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 № 1
по курсу «Криптография»

Группа: М8О-308Б-21

Студент(ка): Т. Ж. Караев

Преподаватель: А. В. Борисов

Оценка:

Дата: 10.04.2025

Москва, 2025

ОГЛАВЛЕНИЕ

1 Тема

Факторизация больших целых чисел.

2 Задание

Строку, в которой записано своё ФИО, подать на вход в качестве аргумента хеш-функции ГОСТ Р 34.11-2012 (Стрибог). Младшие 8 бит выхода интерпретировать как число, которое в дальнейшем будет номером варианта от 0 до 255. В отчёт включить снимок экрана с выбором номера варианта, а также описать шаги решения задачи.

Разложить каждое из чисел `a` и `b` на нетривиальные сомножители.

3 Теория

В данной лабораторной работе рассматривается задача разложения больших чисел на нетривиальные сомножители. Эта задача лежит в основе криптографической стойкости многих современных алгоритмов шифрования, особенно тех, которые основаны на трудности факторизации вроде RSA.

Нетривиальными сомножителями числа называют делители числа, которые не равны единице и самому числу. На практике, чем больше число, тем сложнее найти его делители. Например, если число имеет сотни цифр, известные классические алгоритмы могут работать неприемлемо долго. К таким алгоритмам относятся:

- Алгоритм ρ Полларда (эффективен для малых чисел),
- Квадратичное решето (подходит для чисел до 100 цифр),
- Общий метод числового поля (GNFS, используется для факторизации чисел в несколько сотен цифр),
- Алгоритм Шора (теоретически эффективен на квантовом компьютере)
- И другие.

Для нахождения сомножителей в данной лабораторной использовалась функция factorint из библиотеки sympy, реализующая комбинацию нескольких подходов.

4 Ход лабораторной работы

Для поиска собственного варианта использовалась хэш-функция Стрибог-256 из библиотеки *gostcrypto*. ФИО сначала кодировалось в байты, затем хэшировалось, а после последние два символа интерпретировались в шестнадцатеричное число, что дало мне вариант *94*.

Числа из варианта были скопированы в файл *input.txt*.

Для факторизации, т. е. поиска нетривиальных сомножителей, как и было сказано выше, использовалась функция *sympy.factorint*. Функция возвращает словарь, где ключи — множители, а значения — количество вхождения множителя, поэтому из вывода данной функции была взята только пара ключей.

Далее происходила факторизация чисел *a* и *b* и вывод результатов программы. Нетривиальными сомножителями числа *a* являются числа 12287045022171495721 и 627710173538668076383578942320766641610235544446403451302 9.

Поскольку число *b* огромное и содержит 617 цифр, для него вычисления невозможны. Или у нас нет времени на это времени в обозримом будущем. Если верить интернету, то *b* больше самого большого факторизированного на данный момент числа примерно в три раза и нам понадобится около миллиарда лет с мощностями самых серьёзных суперкомпьютеров для нахождения результата.

Исходя из этого, можно сделать вывод, что это настоящее чудо, что мои коллеги запросто смогли найти сомножители своих чисел *b*.

Или же они заметили закономерность в заведомо сгенерированных данных.

Код программы:

```
from gostcrypto import gosthash
from sympy import factorint

def get_variant(fio: str) -> int:
    fio_bytes = fio.encode('utf-8')
    hash_obj = gosthash.new('streebog256')
    hash_obj.update(fio_bytes)
    hash_result = hash_obj.hexdigest()
    variant_number = int(hash_result[-2:], 16)
    return variant_number

def get_numbers(filepath: str) -> tuple[int, int]:
    with open(filepath, 'r', encoding='utf-8') as f:
        a, b = map(int, f.readlines())
    return a, b

def get_pair_factors(num: int) -> tuple[int, int]:
    return tuple(
        int(num)
        for num in factorint(num).keys()
    )

if __name__ == '__main__':
    variant = get_variant(
        "Караев Тариел Жоомартбекович"
    )
    a, b = get_numbers('input.txt')

    print(f'Мой вариант: {variant}')

    print(f'a[{variant}] = {a}')
    print(f'b[{variant}] = {b}')

    print(f'Цифр в числе a: {len(str(a))}')
    print(f'Цифр в числе b: {len(str(b))}')

    print(
        'Нетривиальные сомножители a: '
        f'{get_pair_factors(a)}'
    )
```

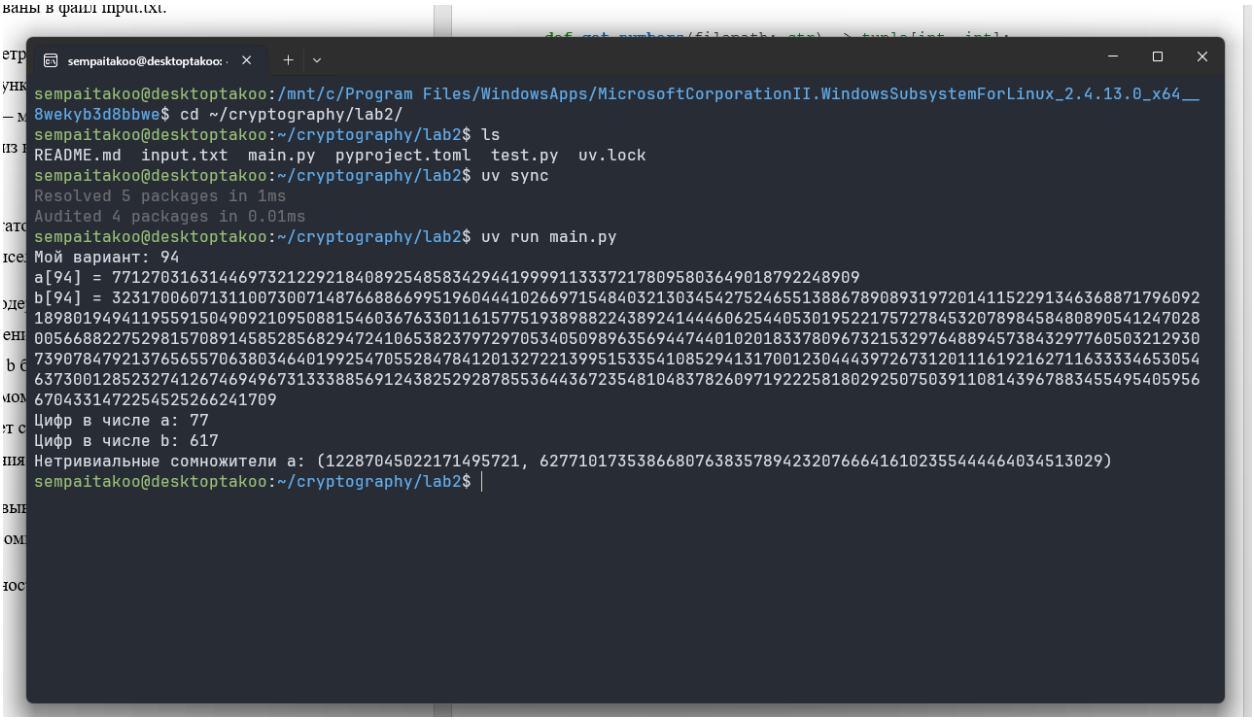

Рисунок 1 Вывод программы

5 Выводы

В ходе выполнения лабораторной работы было выяснено, как сильно усложняется задача факторизации при увеличении заданного числа. Такая трудоёмкая задача может использоваться в криптографии для односторонних вычислений.

6 Список используемой литературы

- https://en.wikipedia.org/wiki/Integer_factorization
- https://en.wikipedia.org/wiki/One-way_function
- https://docs.sympy.org/latest/modules/ntheory.html#sympy.ntheory.factor_.factorint