Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы защиты информации»

|  |  |
| --- | --- |
|  |  |

**ОТЧЕТ**

к лабораторной работе № 3

на тему «Симметричная криптография. Криптосистема Рабина»

Выполнил             Е. А. Киселева

Проверил                           Е. А. Лещенко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc157722973)

[2 Краткие теоретические сведения 4](#_Toc157722974)

[3 Результаты выполнения лабораторной работы 5](#_Toc157722975)

[Выводы](#_Toc157722976) 7

[Список использованных источников 8](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 9](#_Toc157722978)

# **1 ПОСТАНОВКА ЗАДАЧИ**

Целью выполнения данной лабораторной работы является реализация программных средств шифрования и дешифрования текстовых файлов при помощи криптосистемы Рабина.

# **2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Криптосистема Рабина – это криптографический алгоритм, предложенный Майклом Рабином в 1979 году. Рабиновская криптосистема является асимметричной и основана на сложности задачи факторизации больших чисел, аналогично RSA, но с некоторыми ключевыми отличиями.

Принципы криптосистемы Рабина:

1 Основная математическая идея: криптосистема основана на сложности извлечения квадратного корня по модулю произведения двух простых чисел. Этот процесс сложен, если простые числа достаточно велики, что делает криптосистему безопасной.

2 Шифрование: чтобы зашифровать сообщение, получатель использует закрытый ключ, который состоит из 𝑝 и 𝑞, чтобы восстановить исходное сообщение из шифротекста. Для этого решается задача извлечения квадратного корня по модулю числа 𝑛.

Особенности криптосистемы Рабина:

1 Сложность: одним из недостатков системы является то, что на каждый шифротекст может приходиться несколько потенциальных расшифрованных сообщений (обычно 4 возможных значения). Чтобы устранить эту неопределенность, необходимо использовать дополнительные методы для отбора правильного сообщения.

2 Безопасность: если задача факторизации больших чисел остается сложной, криптосистема Рабина безопасна. На практике, если кто-то способен разложить 𝑛 на простые множители, он может взломать систему.

3 Скорость: Рабиновская криптосистема работает быстрее, чем RSA, потому что операция шифрования проще (квадратирование вместо возведения в степень).

Таким образом, криптосистема Рабина является интересной альтернативой RSA с высокой теоретической безопасностью, хотя и с некоторыми практическими трудностями, связанными с неоднозначностью расшифровки.

# **3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ             РАБОТЫ**

В ходе выполнения лабораторной было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Рабина.

Начальный текст находится в файле input.txt или in128.txt. После Запуска программа дает возможность выбрать, что пользователь хочет сделать: сгенерировать ключи, зашифровать файл, дешифровать файл или выйти. В зависимости от выбора программа запрашивает дополнительную необходимую информацию и выводит результат работы в файлы с теми названиями, которые ввел пользователь. Вывод консоли программы представлен на рисунке 3.1.

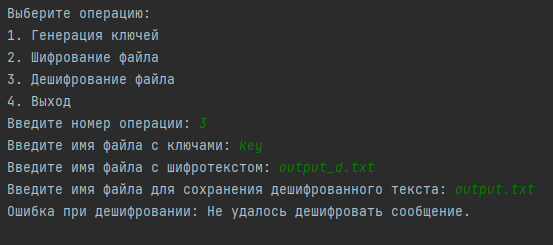


Рисунок 3.1 – Вывод консоли

Таким образом, в ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Рабина.

# **ВЫВОДЫ**

В ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Рабина.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Криптосистема Рабина [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/7880378/page:29/. – Дата доступа: 05.10.2024.

[2] Криптографическая стойкость криптосистемы Рабина [Электронный ресурс]. – Режим доступа: https://rep.bntu.by/bitstream/data/37387/1/Kriptograf  
icheskaya\_stojkost\_kriptosistemy\_Rabina.pdf. – Дата доступа: 06.10.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг 1 – Программный код файла main

import os

import sys

import json

from random import randrange

from math import isqrt, gcd

# Функция для проверки числа на простоту (тест Миллера-Рабина)

def is\_prime(n, k=40):

"""Проверка числа n на простоту с помощью теста Миллера-Рабина."""

if n <= 1:

return False

if n == 2 or n == 3:

return True

if n % 2 == 0:

return False

r, s = 0, n - 1

while s % 2 == 0:

r += 1

s //= 2

for \_ in range(k):

a = randrange(2, n - 1)

x = pow(a, s, n)

if x == 1 or x == n - 1:

continue

for \_ in range(r - 1):

x = pow(x, 2, n)

if x == n - 1:

break

else:

return False

return True

# Генерация случайного простого числа

def generate\_prime(bits):

"""Генерация случайного простого числа заданной битности."""

while True:

prime\_candidate = randrange(2\*\*(bits - 1), 2\*\*bits)

if is\_prime(prime\_candidate):

return prime\_candidate

# Расширенный алгоритм Евклида для вычисления обратного по модулю

def extended\_gcd(a, b):

if a == 0:

return b, 0, 1

gcd\_val, x1, y1 = extended\_gcd(b % a, a)

x = y1 - (b // a) \* x1

y = x1

return gcd\_val, x, y

def mod\_inverse(a, m):

"""Вычисление мультипликативной обратной по модулю."""

gcd\_val, x, \_ = extended\_gcd(a, m)

if gcd\_val != 1:

raise ValueError(f"Обратное значение по модулю не существует для {a} и {m}")

return x % m

# Генерация ключей Рабина

def generate\_keys(bits: int):

p = generate\_prime(bits // 2)

q = generate\_prime(bits // 2)

n = p \* q

return p, q, n

# Сохранение ключей в файл

def save\_keys(p: int, q: int, n: int, filename: str):

keys = {'p': p, 'q': q, 'n': n}

with open(filename, 'w') as f:

json.dump(keys, f)

print(f"Ключи сохранены в {filename}")

# Загрузка ключей из файла

def load\_keys(filename: str):

if not os.path.exists(filename):

raise FileNotFoundError(f"Файл с ключами {filename} не найден.")

with open(filename, 'r') as f:

keys = json.load(f)

try:

p = int(keys['p'])

q = int(keys['q'])

n = int(keys['n'])

return p, q, n

except (KeyError, ValueError):

raise ValueError("Файл с ключами имеет неверный формат.")

# Преобразование текста в число и обратно

def text\_to\_int(text: str) -> int:

return int.from\_bytes(text.encode('utf-8'), byteorder='big')

def int\_to\_text(number: int) -> str:

try:

byte\_length = (number.bit\_length() + 7) // 8

return number.to\_bytes(byte\_length, byteorder='big').decode('utf-8')

except (OverflowError, UnicodeDecodeError):

raise ValueError("Дешифрованное число не может быть преобразовано в текст.")

# Шифрование

def encrypt(message: str, n: int) -> int:

m = text\_to\_int(message)

if m >= n:

raise ValueError("Сообщение слишком длинное для данного ключа.")

return pow(m, 2, n)

# Дешифрование

def decrypt(cipher: int, p: int, q: int) -> str:

n = p \* q

mp = pow(cipher, (p + 1) // 4, p)

mq = pow(cipher, (q + 1) // 4, q)

inv\_q\_mod\_p = mod\_inverse(q, p)

inv\_p\_mod\_q = mod\_inverse(p, q)

# Китайская теорема об остатках для нахождения четырёх возможных корней

r1 = (mp \* q \* inv\_q\_mod\_p + mq \* p \* inv\_p\_mod\_q) % n

r2 = (mp \* q \* inv\_q\_mod\_p - mq \* p \* inv\_p\_mod\_q) % n

r3 = (-mp \* q \* inv\_q\_mod\_p + mq \* p \* inv\_p\_mod\_q) % n

r4 = (-mp \* q \* inv\_q\_mod\_p - mq \* p \* inv\_p\_mod\_q) % n

for root in [r1, r2, r3, r4]:

try:

decrypted\_message = int\_to\_text(root)

return decrypted\_message

except ValueError:

continue

raise ValueError("Не удалось дешифровать сообщение.")

# Работа с файлами

def encrypt\_file(input\_file: str, output\_file: str, n: int):

try:

with open(input\_file, 'r', encoding='utf-8') as f:

message = f.read()

except Exception as e:

raise IOError(f"Не удалось прочитать файл {input\_file}: {e}")

cipher = encrypt(message, n)

with open(output\_file, 'w') as f:

f.write(str(cipher))

print(f"Файл зашифрован и сохранен как {output\_file}")

def decrypt\_file(input\_file: str, output\_file: str, p: int, q: int):

try:

with open(input\_file, 'r') as f:

cipher = int(f.read())

except Exception as e:

raise IOError(f"Не удалось прочитать файл {input\_file}: {e}")

decrypted\_message = decrypt(cipher, p, q)

with open(output\_file, 'w', encoding='utf-8') as f:

f.write(decrypted\_message)

print(f"Файл дешифрован и сохранен как {output\_file}")

def main():

while True:

print("\nВыберите операцию:")

print("1. Генерация ключей")

print("2. Шифрование файла")

print("3. Дешифрование файла")

print("4. Выход")

choice = input("Введите номер операции: ")

if choice == '1':

bits = int(input("Введите битность ключа (например, 512): "))

output\_file = input("Введите имя файла для сохранения ключей: ")

try:

p, q, n = generate\_keys(bits)

save\_keys(p, q, n, output\_file)

except Exception as e:

print(f"Ошибка при генерации ключей: {e}")

elif choice == '2':

keys\_file = input("Введите имя файла с ключами: ")

input\_file = input("Введите имя файла для шифрования: ")

output\_file = input("Введите имя файла для сохранения шифротекста: ")

try:

\_, \_, n = load\_keys(keys\_file)

encrypt\_file(input\_file, output\_file, n)

except Exception as e:

print(f"Ошибка при шифровании: {e}")

elif choice == '3':

keys\_file = input("Введите имя файла с ключами: ")

input\_file = input("Введите имя файла с шифротекстом: ")

output\_file = input("Введите имя файла для сохранения дешифрованного текста: ")

try:

p, q, \_ = load\_keys(keys\_file)

decrypt\_file(input\_file, output\_file, p, q)

except Exception as e:

print(f"Ошибка при дешифровании: {e}")

elif choice == '4':

print("Выход из программы.")

break

else:

print("Неверный ввод. Пожалуйста, выберите правильный вариант.")

if \_\_name\_\_ == '\_\_main\_\_':

main()