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 «Асимметричная криптография. Криптосистема Рабина»

Выполнил             Д. С. Кончик

Проверил                          А. В. Герчик

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель лабораторной работы 3](#_Toc157722973)

[2 Теоретические сведения 4](#_Toc157722974)

[3 Результат выполнения 5](#_Toc157722975)

[Вывод](#_Toc157722976) 6

[Список использованных источников 7](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 8](#_Toc157722978)

# **1 ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ**

Целью выполнения данной лабораторной работы является реализация программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Рабина. Также необходимо добавить интерфейс командной строки, который позволит выбирать файл для шифрования/дешифрования, задавать ключи и режимы операций и предусмотреть обработку возможных ошибок.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Криптосистема Рабина – криптографическая система с открытым ключом, безопасность которой обеспечивается сложностью поиска квадратных корней в кольце остатков по модулю составного числа. Безопасность системы, как и безопасность метода RSA, обусловлена сложностью разложения на множители больших чисел. Зашифрованное сообщение можно расшифровать 4 способами. Недостатком системы является необходимость выбора истинного сообщения из 4-х возможных.

В январе 1979 года Майкл О. Рабин опубликовал описание своей системы. Было доказано, что восстановление исходного текста из зашифрованного столь же трудно, как факторизация больших чисел. Система Рабина стала первой асимметричной криптосистемой, для которой было выполнено такое доказательство. Сложность восстановления связана с трудностью извлечения квадратного корня по модулю составного числа   
N = р \* q. Задача факторизации и задача по извлечению квадратного корня эквивалентны, то есть: зная простые делители числа N можно извлекать квадратные корни по модулю N; умея извлекать квадратные корни по модулю N, можно разложить N на простые множители.

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:

1. Выбираются два случайных числа p и q с учётом следующих требований: числа должны быть большими, числа должны быть простыми, должно выполняться условие: p ≡ q ≡ 3 mod 4. Выполнение этих требований сильно ускоряет процедуру извлечения корней по модулю р и q;
2. Вычисляется число n = p · q;
3. Число n – открытый ключ; числа p и q – закрытый.

Исходное сообщение m (текст) шифруется с помощью открытого   
ключа – числа n по следующей формуле: c = m² mod n. Благодаря использованию умножения по модулю скорость шифрования системы Рабина больше, чем скорость шифрования по методу RSA, даже если в последнем случае выбрать небольшое значение экспоненты.

# **3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

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

После запуска программа предлагает выбрать доступные операции: генерация ключей, шифрация / дешифрация файла и выход. При выборе операций с файлами предложен ввод путей к файлам, если вывод пустой, то программа использует пути по умолчанию. Работа программы представлена на рисунке 1.

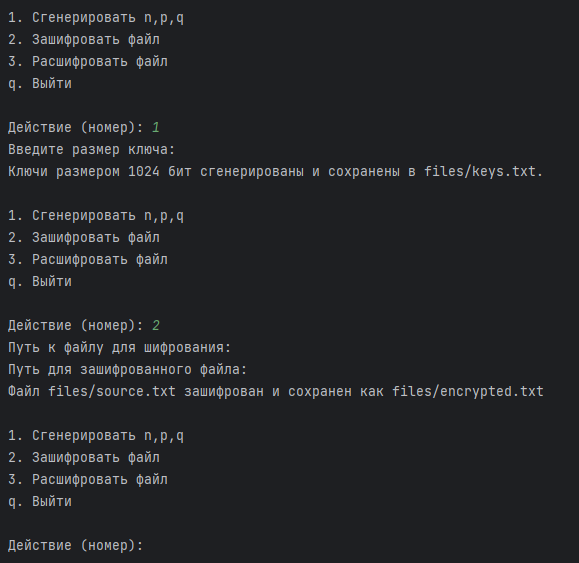


Рисунок 1 – Работа программы

# **ВЫВОД**

В ходе данной лабораторной работы было реализовано программное средство шифрования и дешифрования текстовых файлов при помощи криптосистемы Рабина. Программа реализована на языке программирования Python, предоставляет удобный интерфейс консоли и возможность выбора производимой операции с настройкой ключей и указанием путей к файлам, хранящим исходный, зашифрованный и расшифрованный текст.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Криптосистема Рабина [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Криптосистема\_Рабина. – Дата доступа: 03.11.2024.

[2] Rabin cryptosystem [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Rabin\_cryptosystem. – Дата доступа: 03.11.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

import random

from sympy import isprime

from typing import Tuple, Optional

import os

from constants import SOURCE\_PATH, ENCRYPTED\_PATH, DECRYPTED\_PATH, KEYS\_PATH, OPEN\_KEY\_LENGTH

def generate\_rabin\_keys(bit\_size: int = 256) -> Tuple[int, int, int]:

def find\_prime() -> int:

while True:

candidate: int = random.getrandbits(bit\_size)

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

return candidate

p: int = find\_prime()

q: int = find\_prime()

return p \* q, p, q

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

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

if not (m < n):

raise ValueError("Должно выполняться условие m < n при шифровании")

c: int = pow(m, 2, n)

return c

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

n: int = p \* q

r: int = pow(c, (p + 1) // 4, p)

s: int = pow(c, (q + 1) // 4, q)

# px + qy = 1

\_, x, y = extended\_euclid(p, q)

m1: int = (x \* p \* s + y \* q \* r) % n

m2: int = n - m1

m3: int = (x \* p \* s - y \* q \* r) % n

m4: int = n - m3

i = 0

for mi in [m1, m2, m3, m4]:

i += 1

try:

return mi.to\_bytes((mi.bit\_length() + 7) // 8, 'big').decode('utf-8')

except UnicodeDecodeError:

continue

raise ValueError("Не смогли преобразовать c в m")

# nod(a,b) = ax + by

def extended\_euclid(a: int, b: int) -> Tuple[int, int, int]:

if a == 0:

return b, 0, 1

nod, prev\_x, prev\_y = extended\_euclid(b % a, a)

current\_x = prev\_y - (b // a) \* prev\_x

current\_y = prev\_x

return nod, current\_x, current\_y

def read\_file(file\_path: str) -> str:

if not os.path.exists(file\_path):

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

pass

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

return f.read()

def write\_file(file\_path: str, data: str) -> None:

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

f.write(data)

def display\_menu() -> str:

print()

print("1. Сгенерировать n,p,q")

print("2. Зашифровать файл")

print("3. Расшифровать файл")

print("q. Выйти")

print()

user\_choice: str = input("Действие (номер): ")

return user\_choice

def input\_with\_default(prompt: str, default\_value: str) -> str:

return input(prompt) or default\_value

n: Optional[int] = None

p: Optional[int] = None

q: Optional[int] = None

def main() -> None:

actions: dict = {

"1": lambda: generate\_keys(),

"2": lambda: encrypt\_file(),

"3": lambda: decrypt\_file(),

"q": lambda: exit\_program(),

}

while True:

selected\_action: str = display\_menu()

action = actions.get(selected\_action)

if action:

action()

else:

print("Неверный выбор. Пожалуйста, попробуйте снова.")

def generate\_keys() -> None:

bits: int = int(input\_with\_default("Введите размер ключа: ", OPEN\_KEY\_LENGTH))

global n, p, q

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

write\_file(KEYS\_PATH, f"n: {n}\np: {p}\nq: {q}")

print(f"Ключи размером {bits} бит сгенерированы и сохранены в {KEYS\_PATH}.")

def encrypt\_file() -> None:

global n

if n is None:

print("N is None")

return

file\_path: str = input\_with\_default("Путь к файлу для шифрования: ", SOURCE\_PATH)

try:

message: str = read\_file(file\_path)

ciphertext: int = encrypt(message, n)

enc\_file\_path: str = input\_with\_default("Путь для зашифрованного файла: ", ENCRYPTED\_PATH)

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

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

except Exception as e:

print(f"Ошибка при шифровании: {e}")

def decrypt\_file() -> None:

global p, q

if p is None or q is None:

print("p or q is None")

return

file\_path: str = input\_with\_default("Путь к файлу для расшифрования: ", ENCRYPTED\_PATH)

try:

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

message: str = decrypt(ciphertext, p, q)

dec\_file\_path: str = input\_with\_default("Путь для расшифрованного файла: ", DECRYPTED\_PATH)

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

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

except Exception as e:

print(f"Ошибка: {e}")

def exit\_program() -> None:

print("exit")

exit()

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

main()