Муниципальное автономное общеобразовательное учреждение

«Средняя общеобразовательная школа № 5 им. Л.Н. Гумилева»

города Бежецка Тверской области

Проектно-исследовательская работа

Тема: «Шифр RSA»

Выполнил:

Кудрявцев Александр,

обучающийся 10 класса

Руководитель:

Искра Елена Владимировна,

преподаватель математики

г. Бежецк

2022г.

Оглавление

[Введение 3](#_Toc92499827)

[Глава 1. Математическая теория и принцип шифрования 5](#_Toc92499828)

[Глава 2. Программа 12](#_Toc92499829)

[Заключение 13](#_Toc92499830)

[Список использованных источников 14](#_Toc92499831)

# Введение

Математическая дисциплина нашла применение во всех отраслях нашей жизни, в том числе и защите нашей информации. Особую роль в этом сыграло появление теории чисел – раздела математики, изучающего поведение чисел и их особенности. Именно знания, которые человек извлек из теории чисел, и легли в основу криптографии.

Теория чисел – это сложнейший раздел математики. Во-первых, он призван обобщить все наши математические знания. Во-вторых, здесь используется самые нестандартные методы решения и доказательств: геометрия соседствует с комбинаторикой, а комплексный анализ с дискретной математикой.

Однако в данной исследовательской работе я коснусь только азов теории чисел и больший упор сделаю на основной и главный шифр на данный момент – на шифр RSA.

Шифр RSA – это то, что мы используем, не зная об его наличии. Каждый перевод «Сбербанка» осуществляется через этот шифр. Для нашего же блага все сообщения «Whatsapp» проходят через RSA. Военные активно используют этот шифр: поговаривают, что даже координация наших ядерных вооружений осуществляется через RSA.

В первой главе я коснусь основ теории чисел, необходимые для этого шифра, расскажу про сам шифр и про его создателей, а также детально освящу его проблемы.

Во второй главе я представил вам мою реализацию шифра на языке программирования Python. Там Вы можете увидеть код, который я пометил комментариями для удобства восприятия. В ссылках Вы можете найти мой профиль на сайте Github (это социальная сеть программистов), где можно скачать программу, запустить и зашифровать свое сообщение.

**Цель:**

Разобрать работу шифра RSA и создать программу, реализующую его.

**Актуальность:**

Как говорилось во введении, шифр RSA нашел широкое применение во всех сферах нашей жизни: финансы, военная сфера, хранение данных, mobile, desktop и web разработка и т.д. и т.д.

# Глава 1. Математическая теория и принцип шифрования

***Простое число*** – это число, которое можно разложить только на 1 и само это число.

***Составное число*** – это число, которое можно разложить на множители, отличные от 1 и самого это числа

***Основная теорема арифметики*** – каждое число можно представить в виде произведения простых чисел, притом единственным образом:

***Каноническое разложение*** – это представление числа в виде произведения простых множителей с использованием степени:

***Общий вид числа:***

Зная каноническое разложение можно составить общую формулу всех его делителей:

Тогда все его делители могут иметь вид:

***Формула количества делителей:***

***Взаимно простые числа*** – это пара чисел, у которых значение НОД равно единице.

Свойства взаимно простых чисел:

* Если число делится на пару взаимно простых чисел по отдельности, то оно делится на произведение этих чисел.
* Если произведение взаимно простых чисел делится на какое-то число, то только один из множителей делится на этого число.

***Остаток от деления r*** – это один из результатов деления, когда числа не делятся нацело, остаток от деления всегда меньше по модулю частное q.

Любое числа a можно представить в виде **,** где

Нахождение остатка от деления обозначаются символом процента (это историческая практика, которой начались из-за программирования и внутренней реализации вычисления остатка, однако этот символ не должен Вас путать).

Пример:

Остатки от деления очень любят в криптографии, так как остаток от деления – это ***однонаправленная функция.*** Имея результат функции, мы не можем сказать о ее переменной что-то конкретное.

В школьной математике, все ***функции двунаправлены***.

***Шифр RSA*** – это ассиметричный шифр. В отличие от симметричных шифров, где ключ только один, в нем есть два ключа.

Можно привести жизненную аналогию для осознания ассиметричных шифров: *Вам нужно отправить сообщение своему другу. Однако сначала он вам высылает коробку, в которой Вы положите свое письмо. Ключа от коробки у Вас нет – коробка закроется, когда просто ее захлопните. Открыть ее может только друг с ключом.*

По сути, здесь было два ключа: коробка – это открытый ключ, ключ от коробки – закрытый (приватный) ключ.

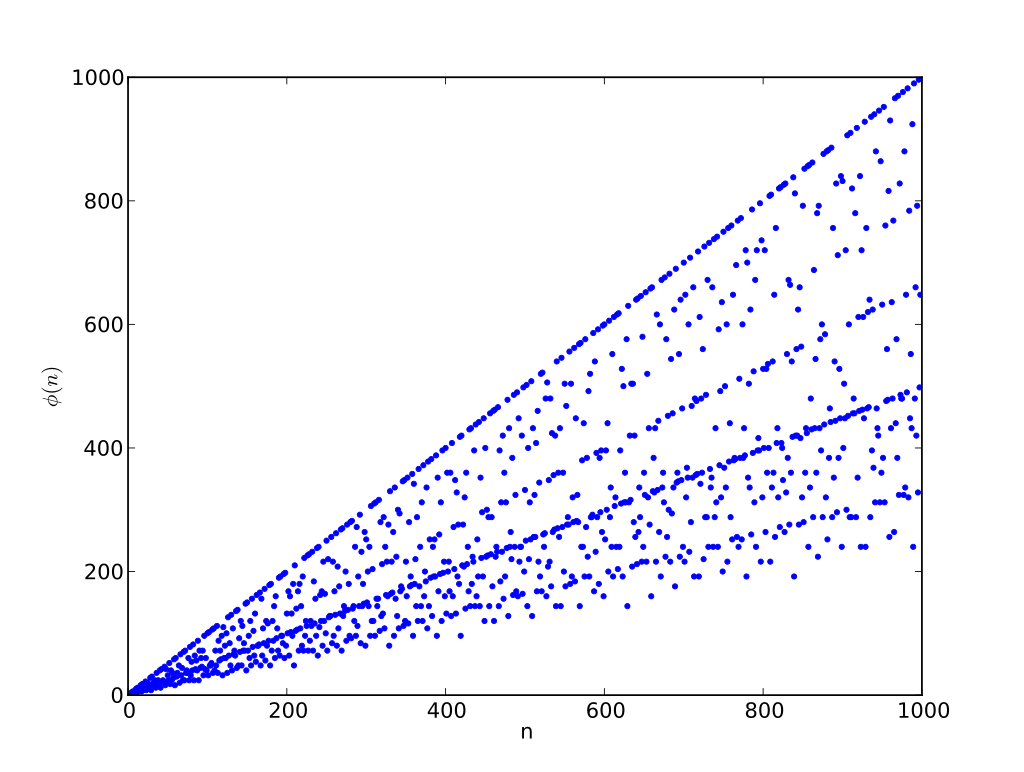
Это все аналогии, которые на самом деле являются сложными математическими понятиями. Так давайте рассмотрим само шифрование:

1. Нам нужно сгенерировать два простых числа и .
2. На основе этих чисел нужно вычислить их ***модуль***

Это число нам в дальнем сильно пригодится, но именно на этом этапе происходит основная защита данного шифра.

Разложить число опять на 2 простых множителей потребует очень много времени. Счет идет на десятки лет и более.

1. Дальше нам нужно поработать с ***функций Эйлера .*** Нам нужно вычислить значение этой функции, подходящей для наших простых чисел:



Однако можно поступить по-человечески:

1. Находим ***открытую экспоненту***  (не путать с числом Эйлером!)

Ее находят методом перебора, основываюсь на 3 условиях:

* Данное число простое
* Данное число меньше
* Данное число взаимно простое с числом

1. Теперь у нас есть открытый ключ , остается найти секретный ключ.
2. Число – это обратное по модулю , т.е. это такое число, при котором:

Данное число можно получить методом подбора.

1. Теперь у нас есть закрытый ключ
2. Шифрование происходит по следующему принципу:

1. Представляем текст в виде чисел, например, «А» - 1, «Б» - 2 и т.д. и т.д.

2. Каждое полученное число возводим в степень , а затем берем остаток от деления на

3. Полученное число и есть зашифрованный символ

1. Дешифрование происходит по следующему принципу:

1. Возводим число в степень и берем остаток от деления на

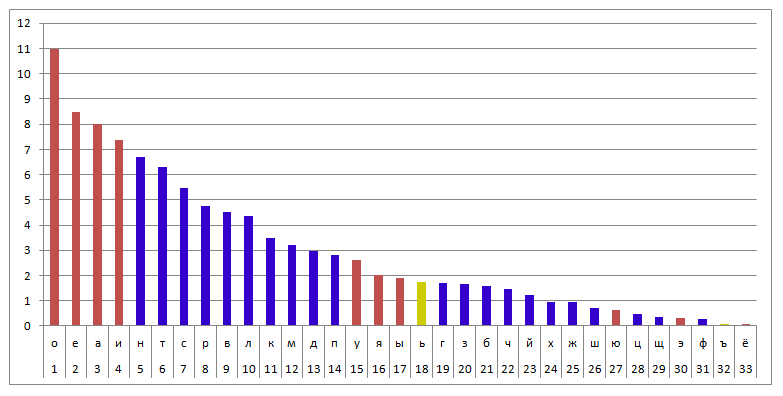
2. Полученное число и есть оригинальный символ.

Это и есть весь алгоритм шифрования RSA, который придумали 1982 Рональд Ривест, Ади Шамир и Леонард Адлеман. В названии шифра зашифрованы первые буквы их фамилий.

Несмотря на все выше перечисленное, у алгоритма есть ряд недостатков, которые могут проявить себя в некоторых ситуациях. Он подвержен частотному анализу, а также его работоспособность могут отменить квантовые компьютеры.

***Частотный анализ***. В рассказе «Пляшущие человечки», Шерлок Холмс, легендарный детектив использовал это для расшифровки сообщения.

Как известно, некоторые буквы встречаются чаще остальных, а некоторые и вообще редки в использовании. Так, «О» мы используем в письме чаще всего, а буквы «Ф» и «Ъ» практически нет.



Узнать частоту символа можно при помощи ***таблиц частотности.*** Для каждого языка они разные, а со временем они могут даже и изменяться.

Имея на руках зашифрованный текст, мы можем восстановить оригинальное содержимое:

* Находим самый часто используемый символ. Сравниваем с таблицей частотности и получаем, что это «О»
* И так до тех пор, пока мы не получим на руки весь текст

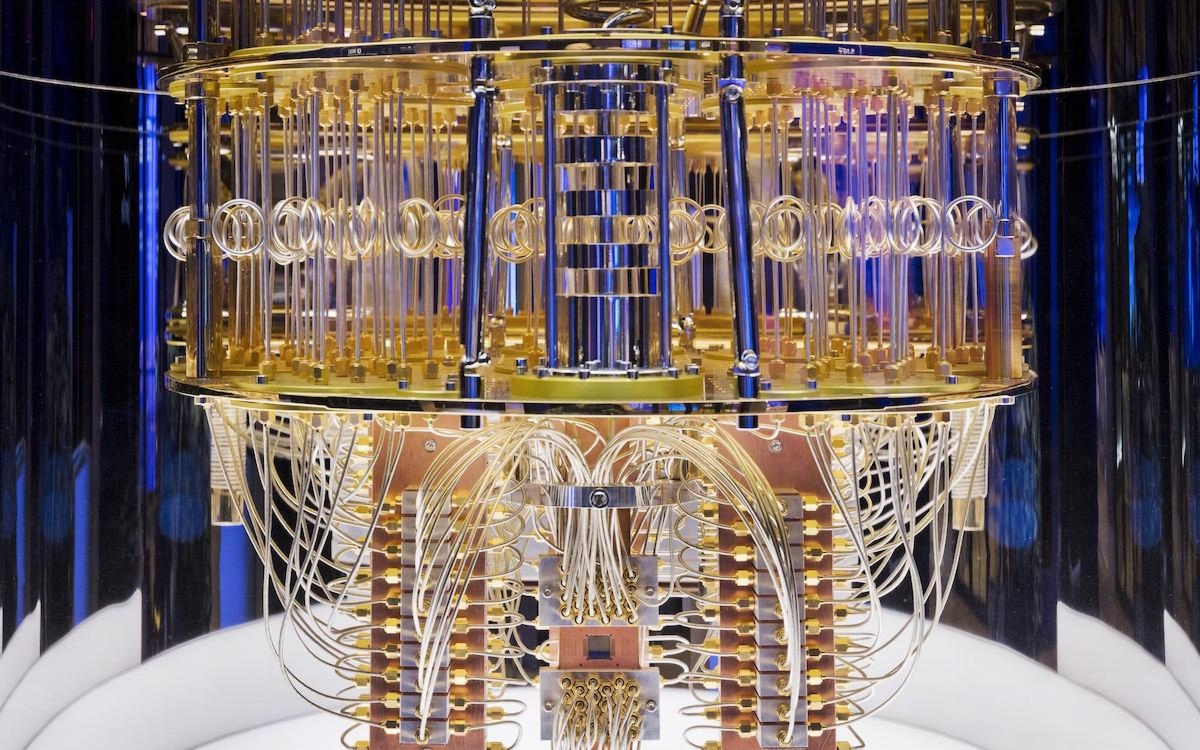
Звучит очень устрашающе, но на самом деле полученный текст может получиться бредом.

***Частотный анализ используется, когда нужно взломать длинный текст.***

Особенно будет хорошо, если бы у нас была таблица частотности конкретного человека: каждый человек использует разный лексикон, поэтому и частность символов его письменной речи будет иной.

Так, я в письме очень активно использую букву «О» - 14,5%, а букву «Ё» я не использую вовсе: 1/250 000 00 от процента.

Данный метод хорошо пойдет, когда защищена только часть информации, а все остальные у нас на руках. Кроме того, он хорошо работает, когда мы говорим про переписки, поскольку там максимально естественная речь, на основе которой и создаются частотные таблицы.



Следующая проблема шифра RSA – это ***квантовые компьютеры*** – это компьютеры, которые основаны на фотонах, благодаря чему в одном реестре может находиться и 0, 1 одновременно. Из-за этого скорость выполнения программ на таких компьютерах в сотни тысяч раз быстрее.

Как говорилось, что ключ есть псевдо простое число, по которому нельзя восстановить его множители, поскольку время, требуемое на перебор всех вариантов, слишком большое.

Однако квантовые компьютеры способны преодолеть эту проблему. Разложить число на множители на них можно будет за максимально короткие сроки: если число бы раскладывалось на обычном цифровом компьютере за пару месяцев, то на квантовом за пару часов.

Но квантовые компьютеры, как и квантовые вычисления, есть часть нашего будущего, которое может и не случиться. Квантовые вычисления не так точны: очень часто они выдают неправильный ответ. А цена квантового компьютера несоизмерима дороже цены обычного устройства.

# Глава 2. Программа

Ниже изложен код программы, реализующий этот шифр:

import random

import math

# Функция, определяющая простоту числа

def is\_prime(n):

    if n % 2 == 0 and n > 2:

        return False

    elif n < 2:

        return False

    for i in range(3, int(math.sqrt(n)) + 1, 2):

        if n % i == 0:

            return False

    return True

# Функция, определяющая взаимную простоту

def is\_coprime\_numbers(a, b):

    while b != 0:

        a, b = b, a % b

    return a == 1

# Генератор простого числа

def generate\_prime():

    n = int(random.random()\*10000)

    while not is\_prime(n):

        n += 1

    return n

# Класс, реализующий публичный ключ

class PublicKey:

    def \_\_init\_\_(self, e, n):

        self.e = e

        self.n = n

    def encrypt(self, num):

        return pow(num, self.e, self.n) # Используем возведение в степень с по модулю

# Класс, реализующий закрытый ключ

class PrivateKey:

    def \_\_init\_\_(self, d, n):

        self.d = d

        self.n = n

    def decrypt(self, num):

        return pow(num, self.d, self.n)

# Генератор ключей:

def createKeys():

    p = generate\_prime()

    q = generate\_prime()

    n = p \* q

    f = (p-1)\*(q-1)

    e = None

    for i in range(1, f):

        if is\_prime(i) and is\_coprime\_numbers(i, f):

            e = i

            break

    d = None

    for i in range(1, n):

        if (i \* e) % f == 1:

            d = i

    return PublicKey(e, n), PrivateKey(d, n)

def convert\_to\_str(list):

    res = ""

    for i in list:

        res += chr(i)

    return res

# Интерфейс программы:

if \_\_name\_\_ == "\_\_main\_\_":

    public, private = createKeys()

    int\_array = [ord(s) for s in input("Input message: ")]

    print("Public keys: {" + str(public.e) + "," + str(public.n) + "}")

    print("Public keys: {" + str(private.d) + "," + str(private.n) + "}")

    en\_int\_array = [public.encrypt(i) for i in int\_array]

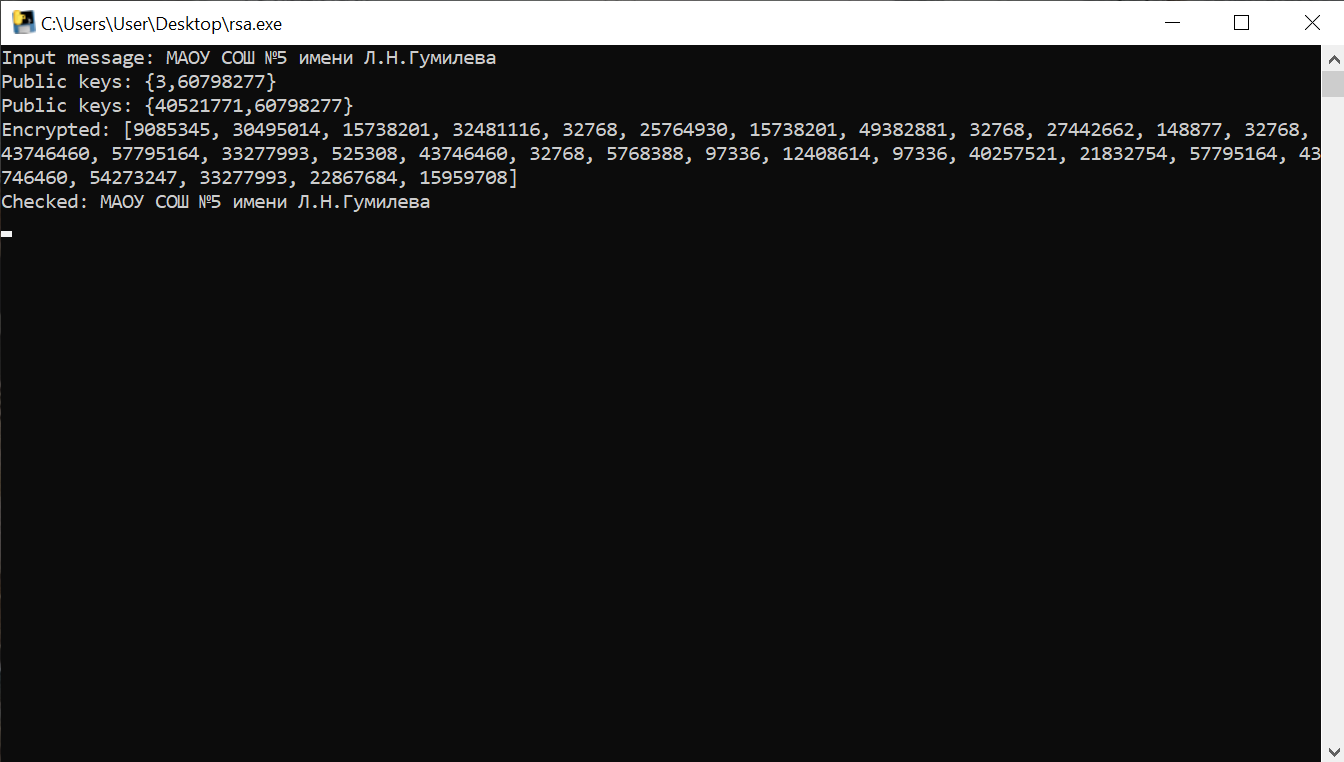
    print("Encrypted:", en\_int\_array)

    dec\_int\_array = [private.decrypt(i) for i in en\_int\_array]

    print("Checked:", convert\_to\_str(dec\_int\_array))

    input()

Пример работы:



# Заключение

В ходе этой исследовательской и практической я вплотную прикоснулся к шифрованию RSA. Мною было описано детальное его устройства и создана программа, которую можно использовать в сторонних проектах.

# Список использованных источников

В основу проекта легла 1 и 6 открытые лекция курса «Защита информации» МФТИ:

<https://www.youtube.com/watch?v=oogljMO_5wo&list=PL2jwxGybEFiuQVQtrLPaH7GNB8ak29634&index=1>

<https://www.youtube.com/watch?v=aupLf4aBjzE&list=PL2jwxGybEFiuQVQtrLPaH7GNB8ak29634&index=6>

Программа:

<https://github.com/alexKudryavtsev-web/project10class>

Методичка по теории чисел:

<https://4ege.ru/matematika/56198-teoriya-chisel.html>