Лабораторная работа 5

Схема Эль-Гамаля и хэш-функция SHA-2

Вариант 18

Раконяц Даниела

7 группа

2023

**Схема Эль-Гамаля** (Elgamal) — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи.

Формально, как и для любой схемы электронно-цифровой подписи (ЭЦП), для работы схемы Эль-Гамаля необходимо реализовать 3 основных алгоритма: генерацию ключей, функцию подписи и функцию проверки подписи. Подробности можно посмотреть в лекции. С учетом того, что в лабораторной не требуется реализовывать генерацию простых чисел, алгоритмы ЭЦП имеют следующий вид:

Генерация ключей (Gen)

Вход: q – простое число. Шаги:

1. Выбрать такое четное число R, что R < 4 (q + 1).
2. Вычислить число p = qR + 1.
3. Если2qR ≠1(modp)или2R =1(modp),товозвратитьсякшагу1,иначеp–  
   простое.
4. Случайным образом выбрать x из Zp и вычислить g = xR (mod p).
5. Если g = 1, то возвратиться к шагу 4, иначе искомое g найдено.
6. Случайным образом выбрать личный ключ d из Zq.
7. Вычислить открытый ключ e = gd (mod p).

Выход: (p, q, g) – параметры ЭЦП; e – открытый ключ; d – личный ключ. Функция подписи (Sign)

Вход: (p, q, g) – параметры ЭЦП; d – личный ключ; M – подписываемое сообщение (в виде строки текста произвольной длины).

Шаги:  
1. Вычислить хэш-значение m от сообщения M: m = h(M) (h() – хэш-функция). 2. Случайным образом выбрать одноразовый личный ключ k из Zq / {0}.  
3. Вычислить r = gk (mod p).  
4. Вычислить s = k-1(m – dr) (mod q).

Выход: (r, s) – подпись.  
Функция проверки подписи (Verify)

Вход: (p, q, g) – параметры ЭЦП; e – открытый ключ; M – подписываемое сообщение (в виде строки текста произвольной длины; (r, s) – подпись.

Шаги:

1. Если r не лежит в Zp / {0} или s не лежит в Zq, то вернуть FALSE.
2. Вычислить хэш-значение m от сообщения M: m = h(M) (h() – хэш-функция).
3. Если errs = gm (mod p), то вернуть TRUE, иначе вернуть FALSE.

Выход: TRUE, если подпись корректна; FALSE, если подпись некорректна.

Задание:

| **№ вар.** | **q** |
| --- | --- |
| 18 | 172653929034048084950674429899193619418057473778703331247504102312524336679143 |

Код:

from hashlib import sha256

from random import randrange

from typing import Tuple

from sympy import isprime, mod\_inverse

##from utils import mod\_inverse, pow\_

import utils

def gen(q: int) -> Tuple[Tuple[int, int, int], int, int]:

if not isprime(q):

raise ValueError(f"Expected q to be prime, but {q} is not prime")

while True:

r = randrange(0, 4 \* (q + 1), 2)

p = q \* r + 1

if pow(2, q \* r, p) != 1 or pow(2, r, p) == 1:

continue

while True:

x = randrange(0, p)

g = pow(x, r, p)

if g == 1:

continue

else:

break

d = randrange(0, q)

e = pow(g, d, p)

break

digital\_signature\_params = (p, q, g)

open\_key = e

personal\_key = d

return digital\_signature\_params, open\_key, personal\_key

def sign(

signature\_params: Tuple[int, int, int], personal\_key: int, message: str

) -> Tuple[int, int]:

p, q, g = signature\_params

d = personal\_key

if not (0 <= personal\_key <= q):

raise ValueError()

m = int(sha256(bytes(message, encoding="utf-8")).hexdigest(), 16)

k = randrange(1, q)

r = pow(g, k, p)

s = (mod\_inverse(k, q) \* (m - d \* r)) % q

return r, s

def verify(

signature\_params: Tuple[int, int, int],

open\_key: int,

message: str,

signature: Tuple[int, int],

) -> bool:

p, q, g = signature\_params

r, s = signature

e = open\_key

if not (0 <= open\_key < p):

raise ValueError()

m = int(sha256(bytes(message, encoding="utf-8")).hexdigest(), 16)

if (pow(e, r, p) \* pow(r, s, p)) % p == pow(g, m, p):

return True

return False

def main():

signature\_params, open\_key, personal\_key = None, None, None

while True:

available\_operations = ["gen", "sign", "verify", "exit"]

operation = input(

"Enter gen for keys signature, sign for signing message, verify to verify signature authenticity"

+ "exit to exit the program: "

)

if operation not in available\_operations:

print(f"Unknown operation, use one of {available\_operations}")

continue

if operation == "exit":

return

if operation == "gen":

q = int(input("Enter q: "))

signature\_params, open\_key, personal\_key = gen(q)

print("Keys are generated")

continue

if operation == "sign":

if signature\_params is None:

print("Keys need to be generated first, use gen command")

continue

else:

message = input("Enter message:")

r, s = sign(signature\_params, personal\_key, message)

print(f"r = {r}")

print(f"s = {s}")

continue

if operation == "verify":

if signature\_params is None:

print("Keys need to be generated first, use gen command")

continue

else:

r = int(input("Enter r: "))

s = int(input("Enter s: "))

message = input("Enter message: ")

print(verify(signature\_params, open\_key, message, (r, s)))

continue

def report():

with open("report.txt", "w") as f:

q = 172653929034048084950674429899193619418057473778703331247504102312524336679143

signature\_params, open\_key, personal\_key = gen(q)

f.write(f"Gen: {signature\_params, open\_key, personal\_key} \n")

message = "I, Rakonjac Daniela, love MiKOZI"

signature = sign(signature\_params, personal\_key, message)

f.write(f"Sign: {signature} \n")

ver\_result = verify(signature\_params, open\_key, message, signature)

f.write(f"Verify: {ver\_result} \n")

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

main()

report()

Результат(из report.txt):

Gen: ((9425868805138302636683961023430036177657734773683307899590846158804085230649170796148747737052860135592605101821840357536715295106314322514178174093418647, 172653929034048084950674429899193619418057473778703331247504102312524336679143, 7697244446764910533439035999465870024624015165009549539237709509561276493745142032855514257884678020205979511241516314780842452478186799801136738477569952), 4029346358246114652067068739766292392118023753655925305867827481498318290795142341938345968717040279506355749099387227664100563171669643366238967415558125, 40668880821025106389898038400811298129185052032777132031473200470122753451754)

Sign: (5703657253897407292490430643453399652579595100365501174401567293134983141197306634740105018554580891346777252140803685759196850583083848267738194247925735, 44898665091535031568480658105565009165530078163431585735513813114273272997373)

Verify: True