**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-312Б-22

Студент: Куценко М.Д.

Преподаватель: А. В. Борисов

Оценка:

Дата: 07.03.2025

Москва, 2025

ОГЛАВЛЕНИЕ

[**1** **Тема** 3](#_Toc194872173)

[**2** **Задание** 3](#_Toc194872174)

[**3** **Теория** 4](#_Toc194872175)

[***ГОСТ Р 34.11-2012 «Стрибог»*** 4](#_Toc194872176)

[**Функция сжатия** 4](#_Toc194872177)

[**Описание** 4](#_Toc194872178)

[**Алгоритмы факторизации** 5](#_Toc194872179)

[**4** **Ход лабораторной работы** 7](#_Toc194872180)

[**Выбор варианта** 7](#_Toc194872181)

[**Факторизация первого числа** 9](#_Toc194872182)

[**Факторизация второго числа** 10](#_Toc194872183)

[**5** **Выводы** 12](#_Toc194872184)

[**6** **Список используемой литературы** 13](#_Toc194872185)

# **Тема**

Разложение чисел на нетривиальные сомножители.

# **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, а также описать шаги решения задачи.

Задача: разложить каждое из чисел `a` и `b` на нетривиальные сомножители.

# **Теория**

## ***ГОСТ Р 34.11-2012 «Стрибог»***

Стрибог — это семейство хеш-функций, включающее в себя всего две функции. Функцию с длиной выходного значения в 256 бит и функцию с длиной выходного значения в 512 бит. Обе эти функции имеют одинаковую структуру и отличаются друг от друга только начальным внутренним состоянием. Входными данными для обеих функций является блок данных длиной 512 бит. В случае, если длина сообщения больше 512 бит, то происходит разбиение сообщения на блоки. В случае же, если длина меньше 512 бит, то производится дополнение сообщения.

### **Функция сжатия**

В хеш-функции важным элементом является функция сжатия. В ГОСТ Р 34.11-2012 функция сжатия основана на конструкции Миагути — Пренеля. Схема конструкции Миагути — Пренеля: h, m — вектора, поступающие на вход функции сжатия; g(h, m) — результат функции сжатия; E — блочный шифр с длиной блока и ключа 512 бит. В качестве блочного шифра в хеш-функции ГОСТ Р 34.11-2012 взят XSPL-шифр. Этот шифр состоит из следующих преобразований:

• сложение по модулю 2;

• преобразование замены или подстановки. Обозначается S-преобразование;

• преобразование перестановки. Обозначается P-преобразование;

• линейное преобразование. Обозначается L-преобразование.

Преобразования, используемые в новой хеш-функции, должны быть хорошо изучены. Поэтому в блочном шифре E используются преобразования X, S, P, L, которые хорошо изучены. Важным параметром блочного шифра является то, как выбирается ключ, который будет использовать на каждом раунде.

### **Описание**

В основу хеш-функции положена итерационная конструкция Меркла — Дамгора с использованием MD-усиления. Под MD-усилением понимается дополнение неполного блока при вычислении хеш-функции до полного путём добавления вектора (0 … 01) такой длины, чтобы получился полный блок. Из дополнительных элементов нужно отметить следующие:

* завершающее преобразование, которое заключается в том, что функция сжатия применяется к контрольной сумме всех блоков сообщения по модулю 2512;
* при вычислении хеш-кода на каждой итерации применяются разные функции сжатия. Можно сказать, что функция сжатия зависит от номера итерации.

Описанные выше решения позволяют противостоять многим известным атакам.

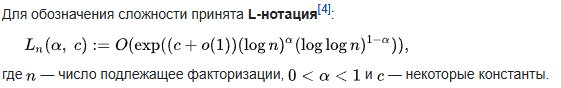
Кратко описание хеш-функции ГОСТ Р 34.11-2012 можно представить следующим образом. На вход хеш-функции подается сообщение произвольного размера. Далее сообщение разбивается на блоки по 512 бит, если размер сообщения не кратен 512, то оно дополняется необходимым количеством бит. Потом итерационно используется функция сжатия, в результате действия которой обновляется внутреннее состояние хеш-функции. Также вычисляется контрольная сумма блоков и число обработанных бит. Когда обработаны все блоки исходного сообщения, производятся ещё два вычисления, которые завершают вычисление хеш-функции:

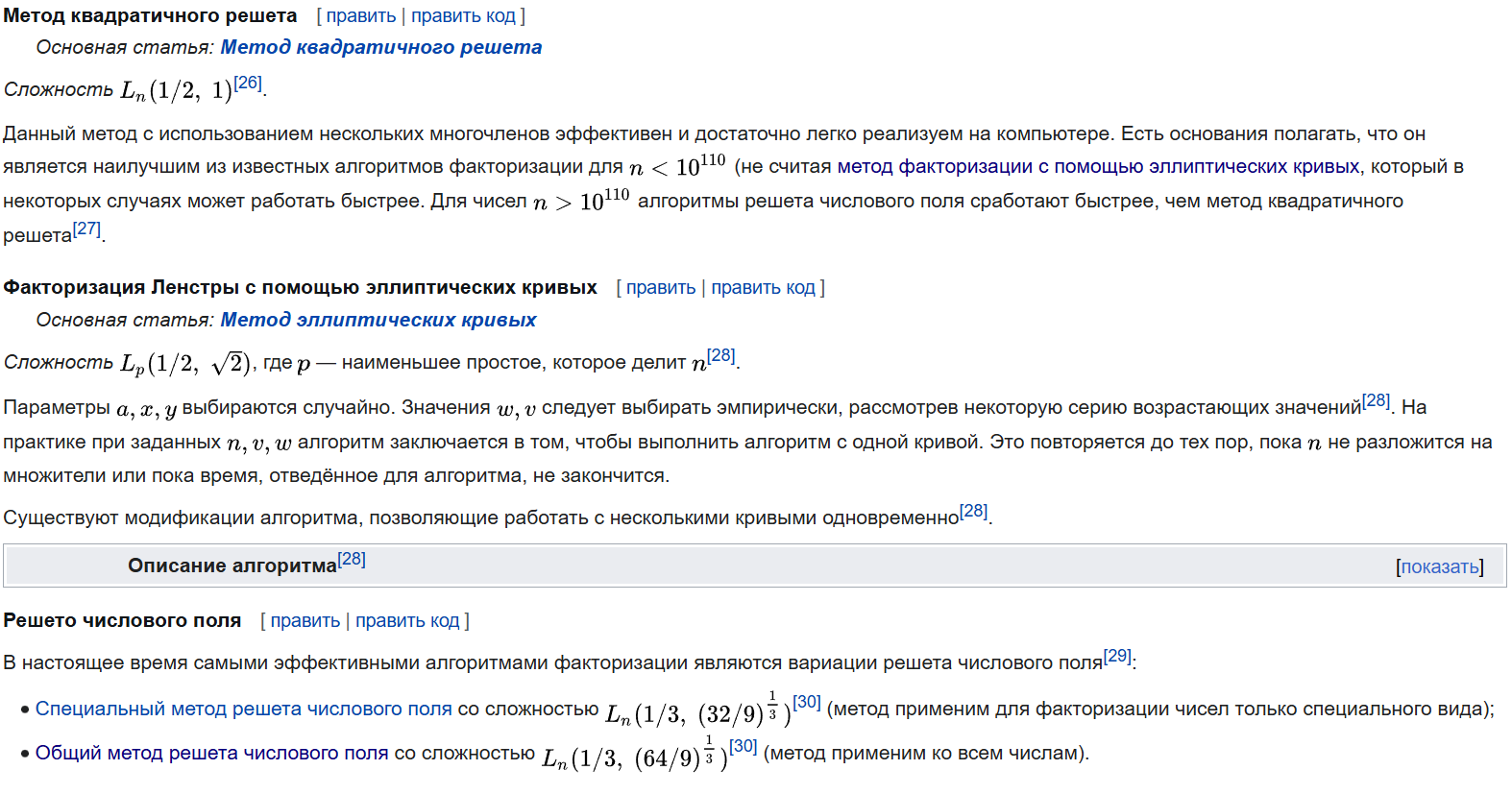
* обработка функцией сжатия блока с общей длиной сообщения.
* обработка функцией сжатия блока с контрольной суммой.

В работе Александра Казимирова и Валентины Казимировой приведена графическая иллюстрация вычисления хеш-функции.

## ***Алгоритмы факторизации***

Рассмотрим 3 наиболее быстрых алгоритма – метод квадратичного решета, метод эллиптических кривых и общий метод решета числового поля:





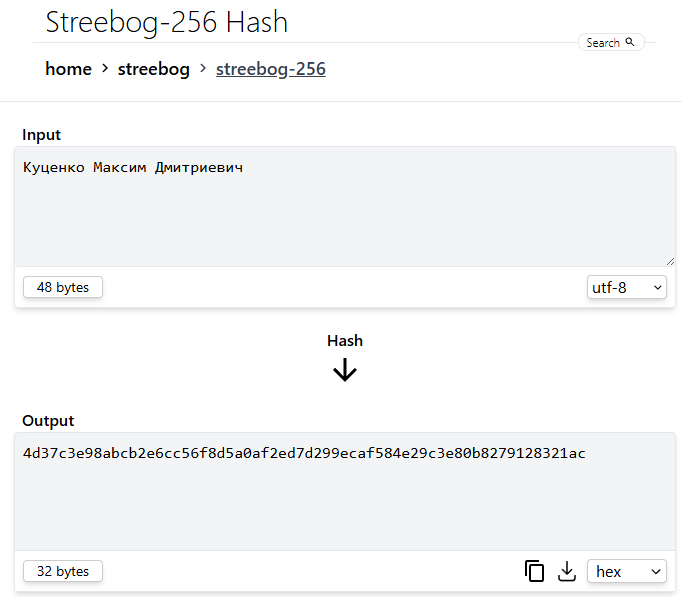
# **Ход лабораторной работы**

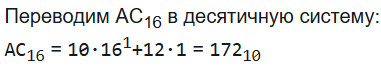
## ***Выбор варианта***

Используем библиотеку gosthash языка Python, имеющую реализации алгоритма Стрибога с длиной выходного сообщения 512 бит. С помощью представленной ниже реализации найдём номер варианта:

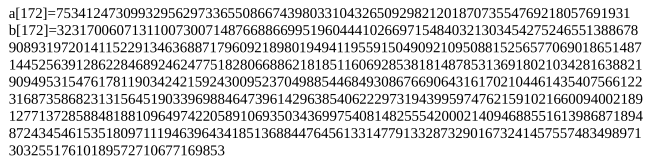


Проверим правильность нахождения при помощи средств сети интернет:

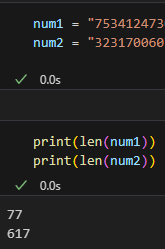




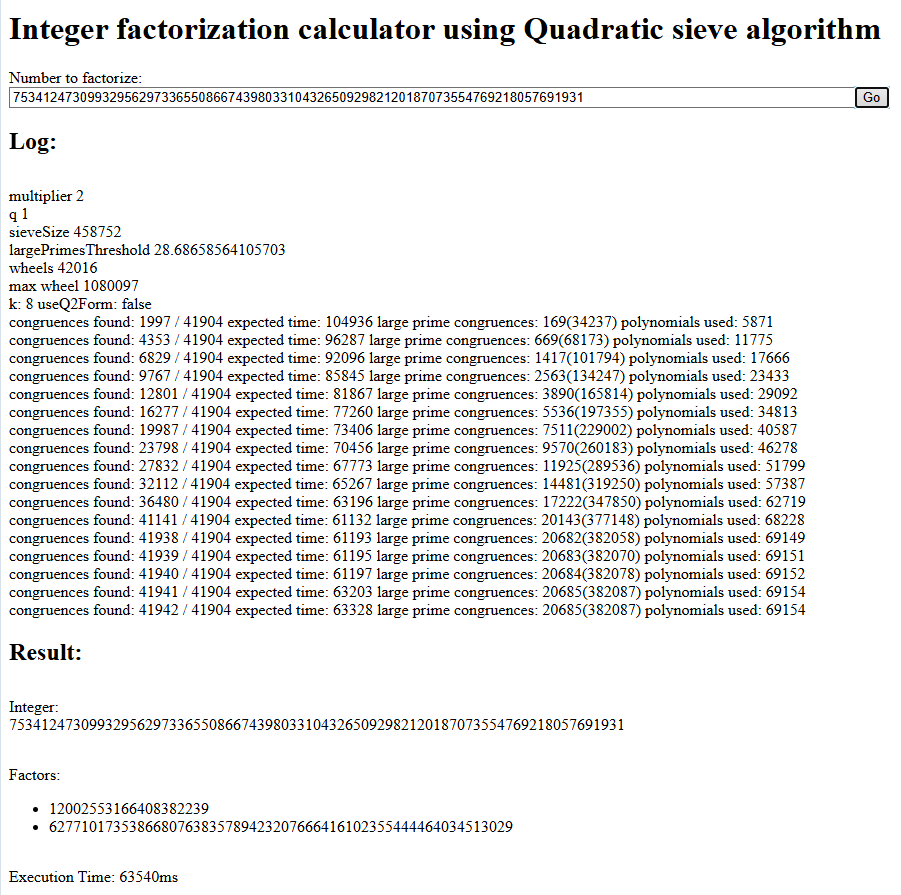
Решения совпали, выбираем вариант 172.



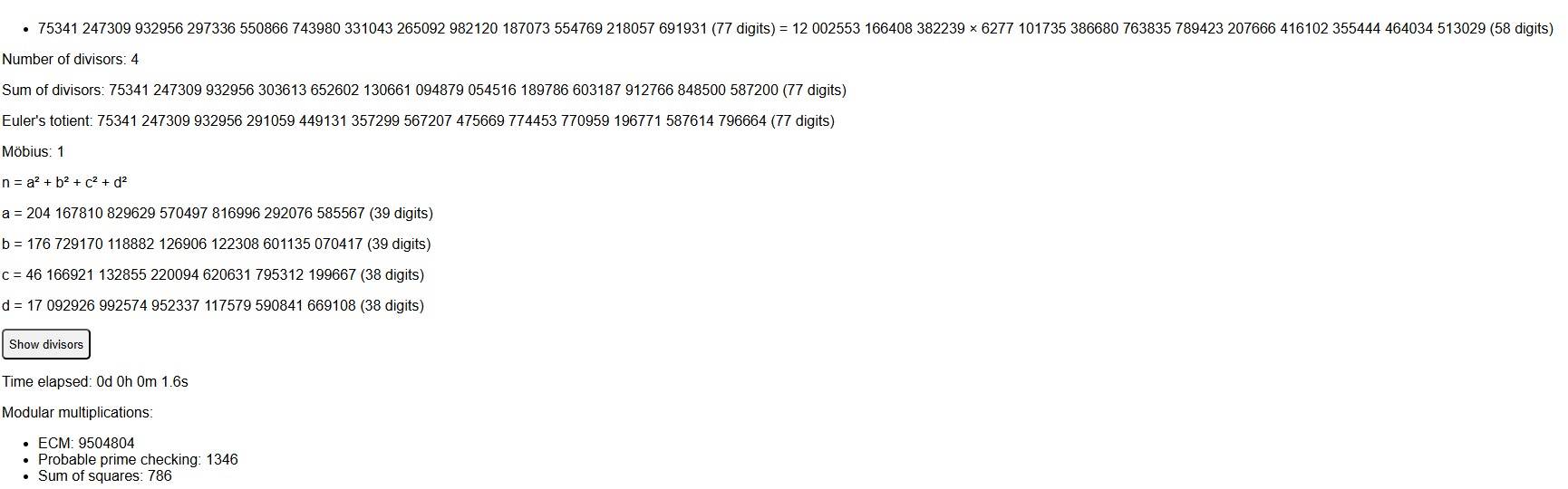
## ***Факторизация первого числа***

Рассмотрим размеры чисел.

Судя по данной информации, быстрее всего для первого числа (меньше 110 символов) должны работать методы квадратичного решета и эллиптических кривых.

Применим метод квадратичного решета:

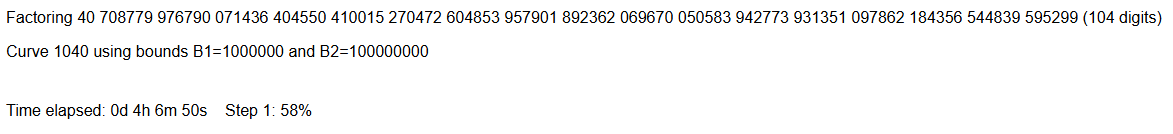
Метод отработал медленно – 63 секунды, возможно причина в низкой производительности серверов. Применим метод эллиптических кривых:



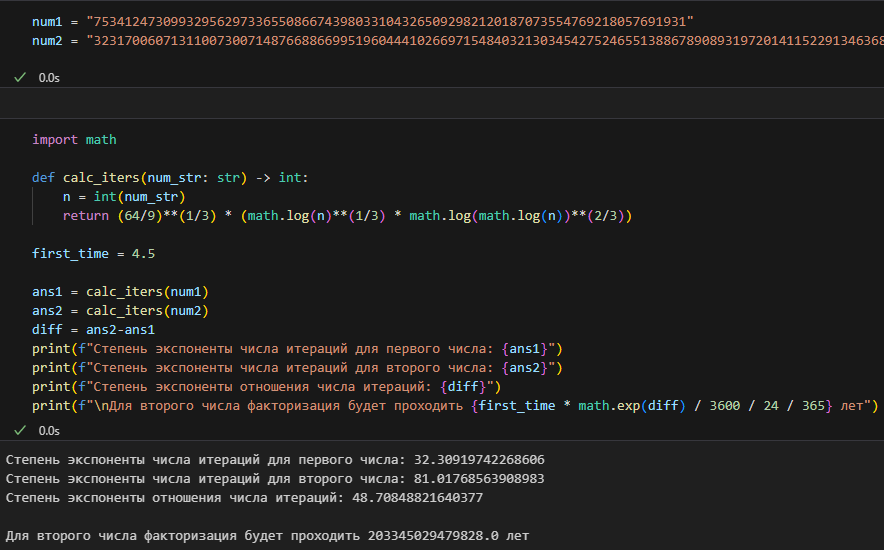
Метод эллиптических кривых решил задачу факторизации за 1.6 секунды.

## ***Факторизация второго числа***

Для второго числа быстрее всего будет работать общий метод решета числового поля – GNFS. При попытке факторизации сталкиваемся с отсутствием результатов в течение долгого времени:



Оценим, сколько будет выполняться факторизация на основе результатов для первого числа. Получим, что для первого числа факторизация заняла 4.5 секунды, тогда используем эту информацию:

Таким образом, факторизация таких чисел невозможна на практике с текущим уровнем развития технологий.

# **Выводы**

Изучил алгоритм хеширования «Стрибог», а также наиболее эффективные методы факторизации чисел. Применил данные знания на практике во время выполнения данной лабораторной работы.

Пришёл к выводу, что факторизация больших чисел практически невозможна вследствие чрезмерно большого периода выполнения, что делает основанный на данном принципе алгоритм шифрования RSA безопасным.

# **Список используемой литературы**

1. Применко Э.А. Алгебраические основы криптографии: Учебное пособие. — М.: Книжный дом «ЛИБРОКОМ», 2013. — 288 с.
2. Грушо А.А., Применко Э.А., Тимонина Е.Е. Теоретические основы компьютерной безопасности: учеб. пособие для студентов высш. учеб. Заведений. — М.: Издательский центр «Академия», 2009. — 272 с.
3. <https://ru.wikipedia.org/wiki/Стрибог_(хеш-функция)>
4. <https://ru.wikipedia.org/wiki/Факторизация_целых_чисел>
5. <https://hashing.tools/streebog/streebog-256>
6. <https://programforyou.ru/calculators/number-systems>
7. <https://yaffle.github.io/QuadraticSieveFactorization/demo.html>
8. <https://www.alpertron.com.ar/ECM.HTM>
9. <https://legacy.cryptool.org/en/cto/msieve>