МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра Інформаційної безпеки

Лабораторна робота 1

з дисципліни
МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ

Виконав студент гр.
ФІ-41мн Поліщук В.О.

Перевірила:
Асистент:
Байденко П.В.

Київ 2025

Підгрупа 2А. Порівняння бібліотек OpenSSL, Crypto++, CryptoLib, PyCrypto для розробки гібрідної криптосистеми під Windows платформу.

Мета роботи

Вибір базових бібліотек для реалізації гібридної криптосистеми (RSA для обміну ключами + AES для шифрування даних) під Windows платформу.

Завдання на лабораторну роботу

Порівняння бібліотек OpenSSL, Crypto++, CryptoLib, PyCrypto для розробки гібридної криптосистеми під Windows. Оформлення: опис функцій, алгоритмів, вхід/вихід, кодів повернення, контрольні приклади, обґрунтування вибору.

Необхідні теоретичні відомості

Гібридна криптосистема поєднує асиметричне (RSA) і симетричне (AES) шифрування для ефективності та безпеки. Джерела: Кнут "Мистецтво програмування" (том 2, алгоритми для великих чисел), Задірака "Комп'ютерна арифметика".

- **RSA**: Асиметричне шифрування. Алгоритм: генерація ключів (p, q — прості, n = p*q, φ(n) = (p-1)(q-1), e — відкритий експонент, d = e^{-1} mod φ(n)). Шифрування: c = m^e mod n. Складність: O((log n)^2) для модульної експоненціації.
- **AES-GCM**: Симетричне шифрування з аутентифікацією. Алгоритм: блочне шифрування (128-біт блоки), GCM додає тег для перевірки цілісності. Складність: O(len(data)) для шифрування.
- Бібліотеки: OpenSSL (C, нативна), Crypto++ (C++, без простих Python bindings), CryptoLib (нестандартна, замінена на cryptography), PyCrypto (застаріла, замінена PyCryptodome).

Вхідні дані: plaintext (bytes), ключі. Вихідні: ciphertext, тег, nonce. Коди повернення: в Python — винятки (ValueError для невалідних даних, DecryptionError для помилки верифікації).

Хід роботи

Досліджено бібліотеки для Windows: PyCryptodome (як наступник PyCrypto) і cryptography (bindings до OpenSSL). Crypto++ виключено через відсутність простих Python bindings; CryptoLib замінено на cryptography як стандартну.

Реалізовано гібридне шифрування в Jupyter Notebook. Тестовий файл: ~200 КБ унікального тексту (українською).

Для реалізації гібридної криптосистеми виконано наступну послідовність дій:

1. **Підготовка середовища** Встановлено необхідні бібліотеки за допомогою pip: pycryptodome, cryptography та pyopenssl. Перевірено успішність імпортування модулів (from Crypto.PublicKey import RSA, import cryptography, import OpenSSL).
2. **Завантаження тестових даних** Завантажено текстовий файл розміром приблизно 200 КБ з унікальним українським текстом у бінарному режимі (open(..., 'rb')). Декодовано вміст у UTF-8 для перевірки коректності читання та виведено перші та останні 200 символів для візуального контролю.
3. **Реалізація гібридного шифрування для PyCryptodome** Написано функцію hybrid_encrypt_pycryptodome(data: bytes), яка:
    - генерує RSA-ключ 2048 біт,
    - створює випадковий сесійний ключ AES-256,
    - шифрує сесійний ключ за допомогою RSA-OAEP,
    - шифрує дані за допомогою AES-GCM з генерацією nonce та тегу аутентифікації. Вихід: словник з зашифрованим сесійним ключем, nonce, тегом, ciphertext (все у base64) та приватним ключем.

Написано функцію hybrid_decrypt_pycryptodome(enc_data: dict), яка виконує зворотні операції та повертає оригінальні байти даних з перевіркою тегу.

4. **Реалізація гібридного шифрування для cryptography (OpenSSL)** Написано функцію hybrid_encrypt_cryptography(data: bytes), яка виконує аналогічні дії з використанням primitives з cryptography: генерація RSA-ключа, шифрування сесійного ключа RSA-OAEP (SHA-256), шифрування даних AES-GCM. Вихід: аналогічний словник у base64.

Написано функцію hybrid_decrypt_cryptography(enc_data: dict) для дешифрування та верифікації.

5. **Перевірка коректності роботи** Виконано шифрування та дешифрування тестового файлу для обох бібліотек. Виведено частини зашифрованих даних (enc_session_key, nonce, ciphertext у base64). Порівняно дешифрований результат з оригіналом побайтово — отримано повну відповідність для обох реалізацій.
6. **Оцінка ефективності** Написано функцію benchmark для вимірювання середнього часу виконання повного циклу (шифрування + дешифрування) за 10 повторів. Проведено тести на даних розміром ~200 КБ. Отримані результати:
    - PyCryptodome: приблизно 0.51 с
    - cryptography (OpenSSL bindings): приблизно 0.11 с

Додатково розраховано співвідношення швидкодії (cryptography швидше в 4–5 разів).

Бібліотека	Час на цикл (с)	Пам'ять
PyCryptodome	0.6955	1mb
cryptography (OpenSSL)	0.1198	1mb