Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине «Криптографические методы защиты информации»

Тема: «Программная реализация ЭЦП»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Пучинский А.А.

**Проверил:**

Хацкевич А. С.

Брест 2023

**Цель:** создать программу, которая реализует учебный вариант схем ЭЦП, используя алгоритмы с открытыми ключами.

**Ход работы:**

**Вариант 3**

Цифровая подпись на базе алгоритма Шнорра

Код сервера:

import socket

import hashlib

import random

def generate\_keys(p, q, g):

x = random.randint(1, q - 1)

y = pow(g, x, p)

return x, y

def sign\_message(p, q, g, x, message):

k = random.randint(1, q - 1)

r = pow(g, k, p) % q

h = int(hashlib.sha256(message.encode()).hexdigest(), 16)

s = (pow(k, -1, q) \* (h + x \* r)) % q

return r, s

def verify\_signature(p, g, y, message, r, s, q):

h = int(hashlib.sha256(message.encode()).hexdigest(), 16)

w = pow(s, -1, q)

u1 = (h \* w) % q

u2 = (r \* w) % q

v = (pow(g, u1, p) \* pow(y, u2, p) % p) % q

return v == r

def start\_server():

host = '127.0.0.1'

port = 12345

p = 23

q = 11

g = 5x, y = generate\_keys(p, q, g)

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:

server\_socket.bind((host, port))

server\_socket.listen()

print(f"Server listening on {host}:{port}")

conn, addr = server\_socket.accept()

with conn:

print(f"Connected by {addr}")

public\_key = f"{p},{q},{g},{y}"

conn.sendall(public\_key.encode())

print(f"Key exchange completed. Public key: {public\_key}")

while True:

message = conn.recv(1024).decode()

if not message:

break

print(f"Received message: {message}")

r, s = sign\_message(p, q, g, x, message)

signature = f"{r},{s}"

conn.sendall(signature.encode())

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

start\_server()

}

Код клиента:

import socket

import hashlib

def verify\_signature(p, g, y, message, r, s, q):

h = int(hashlib.sha256(message.encode()).hexdigest(), 16)

w = pow(s, -1, q)

u1 = (h \* w) % q

u2 = (r \* w) % q

v = (pow(g, u1, p) \* pow(y, u2, p) % p) % q

return v == r

def start\_client():

host = '127.0.0.1'

port = 12345

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as client\_socket:

client\_socket.connect((host, port))

public\_key = client\_socket.recv(1024).decode()

p, q, g, y = map(int, public\_key.split(','))

print(f"Key exchange completed. Public key: {public\_key}")

while True:

message = input("Enter message (or 'exit' to quit): ")

if message.lower() == 'exit':

break

print(f"Message to sign: {message}")

client\_socket.sendall(message.encode())

signature = client\_socket.recv(1024).decode()

r, s = map(int, signature.split(','))

print("Verifying signature...")

result = verify\_signature(p, g, y, message, r, s, q)

if result:

print("Signature is valid.")

else:

print("Signature is invalid.")

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

start\_client()

**Вывод:** в ходе лабораторной работы я изучил работу ЭЦП, создал ее на базе алгоритма Шнорра.