Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
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 III курсу ФТІ
групи ФБ-82
Сумовська Юлія та Руднік Анатолій
Перевірили:
Завадська Л.О.
Савчук М.М.
Чорний О.М.

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 $pq \leq p_1, q_1$; p і q – прості числа для побудови ключів абонента A, p_1, q_1– 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), (e_1, n_1) та секретні d, d_1.
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 $0 < k < n$.

Опис роботи та основні труднощі:

Програма має такі функції:

- gcd - пошук найбільшого загального дільника;
- miller_rabin - тест Міллера-Рабіна з попередніми розподілом;
- choose_random_prime - функція для вибору випадкового простого числа з певного інтервалу;
- findModInverse - пошук зворотного по модулю;
- good_random - генерує хороші прості числа для кращої стійкості;
- gen_p_q - генерує q, p, q_1, p_1, такі що $q * p \leq q_1 * p_1$;
- GenerateKeyPair - генерує відкритий і закритий ключ користувачам A і B;
- Encrypt — шифрування;
- Decrypt — розшифрування;
- Sign - цифровий підпис;

- Verify - перевірка цифрового підпису;
- SendKey - відправити ключ з підтвердженням справжності;
- ReceiveKey - отримати ключ і перевірити справжність відправника.

Труднощі виникли на моменті зашифрування, коли потрібно перетворювати рядок в число. Проблема була вирішена взяттям значення ASCII кожного символу і конкатенацією цих значень.

Значення вибраних чисел p, q, q_1, p_1 із зазначенням кандидатів, що не пройшли тест перевірки простоти і параметрів криптосистеми RSA для абонентів A і B:

Ім'я cookie: JSESSIONID
Значення cookie: Qrd1vbd7QVH53rIYtcA4Kg
q користувача tolik = 0xdb6b61690a51b98b87249a8b43acee0aaeb884cc30d5b99c1db9c64da512e11f
p користувача tolik = 0x8620f84605aa7d36d6e81c987e979a9ecc35e47f925974907f2503c03ad2a8e5
n користувача tolik = 0x72f67735331daf9664a3f2e22e6ba912d0d6a8eb382044701b6497d8a7bd33770cb
a2a1924865b48ef83ff80b23eec4a2052f18b92f67f7eb5b03c86bf0db8bb
e користувача tolik = 0x10001
d користувача tolik = 0x57e9699761135e669f675020ee669dd029ab3d59427e4a57805b1853315259bbad
bf471517c7dbce9c069f6b9287e30fc1ea72d679496e425d802d0d8978ece1
n_1 сайта: 0x87678098e4adce5153c985d53fb018f2dc9b849337b08469711636af37a741c4d60098cfcf69b
ddc60bdd42cc33d1e2a2cdba91ace92d3b41a0c51227a8b4bcd
e_1 сайта: 0x10001

Опис кроків протоколу конфіденційного розсилання ключів з підтвердженням справжності, чисельні значення характеристик на кожному кроці:

Процедура SendKey приймає на вхід відкритий ключ отримувача, зберігає його в змінні e_1, n_1:

e_1 = 10001
n_1 = 0x87678098e4adce5153c985d53fb018f2dc9b849337b08469711636af37a741c4d60098cfcf69bddc60
bdd42cc33d1e2a2cdba91ace92d3b41a0c51227a8b4bcd

В змінні e, n зберігає відкритий ключ відправника:
e_1 = 0x10001
n_1 = 0x72f67735331daf9664a3f2e22e6ba912d0d6a8eb382044701b6497d8a7bd33770cba2a1924865b48e
f83ff80b23eec4a2052f18b92f67f7eb5b03c86bf0db8bb

Генерує випадкове число k в межах від 1 до $n_1 - 1$:
k = 0x4e86205674f50525a089bc7e0950d0a4a1abf5f9676c6e33a6fa0f1002abc8f2e863215c9d9eee3396ebe
f3412291fd0830a1a8ee60b9d16374d57154c13e122

Зашифровує його за допомогою відкритого ключа отримувача і зберігає у змінну k_1:
k_1 = 0x24bb65a42ca8f5ad00f83ab6f3699d82cd7ce4854d2bd0bdd466a0f3ab6d0a025def7b598c80b68d312
aa6527be468490bea02417ae1799347338e075175f3ec

Підписує k за допомогою секретного ключа та модуля відправника та зберігає в змінну s:
s = 0x45cccc975b96729a5ac7589589ffee82990854f2469dd1676803b5f928c05f86e5363b1413770171f986
7ceb32a316cd65bf52b22281901d59b23e7d576b9ad8

Підписує s за допомогою відкритого ключа отримувача та зберігає в s_1:
s_1 = 0x4fcea6f18fe92bbac9944aa0ad1f3331864840db04eae8e875c551a93d5fef086bcd485e34d7dd6b938
25bc191248be23c1fe8e3adc2835594bf9b2f018fcace

Повертає масив [k_1, s_1]

Висновки:

В даному лабораторному практикумі ми ознайомились з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA. Також практично ознайомились з системою захисту інформації на основі криптосхеми RSA, організували з використанням цієї системи засекречений зв'язок і електронний підпис. А також вивчили протокол розсилання ключів. Цей практикум дуже важливий, тому що алгоритм RSA використовується у таких сучасних протоколах як: PGP, TLS/SLL, IPSEC та ін. і треба розуміти як це працює.