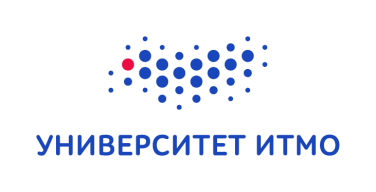
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**



Факультет безопасности информационных технологий

**Дисциплина:**

«Основы информационной безопасности»

Отчет по лабораторной работе RSA

Выполнил: Студент гр. N3147

Чу Ван Доан

Проверил:Еремук В.В

Санкт-Петербург 2023

**Цель: Ознакомление с основами асимметричного шифрования на примере криптосистемы RSA. Разработка программы для шифрования и дешифрования сообщений с использованием RSA.**

**1.Сгенерированные параметры RSA (p, q, открытый/закрытый ключ)**

Параметры RSA генерируются случайным образом и должны оставаться конфиденциальными, чтобы обеспечить безопасность системы шифрования.

Криптосистема RSA использует асимметричный криптографический алгоритм. В отличие от симметричных алгоритмов, у которых используется один ключ для шифрования и дешифрования, у асимметричных алгоритмов есть два разных ключа: открытый и закрытый.

Для генерации ключей в алгоритме RSA выбираются два больших простых числа p и q. Их произведение n = p \* q называется модулем.

Также вычисляется функция Эйлера от числа n: φ(n) = (p - 1) \* (q - 1).

Выбирается целое число e, такое что 1 < e < φ(n), и наибольший общий делитель e и φ(n) равен 1. Пара чисел (n, e) является открытым ключом, а пара чисел (n, d) является закрытым ключом. Число d вычисляется как обратное по модулю φ(n) к числу e, то есть такое, что d \* e ≡ 1 (mod φ(n)).

Для шифрования сообщения M с помощью открытого ключа (n, e) вычисляется зашифрованное сообщение C = M^e mod n.

Для дешифрования зашифрованного сообщения C с помощью закрытого ключа (n, d) вычисляется исходное сообщение M = C^d mod n.

**2.Исходное сообщение**



**3.Результат зашифрования сообщения**

****

**4.Результат расширования сообщения**



**5.Листинг разработанной программы**

**import random**

**def gcd(a, b):**

**while b:**

**a, b = b, a % b**

**return a**

**def multiplicative\_inverse(e, phi):**

**d = 0**

**x1 = 0**

**x2 = 1**

**y1 = 1**

**temp\_phi = phi**

**while e > 0:**

**temp1 = temp\_phi // e**

**temp2 = temp\_phi - temp1 \* e**

**temp\_phi = e**

**e = temp2**

**x = x2 - temp1 \* x1**

**y = d - temp1 \* y1**

**x2 = x1**

**x1 = x**

**d = y1**

**y1 = y**

**if temp\_phi == 1:**

**return d + phi**

**def generate\_keypair(p, q):**

**n = p \* q**

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

**e = random.randrange(1, phi)**

**g = gcd(e, phi)**

**while g != 1:**

**e = random.randrange(1, phi)**

**g = gcd(e, phi)**

**d = multiplicative\_inverse(e, phi)**

**return ((e, n), (d, n))**

**def encrypt(pk, plaintext):**

**key, n = pk**

**cipher = [pow(ord(char),key,n) for char in plaintext]**

**return cipher**

**def decrypt(pk, ciphertext):**

**key, n = pk**

**plain = [chr(pow(char, key, n)) for char in ciphertext]**

**return ''.join(plain)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**print("Генерация открытого и закрытого ключей...")**

**public, private = generate\_keypair(17, 19)**

**print("Открытый ключ: ", public)**

**print("Закрытый ключ: ", private)**

**message = "Hello World"**

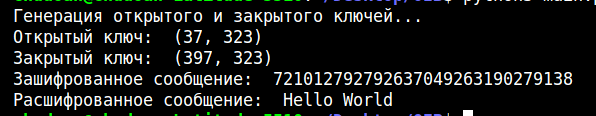
**encrypted\_msg = encrypt(public, message)**

**print("Зашифрованное сообщение: ", ''.join(map(lambda x: str(x), encrypted\_msg)))**

**decrypted\_msg = decrypt(private, encrypted\_msg)**

**print("Расшифрованное сообщение: ", decrypted\_msg)**

**Вывод:**

****