Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

КРИПТОГРАФІЯ

Комп’ютерний практикум

Робота № 4

Виконали

студенти гр. ФБ-06,

Зінов'єв Андрій, Даценко Валерія

Київ - 2022

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 комп'ютерного практикуму

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 A, p_1 і 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 і 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 $0 < k < n$.

Хід роботи

Було написано алгоритм, що генерує прості числа довжиною 256 бітів для змінних p_1, q_1, p_2, q_2. Описано ключові моменти криптографічного алгоритму RSA у вигляді багатьох функцій. А також, застосування його на практиці, на прикладі передачі випадкового числа 256 бітів. Алгоритм збережений у файлі *lab4.py*.

Код складається з декількох функцій:

1. Функція *func()* та *gcd()* - запозичені з 3-ї лабораторної роботи, виконують наступні функції: знайдення оберненого елемента за розширеним алгоритмом Евкліда, знайдення НСД двох чисел за алгоритмом Евкліда.
2. Функція *pow()* - підносить величезне число до величезного степеня за модулем, в основі алгоритму лежить використання бітів показника степеня у двійковій системі відліку, ітеративне множення попереднього результату з числом, піднесеним у степінь 1 чи 0 на квадрат результату минулої ітерації, в кінці множене на саме число без возведення в квадрат.
3. Функція *test_prime()* - перевірка заданого числа на його простоту, за допомогою алгоритму перевірки Міллера-Рабіна.
4. Функція *rand_prime()* - генерує випадкове просте число заданого розміру в бітах.
5. Функція *create_pair()* - генерація випадкових великих чисел для А та Б - значення p_1, q_1 та p_2, q_2.
6. Функція *rsa_pair()* - за допомогою випадкових чисел p і q, для А або Б генеруються набори ключів, такі як відкриті (public keys): e, n; та секретний, приватний ключ (secret key): d.
7. Функція *encrypt()* - зашифровує ВТ за допомогою відкритого ключа {e, n}.
8. Функція *decrypt()* - розшифровує ШТ за допомогою секретного ключа {d, n}.
9. Функція *get_ds()* - створення цифрового підпису для повідомлення.
10. Функція *ds_is_verified()* - перевірка цифрового підпису (автентифікація) для подальшого розшифрування повідомлення.
11. Функція *start()* - тіло програми, усе що відбувається в ній, та і не тільки в ній, описано далі.

Для генерації випадкових чисел у лабораторній роботі використовувався модуль *random* - його функції randint і getrandbits.

Спочатку згенерували ВТ, p_1, q_1 та p_2, q_2, нехай:

M = 4891240494103891742464620858323728947976152089447026415601482779306894546825 8;

p_1= 2184341142982713388683730850320566955389188337753443454828754668222850322 3107;
q_1= 5180522418868225628168497610219848467738647671852226233247556929073779884 8799;

p_2= 5393766801512397283237162524607657267033472031028382278643840605662413539 9151;
q_2= 8527457272703348323022056381068484364552222072056876875152824421095373569 4807;

Нехай Б хоче відправити повідомлення до А. Так як Б хоче надіслати повідомлення А, він намагається запросити у А відкритий ключ шифрування $\{n_1, e_1\}$. А генерує набори ключів: відкритий $\{e_1, n_1\}$, приватний $\{d_1, n_1\}$ і надсилає Б відкритий.

Після того, як Б успішно отримав ключ $\{n_1, e_1\}$: шифрує ВТ за допомогою функції *encrypt(),* передаючи на вхід: ВТ та відкритий (публічний) ключ А - $\{e_1, n_1\}$. ШТ знаходиться за формулою: $C = M^e mod(n)$

C =7215110862770028773864162065449805949086783934980376204828118212941962470229075715910101740309919273535870179057878772756639322004261125974300519222260 8;

Отримавши зашифрований зміст, Б генерує ключі: відкритий $\{e_2, n_2\}$, що надсилається до А, секретний ключ $\{d_2, n_2\}$, за допомогою якого та ШТ, створює цифровий підпис у функції *get_ds(),* що у результаті повертає значення {text, signature}. ЦП знаходиться за формулою: $S = C^d mod(n)$.

S = 2985751640576584508419893159327821685327103246117665907826173189266615216341686199234368855075023042878225863473743573212010956830881030792318106909991484;

Після успішного утворення підпису, повідомлення{text, signature} надсилається до А, де той перевіряє його за допомогою відкритого ключа Б $\{e_2, n_2\}$ - автентифікує та перевіряє на цілісність ШТ. Тобто, за допомогою функції *ds_is_verified()* А перевіряє повідомлення, подаючи на вхід функції повідомлення {text, signature} та ключ $\{e_2, n_2\}$. Якщо процедура автентифікації успішна, то програма продовжує свою роботу.

Щоб А зміг розшифрувати повідомлення, яке йому надіслав Б, він дістає ШТ з повідомлення {text, signature} і застосовує свій приватний (секретний, закритий) ключ $\{d_1, n_1\}$ до функції *decrypt()*. Розшифрування відбувається за формулою $M' = C^d mod(n)$.

M' = 489124049410389174246462085832372894797615208944702641560148277930689454682 58.

Як виявилось, M=M', а отже програма працює.

Висновки: У ході виконання даної лабораторної роботи застосували на практиці алгоритм Міллера-Рабіна для визначення простоти будь-якого обраного цілого числа, згадали операції з модулярної арифметики: знаходження оберненого елемента у кільці за модулем, алгоритми Евкліда. Більш детально ознайомились з особливостями криптографічного алгоритму RSA з використанням цифрового підпису. А також, на практиці за допомогою вищеперелічених алгоритмів змогли передати від Б до А секретну інформацію з автентифікацією за цифровим підписом.