Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
За пятый семестр
По дисциплине: «Криптографические методы защиты информации»

Тема: «Программная реализация ЭЦП»

Выполнила:
Студент 3 курса
Группы ИИ-21(II)
Кирилович А. А.

Проверила:
Хацкевич А. С.

Брест 2023

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

Ход работы:

Вариант 4 (Цифровая подпись Рабина)

Задание:

Файл rabin.py

```
import hashlib

# security level 1 means  512 bits public key and hash length
SECURITY_LEVEL = 1

def gcd(a: int, b: int) -> int:
    if b > a:
        a, b = b, a
    while b > 0:
        a, b = b, a % b
    return a
def next_prime(p: int) -> int:
    while p % 4 != 3:
        p = p + 1
    return next_prime_3(p)
def next_prime_3(p: int) -> int:
    m_ = 3 * 5 * 7 * 11 * 13 * 17 * 19 * 23 * 29
    while gcd(p, m_) != 1:
        p = p + 4
    if pow(2, p - 1, p) != 1 or pow(3, p - 1, p) != 1 or pow(5, p - 1, p) != 1 or
pow(17, p - 1, p) != 1:
        return next_prime_3(p + 4)
    return p
def hash512(x: bytes) -> bytes:
    hx = hashlib.sha256(x).digest()
    idx = len(hx) // 2
    return hashlib.sha256(hx[:idx]).digest() + hashlib.sha256(hx[idx:]).digest()
def hash_to_int(x: bytes) -> int:
    hx = hash512(x)
    for _ in range(SECURITY_LEVEL - 1):
        hx += hash512(hx)
    return int.from_bytes(hx, 'little')
def sign_rabin(p: int, q: int, digest: bytes) -> tuple:
    n = p * q
    i = 0
    while True:
        h = hash_to_int(digest + b'\x00' * i) % n
        if (h % p == 0 or pow(h, (p - 1) // 2, p) == 1) and (h % q == 0 or pow(h,
(q - 1) // 2, q) == 1):
            break
        i += 1
    lp = q * pow(h, (p + 1) // 4, p) * pow(q, p - 2, p)
    rp = p * pow(h, (q + 1) // 4, q) * pow(p, q - 2, q)
    s = (lp + rp) % n
    return s, i
def verify_rabin(n: int, digest: bytes, s: int, padding: int) -> bool:
    return hash_to_int(digest + b'\x00' * padding) % n == (s * s) % n
def write_number(number: int, filename: str) -> None:
    with open(f'{filename}.txt', 'w') as f:
        f.write('%d' % number)
def read_number(filename: str) -> int:
    with open(f'{filename}.txt', 'r') as f:
        return int(f.read())
def sign(hex_message: str) -> tuple:
    p = read_number('p')
    q = read_number('q')
```

```
    return sign_rabin(p, q, bytes.fromhex(hex_message))
def verify(hex_message: str, padding: str, hex_signature: str):
    n = read_number('n')
    return verify_rabin(n, bytes.fromhex(hex_message), int(hex_signature, 16),
int(padding))
def V(mes, pad, sig):
    return verify(mes, pad, sig)
def S(mes):
    sig, pad = sign(mes)
    return hex(sig), pad
def G(seed):
    priv_range = 2 ** (256 * SECURITY_LEVEL)
    p_rabin = next_prime(hash_to_int(bytes.fromhex(seed)) % priv_range)
    q_rabin = next_prime(hash_to_int(bytes.fromhex(seed + '00')) % priv_range)
    write_number(p_rabin, 'p')
    write_number(q_rabin, 'q')
    write_number(p_rabin * q_rabin, 'n')
```

Файл client.py

```
import socket
import rabin

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_address = ('127.0.0.1', 12345)
client_socket.connect(server_address)

rabin.G("01")

message = "00112233445566778899aabbccddeeff"
sig, pad = rabin.S(message)
message = f'{message},{pad},{sig[2:]}'
client_socket.send(message.encode('utf-8'))

data = client_socket.recv(1024)
print("Response from the server:", data.decode('utf-8'))

client_socket.close()
```

Файл server.py

```
import socket
import rabin

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_address = ('127.0.0.1', 12345)
server_socket.bind(server_address)

server_socket.listen(5)
print("The server is listening on {}:{}".format(*server_address))

while True:
    client_socket, client_address = server_socket.accept()
    print("Connection from:", client_address)

    data = client_socket.recv(1024)

    if data:
        response = f"result of verification: {rabin.V(*data.decode('utf-
8').split(','))}"
        client_socket.send(response.encode('utf-8'))

    client_socket.close()
```

Вывод: создал программу, которая реализует алгоритм криптографической системы Рабина.