МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по практическому заданию №12**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил

Студенты 46 группы

Прозоров М.С.

Преподаватель:

Крамаренко А.А.

Краснодар 2024

**Постановка задачи**

Реализовать программный продукт, позволяющий подписывать и проверять подпись вводимого сообщения согласно схеме RSA.

Алгоритм RSA для подписи и проверки подписи сообщений работает следующим образом:

1. Генерация ключей

Выбор двух простых чисел (p) и (q): Для начала необходимо выбрать два больших простых числа. Их размер должен быть достаточным для обеспечения необходимого уровня безопасности.

Вычисление (n = p × q): Произведение p и q дает n, которое используется как часть обоих ключей.

Вычисление функции Эйлера от n: Функция Эйлера (φ(n) = (p-1) × (q-1)).

Выбор открытой экспоненты e: выбирается число e, которое взаимно просто с (φ(n)) и меньше (φ(n)). Обычно используется 65537 за его криптографические свойства.

Вычисление закрытой экспоненты d: выбирается так, чтобы (d × e ≡ 1 mod φ(n)).

После выполнения этих шагов:

Открытый ключ состоит из пары ((n, e)).

Закрытый ключ состоит из пары ((n, d)).

1. Подпись сообщения

Хэширование сообщения: сначала сообщение хэшируется с использованием криптографически стойкой хеш-функции, например SHA-256. Это дает фиксированный размер хеша сообщения.

Шифрование хеша: затем хеш сообщения шифруется с использованием закрытого ключа отправителя (используется значение d). Это шифрование хеша, а не самого сообщения, и является цифровой подписью.

1. Проверка подписи

Дешифрование подписи: Получатель дешифрует подпись с использованием открытого ключа отправителя (используется значение e). Это действие возвращает хеш сообщения.

Хэширование полученного сообщения: Получатель также хэширует полученное сообщение той же хеш-функцией.

Сравнение хешей: Если хеш, полученный путем дешифрования подписи, совпадает с хешем полученного сообщения, подпись считается подлинной, и сообщение — неизменным.

**Вывод**

Реализовал программный продукт, позволяющий подписывать и проверять подпись вводимого сообщения согласно схеме RSA.

**Листинг:**

from Crypto.PublicKey import RSA

from Crypto.Signature import pkcs1\_15

from Crypto.Hash import SHA256

import binascii

# Генерация пары ключей RSA

def generate\_keys():

key = RSA.generate(2048)

private\_key = key.export\_key()

public\_key = key.publickey().export\_key()

return private\_key, public\_key

# Подпись сообщения с использованием приватного ключа

def sign\_message(private\_key, message):

rsakey = RSA.import\_key(private\_key)

msg\_hash = SHA256.new(message.encode())

signer = pkcs1\_15.new(rsakey)

signature = signer.sign(msg\_hash)

return binascii.hexlify(signature).decode()

# Проверка подписи сообщения с использованием публичного ключа

def verify\_signature(public\_key, message, signature):

rsakey = RSA.import\_key(public\_key)

msg\_hash = SHA256.new(message.encode())

signature = binascii.unhexlify(signature)

try:

pkcs1\_15.new(rsakey).verify(msg\_hash, signature)

return True

except (ValueError, TypeError):

return False

# Пример использования

private\_key, public\_key = generate\_keys()

print(f'private key: {private\_key}')

print(f'public key: {public\_key}')

# Подпись сообщения

message = "Hello, World!"

print(f'message: {message}')

signature = sign\_message(private\_key, message)

print(f"Подпись: {signature}")

# Проверка подписи

verification\_result = verify\_signature(public\_key, message, signature)

print(f"Результат проверки подписи: {'Успешно' if verification\_result else 'Ошибка'}")