**Харківський національний економічний університет**

**імені Семена Кузнеця**

**ЗВІТ**

**З ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9**

**за дисципліною: “Теоретичні основи криптографії”**

**Виконав: студент факультету Інформаційних технологій**

**2 курсу, спец. Кібербезпека,**

**групи 6.04.125.010.21.2**

**Бойко Вадим Віталійович**

**Перевірив:**

**Чугай Андрій Михайлович**

**ХНЕУ ім. С. Кузнеця**

**2023**

**Тема:**  Дослідження електронного цифрового підпису (ЕЦП) RSA

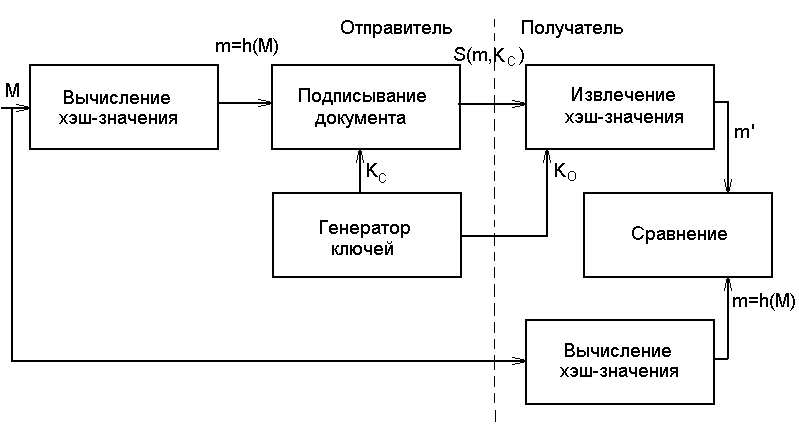
**Мета:** *Дослідити структуру алгоритму та методики практичної реалізації (ЕЦП) RSA.*

**Завдання:**

1. Тему, мету і порядок роботи.
2. Блок-схему алгоритму й програму формування ЕЦП RSA.
3. Висновки: переваги й недоліки ЕЦП RSA.

Хід роботи

Блок схема:



Код:

*import* random  
*import* math  
  
  
*def* is\_prime(*number*):  
 """Перевіряє, чи є число простим."""  
 *if number* <= 1:  
 *return False  
 for* i *in* range(2, int(math.sqrt(*number*)) + 1):  
 *if number* % i == 0:  
 *return False  
 return True  
  
  
def* generate\_prime():  
 """Генерує велике просте число."""  
 *while True*:  
 num = random.randint(100, 100000) # Задаємо діапазон для генерації великого числа  
 *if* is\_prime(num):  
 *return* num  
  
  
*def* gcd(*num1*, *num2*):  
 """Функція для знаходження найбільшого спільного дільника"""  
 *while num2*:  
 num1, num2 = *num2*, *num1* % *num2  
 return num1*p = generate\_prime()  
q = generate\_prime()  
n = p \* q  
euler = (p - 1) \* (q - 1)  
  
  
*def* mod\_inverse(*a*, *mod*):  
 """Функція для знаходження оберненого за модулем числа"""  
  
 *def* extended\_gcd(*num1*, *num2*):  
 """Функція для розширеного алгоритму Евкліда"""  
 *if num2* == 0:  
 *return num1*, 1, 0  
 *else*:  
 g, x, y = extended\_gcd(*num2*, *num1* % *num2*)  
 *return* g, y, x - (*num1* // *num2*) \* y  
  
 nod, value\_x, value\_y = extended\_gcd(*a*, *mod*)  
 *if* nod != 1:  
 *raise* ValueError("Оберненого за модулем числа не існує")  
 *return* value\_x % *mod if* value\_x > 0 *else* value\_x + *mod*# Вибір випадкового числа e, такого, що 1 < e <= eluer і gcd(e, euler) = 1  
*def* finding\_e(*euler\_fun*):  
 """Функція для знаходження чисел e"""  
 *while True*:  
 public = random.randint(2, *euler\_fun* - 1)  
 *if* gcd(public, *euler\_fun*) == 1:  
 *break  
 return* public  
  
  
# Обчислюємо значення секретного ключа d, що задовольняє умові e \* d = 1 (mod (euler))  
*def* finding\_d(*public*, *euler\_fun*):  
 secret = mod\_inverse(*public*, *euler\_fun*)  
 *return* secret  
  
  
*def* hash\_value\_m(*sms*):  
 h = random.randint(1, 1000)  
 list\_index = []  
 *for* letter *in sms*:  
 list\_index.append(ord(letter))  
 m = 0  
 *for* i *in* range(len(list\_index)):  
 m = pow(list\_index[i] + h, 2, n)  
 h = m  
 *return* m  
  
  
*def* encrypt\_m(*secret*, *hash\_m*):  
 digital\_signature = pow(*hash\_m*, *secret*, n)  
 *return* digital\_signature  
  
  
*def* decryption\_m1(*public*, *dig\_signature*):  
 m1 = pow(*dig\_signature*, *public*, n)  
 *return* m1  
  
  
print(f"{'=' \* 50}\n{' ' \* 23}DSA\n{'=' \* 50}")  
print(f"Просте число: p = {p}")  
print(f"Просте число: q = {q}\n{'-' \* 50}")  
print(f"Добуток p \* q: n = {n}")  
print(f"Функція Ейлера: f(n) = {euler}\n{'-' \* 50}")  
e = finding\_e(euler)  
print(f"Значення е = {e}")  
d = finding\_d(e, euler)  
print(f"Значення d = {d}\n{'-' \* 50}")  
message = "Cybersecurity"  
print(f"Повідомлення: {message}")  
m\_hash = hash\_value\_m(message)  
print(f"Хеш-значення m = {m\_hash}")  
dsa = encrypt\_m(d, m\_hash)  
print(f"Електроний цифровий підпис = {dsa}")  
m1\_hash = decryption\_m1(e, dsa)  
print(f"Хеш-значення m1 = {m1\_hash}\n{'-' \* 50}")  
*if* m\_hash == m1\_hash:  
 print("ПІДПИС ДОСТОВІРНИЙ!")  
*else*:  
 print("ПІДПИС НЕ ДОСТОВІРНИЙ!")

Тест:



Проаналізувавши схему RSA, можна зробити висновок, що перевагами данного алгоритму є забезпечення високої криптостійкості при певній довжині ключа та простота алгоритму. Недоліки: складність обчислення ЕЦП, так як необхідні великі обчислювальні ресурси, повільність підписання документу.