Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»
Навчально-науковий Фізико-технічний інститут

Криптографія

Комп’ютерний практикум №4.

Вивчення криптосистеми RSA та алгоритму електронного підпису. Ознайомлення з методами генерації параметрів для асиметричних криптосистем

Варіант №3

Виконав:

Студент групи ФБ-31

Васалатій А. Ю., Яковчук О. С.

Перевірено:

Байденко П. В.

12.11.2025

Виконання лабораторної роботи було почато з ознайомлення з необхідними відомостями з теорії чисел. Першочергово було розглянуто і програмно реалізовано піднесення до степеня за модулем з використанням схеми Горнера.

$$x^{\alpha}(\mathrm{mod}\, m) = (...(...(x^{\alpha_{k-1}})^2 x^{\alpha_{k-2}})^2 ... x^{\alpha_1})^2 x^{\alpha_0} (\mathrm{mod}\, m)\text{, або}$$
$$x^{\alpha}(\mathrm{mod}\, m) = x^{\alpha_0} (x^2)^{\alpha_1} (x^{2^2})^{\alpha_2} ... (x^{2^{k-1}})^{\alpha_{k-1}} (\mathrm{mod}\, m)$$

(x, a – цілі невід'ємні числа, $x < m$;
a перед безпосереднім початком обчислень подається у двійковій формі)

Після цього ми перейшли до поняття псевдопростих чисел - складених чисел, що зберігають певні властивості простих. Серед пройдених - псевдопрості Ферма, Ойлера-Якобі та сильні псевдопрості числа. Базуючись на даних відомостях, ми почали розгляд ймовірнісних тестів для перевірки числа на простоту - Ферма, Соловея-Штрассена і Міллера-Рабіна. Після ознайомлення з постулатом Белтрана і алгоритмом генерації простих чисел, ми перейшли до програмної реалізації генерації “гарних” простих чисел з використанням вищезгаданих тестів (у користувача є можливість використати будь-який з трьох) на простоту для подальшого застосування у нашій імплементації криптосистеми RSA.

Серед реалізованого функціоналу:

1) Функція генерації пари відкритий/секретний ключ на основі пари двох великих простих чисел (довжина обиралася 256 біт+). Як експонента використовувалася одна з найбільш поширених у сучасних реалізаціях криптосистема RSA - 65537.
2) Функція зашифрування, що базується на перетворенні $C = M^e mod\, n$.
3) Функція розшифрування, що базується на перетворенні - $M = C^d mod\, n$.

4) Функція генерації цифрового підпису - $S = M^d mod\ n$.
5) Функція перевірки цифрового підпису - як перевірки чи є однаковими M та $S^e mod\ n$.

Також було імплементовано модель протоколу конфіденційного розсилання ключа по відкритим каналам з підтвердженням відправника у вигляді двох функції $((e, n), d$ і $(e_1, n_1), d_1$ є відповідно відкритими і секретними ключами абонентів A/B відповідно, при чому $n <= n_1$):

1) Створення повідомлення виду (k_1, S_1), де

$k_1 = k^{e_1} mod\ n_1;\ S_1 = S^{e_1} mod\ n_1;\ S = k^d mod\ n;$

2) Отримання ключа з повідомлення вище з перевіркою підпису.

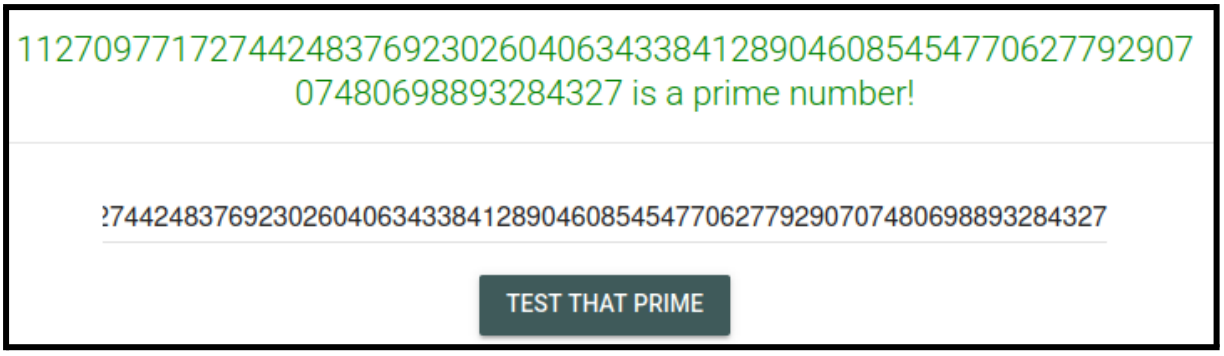
$k = k_1^{d_1} mod\ n_1,\quad S = S_1^{d_1} mod\ n_1$.

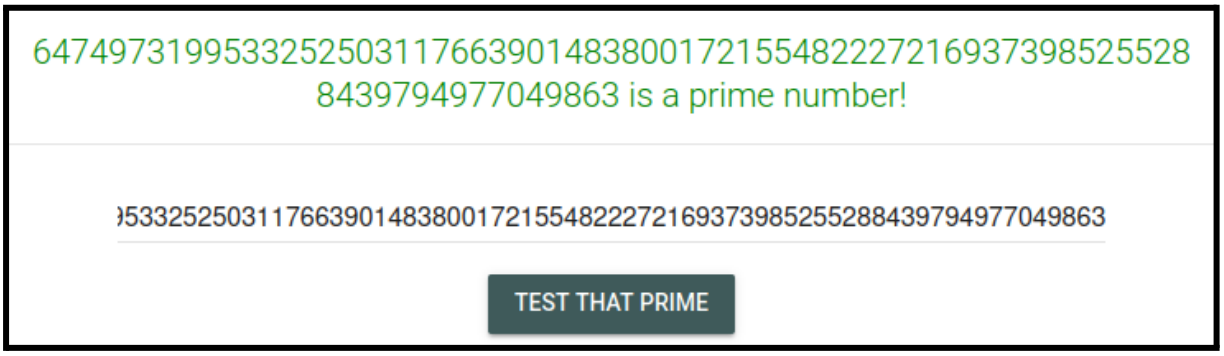
Тестування

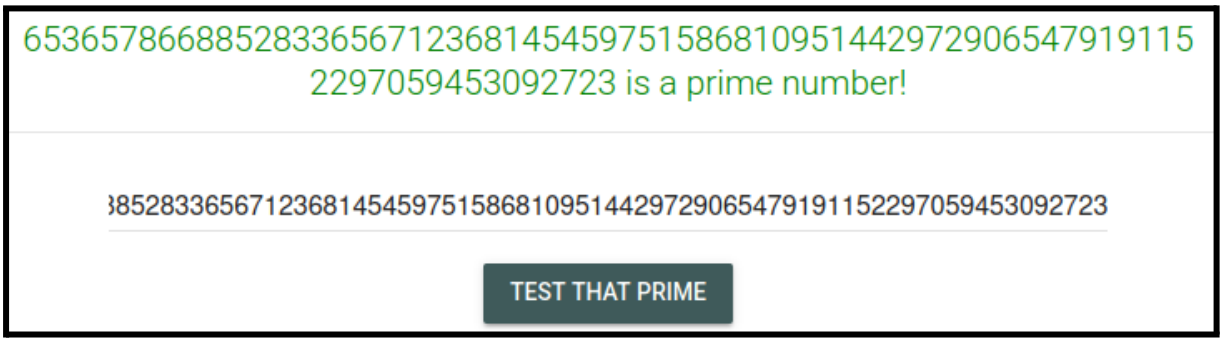
Перевірка коректності реалізованого здійснюється двома способами - локальним тестування через створення двох абонентів, а також використанням RESTful API веб-ресурсу, що був запропонований в методичних рекомендаціях (http://asymcryptwebservice.appspot.com/docs).

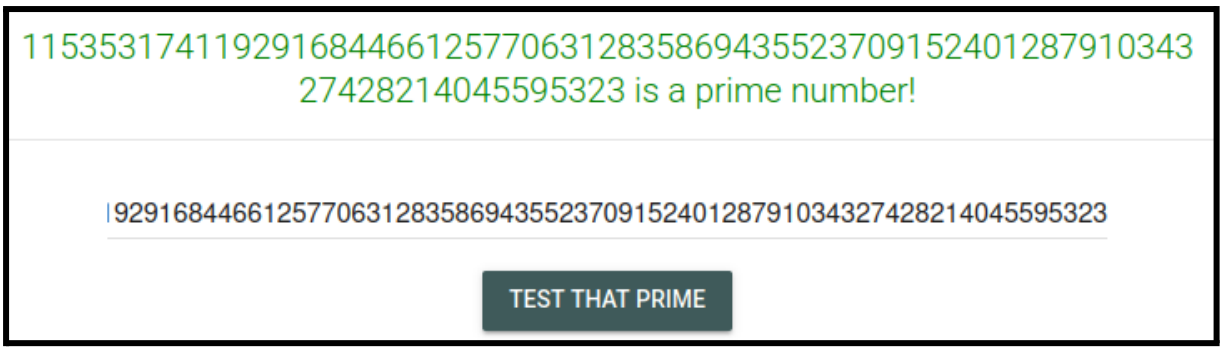
При локальному тестуванні першочергово генеруються дві пари з відкритого й секретного ключів для абонентів A та B, при чому n з відкритого ключа абонента A не більший за B, що необхідно для коректної роботи протоколу розсилання ключа.

Обрані числа $p,\ q,\ p_1,\ q_1$ були додатково перевірені на простоту з використанням ресурсу https://bigprimes.org/primality-test.



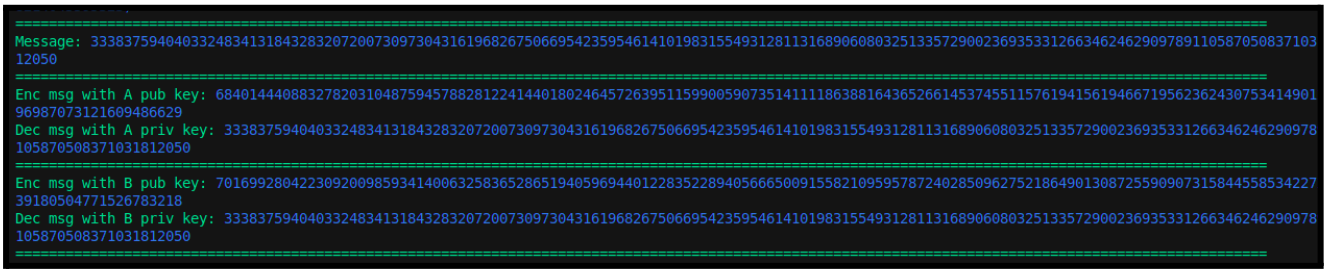






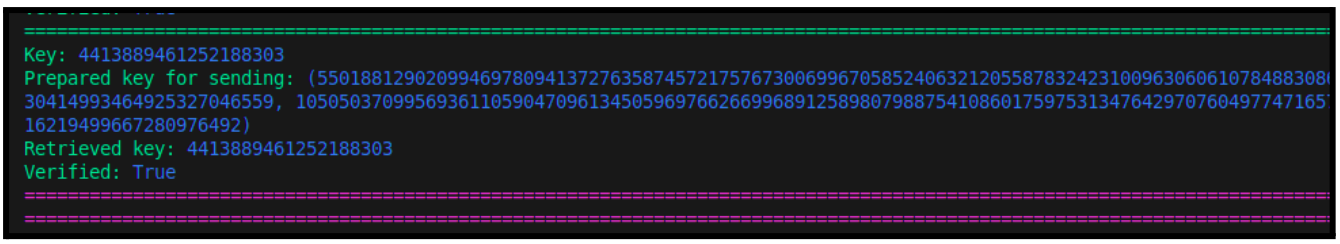
Далі обирається випадкове повідомлення в межах [1, n-1], після чого воно зашифровується і розшифрується з використанням відкритих і секретних ключів обох абонентів.

Також відбувається підпис і його перевірка з використанням ключів обох абонентів.



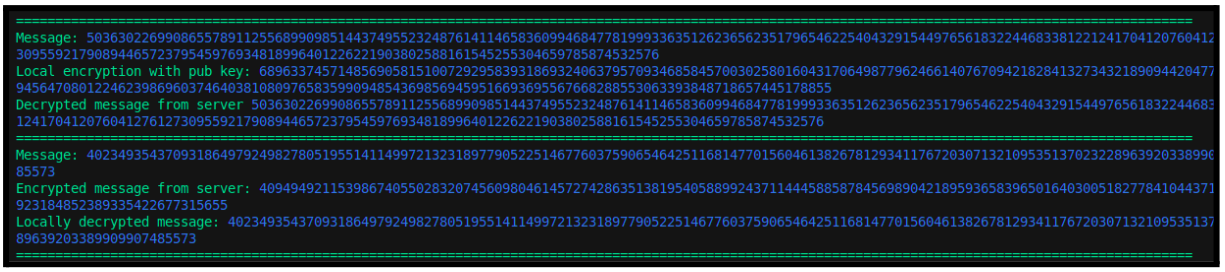
Далі обирається випадкове число - “ключ” - і здійснюється створення повідомлення з зашифрованим ключем та його підписом для абонента B від A.

Після чого відбувається отримання ключа і підпису, з подальшою перевіркою, з використанням секретного ключа абонента B і відкритого A.



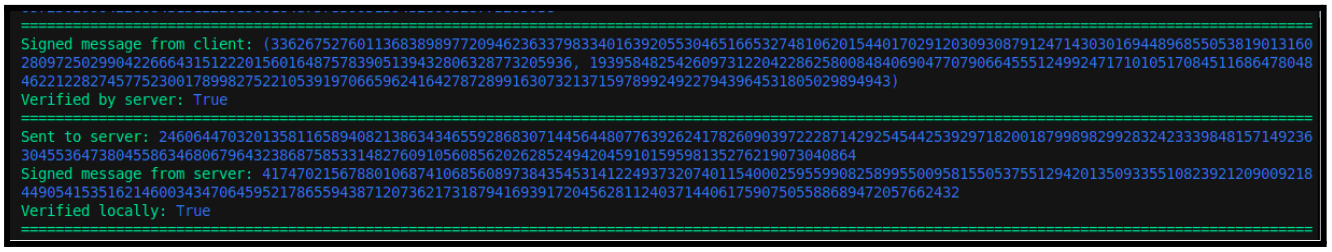
Для тестування з використанням API (http://asymcryptwebservice.appspot.com/rsa/) спочатку здійснюється отримання відкритого ключа серверу через GET-запит за шляхом http://asymcryptwebservice.appspot.com/rsa/serverKey. Після цього відбувається генерація пари ключів локального абонента (модуль ключа локального абонента спочатку обирається як не більший за серверного).

Далі обране клієнтом повідомлення спочатку шифрується локально з використанням відкритого ключа серверу і відправляється йому на розшифрування (http://asymcryptwebservice.appspot.com/rsa/decrypt) для перевірки коректності локального шифрування. Потім інше повідомлення відправляється на сервер для шифрування відкритим ключем клієнта (http://asymcryptwebservice.appspot.com/rsa/encrypt), а результат локально розшифровується для перевірки коректності реалізації розшифрування.

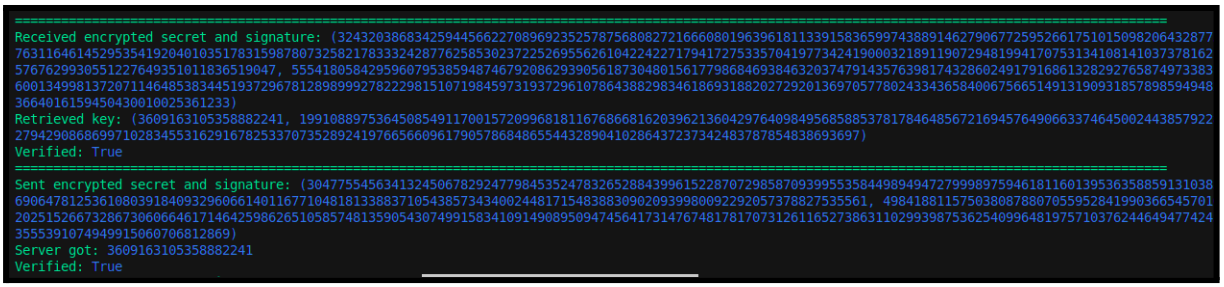


Також відбувається перевірка коректності реалізації створення і перевірки цифрового підпису. Спочатку повідомлення підписується на клієнті і перевіряється на сервері (http://asymcryptwebservice.appspot.com/rsa/verify),

а потім інше повідомлення підписується на сервері (http://asymcryptwebservice.appspot.com/rsa/sign) і перевіряється на клієнті.



Також з використанням імплементації протоколу конфіденційного надсилання ключів спочатку здійснюється отримання такого ключа від серверу (http://asymcryptwebservice.appspot.com/rsa/sendKey) і локальна перевірка його цілісності на основі підпису, а потім повернення назад з перевіркою цілісності з боку серверу (http://asymcryptwebservice.appspot.com/rsa/recieveKey/).



Висновки

Першочергово в межах даного практикуму відбулося ознайомлення з необхідними відомостями з теорії чисел - піднесенням до степеня за модулем за схемою Горнера; поняттям псевдопростого числа і зокрема псевдопростими числами Ферма, Ойлера-Якобі й сильними псевдопростими; ймовірнісними тестами Ферма, Соловея-Штрассена та Міллера-Рабіна, а також алгоритмом генерації простих чисел загалом. Після цього було розглянуто криптосистему RSA, а також протокол конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника

Базуючись на отриманих теоретичних знаннях, було реалізовано програмну імплементацію вищеописаного з перевіркою коректності роботи, базуючись на відкритому RESTful API веб-ресурсу, що був запропонований в методичних рекомендаціях (http://asymcryptwebservice.appspot.com/docs). Зокрема було використано всі запропоновані розробником запити, що стосуються криптосистеми RSA.