dim = 26805)

recovered 14 nontrivial dependencies

p39 factor: 537228079155448813380781027030896715807

p39 factor: 583050117352260532679885280778162124743

elapsed time 00:01:54

Второе число:

НОД с числом из второго варианта:
120050505373548699525427072160307080347616964834042886652
247670626117267458276794506042864873487917519688827578324
240353041091749676693534016592215307728 27

Ответы:

Факторизация первого числа:

1) 537228079155448813380781027030896715807

2) 583050117352260532679885280778162124743

Факторизация второго числа:

1)

163293273491323423813718250415724354506272599158350870439
971669103635652659935643004482831489242678221800658262859
359551639300440700014162773951243513304159307962059110327
063693116472159225989885945735405828148563381462677904094
802373237140070461921154426170136349806758308479922324825
98124424978876686764212 3

2)
120050505373548699525427072160307080347616964834042886652
247670626117267458276794506042864873487917519688827578324
240353041091749676693534016592215307728 27

Выводы

Выполнив первую лабораторную работу по криптографии, я получил свой первый опыт факторизации больших чисел. Я узнал об алгоритмах и библиотеках, облегчающих этот процесс. Я рад, что в начале лабораторной работы, не сразу узнав про msieve, мне пришлось реализовывать алгоритм Полларда. В дальнейшем я бы хотел подробно изучить реализацию msieve.