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 № 4

на тему «Асимметричная криптография. Алгоритм Мак-Элиса»

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, однако пользователь может создать и выбрать другой. 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. Полная работа программы представлен на рисунке 3.1.

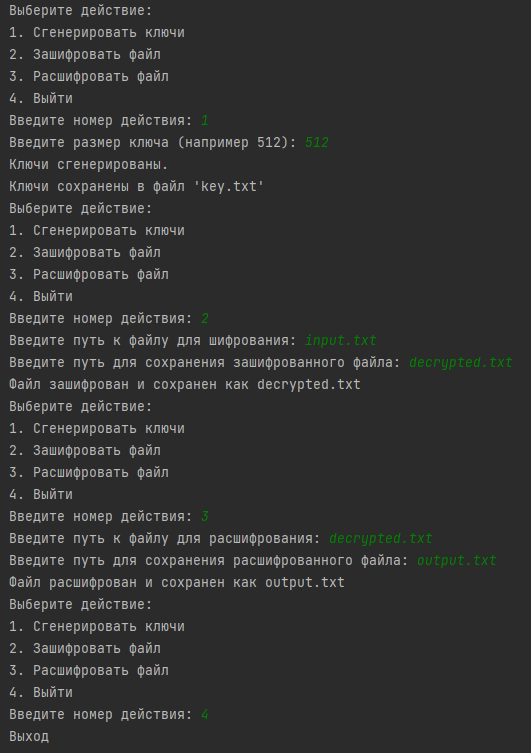


Рисунок 3.1 – Вывод консоли

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 random

from sympy import isprime

# Генерация ключей для системы Рабина

def generate\_keys(bits=512):

while True:

p = random.getrandbits(bits)

if isprime(p) and p % 4 == 3:

break

while True:

q = random.getrandbits(bits)

if isprime(q) and q % 4 == 3:

break

N = p \* q

return (N, p, q)

def encrypt(message, N):

m = int.from\_bytes(message.encode('utf-8'), 'big')

if m >= N:

raise ValueError("The message is too large for the key size")

c = pow(m, 2, N)

return c

def decrypt(ciphertext, p, q):

N = p \* q

m\_p = pow(ciphertext, (p + 1) // 4, p)

m\_q = pow(ciphertext, (q + 1) // 4, q)

\_, yp, yq = extended\_gcd(p, q)

r1 = (yp \* p \* m\_q + yq \* q \* m\_p) % N

r2 = N - r1

r3 = (yp \* p \* m\_q - yq \* q \* m\_p) % N

r4 = N - r3

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

try:

decrypted\_message = r.to\_bytes((r.bit\_length() + 7) // 8, 'big').decode('utf-8')

return decrypted\_message

except UnicodeDecodeError:

continue

raise ValueError("Decryption failed, none of the roots produced a valid message")

# Алгоритм Евклида для нахождения gcd и коэффициентов

def extended\_gcd(a, b):

if a == 0:

return b, 0, 1

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

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

y = x1

return gcd, x, y

def read\_file(file\_path):

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

return f.read()

def write\_file(file\_path, data):

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

f.write(data)

def menu():

print("Выберите действие:")

print("1. Сгенерировать ключи")

print("2. Зашифровать файл")

print("3. Расшифровать файл")

print("4. Выйти")

choice = input("Введите номер действия: ")

return choice

# m\_p = c^(p+1)/4 mod p

def main():

N, p, q = None, None, None

while True:

choice = menu()

if choice == "1":

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

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

print(f"Ключи сгенерированы.")

with open('key.txt', 'w') as f:

f.write(f"{N}\n{p}\n{q}")

print("Ключи сохранены в файл 'key.txt'")

elif choice == "2":

if N is None:

print("Сначала необходимо сгенерировать ключи!")

continue

file\_path = input("Введите путь к файлу для шифрования: ")

try:

message = read\_file(file\_path)

ciphertext = encrypt(message, N)

enc\_file\_path = input("Введите путь для сохранения зашифрованного файла: ")

write\_file(enc\_file\_path, str(ciphertext))

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

except Exception as e:

print(f"Ошибка при шифровании: {e}")

elif choice == "3":

if p is None or q is None:

print("Сначала необходимо сгенерировать ключи!")

continue

file\_path = input("Введите путь к файлу для расшифрования: ")

try:

ciphertext = int(read\_file(file\_path))

message = decrypt(ciphertext, p, q)

dec\_file\_path = input("Введите путь для сохранения расшифрованного файла: ")

write\_file(dec\_file\_path, message)

print(f"Файл расшифрован и сохранен как {dec\_file\_path}")

except Exception as e:

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

elif choice == "4":

print("Выход")

break

else:

print("Заново")

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

main()