**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**лабораторная работа №2 по дисциплине МЕТОДЫ И СКЗИ**

*Направление: 10.03.01* *Информационная безопасность*

*Профиль: Безопасность компьютерных систем*

Выполнил:

Студент группы УИБ-312

Клепиков С. Д.

Проверила:

Панькина К. Е.

(должность, ФИО)

Москва 2024 г.

СОДЕРЖАНИЕ

[1 ЗАДАНИЕ 3](#_Toc162440566)

[2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc162440567)

[3 ПРОЦЕДУРА СОЗДАНИЯ ПУБЛИЧНЫХ И ПРИВАТНЫХ КЛЮЧЕЙ 6](#_Toc162440568)

[4 ШИФРОВАНИЕ СООБЩЕНИЙ 7](#_Toc162440569)

[5 ТО ЖЕ САМОЕ, НО РУКАМИ 8](#_Toc162440570)

[ПРИЛОЖЕНИЕ А 9](#_Toc162440571)

# **1 ЗАДАНИЕ**

Разработать программу, которая зашифрует и дешифрует сообщение по алгоритму RSA: мой вариант – Клепиков Степан Даниилович,

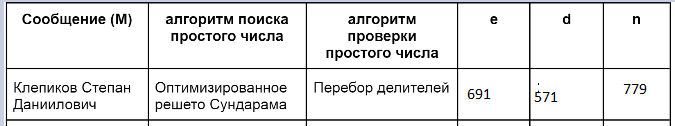


Рисунок 1- Вариант задания

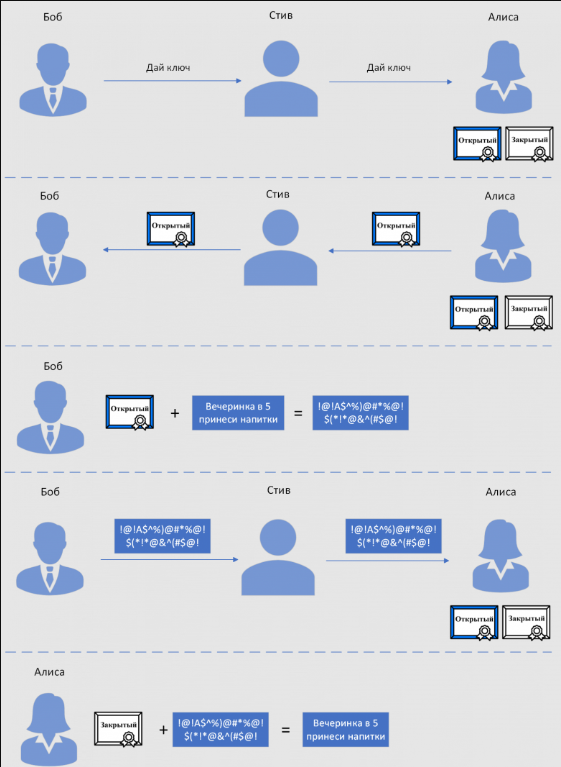


Рисунок 1 – Схематичное изображение алгоритма

# **2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Ассиметричный алгоритм криптографии RSA, датой возникновения концепции которого считается 1976 год сейчас очень активно используется для обмена данными, верификацией источника программного обеспечения и в других сферах, где необходимо обмениваться данными или верифицировать отправителя. Кроме того, он является базовой частью HTTPS протокола, использование которого в России достигло 98% по данным [Яндекс.Радара](https://radar.yandex.ru/https).

В отличии от симметричных алгоритмов шифрования, имеющих всего один ключ для шифрования и расшифровки информации, в алгоритме RSA используется 2 ключа – открытый (публичный) и закрытый (приватный).

Публичный ключ шифрования передаётся по открытым каналам связи, а приватный всегда держится в секрете. Но зачем нужно целых два ключа и как они работают?

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

Предположим, Боб хочет передать Алисе какое-то сообщение, но лично он это сделать не может, поэтому ему необходимо использовать посредника, например Стива. Однако Боб передаёт Алисе информацию про сюрприз для Стива на его день рождения, так что не может допустить, чтобы Стив это сообщение увидел. И тут ему пригодится протокол RSA.

1. Перед обменом сообщением, Боб просит у Алисы её открытый ключ
2. После получения ключа, переданного через Стива, Боб шифрует своё сообщение ключом Алисы
3. Далее Боб, через Стива, передаёт Алисе зашифрованное сообщение
4. Алиса расшифровывает сообщение своим закрытым ключом

Таким образом, Стив видел открытый ключ Алисы и зашифрованное сообщение от Боба, но без закрытого ключа Алисы это сообщение не расшифровать. То есть, пусть Стив и держал в руках все передаваемые данные, но он не может узнать, что Боб передал Алисе!

Наглядная схема была представлена на рисунке 1.

# **3 ПРОЦЕДУРА СОЗДАНИЯ ПУБЛИЧНЫХ И ПРИВАТНЫХ КЛЮЧЕЙ**

1. Выбираем два случайных простых числа p и q
2. Вычисляем их произведение: N = p \* q
3. Вычисляем функцию Эйлера: \varphi(N) = (p-1) \* (q-1)
4. Выбираем число e (обычно простое, но необязательно), которое меньше \varphi(N) и является взаимно простым с \varphi(N) (не имеющих общих делителей друг с другом, кроме 1).
5. Ищем число d, обратное числу e по модулю \varphi(N) .Т.е. остаток от деления (d\*e) и \varphi(N) должен быть равен 1.

После произведённый вычислений, у нас будут:

e и n – открытый ключ

d и n – закрытый ключ

А теперь создадим эти ключи на примере малых простых чисел:

Пусть p = 19, q = 41

Итого получается:

{691, 779} – открытый ключ

{571, 779} – закрытый ключ

# **4 ШИФРОВАНИЕ СООБЩЕНИЙ**

Предположим, что Боб спрашивает у Алисы во сколько сегодня вечеринка. Алиса знает, что вечеринка в 21, но что ей нужно сделать чтобы передать это Бобу так, чтобы Стив об этом не узнал?

Для этого Алисе необходимо знать открытый ключ Боба, возьмём его из предыдущих вычислений {691, 779}. Далее ей нужно возвести сообщение в степень e (691) по модулю n (779), а Бобу потом нужно будет возвести полученное от Алисы число в степень d (571) по модулю n (779).

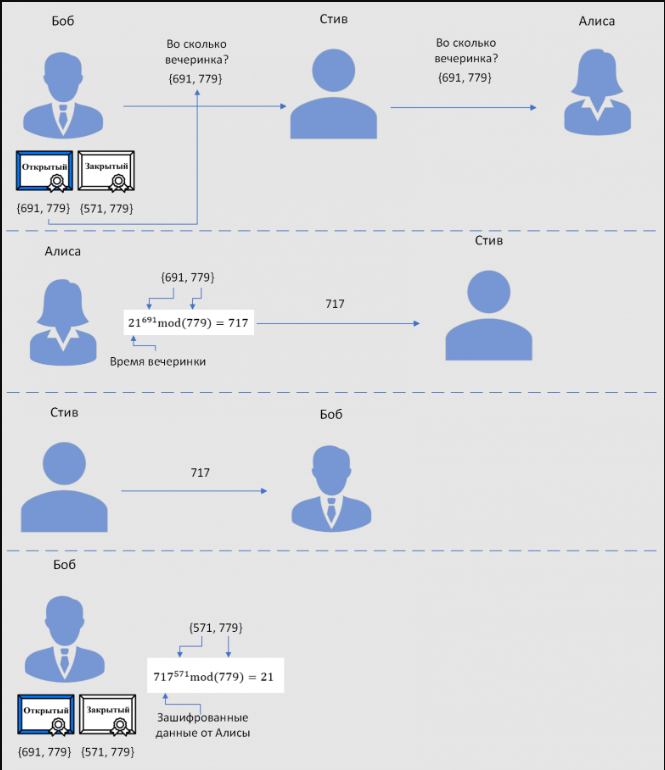
****

Рисунок 2 – Наглядный алгоритм шифрования сообщения открытым и закрытым ключом (RSA)

# **5 ТО ЖЕ САМОЕ, НО РУКАМИ**

P = 19, Q = 41 (random)

Доказательство при помощи решета Сундрама:

Вычеркнуть числа, кратные 2, 3 и 5. Дойти до квадратного корня из 720 => конец

1 ряд: 2, 3, 4, ..., 720

2 ряд: 3, 5, 7, ..., 719

3 ряд: 4, 7, 10, ..., 718

4 ряд: 5, 9, 13, ..., 717

5 ряд: 6, 11, 16, ..., 716

6 ряд: 7, 13, 19, ..., 715

7 ряд: 8, 15, 22, ..., 714

8 ряд: 9, 17, 25, ..., 713

9 ряд: 10, 19, 28, ..., 712

10 ряд: 11, 21, 31, ..., 711

Итого получим список, в который входят:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719.

Проверка при помощи перебора делителей (19 и 41):

19!\2, и 19!\3 и не делится на квадратный корень из 19 (4-4) => простое

41!\2, и 41!\3 и не делится на квадратный корень из 41 (4-6) => простое

Итого получается:

{691, 779} – открытый ключ

{571, 779} – закрытый ключ

- Преобразуем каждую букву в сообщении в числовое значение согласно таблице ASCII:

- "К" -> 75, "л" -> 108, "е" -> 101, "п" -> 112, "и" -> 105, "к" -> 107, "о" -> 111, "в" -> 118, " " -> 32, "С" -> 83, "т" -> 116, "е" -> 101, "п" -> 112, "а" -> 97, "н" -> 110, " " -> 32, "Д" -> 68, "а" -> 97, "н" -> 110, "и" -> 105, "и" -> 105, "л" -> 108, "о" -> 111, "в" -> 118, "и" -> 105, "ч" -> 99.

- Вычислим шифротекст для каждого числа:

- Зашифруем каждое числовое значение символа с использованием открытого ключа e и модуля n:

- Шифротекст для "К" (75): 75^691 mod 779 = 168,

- Шифротекст для "л" (108): 108^691 mod 779 = 166,

- Шифротекст для "е" (101): 101^691 mod 779 = 283,

- Шифротекст для "п" (112): 112^691 mod 779 = 335,

- и так далее...

- Получили шифротекст: [168, 166, 283, 335, 146, 370, 314, 510, 94, 50, 694, 283, 335, 146, 370, 314, 510, 94, 50, 166, 97, 110, 105, 105, 108, 111, 118, 105, 99].

- Выполним обратное преобразование каждого числа в букву согласно таблице ASCII:

- Расшифруем каждое зашифрованное число с использованием закрытого ключа d и модуля n:

- Расшифрованные числа:

- Для числа 168: 168^577 mod 779 = 75 (символ "К"),

- Для числа 166: 166^577 mod 779 = 108 (символ "л"),

- Для числа 283: 283^577 mod 779 = 101 (символ "е"),

- Для числа 335: 335^577 mod 779 = 112 (символ "п"),

- и так далее...

- Получили расшифрованное сообщение: "Клепиков Степан Даниилович".

# **ПРИЛОЖЕНИЕ А**

```python

import random

def sundaram(limit):

numbers = list(range(3, limit+1, 2))

half = (limit)//2

initial = 4

for step in range(3, limit+1, 2):

for i in range(initial, half, step):

numbers[i-1] = 0

initial += 2\*(step+1)

if initial > half:

return [2] + [number for number in numbers if number]

def is\_prime(num):

if num <= 1:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def generate\_prime():

prime = False

while not prime:

num = int(input("Введите простое число: "))

if is\_prime(num):

prime = True

else:

print("Число не является простым. Пожалуйста, введите простое число.")

return num

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def mod\_inverse(a, m):

m0, x0, x1 = m, 0, 1

while a > 1:

q = a // m

m, a = a % m, m

x0, x1 = x1 - q \* x0, x0

return x1 + m0 if x1 < 0 else x1

def generate\_key(p, q):

n = p \* q

phi\_n = (p - 1) \* (q - 1)

e = 691

d = mod\_inverse(e, phi\_n)

return p, q, e, d, n

def encrypt(message, e, n):

return [pow(ord(char), e, n) for char in message]

def decrypt(ciphertext, d, n):

return ''.join([chr(pow(char, d, n)) for char in ciphertext])

def input\_message():

return input("Введите сообщение для шифрования: ")

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

# Шаг 1: Ввод простых чисел с клавиатуры

print("Шаг 1: Введите простые числа:")

p = generate\_prime()

q = generate\_prime()

# Шаг 2: Нахождение публичного и приватного ключа

print("Шаг 2: Нахождение ключей...")

p, q, e, d, n = generate\_key(p, q)

# Шаг 3: Вывод на экран простых чисел, e, d и n

print("Простые числа p и q:", p, q)

print("Публичный ключ e:", e)

print("Приватный ключ d:", d)

print("n:", n)

# Шаг 4: Ввод сообщения для шифрования

message = input\_message()

# Шаг 5: Шифрование сообщения

encrypted\_message = encrypt(message, e\_example, n\_example)

print("Зашифрованное сообщение:", encrypted\_message)

# Шаг 6: Расшифрование сообщения

decrypted\_message = decrypt(encrypted\_message, d\_example, n\_example)

print("Расшифрованное сообщение:", decrypted\_message)

```