

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

Виконали:

студенти групи ФБ-32

Грабовецький Микита

Драбок Алла

Київ - 2025

Мета роботи:

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 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 p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p_1q_1$; p і q – прості числа для побудови ключів абонента А, p_1 і q_1 – абонента В.
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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , (e_1, n_1) та секретні d і d_1 .
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. 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 А і В, перевірити правильність розшифрування. Скласти для А і В 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 $0 < k < n$.

Варіант: 13

Хід роботи:

1. Генерація простих чисел:

Для генерації простих чисел p та q використовувався імовірнісний підхід. Спочатку генерувалося випадкове непарне число заданої довжини біт. Першим етапом число проходило тест пробних ділень на малі прості числа (3, 5, 7...). Якщо число проходило цей етап, до нього застосовувався тест Міллера-Рабіна з кількістю раундів $k=40$, що забезпечує ймовірність помилки не більше $4^{(-40)}$.

2. Реалізація RSA:

- Генерація ключів: $n = p \cdot q$, функція Ойлера $\phi(n) = (p-1)(q-1)$. Випадкове число $e=2^{16}+1$ (зазвичай обирається). Секретний ключ - для e обернений за $\text{mod } \phi(n)$ елемент d : $ed \equiv 1 \pmod{\phi(n)} \Rightarrow d = e^{-1} \pmod{\phi(n)}$.
- Шифрування/Розшифрування: $C = M^e \pmod{n}$, $M = C^d \pmod{n}$.

3. Цифровий підпис:

$S = M^d \pmod{n}$, перевірка $M = S^e \pmod{n}$.

4. Протокол конфіденційного розсилання ключів:

Схема передачі секретного ключа k від користувача А до В:

- Перед початком перевіряється умова $n_B \geq n_A$. Це необхідно, щоб при операціях за модулем n_B не відбулася втрата інформації, якщо значення зашифрованого повідомлення або підпису перевищуватиме n_B .
- Крок відправлення (А):
 - Абонент А підписує ключ k своїм секретним ключем: $S = k^d_A \pmod{n_A}$
 - Абонент А шифрує ключ k відкритим ключем В: $k1 = k^e_B \pmod{n_B}$.
 - Абонент А шифрує свій підпис S відкритим ключем В: $S1 = S^e_B \pmod{n_B}$ (забезпечення конфіденційності підпису).
 - Відправляється пакет $(k1, S1)$.
- Крок отримання (В):
 - Абонент В розшифровує $k1$ своїм секретним ключем: $k = k1^d_B \pmod{n_B}$.
 - Абонент В розшифровує $S1$ своїм секретним ключем: $S = S1^d_B \pmod{n_B}$.
 - Абонент В перевіряє автентичність ключа k , використовуючи відкритий ключ А: $k = S^e_A \pmod{n_A}$.

```
C:\Users\grahm\PycharmProjects\pythonProject(CySec)\venv\Scripts\python.exe "C:\Users\grahm\PycharmProjects\pythonProject(Labs)\Cryptography\Lab4\main.py"
[1] Генерація ключів для абонентів A і B...
A Public Key (n, e): (9627866473137615265257705720264815611274741544835382383495676143889581682480060215949013113985222663155825266359613328820804063156387855479643845127799, 65537)
(Перегенерація ключів B для виконання умови n_B >= n_A)...
(Перегенерація ключів B для виконання умови n_B >= n_A)...
(Перегенерація ключів B для виконання умови n_B >= n_A)...
B Public Key (n1, e1): (10246701164049415234803280310136720029906281009800275530176230919174082405356955266135478951795835513663180388414184812432047283138483941162261740969998249, 65537)

[2] Тест шифрування/розшифрування (Confidentiality)
Відкрите повідомлення: 123456789
Зашифроване (C): 3079623870428460118762466246853225423107529674913137282917130089776804176930761043630432012375968754563551372806207459402171330381307108868751098701131078
Розшифроване (M): 123456789
>>> Статус: Успішно

[3] Тест цифрового підпису (Integrity & Authentication)
Повідомлення для підпису: 987654321
Підпис A (S): 3576023547622301882915363938460732509002635520623835833963139133771599910943171100395521006121611421991077893741993937274691765333992856535581526808756571
Результат перевірки підпису: True
>>> Статус: Підпис вірний

[4] Протокол конфіденційного розсилання ключів (A -> B)
Секретний ключ k, що передається: 89356170860231864208534522038864642078009962484426756245544926350113617341865513631409727775694946760487743680204911113604201999237679952886795054566998
Відправлено пакет (k1, S1): (80468211617009477096896521478736915490277526220697113622572331848469586716882311109547536885690183824891750828360259416995150899845253338876312087293478297, 5698534420804647330125339)
Отримано k: 89356170860231864208534522038864642078009962484426756245544926350113617341865513631409727775694946760487743680204911113604201999237679952886795054566998
Автентифікація підтверджена: True
>>> Висновок: Протокол виконано успішно.
```

Перевірка роботи алгоритму

Перевірка відбувається за допомогою згенерованих ключів у скрипті та за допомогою сайту asymcryptwebservice.appspot.com

```
>>> Генеруємо пару ключів...
```

```
[1] ШИФРУВАННЯ НА САЙТІ
Modulus (n): 5502d819f82b8c82721d3c25ca6a949c2f4b4697cf833de50dee68862af12e889f986fc5d0be3ebc40cc6611a1baff98c332f49963ba0fb8e17547d5267e823
Public Exponent (e): 10001
```

Encryption

✕ Clear

Modulus

5502d819f82b8c82721d3c25ca6a949c2f4b4697cf833de50dee68862af12e889f986fc5d0be3ebc40cc6611a1baff98c332f49963ba0fb8e17547d5267e823

Public exponent

10001

Message

Hello World

Text

Encrypt

Ciphertext

3479155A0D633E29F6B02106EF5FEE866AF8B025BD0B8F0DD4

```
Ciphertext з сайту (HEX) >>> 3479155A0D633E29F6B02106EF5FEE866AF8B025BD0B8F0DD477D0B1428C9472D888C1688659B7676951369F8D496D32563A2126E98BEF5720279AF43FF627AE
Розшифровано (число): 87521618088882533792115812
Розшифровано (текст): Hello World
```

Публічний ключ «передається» сайту, де ми шифруємо повідомлення Hello World за його допомогою. Отриманий шифротекст вставляється у код й розшифровується за допомогою приватного ключа.

```
[2] ПЕРЕВІРКИ ПІДПИСУ
Message (Text): Hello World
Signature: 2ff8c1ef5dbb7cabacd63bf1e64107f4227a063bb51f279a4efe0e1a36cf24c72fb06f2e0fe110d426495b059f2b51062298d518e77cf0d80b8e2d5fad494d
Modulus: 5502d819f82b8c82721d3c25ca6a949c2f4b4697cf833de50dee688862af12e889f986fc5d0be3ebc40cc6611a1baff98c332f49963ba0fb8e17547d5267e823
Public Exponent: 10001
```

Verify

Message

Text ▾

Signature

Modulus

Public exponent

Verification

✓

Для перевірки підпису, код проводить математичне перетворення повідомлення за допомогою приватного ключа й разом з публічним ключем передається на сайт.

Зі скріншоту можна зробити висновок, що підпис дійсний й код успішно пройшов тестування.

Висновки:

Було реалізовано криптосистему RSA з повним набором необхідних процедур: генерація ключових пар, шифрування та розшифрування повідомлень, створення і перевірка цифрового підпису, а також протокол конфіденційної передачі ключів між абонентами.

На тестових повідомленнях програма коректно виконала всі операції: повідомлення 123456789 після шифрування та розшифрування співпадає з оригіналом, а підпис для 987654321 успішно підтверджено відкритим ключем.

Протокол передачі секретного значення к також виконано успішно – отриманий ключ співпав із відправленим, а автентичність підтверджено.