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-307Б-21

Студент: Ф.А.Меркулов

Преподаватель: А. В. Борисов

Оценка:

Дата: 26.03.2024

Москва, 2024

ОГЛАВЛЕНИЕ

1 Тема

Факторизация чисел

2 Задание

Разложить каждое из чисел n_1 и n_2 на нетривиальные сомножители.

15)

n1=611219701749111463195451937544251195208759452152827846401772765
23929376501913,

n2=166981202821111487603574159347402180221234004474088470134427127
019583208585679731493672560996991989288043247004768446454156726533
680678895840262535052207221535688754234509196536441271441614772300
782485294043921034753549207993093871530185166350490763327117821598
668749628167305979543150020080011233737488865764293201137701077973
963199041171488573736171460271539763898264613648163023841948180886
443891137180408521294684019855844147917625683268960047666893086522
2709

3 Теория

Факторизация числа — это запись числа как произведения нескольких факторов, обычно меньших или более простых чисел.

Например, 3 × 5 — это факторизация целого числа 15

Задача факторизации целого числа N заключается в нахождении разложения его в произведение простых сомножителей.

$$N = p * q, N \approx 10^{60}$$

Сложность решения задачи факторизации лежит в основе криптостойкости некоторых алгоритмов шифрования с открытым ключом, таких как RSA.

4 Ход лабораторной работы

В самом начале я был наивен и как понял, что задача состоит в нахождение простых делителей числа решил написать код, находящий делители с помощью Решета Эратосфена, не подумав с насколько большими числами мне придётся работать, в итоге словил Memory Limit и переписал программу через проверку всех простых чисел с шагом 6 (все простые числа кроме 2, 3 находятся на позициях 6k – 1 или 6k + 1, $k \in \mathbb{N}$) являются ли они делителями исходного числа, но и тут я допустил фатальную ошибку так как при выполнении $\sim 10^9$ операций в секунду моя программа потребовала бы $\sim 10^{29}$ секунд, то есть примерно 10^{23} лет, что чуть-чуть больше чем я могу выделить под работу программы.

В итоге я понял, что даже если можно написать алгоритм своими руками, который бы делал это, то это было бы не тривиально и это 100% кто-то делал уже до меня и я решил не изобретать велосипед и обратиться к сайтам, которые занимаются факторизацией чисел:

1) Integer factorization calculator

List of divisors:

- 1
- 212 453377 902490 714807 152598 462337 631023 (39 digits)
- 287 695920 763209 375310 868462 625502 197431 (39 digits)
- 61121 970174 911146 319545 193754 425119 520875 945215 282784 640177 276523 929376 501913 (77 digits)

Откуда видно, что простыми делителями n_1 являются числа:

- 212453377902490714807152598462337631023
- 287695920763209375310868462625502197431

2) [Msieve Factorizer – CryptoTool Portal]

Found factors: 2

Factorized number: 611219701749111463195451937544251195208759452152827846401772765239293765019 13

212453377902490714807152598462337631023

287695920763209375310868462625502197431

На всякий случай я решил проверить является ли произведение этих чисел исходным числом n_1:

main.py +

```
1 a = 212453377902490714807152598462337631023 * 287695920763209375310868462625502197431
2 print(a == 611219701749111463195451937544251195208759452152827846401772765239293765019 13)
```

И в консоли получил заветный:

> True

Всё замечательно отработало, моей радости не было конца, только меня удручало, что на каком бы сайте я не делал такое маленькое число, по сравнению с n_2, оно отрабатывало в районе 5 минут, что достаточно плохо, так как n_2 на 386 порядков больше n_1. То есть даже если не учитывать ошибки, которые выдают сайты при попытке факторизовать n_2 эта операция заняла бы больше лет, чем есть атомов во всей видимой вселенной. Я читал разные статьи про квадратичные методы, асимптотики методов факторизации чисел и с таким большим числом ни один алгоритм не мог отработать за время, отведённое до дедлайна лабораторной работы.

В итоге я услышал, что мои однокурсники уже сдали эту лабораторную работу, я посмотрел в варианты и понял, что все в одинаковых условиях и мне стало очень интересно как же они смогли факторизовать второе число. В одном из разговоров со знакомыми выяснилось, что при факторизации второго числа первый множитель находится как НОД с числом другого варианта, а второй множитель – простым делением изначального числа n_2 и первого множителя. После такой подсказки я очень быстро смог найти ответ на поставленную задачу.

Я сделал программу на Python реализующую данную идею и получил ответ:

1)

164002213211582449439345103101010580261572585169552138300989634438730183511532326279004028254034511611436674562580240536376226692152792624551569104138918952496750359796132275664874619889146598902097542761623680838097442518495171412837429189216783713021962416264736667759184361105350899864242921476311929824199

2)

101816432565873843207798197825302055928241499243422284060419134254479162392047499093755438933732248628951777130923340269983780178064697032344900243462594 91

5 Выводы

В ходе выполнения данной лабораторной работы я ознакомился с основными алгоритмами и методами решения задачи факторизации числа. Я рассмотрел использование такой задачи на практике и ознакомился с реальными примерами.

Также, я на практике убедился, что не всегда «прямое» решение сможет привести к какому-либо результату за разумное время.

Задача факторизации числа с использованием простых делителей является отличным примером одной из главный идей криптографических функций, которые часто ассоциируют с их кратким описанием «легко вычисляются в одну сторону и неразумно сложно в обратную».

Использование таких функций, подходов, методов оказывает огромное влияние на криптостойкость и эффективность шифров, так как, с одной стороны, мы можем быстро зашифровать или расшифровать данные, имея нужную нам информацию, но получить такую информацию для злоумышленников становится практически нерешаемой задачей за отведённое им время.

6 Список используемой литературы

1. Видео про ассиметричное шифрование: https://youtu.be/qgofSZFTuVc?si=sBlzrXmsqN6nXnik
2. Статья с описанием алгоритма RSA: https://habr.com/ru/articles/745820/
3. https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%86%D0%B5%D0%BB%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB
4. Статья с описанием алгоритма Ферма на Wikipedia: https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%A4%D0%B5%D1%80%D0%BC%D0%B0
5. https://habr.com/ru/sandbox/163811/
6. https://algorithmica.org/ru/pollard