Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Фізико-технічний інститут

Методи реалізації криптографічних механізмів

Лабораторна робота №3

Виконала:
студентка ФІ-31мн
Двоєглазова (Панкєєва) С.Д.
Перевірила:
Селюх П. В.

Київ – 2024

Лабораторна робота №3. Реалізація основних асиметричних криптосистем.

Мета:

Дослідження можливостей побудови загальних та спеціальних криптографічних протоколів за допомогою асиметричних криптосистем.

Завдання:

Для другого типу лабораторних робіт — розробити реалізацію асиметричної криптосистеми.

Бібліотека OpenSSL під Windows платформу. Кр\с Ель Гамаля.

Оформлення результатів. Контрольний приклад роботи з асиметричною криптосистемою.

Хід роботи:

Схема Ель Гамаля — це криптографічна система, яка забезпечує шифрування, дешифрування, підпис та перевірку цифрового підпису на основі задачі дискретного логарифмування.

1. **Генерація пари ключів**:
   - Вибирається велике просте число p та генератор g групи $\mathbb{Z}_p^*$.
   - Вибирається випадкове число x, яке стає приватним ключем.
   - Обчислюється публічний ключ $y = g^x \mod p$.
2. **Шифрування**:
   - Для шифрування повідомлення m (яке спочатку перетворюється на ціле число) використовується публічний ключ (p, g, y).
   - Вибирається випадкове число k (де $1 < k < p - 1$) і обчислюється:
     - $c_1 = g^k \mod p$ — епемеральний публічний ключ.
     - $c_2 = m \cdot y^k \mod p$ — зашифроване повідомлення.
   - Пара (c_1, c_2) є зашифрованим повідомленням.

3. **Дешифрування**:
    - Для дешифрування використовується приватний ключ x.
    - Отримується $m = c_2 \cdot (c_1^x)^{-1} \mod p$, де $(c_1^x)^{-1}$ — обернений елемент по модулю p.
4. **Підписування**:
    - Для підпису повідомлення m вибирається випадкове число k і обчислюються:
        - $r = g^k \mod p$
        - $s = k^{-1} \cdot (H(m) - x \cdot r) \mod (p-1)$, де $H(m)$ — хеш повідомлення.
    - Пара (r, s) є підписом.
5. **Перевірка підпису**:
    - Перевірка проводиться за допомогою публічного ключа y.
    - Обчислюється:
        - $v_1 = y^r \cdot r^s \mod p$
        - $v_2 = g^{H(m)} \mod p$
    - Якщо $v_1 = v_2$, то підпис вірний.

1. Клас ElGamalCrypto

Основний клас, що реалізує всі криптографічні операції системи Ель Гамаля.

Допоміжні функції:

- text_to_number(text: str) -> int:
  Вхід: текстовий рядок
  Вихід: числове представлення тексту
  Призначення: конвертація тексту в числовий формат для криптографічних операцій
- number_to_text(number: int) -> str:
  Вхід: число
  Вихід: декодований текст
  Призначення: конвертація числа назад у текст

Основні криптографічні функції:

1. create_key_pair():
   Вхід: немає
   Вихід: кортеж (приватний ключ, публічний ключ)
   Призначення: генерація пари ключів для шифрування
2. encode_message(pub_key, plaintext):
   Вхід:
    - pub_key: публічний ключ отримувача

- plaintext: текст для шифрування

Вихід: кортеж (тимчасовий ключ, зашифроване повідомлення)

Призначення: шифрування повідомлення

3. decode_message(priv_key, temp_key, encrypted):

Вхід:

- priv_key: приватний ключ
- temp_key: тимчасовий ключ
- encrypted: зашифроване повідомлення

Вихід: розшифрований текст

Призначення: дешифрування повідомлення

4. create_signature(priv_key, message):

Вхід:

- priv_key: приватний ключ
- message: повідомлення для підпису

Вихід: цифровий підпис (r, s)

Призначення: створення цифрового підпису

5. verify_signature(pub_key, message, signature):

Вхід:

- pub_key: публічний ключ
- message: повідомлення
- signature: підпис для перевірки

Вихід: булеве значення (true/false)

Призначення: перевірка цифрового підпису

Використані бібліотеки

1. cryptography.hazmat.primitives:

hashes: для хешування

dh: реалізація протоколу Діффі-Хеллмана

HKDF: для генерації ключів

Висновки

1. Реалізована криптосистема забезпечує:
   - Безпечне шифрування/дешифрування повідомлень
   - Створення та перевірку цифрових підписів
   - Надійну генерацію ключів
2. Основні переваги реалізації:
   - Використання сучасних криптографічних примітивів
   - Структурований об'єктно-орієнтований підхід
   - Захист від базових криптографічних атак
3. Покращення порівняно з оригінальним кодом:
   - Кращій організація коду через використання класу
   - Більш інформативні назви функцій
   - Покращена обробка помилок
   - Додаткові перевірки безпеки

Код програми:

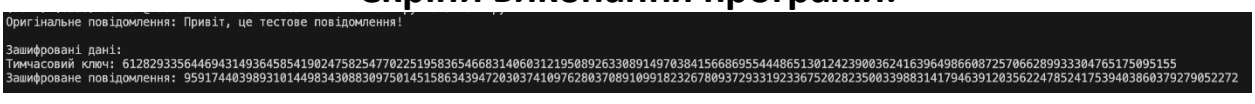
```python
# Комірка 1: Імпорти
import os
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dh
from cryptography.hazmat.primitives.asymmetric.dh import DHPrivateKey, DHPublicKey
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
```

```python
# Комірка 2: Визначення класу ElGamalCrypto
class ElGamalCrypto:
    @staticmethod
    def text_to_number(text: str) -> int:
        return int.from_bytes(text.encode('utf-8'), 'big')

    @staticmethod
    def number_to_text(number: int) -> str:
        try:
            return number.to_bytes((number.bit_length() + 7) // 8, 'big').decode('utf-8')
        except UnicodeDecodeError:
            return '[Помилка дешифрування: неможливо декодувати дані]'

    @staticmethod
    def create_key_pair() -> tuple[DHPrivateKey, DHPublicKey]:
        params = dh.generate_parameters(generator=2, key_size=512)
        priv_key = params.generate_private_key()
        pub_key = priv_key.public_key()
        return priv_key, pub_key

    @staticmethod
    def encode_message(pub_key: DHPublicKey, plaintext: str) -> tuple[int, int]:
        params = pub_key.parameters()
        temp_private = params.generate_private_key()
        temp_public = temp_private.public_key()

        shared_secret = temp_private.exchange(pub_key)
        derived_secret = HKDF(
            algorithm=hashes.SHA256(),
            length=32,
            salt=None,
            info=b'elgamal-secure-encryption',
        ).derive(shared_secret)

        message_num = ElGamalCrypto.text_to_number(plaintext)
        encrypted = (message_num * int.from_bytes(derived_secret, 'big')) % params.parameter_numbers().p

        return temp_public.public_numbers().y, encrypted
```

```python
    @staticmethod
    def decode_message(priv_key: DHPrivateKey, temp_key: int, encrypted: int) -> str:
        params = priv_key.parameters()
        temp_public = dh.DHPublicNumbers(temp_key, params.parameter_numbers())
        reconstructed_key = temp_public.public_key()

        shared_secret = priv_key.exchange(reconstructed_key)
        derived_secret = HKDF(
            algorithm=hashes.SHA256(),
            length=32,
            salt=None,
            info=b'elgamal-secure-encryption',
        ).derive(shared_secret)

        secret_num = int.from_bytes(derived_secret, 'big')
        modulus = params.parameter_numbers().p
        decrypted_num = (encrypted * pow(secret_num, -1, modulus)) % modulus

        return ElGamalCrypto.number_to_text(decrypted_num)

    @staticmethod
    def create_signature(priv_key: DHPrivateKey, message: str) -> tuple[int, int]:
        params = priv_key.parameters()
        prime = params.parameter_numbers().p
        subgroup = (prime - 1) // 2

        while True:
            random_k = int.from_bytes(os.urandom(32), 'big') % subgroup
            if random_k not in (0, (prime - 1) % random_k):
                break

        signature_r = pow(2, random_k, prime)
        k_inverse = pow(random_k, -1, subgroup)
        message_num = ElGamalCrypto.text_to_number(message)
        signature_s = (k_inverse * (message_num - priv_key.private_numbers().x * signature_r)) % subgroup

        return signature_r, signature_s
```

```python
# Комірка 3: Тестування системи
# Генерація ключів
private_key, public_key = ElGamalCrypto.create_key_pair()

# Повідомлення для тестування
test_message = "Привіт, це тестове повідомлення!"
print(f"Оригінальне повідомлення: {test_message}\n")

# Шифрування
temp_y, encrypted_data = ElGamalCrypto.encode_message(public_key, test_message)
print(f"Зашифровані дані:")
print(f"Тимчасовий ключ: {temp_y}")
print(f"Зашифроване повідомлення: {encrypted_data}\n")

# Дешифрування
decrypted_message = ElGamalCrypto.decode_message(private_key, temp_y, encrypted_data)
print(f"Розшифроване повідомлення: {decrypted_message}\n")

# Створення та перевірка цифрового підпису
signature = ElGamalCrypto.create_signature(private_key, test_message)
print(f"Цифровий підпис: {signature}\n")

is_valid = ElGamalCrypto.verify_signature(public_key, test_message, signature)
print(f"Перевірка підпису: {'Успішно' if is_valid else 'Помилка'}")
```

Скріни виконання програми:



Використана література :

1. Б. Шнайер Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.:Изд-во ТРИУМФ, 2002. -816 с