**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО 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 № 2   
по курсу «Криптография»

Группа: М8О-310Б-21

Студент: А. С. Личковаха

Преподаватель: А. В. Борисов

Оценка:

Дата: 05.06.24

Москва, 2024

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc158983147)

[2 Задание 3](#_Toc158983148)

[3 Теория 4](#_Toc158983149)

[4 Ход лабораторной работы 5](#_Toc158983150)

[5 Выводы 6](#_Toc158983151)

# **Тема**

Разложение на нетривиальные сомножители

# **Задание**

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители. Вариант 2-ой.

n1=313230894596513941163065516500542159481861849753982064716706926040955753912601  
n2=196034400067344801010996612379825913878831222300011028544413898468704368209191843 7726564873652655959337927213942829283843615252926281789196372471730892422452230531118 2653859231485873649563920450252677624041195978388744710390172532363083063745412743753 5567150099119639452450919227848747342902206784846015011491899683841540164482032449394 1862061208584686840594025223786924079444262714095490301772077126395790235999836003971 290616988894725373002042174148527448991721

# **Теория**

Факторизация - это процесс нахождения всех простых множителей, составляющих данное число. Она является важной задачей в математике и криптографии.

В современной криптографии активно используется алгоритм шифрования RSA. 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. Задача дешифровки сообщения или ключа, зашифрованного с помощью RSA сводится к проблеме факторизации целых чисел.

В алгоритме RSA используется два больших простых числа, которые умножаются друг на друга, чтобы получить очень большое число, которое служит открытым ключом. Факторизация этого большого числа обратно на его простые множители является трудной задачей и основывается на том, что нельзя эффективно найти простые множители очень больших чисел.

Когда вы пытаетесь взломать систему RSA, вы, по сути, пытаетесь найти простые множители этого большого числа (открытого ключа), которое было создано путем умножения двух простых чисел.

Именно сложность задачи факторизации больших чисел является ключевой для безопасности алгоритма RSA: если бы эта задача была решена эффективно, безопасность RSA была бы нарушена.

# **Ход лабораторной работы**

Увидев число n1, я понял, что в инт32 оно не помещается, и решил сразу писать код на питоне

import math

from datetime import timedelta,datetime

from sympy import factorint

n1=313230894596513941163065516500542159481861849753982064716706926040955753912601

last\_time = datetime.now()

cur = 1

with open ('logs.txt','w') as file:

    sqrt = math.ceil(math.sqrt(n1))

    for i in range(2,sqrt):

        if n1 % i == 0:

            print(i)

        if datetime.now() - last\_time >= timedelta(*seconds*=1):

            print("[x]", datetime.now().strftime("%H:%M"),f': iter:\"{i}\", last\_factor:\"{cur}\"',*file*=file,*flush*=True)

            last\_time = datetime.now()

Потом я решил посчитать ассимптотику, мой алгоритм работает за O(N^1/2). Допустим, в лучшем случае питон считает 10^7 операций за секунду, когда речь идёт о самых наивных операциях (сложение, вычитание и т.д). Примерно алгоритм будет выполнять 5 \* 10^38 операций. Если всё подсчитать, то в лучшем случае компьютер посчитает это за 5\*10^26 дней и если учесть что операция братия остатка достаточно долгая операция, то действительное время работы будет напорядок больше. Даже если я буду использовать алгоритм поларда, работающего за O(N^1/4), я всё равно не успею сдать лабораторную к концу своего обучения.   
Начал искать сервер, который бы смог мне посчитать это число, вышел на сайт alpetron.com. Он вычислил мне два простых числа, на которое раскладывается наша n1 за 2 минуты 56 секунд:

p = 537 228079 155448 813380 781027 030896 715807

q = 583 050117 352260 532679 885280 778162 124743

Мне стало очевидно, что второе число я не смогу дождаться, так как оно содержит 463 цифры. Если сервер не в состоянии мне посчитать за приемлемое время, то я решил обратиться к базе данных, зашёл на сайт factordb.com и достал оттуда множители моего числа.

P:

12005050537354869952542707216030708034761696483404288665224767062611726745827679450604286487348791751968882757832424035304109174967669353401659221530772827

q:

163293273491323423813718250415724354506272599158350870439971669103635652659935643004482831489242678221800658262859359551639300440700014162773951243513304159307962059110327063693116472159225989885945735405828148563381462677904094802373237140070461921154426170136349806758308479922324825981244249788766867642123

# **Выводы**

Исходя из проделанной работы, я увидел, что процесс факторизации простых чисел является очень трудоёмкой задачей даже для современных компьютеров. Поэтому, если злоумышленник получит открытый ключ, дешифрование без закрытого ключа приведёт к решению задачи, которую его компьютер будет выполнять десятки лет.

# **Список используемой литературы**

<https://habr.com/ru/companies/virgilsecurity/articles/459370/>

<https://softwaredominos.com/home/science-technology-and-other-fascinating-topics/integer-factorization-algorithms-a-comparative-analysis/>

https://habr.com/ru/companies/virgilsecurity/articles/459370/