МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Схема подписи DSA

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Норикова Павла Сергеевича

Преподаватель
аспирант _____ Р. А. Фарахутдинов
подпись, дата

Саратов 2023

ВВЕДЕНИЕ

Цель работы – изучение и реализация схемы подписи DSA.

1 Теория

DSA (англ. Digital Signature Algorithm – алгоритм цифровой подписи) – криптографический алгоритм с использованием закрытого ключа (из пары ключей: <открытый; закрытый>) для создания электронной подписи, но не для шифрования (в отличие от RSA и схемы Эль-Гамаля). Подпись создается секретно (закрытым ключом), но может быть публично проверена (открытым ключом). Это означает, что только один субъект может создать подпись сообщения, но любой может проверить её корректность. Алгоритм основан на вычислительной сложности взятия логарифмов в конечных полях.

1.1 Параметры схемы цифровой подписи

1. Выбирается простое число p.
2. Выбор криптографической хеш-функции $H(x)$.
3. Выбор простого числа q, размерность которого в битах совпадает с размерностью в битах значений хеш-функции $H(x)$.
4. Выбор простого числа p, такого, что $(p-1)$ делится на q.
5. Выбор числа $g > 1$ такого, что его мультипликативный порядок по модулю p равен q. Для его вычисления можно воспользоваться формулой

$$g = h^{\frac{p-1}{q}} \bmod p,$$

где $1 < h < p-1$ — некоторое произвольное число.

1.2 Генерация ключей

1. Секретный ключ представляет собой число $x \in (0, q)$
2. Открытый ключ вычисляется по формуле $y = g^x \bmod p$

1.3 Подпись сообщения

1. Выбор случайного числа $k \in (0, q)$
2. Вычисление $r = (g^k \bmod p) \bmod q$

3. Выбор другого k, если $r = 0$
4. Вычисление $s = k^{-1}(H(m) + xr)\ mod\ q$
5. Выбор другого k, если $s = 0$
6. Подписью является пара (r, s) общей длины $2N$

1.4 Проверка подписи

1. Вычисление $w = s^{-1}\ mod\ q$
2. Вычисление $u_1 = H(m) * w\ mod\ q$
3. Вычисление $u_2 = r * w\ mod\ q$
4. Вычисление $v = (g^{u_1} * y^{u_2}\ mod\ p)\ mod\ q$
5. Подпись верна, если $v = r$

2 Практическая реализация

2.1 Описание программы

Функции *gen_p*, *gen_q* и *gen_g* генерируют числа p, q и g соответственно.

Функции *gcd_ex* и *gcd* реализуют расширенный и обычный алгоритм Евклида соответственно.

На вход программе подается сообщение m.

2.2 Тестирование программы

```
Введите сообщение: Norikov
Генерация подписи:
h = 2709995991867635711
Общие элементы:
p = 35624832104095143011221074150535257437463125528237257323896329350078791 69
q = 2324628761005493143
g = 17930889851684990627018184977320413594864207058339426845472001384315023 10

Открытый ключ:
y = 14497183712497861782967120766193624513486787645986693832847147889393856 03

Закрытый ключ:
x = 2095770750760911025

k = 767992980135852696
r = 343684577772259029
s = 266269337652550206
A -> B: { Norikov 343684577772259029 266269337652550206 }


Проверка подписи:
u = 303256335581249474
a = 1109051734254173348
b = 812705177706261946
v = 343684577772259029
Результат проверки: True
```

Рисунок 1 - Работа протокола

ПРИЛОЖЕНИЕ А

Листинг программы

```
import random
from sympy import isprime


def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def gcd_ex(a, b):
    if b == 0:
        return a, 1, 0
    else:
        gcd, x1, y1 = gcd_ex(b, a % b)
        x = y1
        y = x1 - (a // b) * y1
        return gcd, x, y


def gen_p(q):
    res = 2 * q
    while not isprime(res + 1):
        res *= 2
    return res + 1


def gen_q(L):
    res = "0"
    while not isprime(int(res, 2)):
        res = ""
        for i in range (1, L - 1):
            random.seed()
            res += str(random.randint(0,100)%2)
        res = '1' + res + '1'
    return int(res, 2)



def gen_g(q, p):
    res = 1
    h = 2
    while res == 1:
        res = pow(h, (p - 1) // q, p)
        h = random. randint(2, p - 2)
    return res



def main(m):
    print('Генерация подписи:')
    #h = m
    h = abs(int(hash(str(m))))
```

```
    print('h =', h)
    q = gen_q(len(bin(h)) - 2)
    p = gen_p(q)
    g = gen_g(q, p)
    x = random.randint(1, q - 1)
    y = pow(g, x, p)
    print('Общие элементы:', '\np =', p, '\nq =', q, '\ng =', g, '\n')
    print('Открытый ключ:', '\ny =', y, '\n')
    print('Закрытый ключ:', '\nx =', x, '\n')

    s = 0
    k = 0
    while s == 0 or r == 0:
        k = random.randint(1, q - 1)
        #k = 3
        r = pow(g, k, p) % q
        s = (gcd_ex(k, q)[1] * (h + x*r)) % q
        #print(s)

    print('k =', k)
    print('r =', r)
    print('s =', s)
    print('A -> B: {', m, r, s, '}\n')

    print('\nПроверка подписи:')
    u = gcd_ex(s, q)[1]
    print('u =', u)
    a = (h * u) % q
    print('a =', a)
    b = (r * u) % q
    print('b =', b)
    v = (pow(g, a, p) * pow(y, b, p)) % p % q
    print('v =', v)
    print('Результат проверки:', v == r)


s = str(input('Введите сообщение: '))
main(s)
```