МІНІСТЕРСТВО ОСВІТИ І НАУКИ КРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського”
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГР АФІЧНИХ МЕХАНІЗМІВ

ЛАБОР АТОРНА РОБОТА №4

“Дослідження особливостей реалізації існуючих програмних систем, які використовують криптографічні механізми захисту інформації.”

Виконали:
студенти 6-го курсу
групи: ФІ-31мн
Коробан Ольга
Каюк Ксенія
Кухар Богдан

Київ - 2023

Мета роботи: Отримання практичних навичок побудови гібридних криптосистем.

Підгрупа 2С. Бібліотека PyCrypto під Crypto++ під Android/MacOs/Ios платформу. Реалізація несуперечного цифрового підпису.

Використаємо бібліотеку **PyCryptodome** для побудови асиметричної криптосистеми

Перевіримо версію Python

PyCryptodome вже встановлено тож можна починати

Реалізація криптосистеми

План необхідних дій:

1. Генерація пари ключів(відкритий, закритий)
2. Шифрування та дешифрування
3. Цифровий підпис та його перевірка

Створимо код для генерації ключів

```python
from Crypto.PublicKey import RSA

# Генерація ключей
key = RSA.generate(2048)
private_key = key.export_key()
public_key = key.publickey().export_key()

print("Закритий ключ:")
print(private_key.decode()[:100] + "...")
print("\nВідкритий ключ:")
print(public_key.decode())
```

Протестуємо генерацію приховавши частину закритого ключа

```
❯ cd /Users/a1/Downloads

❯ python3 Lab4.py

Закритий ключ:
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAz6cog8k4WrElWhAMV0rB24wiEl2Pro3HZokILSsMoYwlEXbt
ZM0...

Відкритий ключ:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAz6cog8k4WrElWhAMV0rB
24wiEl2Pro3HZokILSsMoYwlEXbtZM00mZDV82nfTSfnsI5l5QOeuyUdyPvVoNMf
wbsQcZLyLAne7gszKBpTQyc/Rky2Adw/pEi6WI0y8Z7noHYsY5PxtCxN86vwmYjY
Pnt4WzNjMbUScQCqnLPozeYkmJl5kpLX0OH1MP4y++SgFfAFd+gA4N9fBjyD/yO5
Xg7BU9WDiiawnnP367WIHiIhP93bHbXnlltsAL95xm52d7qhcQCE3GK5et4N/yDf
5ELJxNzlIyGEGsM1WTGGnQOyPK5N1Rq2/fsJCcQ+U3PMPp206EpDbR7qrYwhPC96
owIDAQAB
-----END PUBLIC KEY-----
```

Створимо код для шифрування та дешифрування на основі створених раніше ключів

```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

# Генерація ключей
key = RSA.generate(2048)
private_key = key.export_key()
public_key = key.publickey().export_key()
"""
print("Закритий ключ:")
print(private_key.decode()[:100] + "...")
print("\nВідкритий ключ:")
print(public_key.decode())
"""

message = b"super secret information for Lab/super secret information for Lab"
print("початкове повідомлення: ", message)
# Шифрування
cipher = PKCS1_OAEP.new(key.publickey())
encrypted_message = cipher.encrypt(message)
print("\nЗашифроване повідомлення:")
print(encrypted_message)

# Дешифрування
decipher = PKCS1_OAEP.new(key)
decrypted_message = decipher.decrypt(encrypted_message)
print("\nРазшифроване повідомлення:")
print(decrypted_message.decode())
```

Протестуємо шифрування та дешифрування

```
❯ python3 Lab4.py

початкове повыдомлення:  b'super secret information for Lab/super secret information for Lab'

Зашифроване повідомлення:
b's\xea \xb3d\xcc\xf6\xae\xf4\x0f\xa3\x04\xea\xe8\xdb\x93\xadH&T\x8b\xcca\xee&1\xc3\x85\xb7$\x17\x15\xbd\x1fo@?\xfb\
xa4\xe71gP\xa6\xdd\xe1/1\x81*\x8a\xf7X\xa9\x92\xa6H-.\xec\x8f\xf8\x00\'\xe2\x8b1\xc8\xc6<4)Cy\x80\xe5\xe7\x9bn0\xe3#
\x0bJ\\\xbdiUb\'\xb7!\xf77\xd6\x9d\x07\x0e\x97\x12\x92%\x8d\xa6|\xd8\t\xe86#j\xdd\xd0\x9cN\xfb)\x9e+\x94y\x04\xc4\x1
cD\x90\xcc\x06\x90r;0n\x06l\x19\x96>\x10\x0cZ\xca\xca\xc3\xbb\x0f\xf9\xf5a9\xc7\xb5S\xa2O"\xba\xf7Egh\xd6*\xe6.\xc8+
P\x1aQ\xee\x06\x1b\x8c\x0c\xe0\tT\xb6\xee{\x15\t\xeb\xd6{\xd2g\xd0/\xac\xeb\x0c\x9c\x14\xa5(X\xd76\r\xfd]\xfd\xe1p/\
x8e<?\xd1\xdevhXd0\x91aI\x03\xba\x0ff\xbc\xcd\xa6:N\xc9\xc4P0\xf98\x11\xf2\x9b\x9a\x9f\xe6t\xc2\'\xb5\xe8QG\xc6\xc1K
\xbb?\x99\xf8\x12'

Разшифроване повідомлення:
super secret information for Lab/super secret information for Lab
  ~/Downloads                                                                                                19:52:40
```

Створимо код для цифрового підпису та його перевірки

```python

# Генерація ключей
key = RSA.generate(2048)
private_key = key.export_key()
public_key = key.publickey().export_key()
"""
print("Закритий ключ:")
print(private_key.decode()[:100] + "...")
print("\nВідкритий ключ:")
print(public_key.decode())
"""

message = b"super secret information for Lab/super secret information for Lab"
print("початкове повідомлення: ", message)
# Шифрування
cipher = PKCS1_OAEP.new(key.publickey())
encrypted_message = cipher.encrypt(message)
print("\nЗашифроване повідомлення:")
print(encrypted_message)

# Дешифрування
decipher = PKCS1_OAEP.new(key)
decrypted_message = decipher.decrypt(encrypted_message)
print("\nРазшифроване повідомлення:")
print(decrypted_message.decode())

# Створення цифрового підпису
hash_obj = SHA256.new(message)
signature = pkcs1_15.new(key).sign(hash_obj)
print("\nЦифровий підпис:")
print(signature.hex())

# Перевірка цифрового підпису
try:
    pkcs1_15.new(key.publickey()).verify(hash_obj, signature)
    print("Підпис дійсний!")
except (ValueError, TypeError):
    print("Підпис не дійсний!")
```

Протестуємо цифровий підпис

```
❯ python3 Lab4.py

початкове повідомлення:  b'super secret information for Lab/super secret information for Lab'

Зашифроване повідомлення:
b"PD\xb9\xa1\xec+y\xd1,o\xc8!\x93\xcc\x0b\xec\xe5\xc9TJ\xebF\x07nT\x072\xa0\x1c\x89\xe2\x12}\x8f\xc1~H\xac\xff~\xb6\
x14\xfe\x88\xd8h\x19\xecH\x17J\x90A\xec\xe7\xcc\xf1\x85\xbf\x8f,Z\x96\xff\x98\xf0;\xe7\xbc\xb4b\xc4\\\xaf\xc8 \x1fl\
x18\x1f\xe5\xd4AE\xdeJ!\x11!?\xa4\x9c\x89\xc4g\xe6\xd2\xe9:\x0e\xef\xeeO\x86Y>\x16\x98\xf7\xd6\x11\xf4\x11\x9cM\xcb\
xf7\xec\x08;@k'\xe3l)X\xe4\xc6\x18\xbfe\x05\xd6NQ\xb3^\xe8\xe3\xb8\xed\x100\xda\xa38\xc7-\x05\xd5\xbc\xce\xa2(e\x9c\
x936\x05\x9d\x857\xb2n\xb1a\x9d\xb1H\xda\x8cB\xcbX:\x97\x9e\x0c\xec\x8f\xc3^\x99\x01vy\xfa\xc62\x97p,\x19\xbfb78\xa4
\xe3\xb4U:)\xe9fF\xed\xec\xe4\x80;\xfas\x15h\xc3\xfe\xdc\x8b\xd7\xf3}\x80\x02Z\xe2n\xfd\xd0\xc1K\x04\xb0\x02a/\\\xa4
4j\xbf^\xaf\x8c'>\x9c/e\x1f_\xdd\r5\xb2"

Разшифроване повідомлення:
super secret information for Lab/super secret information for Lab

Цифровий підпис:
a2320dc3a32bd2ea1533390d57d036b8647425ebbfa7f62fcd170004b8fccb43249dce6250645db5ed9b022c5cda60cdfc688e15e50afc409c94
c632298676ae2cf712b379c36f3f462d0ddb8edd8acb65b7ccf97d4cd5f91551852c7dddb649394343adb247408d2a506ad6e9d06b3c1a750736
920340d9e460347cb75df9d861d0ea4971df0f1aa9cd859f31f128e3c7537586b3d7169e7892558ef69b7ae30411a27074386ea2fb001b2befbd
b8f0b2e69376cc4e389bb4ec2d5ac5f5469af5b7db4503451dbf4ed6f473f6d5a9e0e9bd43c04cbc2d9648e7ffa48869f037cb972b272ff51afb
21ec7343d44cb0cd2a0aa34b6d9d068ef1e139fac590c1ce
Підпис дійсний!
```

Атакую

Розробити код для симуляції Padding Oracle атаки

```python
# Симуляція атаки Padding Oracle
def padding_oracle_simulation(key, encrypted_message):
    print("\n[PADDING ORACLE ATAKA]")

    try:
        modified_message = bytearray(encrypted_message)

        # Навмисне пошкодження частини зашифрованого повідомлення
        index = random.randint(0, len(modified_message) - 1)
        modified_message[index] ^= 0xFF

        decipher = PKCS1_OAEP.new(key)
        decipher.decrypt(bytes(modified_message))
    except Exception as e:
        print("Результат атаки:", e)
        print("Система захисту спрацювала!")
```

Перевіримо чи спрацює атака

```
[PADDING ORACLE ATAKA]
Результат атаки: name 'random' is not defined
Система захисту спрацювала!
 ~/Downloads                                                                    20:11:03
```

Атака не принесла результатів

Атака брут форс

```python
# Симуляція атаки brute force
def brute_force_simulation(message, key):
    print("\n[СИМУЛЯЦІЯ АТАКИ ПІДБОРУ]")

    for _ in range(5):
        test_key = RSA.generate(2048)

        try:
            cipher = PKCS1_OAEP.new(test_key.publickey())
            encrypted_test = cipher.encrypt(message)

            decipher = PKCS1_OAEP.new(test_key)
            decrypted_test = decipher.decrypt(encrypted_test)

            print("Увага: Знайдено збіг ключа!")
        except Exception:
            print("Спроба підбору не вдалася")
```

Перевіримо результат атаки

```
❯ python3 Lab4.py

початкове повідомлення:  b'super secret information for Lab/super secret information for Lab'

Зашифроване повідомлення:
b'\xc7\x9c\x8cu]\xcb\x0b\x90\x11\x1d,r\xab\xe5\xed\xdc\xe9"\x110\x97\x9e\xfb\x1b\xff\xfa\xf2\xdfs=i\xdb\xe5r\xbe|J"\
xe5B\xd8\xfa\xcd\x11\xb2\xbfN\xc7\xca\x9f\x16\x95\xa2|):\x14\x0b\x9b\xf1\xb6\x86\xbbMo\xe2T9%^\x9c\x03\xed\x86c\xbdi
\xf1\x0e\xd1\xb0\xfa\x95\xb1\xbad\xd1\xafj(\xdab\xed$\xe1\xb1\xa3\xd1\xd2\x1c{\x9b1\xca\x15_\x08\xa8c\xc3\x186c4\ny\
xa9a\xb5\x1cwK\xb4\xaa\x1b\xa4\x9f\x1et#\xf0R\x15\xe7D\x08\\_\x12@Q\xe9pO\x89\xff#_\x0b\x08\xdaX\xfa\xf5a\xf2V\xe9sl
#l\xa3\x96\x1bE\x0c\xf44A\xa2\x7f,\xa9K\xe0\xf4W!\xc0\x98h\xc7,E\x06\xb9\xb5\x0f\xd4\xb9\xc1:eF\x8fa\xc4\xf5\xa6\xd8
\xac9\x7fS&4\xf8\xdf\x97\xe5\xc6\xd7b\xdf\xaa\xca_6\x905\xf5B\xb3K\x89\xbc\x18\xfe\x0e\xec\xc06\xf7=\xe8\x8do\x9eYW\
x92\xb5\xe6\x15\x9c\x0e]\x06\xac\xf0\xb5\x80\x87u\xa5'

Разшифроване повідомлення:
super secret information for Lab/super secret information for Lab

Цифровий підпис:
aa76da75ea478e2539a92f5cc23ba1dff32860c6857315d67cd85fcb2f35a2197bd662c4e42ec54dbec712a8eb5fac37863c7e55d08336a2a2e3
8f2c730c29af09a449d76a6752ebbae68a239eb55065983f8e16bfe34df375ddbc274f272d3ad2ea2bd3674b2c35179a7bbfacc39046629bf309
dfbb726aa3fd69c7df8d2870086187d2e2360b0d6f5b44c9507716bca36c14b71098a0e5ca53c42db238371afbdabbd3426b58831b0fb2c856a3
9e0d30756ad89ae5f3c82606e197b981865465dce41ea722e32e4755895c9ec6d209c2eed0bfba3876dac837127967aeec1bb761cf3c47cd9809
1a1f7746962fed8e7078027925d8237d3e95aa915204dd05
Підпис дійсний!

[СИМУЛЯЦІЯ АТАКИ ПІДБОРУ]
Спроба підбору не вдалася
```

Тепер розробимо сценарії атаки на цифровий підпис

Подивимось атаки на цілісність повідомлення та підробки підпису

Розробити код

```python
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256
import random
import math
from decimal import Decimal, getcontext

def digital_signature_attacks(key, message):
    print("\n[АТАКИ НА ЦИФРОВИЙ ПІДПИС]")

    # Створення підпису
    hash_obj = SHA256.new(message)
    original_signature = pkcs1_15.new(key).sign(hash_obj)

    # Атака на цілісність повідомлення
    def message_tampering_attack():
        print("\n1. Атака на цілісність повідомлення:")
        tampered_message = message + b" additional information for lab"

        try:
            # Перевірка, чи підпис валідний для зміненого повідомлення
            hash_tampered = SHA256.new(tampered_message)
            pkcs1_15.new(key.publickey()).verify(hash_tampered, original_signature)
            print("ВРАЗЛИВІСТЬ: Підпис валідний для зміненого повідомлення!")
        except (ValueError, TypeError):
            print("Система захищена від атаки підміни повідомлення")

    # Демонстрація складності підробки підпису
    def signature_forgery_attack():
        print("\n2. Аналіз складності підробки підпису:")

        key_size = key.size_in_bits()
        print(f"Розмір ключа: {key_size} біт")

        #кількість комбінацій
        possible_combinations = 2 ** key_size
        print(f"Теоретична кількість можливих комбінацій: {possible_combinations}")

        # Оцінка часу підбору
        getcontext().prec = 500
        possible_combinations_decimal = Decimal(possible_combinations)
        estimated_time_seconds = possible_combinations_decimal / Decimal(10**12)
        if estimated_time_seconds > Decimal(10**9):
            estimated_time_years = estimated_time_seconds.ln() / Decimal(60 * 60 * 24 * 365).ln()
            print(f"Приблизний час підробки підпису: {estimated_time_years:.1f} ")
        else:
            years = estimated_time_seconds / (365 * 24 * 60 * 60)
            print(f"Приблизний час підробки підпису: {years:.1f} років")

        print("Висновок: Пряма підробка підпису практично неможлива")
    # Виклик функцій атак
    message_tampering_attack()
    signature_forgery_attack()

# Генерація ключа
key = RSA.generate(2048)
message = b"Info for lab"

```

Результат роботи

```
[АТАКИ НА ЦИФРОВИЙ ПІДПИС]

1. Атака на цілісність повідомлення:
Система захищена від атаки підміни повідомлення

2. Аналіз складності підробки підпису:
Розмір ключа: 2048 біт
Теоретична кількість можливих комбінацій: 323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459972683529986382155251663894373355436021354332296046453184786049521481935558536110595962306560
Приблизний час підробки підпису: 80.6 років
Висновок: Пряма підробка підпису практично неможлива
 ~/Downloads                                                                                                    20:50:00
```

Висновок

У результаті виконання лабораторної роботи було успішно реалізовано асиметричну криптосистему за допомогою бібліотеки **PyCryptodome** на платформі **MacOS**. Описана система включала генерацію пари RSA-ключів (приватного та публічного), шифрування і розшифрування повідомлення, а також створення і перевірку цифрового підпису.

Крім того, було проведено дослідження стійкості системи до атак, спрямованих на експлуатацію недоліків механізмів захисту операційної системи.

Проведені експерименти з цифровим підписом показали, що система RSA забезпечує надійний захист від підробки підписів при коректному зберіганні і використанні ключів. Однак, для подальшого підвищення стійкості до атак важливо використовувати додаткові механізми захисту, такі як шифрування ключів і багатофакторна аутентифікація.

У підсумку, лабораторна робота показала ефективність асиметричних криптосистем для захисту даних