

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

ОТЧЕТ  
по лабораторной работе  
на тему  
**Асимметричная криптография. Криптосистема Рабина**

Выполнил студент группы 053501  
Криштафович Карина Дмитриевна

Проверил  
ассистент кафедры информатики  
Лещенко Евгений Александрович

Минск 2023

## СОДЕРЖАНИЕ

Введение.....	3
1 Демонстрация работы программы.....	4
1.1 Шифрование.....	4
1.2 Дешифрование.....	4
2 Описание блок-схемы алгоритма.....	5
Заключение.....	7
Приложение А (обязательное) Листинг программного кода.....	8

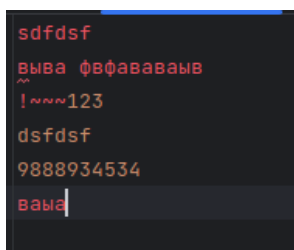
## **ВВЕДЕНИЕ**

Криптография является неотъемлемой частью информационной безопасности в современном цифровом мире. Одним из ключевых аспектов криптографии является защита информации с помощью шифрования, которое позволяет передавать данные так, чтобы они были недоступны несанкционированным лицам. Асимметричная криптография представляет собой одну из наиболее важных и широко используемых техник шифрования, которая обеспечивает высокий уровень безопасности в обмене информацией.

Одним из классических примеров асимметричной криптосистемы является криптосистема Рабина, разработанная Рональдом Л. Рабином в 1979 году. Криптосистема Рабина отличается от симметричных алгоритмов шифрования тем, что в ней используются два различных ключа: открытый и закрытый. Открытый ключ используется для шифрования информации, в то время как закрытый ключ используется для расшифровки.

# 1 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

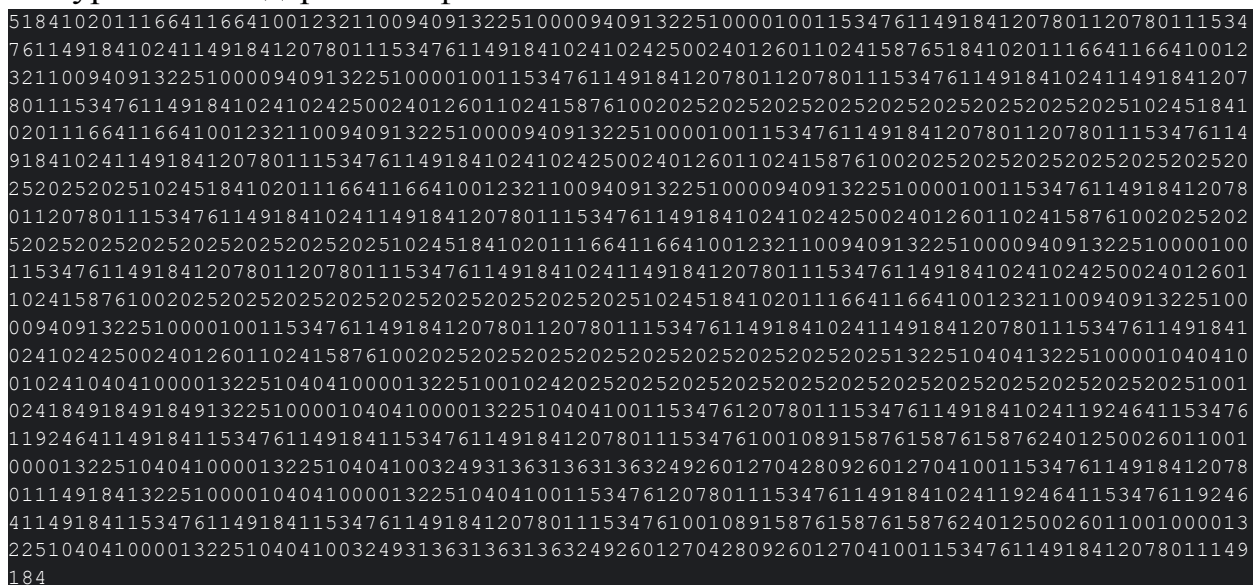
## 1.1 Шифрование



```
sdfdsf
ываа фвфавававыв
^
!~~~123
dsfdsf
9888934534
vava
```

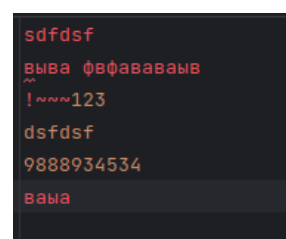
Рисунок 1 – Запись исходного текста в файл input.txt

В результате выполнения зашифрованный текст сохраняется в файл encrypted.txt. Содержимое файла:



```
5184102011166411664100123211009409132251000094091322510000100115347611491841207801120780111534
7611491841024114918412078011153476114918410241024250024012601102415876518410201116641166410012
3211009409132251000094091322510000100115347611491841207801120780111534761149184102411491841207
80111534761149184102410242500240126011024158761002025202520252025202520252025202520252025102451841
0201116641166410012321100940913225100009409132251000010011534761149184120780112078011153476114
9184102411491841207801115347611491841024102425002401260110241587610020252025202520252025202520
2520252025102451841020111664116641001232110094091322510000940913225100001001153476114918412078
0112078011153476114918410241149184120780111534761149184102410242500240126011024158761002025202
520252025202520252025202510245184102011166411664100123211009409132251000094091322510000100
1153476114918412078011207801115347611491841024114918412078011153476114918410241024250024012601
1024158761002025202520252025202520252025202520251024518410201116641166410012321100940913225100
0094091322510000100115347611491841207801120780111534761149184102411491841207801115347611491841
02410242500240126011024158761002025202520252025202520252025202520252025132251040413225100001040410
0102410404100001322510404100001322510010242025202520252025202520252025202520252025202520251001
0241849184918491322510000104041000013225104041001153476120780111534761149184102411924641153476
1192464114918411534761149184115347611491841207801115347610010891587615876158762401250026011001
0000132251040410000132251040410032493136313631363249260127042809260127041001153476114918412078
0111491841322510000104041000013225104041001153476120780111534761149184102411924641153476119246
4114918411534761149184115347611491841207801115347610010891587615876158762401250026011001000013
2251040410000132251040410032493136313631363249260127042809260127041001153476114918412078011149
184
```

## 1.2 Дешифрование



```
sdfdsf
ываа фвфавававыв
^
!~~~123
dsfdsf
9888934534
vava
```

Рисунок 2 – Запись расшифрованного текста в файл decrypted.txt

## 2 ОПИСАНИЕ БЛОК-СХЕМЫ АЛГОРИТМА

Блок-схема алгоритма представлена на рисунке 3.

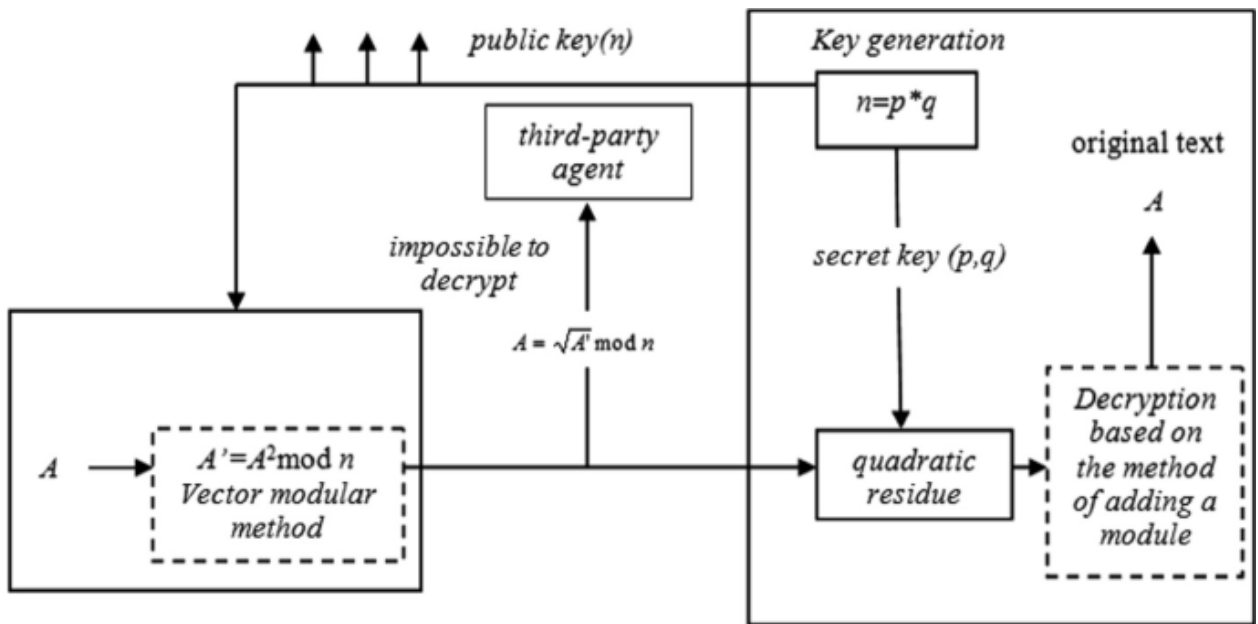


Рисунок 3 – Блок-схема работы алгоритма

Система Рабина, как и любая асимметричная криптосистема, использует открытый и закрытый ключи. Открытый ключ используется для шифрования сообщений и может быть опубликован для всеобщего обозрения. Закрытый ключ необходим для расшифровки и должен быть известен только получателям зашифрованных сообщений.

Процесс генерации ключей следующий:

- выбираются два случайных числа  $p$  и  $q$  с учётом следующих требований:
  - числа должны быть большими;
  - числа должны быть простыми;
  - должно выполняться условие:  $p \equiv q \equiv 3 \bmod 4$ .

Выполнение этих требований сильно ускоряет процедуру извлечения корней по модулю  $p$  и  $q$ ;

- вычисляется число  $n = p \cdot q$ ;
- число  $n$  — открытый ключ; числа  $p$  и  $q$  — закрытый.

Исходное сообщение  $m$  (текст) шифруется с помощью открытого ключа — числа  $n$  по следующей формуле:  $c = m^2 \bmod n$ .

Для расшифровки сообщения необходим закрытый ключ — числа  $p$  и  $q$ . Процесс расшифровки выглядит следующим образом: сначала, используя алгоритм Евклида, из уравнения  $y_p \cdot p + y_q \cdot q = 1$ . Далее, используя китайскую теорему об остатках, вычисляют четыре числа:

$$\begin{aligned} r &= (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \bmod n \\ -r &= n - r \\ s &= (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \bmod n \\ -s &= n - s \end{aligned}$$

Одно из этих чисел является истинным открытым текстом  $m$ .

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов с использованием Криптосистемы Рабина. Этот процесс включал в себя несколько важных шагов, включая генерацию ключей, шифрование и последующую дешифрацию данных.

В итоге, выполнение данной лабораторной работы позволило нам приобрести практические навыки в области асимметричной криптографии и ознакомиться с принципами работы Криптосистемы Рабина. Не смотря на свои плюсы, криптосистема обладает большим количеством минусов, в связи с чем данная криптосистема на данный момент широко не применяется, в частности стоит отметить что данная криптосистема часто применяется в академических целях. Эти знания могут быть полезными при решении задач по защите конфиденциальности данных в современном информационном мире.

**ПРИЛОЖЕНИЕ А**  
**(обязательное)**  
**Листинг программного кода**

```
import random

def modulo(a, b):
    if a >= 0:
        return a % b
    else:
        return (b - abs(a % b)) % b

def generate_key(bits):
    def is_prime(n):
        if n <= 1:
            return False
        for i in range(2, int(n ** 0.5) + 1):
            if n % i == 0:
                return False
        return True

    def generate_prime(bits):
        while True:
            num = random.getrandbits(bits)
            if num % 4 == 3 and is_prime(num):
                return num

    p = generate_prime(bits)
    q = generate_prime(bits)
    open_key = p * q
    close_key = (p, q)
    return open_key, close_key

def number_to_text(numbers):
    result = []
    for item in numbers:
        for i in range(0, len(item), 4):
            number = item[i:i + 4]
            while number[0] == '0':
```



```

        number = number[1:]

        result.append(chr(int(number)))

    return result

def extended_gcd(a, b):
    if a == 0:
        return (0, 1)
    else:
        x, y = extended_gcd(b % a, a)
        return (y - (b // a) * x, x)

def find_Yp_Yq(p, q):
    x, y = extended_gcd(p, q)
    if x < 0:
        x += q
    Yp = x
    Yq = (1 - Yp * p) // q
    return Yp, Yq

def encrypted(text, open_key):
    number = ord(text)
    c = (number ** 2) % open_key

    return c

def mod(k, b, m):
    i = 0
    a = 1
    v = []
    while k > 0:
        v.append(k % 2)
        k = (k - v[i]) // 2
        i += 1
    for j in range(i):
        if v[j] == 1:
            a = (a * b) % m
            b = (b * b) % m

```

```

        else:
            b = (b * b) % m

    return a

def decrypted(c, open_key, close_key):
    p = close_key[0]
    q = close_key[1]
    # алгоритм Евклида
    x, y = find_Yp_Yq(*close_key)
    while x * p + y * q != 1:
        x, y = find_Yp_Yq(*close_key)
    # китайская теорема об остатках
    r = mod((p+1)/4, c, p)
    s = mod((q+1)/4, c, q)
    r1 = (x*p*s + y*q*r) % open_key
    r2 = (open_key - r1)
    r3 = (x * p * s - y * q * r) % open_key
    r4 = (open_key - r3)
    # print(r1, r2, r3, r4)
    for item in (r1, r2, r3, r4):
        if item <= 1200:
            return chr(item)

with open("input.txt", "r", encoding='utf-8') as f:
    text = f.read()

open_key, close_key = generate_key(42)
while close_key[0] == close_key[1]:
    open_key, close_key = generate_key(42)

for item in text:
    encrypted_text = encrypted(item, open_key)
    with open("encrypted.txt", "a", encoding='utf-8') as f:
        f.write(str(encrypted_text))
    decrypted_text = decrypted(encrypted_text, open_key, close_key)
    with open("decrypted.txt", "a", encoding='utf-8') as f:
        f.write(decrypted_text)

```