НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Криптографія

КОМП’ЮТЕРНИЙ ПРАКТИКУМ 4

«Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем»

ФБ-32 Дорошенко Ілля

Мета: Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Порядок і рекомендації щодо виконання роботи

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq ≤ p1q1 ; p і q – прості числа для побудови ключів абонента A, 1 p і q1 – абонента B.

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p,q) та відкритий ключ (n,e) . За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e,n) , (,) 1 n1 e та секретні d і d1 .

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання. За допомогою датчика випадкових чисел вибрати відкрите повідомлення M і знайти криптограму для абонентів A и B, перевірити правильність розшифрування. Скласти для A і B повідомлення з цифровим підписом і перевірити його.

5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа $0 < k < n$.

Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція Encrypt(), яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Відповідно, програмний код повинен містити сім високорівневих процедур: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey().

Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою http://asymcryptwebservice.appspot.com/?section=rsa

Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

Хід роботи:

Завдання 1:

Схема Горнера (Modular Exponentiation): Функція для швидкого піднесення до степеня за модулем $x^a \pmod m$. Ця функція є необхідною основою для всіх імовірнісних тестів простоти.

Пробні Ділення (Trial Division): Функція для швидкої перевірки числа на подільність малими простими числами (наприклад, до 1000). Це дозволить швидко відкинути більшість складених чисел.

Тест Міллера-Рабіна (Miller-Rabin Primality Test): Імовірнісний тест Міллера-Рабіна, який базується на властивостях сильних псевдопростих чисел.

- Розкласти $p - 1 = d * 2^s$, де d непарне.
- Вибрати випадкову основу x.
- Перевірити умови сильного псевдопростого числа: $x^d \equiv 1 \pmod p$ або $x^{d*2^{\wedge}r} \equiv -1 \pmod p$ для деякого $0 \leq r < s$.
- Повторити тест k разів для зменшення ймовірності помилки до 4^{-k}.

Функція Генерації Простого Числа:

- Генерувати випадкове непарне число x у заданому інтервалі n_0, n_1 або заданої довжини.
- Починаючи з x (або найменшого непарного $\geq x$), перевіряти послідовні непарні числа p: $p = x, x + 2, x + 4, \ldots$.
- Для кожного p:
    - Виконати пробні ділення. Якщо складене, перейти до наступного непарного числа.
    - Виконати тест Міллера-Рабіна k разів (наприклад, k=40).
- Перше число, яке пройде обидва тести, є шуканим простим числом p.

Завдання 2:

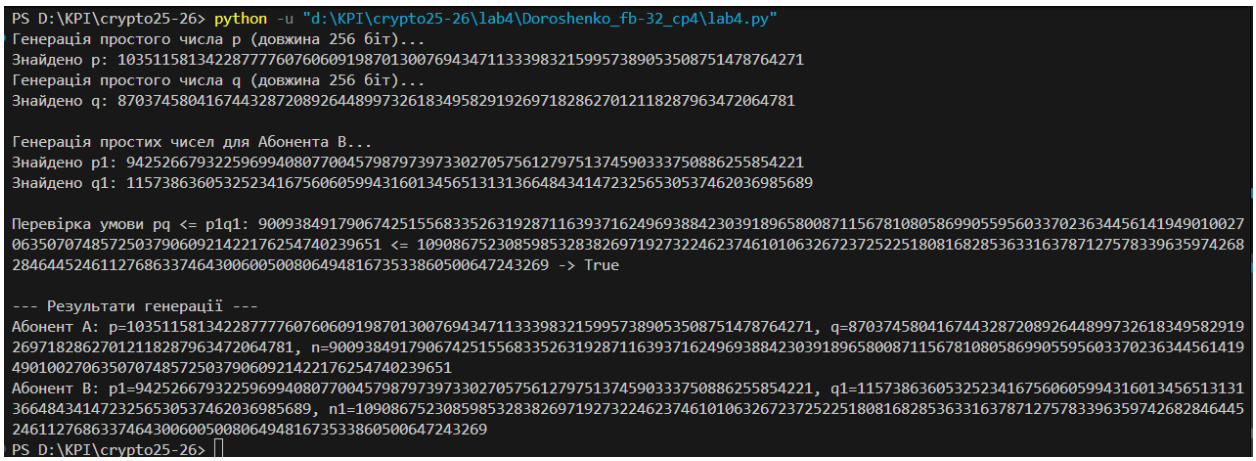
Вибір Довжини: Визначити необхідну бітову довжину для p, q, p_1, q_1. Оскільки модуль $n = pq$ має бути довжиною щонайменше $2 * 256 = 512$ біт, кожне з p, q повинно мати довжину близько 256 біт.

Генерація p та q (Абонент A): Використовуючи функцію find_prime з Завдання 1, згенерувати два випадкові прості числа p та q довжиною, наприклад, 256 біт.

Генерація p_1 та q_1 (Абонент B):

- Згенерувати p_1 та q_1 довжиною, наприклад, 256 біт.
- Перевірити умову: $pq \leq p_1q_1$.
- Якщо умова не виконується, повторно згенерувати p_1 та q_1 або збільшити їхню бітову довжину (наприклад, до 257 біт), щоб забезпечити $n_1 \geq n$. (Умова $n_1 \geq n$ (де $n = pq$ і $n_1 = p_1q_1$) необхідна для протоколу розсилання ключів.

Результат виконання коду:



Завдання 3:

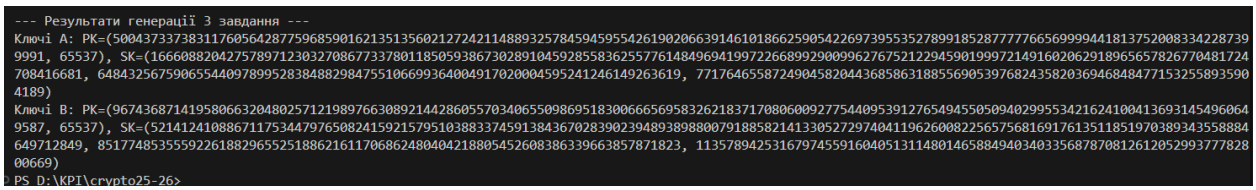
Обчислення n та φ(n): Обчислити модуль n = pq та функцію Ейлера φ(n) = (p -1)(q - 1).

Вибір e: Обрати відкриту експоненту e. Рекомендовано використовувати $e=2^{16}+1$.

Перевірка gcd(e, φ(n)): Перевірити, чи e і φ(n) є взаємно простими: gcd(e, φ(n))=1. Якщо ні, обрати інше e (або використовувати фіксоване e та повторно генерувати p, q).

Знаходження d: Знайти секретну експоненту d як обернений елемент до e за модулем φ(n). Для цього використовується Розширений алгоритм Евкліда.

Результат виконання коду:



Завдання 4:

Шифрування (Encrypt): $C = M^e \pmod n$.

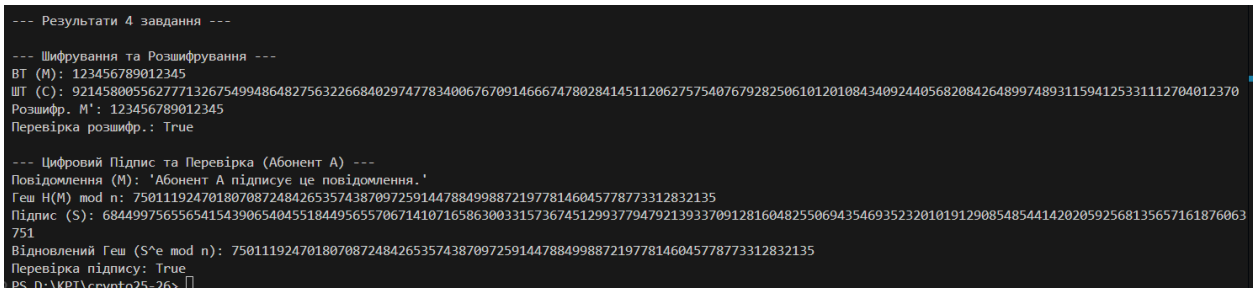
Розшифрування (Decrypt): $M = C^d \pmod n$.

Створення Підпису (Sign): $S = M^d \pmod n$.

Перевірка Підпису (Verify): Перевірити, чи $M = S^e \pmod n$.

Для практичної реалізації підпису повідомлення M спочатку піддається стисненню за допомогою геш-функції H(M)=m. У цілях цього практикуму ми будемо використовувати просту вбудовану геш-функцію Python (наприклад, hashlib.sha256) для отримання m.

Результат виконання коду:



Завдання 5:

Ролі та Ключі

- **Абонент А (Відправник):** має ключі (e, n) та d (власні), а також (e_1, n_1) (відкритий ключ В).
- **Абонент В (Отримувач):** має ключі (e_1, n_1) та d_1 (власні), а також (e, n) (відкритий ключ А).
- **Умова:** $n_1 \geq n$.

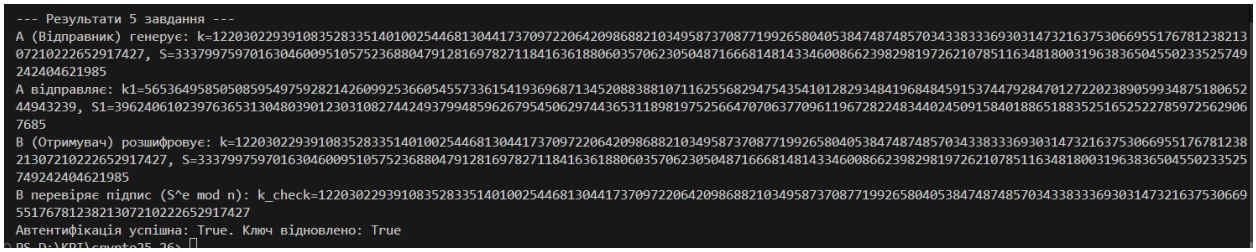
Відправник (SendKey):

- Обчислює підпис ключа k: $S = k^d \pmod n$.
- Шифрує ключ k: $k_1 = k^e{}_1 \pmod{n_1}$.
- Шифрує підпис S: $S_1 = S^e{}_1 \pmod{n_1}$.
- Відправляє (k_1, S_1).

Отримувач (ReceiveKey):

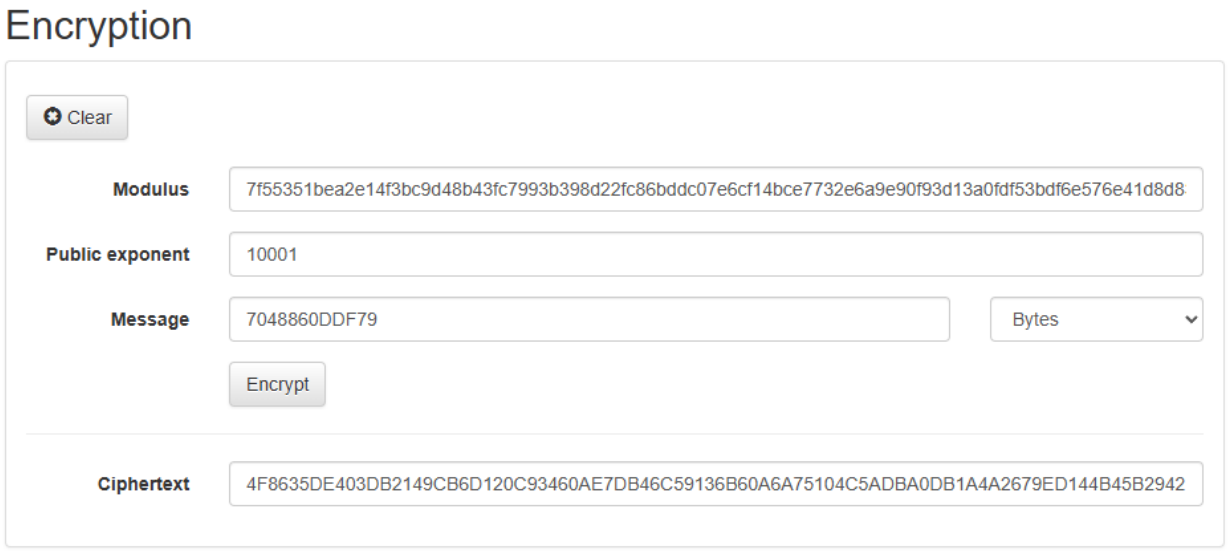
- Розшифровує ключ: $k = k_1{}^d{}_1 \pmod{n_1}$.
- Розшифровує підпис: $S = S_1{}^d{}_1 \pmod{n_1}$.
- Перевіряє підпис А: $k_{check} = S^e \pmod n$.
- Автентифікація: Якщо $k_{check} = k$, підпис вірний, відправник автентифікований.

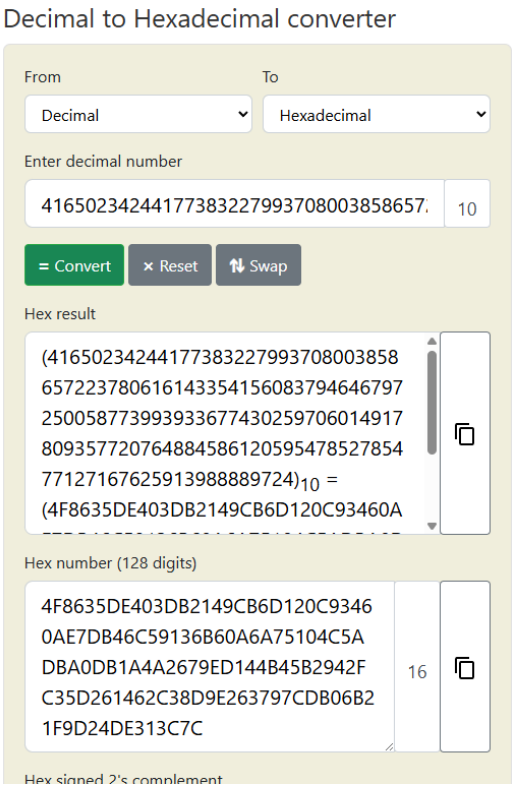
Результат виконання коду:

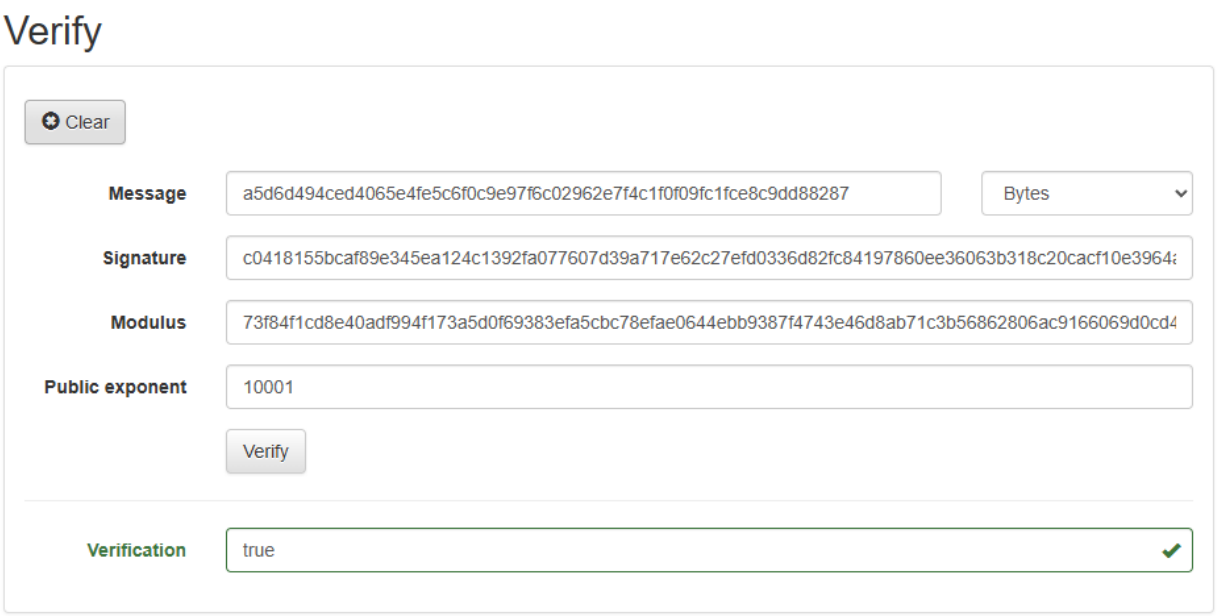


Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою http://asymcryptwebservice.appspot.com/?section=rsa

```
--- Шифрування та Розшифрування ---
ВТ (M): 123456789012345
ШТ (C): 4165023424417738322799370800385865722378061614335415608379464679725005877399393367743025970601491780935772076488458612059547852785477127167625913988889724
Розшифр. M': 123456789012345
Перевірка розшифр.: True
```







Висновок:

У ході виконання практикуму було досліджено принципи роботи асиметричної криптосистеми RSA, методи генерації параметрів для неї та протоколи захищеного обміну даними.

Отримано практичні навички розробки програмного забезпечення для асиметричної криптографії, засвоєно особливості роботи алгоритму RSA та важливість коректного вибору параметрів криптосистеми (зокрема, простих чисел) для забезпечення стійкості захисту інформації.