Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем
Математичні основи захисту інформації

Лабораторна робота №13
“Електронний підпис на основі протоколу Ель-Гамаля”

Виконали студенти 3-го курсу
Групи ІПС-32

Роботу виконали:
Ольховатий Ігор
Ковальов Володимир
Тряско Софія
Цілинко Олександр
Бондар Юлія
Волик Артем

Київ 2023

Тема: реалізувати алгоритм електронного підпису на основі протоколу Ель-Гамаля

Розв'язок:

Робота Аліси:

Генеруємо випадкове велике просте число

$p = 147031$ - зберігається у відкритому вигляді

$g = 60051$ - зберігається у відкритому вигляді

$x = 67319$ - приватний ключ Аліси

$y = g^x \bmod p = 49258$ відкритий ключ Аліси

Повідомлення:

$m = This\ is\ Alice$

Випадково генеруємо число взаємно просте з $p - 1$

$k = 10333$

$r = g^k \bmod p = 114595$ - передається як елемент електронного підпису Аліси

Обчислюємо хеш-функцію повідомлення m. Ця хеш-функція може бути довільною

$h = 116334$ - хеш функція повідомлення

$u = (h - xr) \bmod (p - 1) = 108741$

$k^{-1} = 146347$

$s = k^{-1} u \ (mod(p - 1)) = 60523$ - передається як елемент електронного підпису Аліси

Підпис Аліси: $(m, s, r) = (This\ is\ Alice,\ 60523,\ 114595)$

Робота Боба

Перевіримо підпис

$y^r * r^s \ (mod\ p) = 2567$

$g^h \bmod p = 2567$

Ці числа співпадають, а отже алгоритм виконано правильно

Програмна реалізація

```
import random

def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
```

```
            return False
    return True

def generate_random_prime():
    while True:
        random_num = random.randint(2, 10**6)  # Adjust the
range as needed
        if is_prime(random_num):
            return random_num
def power(a:int, d:int,m:int):
    d_binary = bin(d)

    d_binary = d_binary[2:]
    y =1
    s = a

    for i in reversed(d_binary):
        if i == '1':
            y = (y*s)%m

        s = (s*s)%m

    return y
def simple_hash_function(input_string):
    # Using the built-in hash() function
    hashed_value = hash(input_string)
    return hashed_value
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
def extended_gcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = extended_gcd(b % a, a)
        return (g, y - (b // a) * x, x)

def mod_inverse(a, p):
    g, x, _ = extended_gcd(a, p)
    if g != 1:
```

```
        raise ValueError(f"The modular inverse does not
exist for {a} modulo {p}")
    else:
        return x % p
if __name__ == '__main__':
    m = "This is Alice"

    random_prime = generate_random_prime()
    x = random.randint(2, random_prime)
    print("Random prime number:", random_prime)

    g  = random.randint(2, random_prime)
    y = power(g,x,random_prime)

    print('g:',g)
    print('x: ', x)
    print('y:', y)
    while True:
        k  = random.randint(2, random_prime)
        if gcd(k,random_prime-1)==1:
            break
    print('k:', k)
    r = power(g,k,random_prime)
    print("r:", r)
    h = simple_hash_function(m) % random_prime
    print('h:', h)
    u = (h - x*r) % (random_prime-1)
    print("u:", u)
    k_inv = mod_inverse(k,random_prime-1)
    print('k_inv:', k_inv)
    s = (k_inv*u) % (random_prime-1)
    print("s:", s)

    print('Підпис Аліси: ', m,s,r)

    print('------------')
    l = (power(y,r,random_prime) * power(r,s,random_prime))
% random_prime
    r = power(g,h,random_prime)% random_prime
    print(l,r)
    if l == r:
```

```
        print('Електронний підпис коректний')
```

Література

- Лекції з предмету “Математичні основи захисту інформації”
- https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%AD%D0%BB%D1%8C-%D0%93%D0%B0%D0%BC%D0%B0%D0%BB%D1%8F