

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы защиты информации

ОТЧЁТ
к лабораторной работе №3
на тему

АСИММЕТРИЧНАЯ КРИПТОГРАФИЯ. КРИПТОСИСТЕМА РАБИНА

Выполнил: студент гр.253504
Фроленко К.Ю.
Проверил: ассистент кафедры информатики
Герчик А.В.

Минск 2025

СОДЕРЖАНИЕ

1 Цель работы	3
2 Этапы выполнения работы.....	4
2.1 Краткие теоретические сведения	4
2.2 Пример работы программы.....	4
Заключение	6

1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является практическое изучение и реализация асимметричной криптографической системы Рабина, которая представляет собой одну из фундаментальных криптосистем с открытым ключом. Работа направлена на глубокое понимание математических основ современной криптографии, в частности задач, связанных с теорией чисел и сложностью факторизации больших целых чисел.

В рамках лабораторной работы предстоит разработать программное обеспечение на языке Python, реализующее полный цикл работы криптосистемы Рабина: от генерации ключевой пары до шифрования и дешифрования текстовых данных. Особое внимание уделяется изучению алгоритма генерации простых чисел специального вида $(4k + 3)$, механизма возведения в квадрат по модулю и процедуры извлечения квадратных корней с использованием китайской теоремы об остатках.

Важной составляющей работы является исследование особенности криптосистемы Рабина, связанной с неоднозначностью дешифрования, когда каждое зашифрованное сообщение может быть корректно расшифровано четырьмя различными способами. Практическая задача включает разработку механизма выбора правильного варианта среди нескольких возможных решений на основе анализа целостности и семантической корректности данных.

Результатом работы должно стать законченное программное решение, сопровождаемое подробной документацией и тестовыми примерами, демонстрирующими корректность реализации всех компонентов системы. Полученные знания и навыки имеют важное значение для понимания принципов построения современных криптографических систем и могут быть применены в дальнейшем при изучении более сложных криптографических протоколов и алгоритмов.

2 ЭТАПЫ ВЫПОЛНЕНИЯ РАБОТЫ

2.1 Краткие теоретические сведения

Криптосистема Рабина, впервые описанная Майклом Рабином в 1979 году, представляет собой асимметричную криптографическую схему, безопасность которой напрямую связана с вычислительной сложностью разложения больших составных чисел на простые множители. В отличие от многих других асимметричных алгоритмов, для криптосистемы Рабина существует строгое математическое обоснование её стойкости: успешный криптоанализ эквивалентен решению задачи факторизации целого числа, что до сих пор остаётся одной из центральных нерешённых проблем теории чисел.

Основу алгоритма составляет трудность вычисления квадратных корней по модулю составного числа $n = p \cdot q$, где p и q — большие простые числа, удовлетворяющие специальному условию: $p \equiv 3 \pmod{4}$ и $q \equiv 3 \pmod{4}$. Это требование обеспечивает, что любой квадратичный вычет по модулю n имеет ровно четыре квадратных корня, что является фундаментальной характеристикой данной системы и напрямую влияет на процесс дешифрования.

Шифрование в схеме Рабина отличается исключительной простотой и эффективностью: для открытого текста m шифротекст вычисляется как $c = m^2 \bmod n$. Возведение в квадрат по модулю — чрезвычайно быстрая операция даже при работе с очень большими числами, что делает этап шифрования высокопроизводительным. Однако именно эта простота создаёт асимметрию в сложности: обратная задача — извлечение квадратного корня по модулю n без знания разложения n на простые множители — является вычислительно трудной и эквивалентна факторизации.

Одной из ключевых особенностей криптосистемы Рабина является неоднозначность расшифровки: каждое зашифрованное значение соответствует четырём различным кандидатам на исходное сообщение. Чтобы однозначно определить правильный вариант, необходимо внедрять дополнительные механизмы — например, добавлять избыточность в сообщение (такую как фиксированный заголовок или контрольную сумму) или использовать известные структурные свойства данных. По этой причине схема Рабина редко применяется напрямую для шифрования реальных сообщений, однако она играет важную роль в теоретической криптографии и служит основой для построения более сложных и практических криптографических протоколов.

2.2 Пример работы программы

На вход программы подается текстовый файл с исходным текстом для шифрования. На рисунке 2.2.1 представлено содержимое исходного файла перед шифрованием. Программа выполняет генерацию ключевой пары,

включающей два больших простых числа p и q вида $4k+3$ и их произведение n , которое используется в качестве открытого ключа.

На рисунке 2.2.2 демонстрируется процесс генерации ключей и основные параметры криптосистемы. Особенностью криптосистемы Рабина является неоднозначность дешифрования, что показано на рисунке 2.2.3, где представлены четыре возможных варианта восстановления исходного сообщения, три из которых содержат случайные данные и только один является корректным.

Программа автоматически определяет корректный вариант дешифрования на основе анализа целостности данных и проверки семантической содержательности восстановленного текста. На рисунке 2.2.4 показан окончательный результат работы программы — успешно расшифрованный текст, полностью соответствующий исходному сообщению.

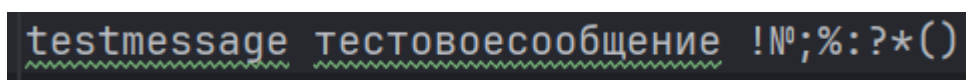


Рисунок 2.2.1 – Исходный текст

```
p = 11997450121741541311068561764398512882136533076131095119117203427799973962057320710211085523849971
q = 11857973658195771311130371789622790843417022459001662284726485642517807451676874279161490874649502
n = 14226544750912884649152436942309373373476392516123428758201026076712805302044152503954798925134306
```

Рисунок 2.2.2 – Демонстрация генерации ключей

```
Найдено 4 возможных решений
g00ph0000004q0R0-0000"0q00
0
f00=000%0000±00X00000KR... (128 байт)
$0V0
n0... (128 байт)
00>0C?Q00#3000Г00000LCE0'A0w^00000Z0020000000[0,00000
00 .00000.00... (128 байт)
Вариант 4: testmessage тестовое сообщение !№;%:?*()... (58 байт)
```

Рисунок 2.2.3 – Четыре возможных варианта восстановления исходного сообщения

```
Исходный текст: 'testmessage тестовое сообщение !№;%:?*()'
Расшифрованный текст: 'testmessage тестовое сообщение !№;%:?*()'
```

Рисунок 2.2.4 – Расшифрованный текст

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была успешно реализована и протестирована криптосистема Рабина — одна из классических асимметричных криптографических схем. Практическая реализация позволила детально изучить её математическую основу, связанную с вычислительной сложностью факторизации больших целых чисел и задачей извлечения квадратных корней по составному модулю.

В процессе работы были отработаны ключевые компоненты алгоритма: генерация простых чисел вида $p \equiv 3 \pmod{4}$ и $q \equiv 3 \pmod{4}$, шифрование путём возведения сообщения в квадрат по модулю $n = p \cdot q$, а также процедура дешифрования, приводящая к четырём возможным кандидатам на исходное сообщение. Для решения проблемы неоднозначности расшифровки была разработана и внедрена стратегия автоматического выбора корректного варианта на основе анализа семантической и структурной целостности данных.

Практическая ценность выполненной работы заключается в создании полнофункционального программного инструмента, способного корректно выполнять шифрование и дешифрование текстовой информации в соответствии с оригинальной спецификацией криптосистемы Рабина. При этом были приобретены важные навыки работы с большими целыми числами, модульной арифметикой и реализацией криптографических примитивов — компетенции, имеющие значительную актуальность в сфере информационной безопасности.

Экспериментальные результаты полностью подтвердили теоретические свойства схемы: преобразования оказались обратимыми при условии знания секретного ключа, а криптостойкость — обусловленной трудностью факторизации модуля n . Разработанное решение может эффективно использоваться в учебных целях для демонстрации принципов асимметричной криптографии и применения теоретико-числовых методов в задачах защиты информации.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код программы

```
import random
from typing import Tuple, List

class RabinCryptosystem:
    def __init__(self, bit_length: int = 512):
        self.bit_length = bit_length
        self.p, self.q, self.n = self._generate_keys()

    def _is_prime(self, n: int, k: int = 20) -> bool:
        if n <= 1:
            return False
        if n <= 3:
            return True
        if n % 2 == 0:
            return False
        d = n - 1
        r = 0
        while d % 2 == 0:
            d //= 2
            r += 1
        for _ in range(k):
            a = random.randint(2, n - 2)
            x = pow(a, d, 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

    def _generate_prime(self) -> int:
        while True:
            p = random.getrandbits(self.bit_length)
            p |= (1 << (self.bit_length - 1)) | 1
            p = (p // 4) * 4 + 3
            if self._is_prime(p):
                return p

    def _generate_keys(self) -> Tuple[int, int, int]:
        p = self._generate_prime()
        q = self._generate_prime()
        while p == q:
            q = self._generate_prime()
        n = p * q
        print(f"    p = {p}")
        print(f"    q = {q}")
        print(f"    n = {n}")
        print(f"    Размер ключа: {n.bit_length()} бит")
        return p, q, n

    def _extended_gcd(self, a: int, b: int) -> Tuple[int, int, int]:
        if a == 0:
            return b, 0, 1
        gcd, x1, y1 = self._extended_gcd(b % a, a)
        x = y1 - (b // a) * x1
```

```

        y = x1
        return gcd, x, y
def _crt(self, a1: int, a2: int) -> int:
    _, x, y = self._extended_gcd(self.p, self.q)
    return (a1 * self.q * y + a2 * self.p * x) % self.n

def encrypt(self, message: int) -> int:
    if message >= self.n:
        raise ValueError("Сообщение должно быть меньше n")
    return pow(message, 2, self.n)

def decrypt(self, ciphertext: int) -> List[int]:
    mp = pow(ciphertext, (self.p + 1) // 4, self.p)
    mq = pow(ciphertext, (self.q + 1) // 4, self.q)
    solutions = [
        self._crt(mp, mq),
        self._crt(mp, self.q - mq),
        self._crt(self.p - mp, mq),
        self._crt(self.p - mp, self.q - mq)
    ]
    return solutions

def encrypt_bytes(self, data: bytes) -> bytes:
    message_int = int.from_bytes(data, 'big')
    if message_int >= self.n:
        raise ValueError("Данные слишком велики для шифрования")
    cipher_int = self.encrypt(message_int)
    cipher_bytes = cipher_int.to_bytes((cipher_int.bit_length() + 7) // 8,
'big')
    return cipher_bytes

def decrypt_bytes(self, cipher_data: bytes) -> List[bytes]:
    cipher_int = int.from_bytes(cipher_data, 'big')
    solutions = self.decrypt(cipher_int)
    result = []
    for solution in solutions:
        try:
            byte_length = (solution.bit_length() + 7) // 8
            if byte_length > 0:
                decrypted_bytes = solution.to_bytes(byte_length, 'big')
                result.append(decrypted_bytes)
        except:
            continue
    return result

def rabin_encrypt_file(input_file: str, output_file: str, rabin:
RabinCryptosystem):
    with open(input_file, 'rb') as f:
        plaintext = f.read()
    print(f"    Размер исходных данных: {len(plaintext)} байт")
    print(f"    Максимальный размер блока: {(rabin.n.bit_length() - 1) // 8}
байт")
    max_block_size = (rabin.n.bit_length() - 1) // 8
    if len(plaintext) > max_block_size:
        raise ValueError(f"Файл слишком большой. Максимум: {max_block_size}
байт")
    ciphertext = rabin.encrypt_bytes(plaintext)
    with open(output_file, 'wb') as f:
        f.write(ciphertext)
    print(f"Файл зашифрован в '{output_file}'")
    print(f"Размер зашифрованного файла: {len(ciphertext)} байт")
def rabin_decrypt_file(input_file: str, output_file: str, rabin:
RabinCryptosystem):
    with open(input_file, 'rb') as f:

```



```

        ciphertext = f.read()
    print(f"Размер зашифрованных данных: {len(ciphertext)} байт")
    possible_results = rabin.decrypt_bytes(ciphertext)
    print(f"Найдено {len(possible_results)} возможных решений")
    for i, data in enumerate(possible_results):
        try:
            decoded = data.decode('utf-8', errors='replace')
            print(f"    Вариант {i+1}: {decoded[:100]}... ({len(data)} байт)")
# Исправлено: decoded вместо decrypted
        except:
            print(f"    Вариант {i+1}: [бинарные данные] ({len(data)} байт)")
    correct_data = None
    for i, data in enumerate(possible_results):
        try:
            decoded = data.decode('utf-8')
            if decoded.isprintable() or len(decoded) > 0:
                correct_data = data
                print(f"    Выбран вариант {i+1} как корректный")
                break
        except:
            continue
    if correct_data is None and possible_results:
        correct_data = possible_results[0]
        print(f"    Используем первый вариант")
    if correct_data is None:
        raise ValueError("Не удалось дешифровать данные")
    with open(output_file, 'wb') as f:
        f.write(correct_data)
    print(f"Файл расшифрован в '{output_file}'")

def save_keys(rabin: RabinCryptosystem, filename: str):
    with open(filename, 'w') as f:
        f.write(f"p={rabin.p}\n")
        f.write(f"q={rabin.q}\n")
        f.write(f"n={rabin.n}\n")
    print(f"Ключи занесены в '{filename}'")

def load_keys(filename: str) -> Tuple[int, int, int]:
    with open(filename, 'r') as f:
        lines = f.readlines()
    p = int(lines[0].split('=')[1])
    q = int(lines[1].split('=')[1])
    n = int(lines[2].split('=')[1])
    print(f"Ключи загружены из '{filename}'")
    return p, q, n

if __name__ == "__main__":
    rabin = RabinCryptosystem(bit_length=512)
    save_keys(rabin, "rabin_keys.txt")
    test_text = "testmessage тестовое сообщение !№;%:~*()"
    with open('test_rabin.txt', 'w', encoding='utf-8') as f:
        f.write(test_text)
    print(f"\nСоздан тестовый файл: {len(test_text)} СИМВОЛОВ")
    print(f"Исходный текст: {test_text}")
    try:
        rabin_encrypt_file('test_rabin.txt', 'encrypted_rabin.bin', rabin)
    except ValueError as e:
        print(f"Ошибка: {e}")
        max_size = (rabin.n.bit_length() - 1) // 8
        test_text = test_text[:max_size]
        with open('test_rabin_small.txt', 'w', encoding='utf-8') as f:
            f.write(test_text)
        rabin_encrypt_file('test_rabin_small.txt', 'encrypted_rabin.bin',
rabin)
        rabin_decrypt_file('encrypted_rabin.bin', 'decrypted_rabin.txt', rabin)

```

```
with open('decrypted_rabin.txt', 'r', encoding='utf-8') as f:
    decrypted_text = f.read()
print(f"    Исходный текст: '{test_text}'")
print(f"    Расшифрованный текст: '{decrypted_text}'")
```