



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.
Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа №2 по курсу "Защита информации"

Тема Алгоритм шифрования RSA

Студент Нисуев Н. Ф.

Группа ИУ7-72Б

Преподаватель Руденкова Ю.С.

Москва — 2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Аналитическая часть	4
1.1 Алгоритм RSA	4
2 Конструкторская часть	5
2.1 Схема алгоритма	5
3 Технологическая часть	6
3.1 Средства реализации	6
3.2 Реализация алгоритма	6
3.3 Пример работы программы	10
ЗАКЛЮЧЕНИЕ	12

ВВЕДЕНИЕ

Шифрование с открытым ключом, в отличие от симметричного шифрования, использует два различных ключа — открытый и закрытый, что решает проблему безопасной передачи ключа. Алгоритм RSA — один из первых и наиболее широко применяемых алгоритмов асимметричного шифрования.

Цель лабораторной работы: проектирование и разработка программной реализации алгоритма шифрования RSA. Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ работы алгоритма RSA;
- описать алгоритм шифрования с открытым ключом;
- реализовать и протестировать реализацию алгоритма шифрования.

1 Аналитическая часть

В этом разделе будет рассмотрен криптографический алгоритм RSA.

1.1 Алгоритм RSA

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — ассиметричный алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших полупростых чисел. В алгоритме RSA используется 2 ключа — открытый (публичный) и закрытый (приватный).

В ассиметричной криптографии и алгоритме RSA, в частности, открытый и закрытый ключи являются двумя частями одного целого и неразрывны друг с другом. Для шифрования информации используется открытый ключ, а для её расшифровки закрытый.

RSA ключи генерируются следующим образом:

- 1) выбираются два отличающихся друг от друга случайных простых числа p и q , лежащие в установленном диапазоне;
- 2) вычисляется их произведение $n = p \cdot q$, называемое модулем;
- 3) вычисляется значение функции Эйлера от числа n : $\phi(n) = (p - 1) \cdot (q - 1)$;
- 4) выбирается целое число e ($1 < e < \phi(n)$), взаимно простое со значением $\phi(n)$, оно называется открытой экспонентой;
- 5) вычисляется число $d \cdot e \equiv 1 \pmod{\phi(n)}$, оно называется закрытой экспонентой.

Пара (e, n) публикуются в качестве открытого ключа RSA, а пара (d, n) — в виде закрытого ключа.

Шифрование сообщения m ($0 < m < n - 1$) в зашифрованное сообщение c производится по формуле $c = E(m, k_1) = E(m, n, e) = m^e \bmod(n)$.

Расшифровка: $m = D(c, k_2) = D(c, n, d) = c^d \bmod(n)$

2 Конструкторская часть

В этом разделе будут представлена схема алгоритма шифрования RSA.

2.1 Схема алгоритма

Схема алгоритма RSA представлена на рисунке 2.1.

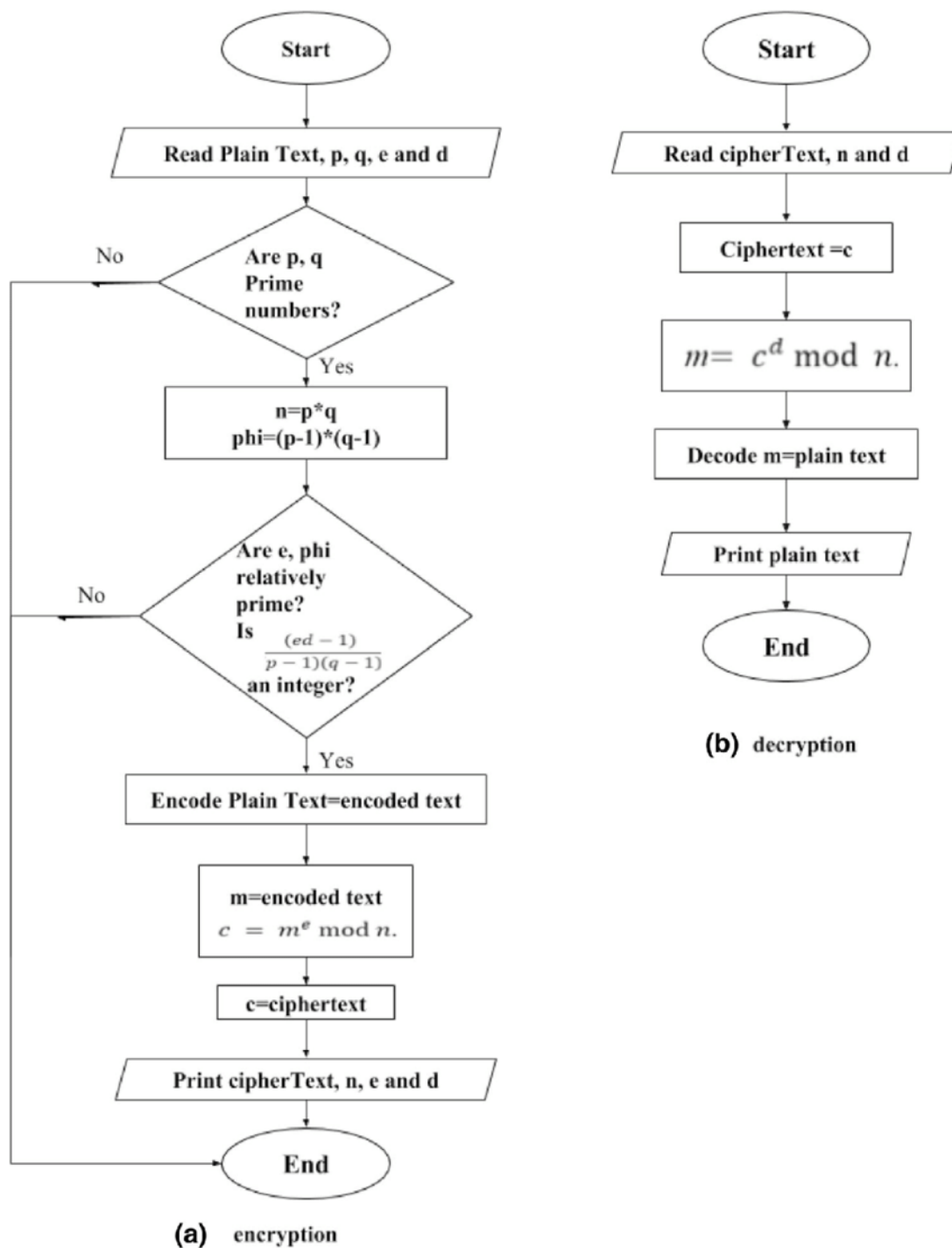


Рисунок 2.1 – Схема алгоритма шифрования RSA

3 Технологическая часть

3.1 Средства реализации

Для программной реализации шифровальной машины был выбран язык Python. В данном языке есть все требующиеся инструменты для данной лабораторной работы.

3.2 Реализация алгоритма

В листингах 3.1 — 3.2 представлена реализация алгоритма RSA.

Листинг 3.1 – Генерация ключей

```
1 def is_prime(n: int, k: int = 5) -> bool:
2     if n <= 1:
3         return False
4     if n <= 3:
5         return True
6     if n % 2 == 0:
7         return False
8
9     d = n - 1
10    r = 0
11    while d % 2 == 0:
12        d //= 2
13        r += 1
14
15    for _ in range(k):
16        a = random.randint(2, n - 2)
17        x = pow(a, d, n)
18        if x == 1 or x == n - 1:
19            continue
20        for _ in range(r - 1):
21            x = pow(x, 2, n)
22            if x == n - 1:
23                break
24    else:
25        return False
```

```

26     return True
27
28 def generate_prime(bits: int) -> int:
29     while True:
30         num = random.getrandbits(bits)
31         num |= (1 << bits - 1) | 1
32         if is_prime(num):
33             return num
34
35
36 def gcd(a: int, b: int):
37     while b:
38         a, b = b, a % b
39     return a
40
41 def mod_inverse(a: int, m: int):
42     def extended_gcd(a, b):
43         if a == 0:
44             return b, 0, 1
45         gcd, x1, y1 = extended_gcd(b % a, a)
46         x = y1 - (b // a) * x1
47         y = x1
48         return gcd, x, y
49
50     gcd, x, _ = extended_gcd(a, m)
51     if gcd != 1:
52         raise ValueError("Обратный элемент не существует")
53     return x % m
54
55 KEY_SIZE = 1024
56 def generate_keys() -> tuple[tuple[int, int], tuple[int, int]] | None:
57     try:
58         bits = KEY_SIZE
59         p = generate_prime(bits // 2)
60         q = generate_prime(bits // 2)
61
62         n = p * q
63         phi = (p - 1) * (q - 1)
64
65         e = 65537
66         while gcd(e, phi) != 1:

```

```

67         e = random.randint(2, phi - 1)
68
69         d = mod_inverse(e, phi)
70
71         public_key = (e, n)
72         private_key = (d, n)
73
74         print("Ключи успешно сгенерированы")
75         return (public_key, private_key)
76     except Exception as ex:
77         print(f"Ошибка при генерации ключей: {str(ex)}")
78         return None

```

Листинг 3.2 – Шифрование/расшифровка алгоритмом RSA

```

1 STD_PUB_KEY_FILE = ".rsa.pub"
2 STD_PRIV_KEY_FILE = ".rsa.priv"
3
4 def encrypt_file(fininput: str, foutput: str, fpub_key: str =
   STD_PUB_KEY_FILE):
5     try:
6         e, n = load_key(fpub_key)
7         block_size = n.bit_length() // 8 - 1
8
9         with open(fininput, 'rb') as f:
10             data = f.read()
11
12             encrypted_blocks = []
13             for i in range(0, len(data), block_size):
14                 block = data[i:i+block_size]
15                 num = int.from_bytes(block, 'big')
16                 encrypted_num = pow(num, e, n)
17                 encrypted_block = encrypted_num.to_bytes((n.bit_length() +
18                     7) // 8, 'big')
19                 encrypted_blocks.append(encrypted_block)
20
21             with open(foutput, 'wb') as f:
22                 for block in encrypted_blocks:
23                     f.write(block)
24
25             print("Файл успешно зашифрован!")
26     except Exception as ex:

```



```

26         print(f"Ошибка при шифровании: {str(ex)}")
27
28 def decrypt_file(fininput: str, foutput: str, fpriv_key: str =
    STD_PRIV_KEY_FILE):
29     try:
30         d, n = load_key(fpriv_key)
31         block_size = (n.bit_length() + 7) // 8
32
33         with open(fininput, 'rb') as f:
34             data = f.read()
35
36         decrypted_data = bytearray()
37         for i in range(0, len(data), block_size):
38             block = data[i : i + block_size]
39             num = int.from_bytes(block, 'big')
40             decrypted_num = pow(num, d, n)
41             decrypted_block = decrypted_num.to_bytes((n.bit_length() //
                8 - 1), 'big')
42             decrypted_data.extend(decrypted_block)
43
44         with open(foutput, 'wb') as f:
45             f.write(decrypted_data)
46
47         print("Файл успешно расшифрован!")
48     except Exception as ex:
49         print(f"Ошибка при шифровании: {str(ex)}")

```

3.3 Пример работы программы

На рисунке 3.1 представлен пример работы программы на текстовом файле.

```
PS D:\Programms\InfSecurity\lab_02> python .\src\main.py keygen
Ключи успешно сгенерированы
Ключи успешно сохранены!
Ключи успешно сохранены в папку 'keys'
PS D:\Programms\InfSecurity\lab_02> cat .\test\file.txt
vhwuvwovouwevuebvuew
PS D:\Programms\InfSecurity\lab_02> python .\src\main.py encrypt .\test\file.txt .\test\efile.txt -k .\keys\.rsa.pub
Файл успешно зашифрован!
PS D:\Programms\InfSecurity\lab_02> cat .\test\efile.txt
7000001Waf300;105(+00000Y|00407{00000000u.g00T0aG$0%00ã]00ZDRUOY0*{0S0=00000@0w0<0%0 |
4060r0p0
090^0NuG>I\
PS D:\Programms\InfSecurity\lab_02> python .\src\main.py decrypt .\test\efile.txt .\test\dfile.txt -k .\keys\.rsa.priv
Файл успешно расшифрован!
PS D:\Programms\InfSecurity\lab_02> cat .\test\dfile.txt
vhwuvwovouwevuebvuew
PS D:\Programms\InfSecurity\lab_02>
```

Рисунок 3.1 – Пример работы программы на текстовом файле

На рисунках 3.2 — 3.5 представлен пример работы программы на zip-файле.

```
PS D:\Programms\InfSecurity\lab_02> python .\src\main.py encrypt .\test\example.zip .\test\ezip.zip -k .\keys\.rsa.pub
Файл успешно зашифрован!
PS D:\Programms\InfSecurity\lab_02> python .\src\main.py decrypt .\test\ezip.zip .\test\dzip.zip -k .\keys\.rsa.priv
Файл успешно расшифрован!
PS D:\Programms\InfSecurity\lab_02>
```

Рисунок 3.2 – Шифрация/дешифрация zip-файла

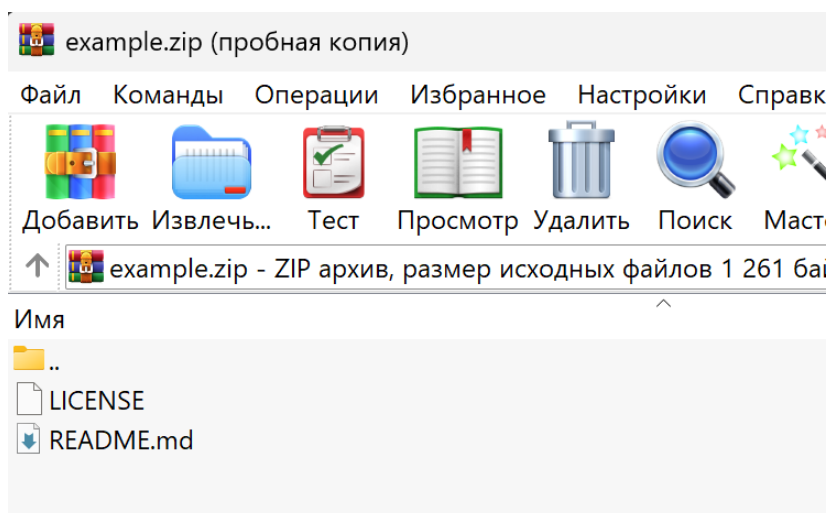


Рисунок 3.3 – Пример zip-файла

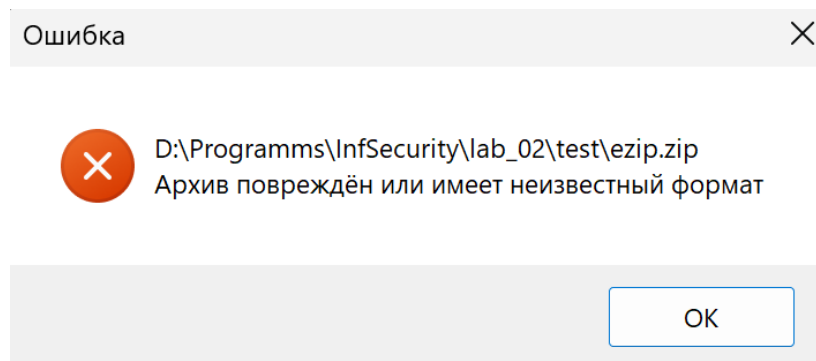


Рисунок 3.4 – Зашифрованный zip-файл

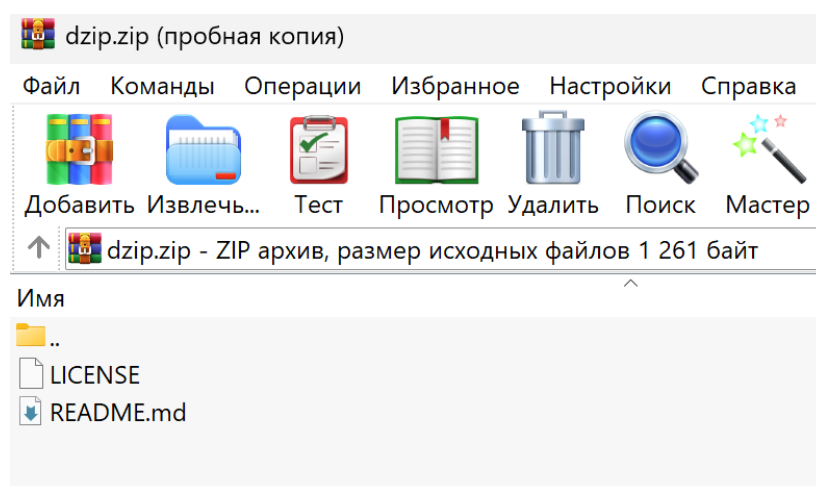


Рисунок 3.5 – Дешифрованный zip-файл

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы был изучен алгоритм шифрования RSA и разработана программная реализация. Были решены следующие задачи:

- проведен анализ работы алгоритма RSA;
- описан алгоритм шифрования RSA;
- реализована и протестирована реализация алгоритма шифрования RSA.