

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



Лабораторна робота №3

з дисципліни: «Технології захисту інформації»

на тему: «Захист даних з допомогою комбінованих алгоритмів шифрування та застосування електронного цифрового підпису»

Варіант №6

Виконала:

студентка групи ІТ-32

Моляща Ю.А.

Прийняв:

Табачишин Д.Р.

Львів - 2023

### Лабораторна робота №3

#### «Захист даних з допомогою комбінованих алгоритмів шифрування та застосування електронного цифрового підпису»

**Мета роботи:** вивчити принцип роботи асиметричного алгоритму шифрування на прикладі алгоритму RSA. Освоїти методику створення комбінованих алгоритмів шифрування, які поєднують переваги методів симетричної та асиметричної криптографії та навчитись застосовувати електронний цифровий підпис.

#### Завдання на лабораторну роботу.

**Завдання 1.** Створення відкритого та закритого ключа при заданих для конкретного варіанту значеннях  $p$  та  $q$  (табл. 2). В якості числа  $e$ , яке входить в склад відкритого ключа, необхідно взяти найбільше просте число, менше  $p$ .

**Завдання 2.** Використовуючи алфавіт (табл.1.), зашифруйте повідомлення з допомогою створеного відкритого ключа. Повідомлення вибирається згідно з варіантом з табл.2.

$p=23$ ,  $q=31$ , СЕЙФ02

**Завдання 3.** Дешифрування зашифрованого тексту з допомогою створеного в 1.1. закритого ключа. Шифрограма вибирається згідно з варіантом з табл. 3.

Криптограма: 468, 409, 568, 1, 286, 82, 297, 40

**Завдання 4.** Оформити алгоритм функціонування програми та здійснити його опис.

**Завдання 5.** Оформити звіт в згідно вимог, результати подати у вигляді текстів програми, відповідних пояснень та скріншотів.

#### Хід роботи.

Код програми:

```
import random
def is_prime(n):
    """Check if number is prime"""
    if n <= 1:
        return False
    elif n > 1:
        for number in range(2, n - 1):
            if n % number == 0:
                return False
        return True
def euler_function(p, q):
    return (p - 1) * (q - 1)
def gcd(a, b):
    """Greatest common divisor"""
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
```

```

def generate_public_key(p, q):
    """Generate public key (e, n)

    Approach 1:
    e is max prime number less than p (With respect to the task)
    Approach 2:
    e is a random return value (Common case)

    Args:
        p (int): First prime number
        q (int): Second prime number

    Returns:
        tuple: (e, p * q)
    """
    e = p * q
    phi = euler_function(p, q)

    # Approach 1
    while not is_prime(e):
        e -= 1

    # Approach 2
    # while True:
    #     e = random.randint(2, phi - 1)
    #     if gcd(e, phi) == 1:
    #         return (e, p * q)

    return (e, p * q)

def generate_private_key(p, q, e):
    """Generate private key (d, n)

    Iterate over k until d is integer

    Args:
        p (int): First prime number
        q (int): Second prime number
        e (int): Public key

    Returns:
        tuple: (d, p * q)
    """
    k = 1
    phi = euler_function(p, q)

    while True:
        d = (k * phi + 1) / e
        if d.is_integer():

```

```

        return (int(d), p * q)
    k += 1

def message_to_list(message):
    """Convert message to list of indices"""
    message_list = [alphabet_dict[char] for char in message if char in
alphabet_dict]
    return message_list
def list_to_message(indices_list):
    """Convert list of indices to message"""
    message = ""
    index = ''

    for item in indices_list:
        for key, value in alphabet_dict.items():
            if str(item) == value:
                index = key
        message += index

    return message
def encrypt_message(indices_list):
    """Encrypt message using public key (e, n)"""
    encrypted_message = [ int(item) ** e % n for item in indices_list ]
    return encrypted_message
def decrypt_message(indices_list):
    """Decrypt message using private key (d, n)"""
    decrypted_message = [ int(item) ** d % n % ALPHABET_LENGTH for item in
indices_list]
    return decrypted_message

# Task 1
p = 23
q = 31
e, n = generate_public_key(p, q)
d, n = generate_private_key(p, q, e)
print(f"Public key: ({e}, {n})")
print(f"Private key: ({d}, {n})")

# Task 2
# String to store ukrainian alphabet
alphabet = "АБВГГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ_0123456789"
ALPHABET_LENGTH = len(alphabet)
# Create a dictionary to map each character to its index
alphabet_dict = {char: str(i) for i, char in enumerate(alphabet, start=1)}
message = "СЕЙФ02"

indices_list = message_to_list(message)
encrypted_message = encrypt_message(indices_list)
decrypted_message = decrypt_message(encrypted_message)
decrypted_message_str = list_to_message(decrypted_message)
print(indices_list)

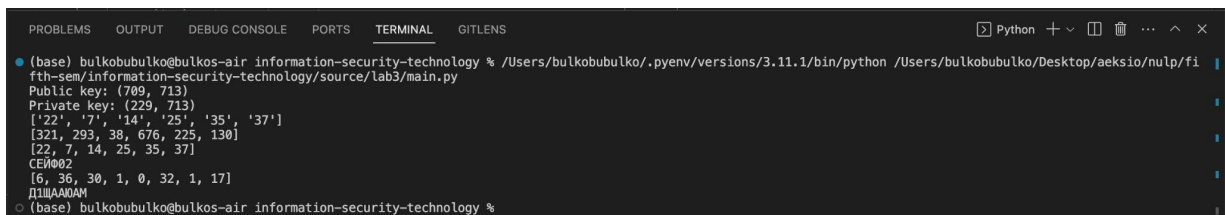
```

```

print(encrypted_message)
print(decrypted_message)
print(decrypted_message_str)
# Task 3
cryptogram = [468, 409, 568, 1, 286, 82, 297, 40]
decrypted_message = decrypt_message(cryptogram)
ddecrypted_message_str = list_to_message(decrypted_message)
print(decrypted_message)
print(ddecrypted_message_str)

```

Приклад виконання зображено на Рис. 1.



```

(base) bulkobubulko@bulkos-air information-security-technology % /Users/bulkobubulko/.pyenv/versions/3.11.1/bin/python /Users/bulkobubulko/Desktop/aeksio/nulp/fi
fth-sem/information-security-technology/source/lab3/main.py
Public key: (709, 713)
Private key: (229, 713)
['22', '7', '14', '25', '35', '37']
[321, 293, 38, 676, 225, 130]
[22, 7, 14, 25, 35, 37]
СЕЙМ02
[6, 36, 30, 1, 0, 32, 1, 17]
ДІЩААЮАМ
(base) bulkobubulko@bulkos-air information-security-technology %

```

Рис. 1. Приклад виконання

**Посилання на GitHub-репозиторій:** <https://github.com/bulkobubulko/nulp-ist>

**Висновок:** під час виконання лабораторної роботи було вивчено принцип роботи асиметричного алгоритму шифрування на прикладі алгоритму RSA. Освоєно методику створення комбінованих алгоритмів шифрування, які поєднують переваги методів симетричної та асиметричної криптографії та вивчено застосування електронний цифровий підпис.