Міністерство освіти і науки України Національний технічний університет України
"Київський політехнічний інститут ім. 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:

Студент групи ФБ-05 Даниленко Данило,

Студентка ФБ-05 Мірошніченко Ілона

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 p1 , q1 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), (e1, n1) та секретні d і d1.
- 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 М і знайти криптограму для абонентів А и 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. 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 Епстурt(), 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(). 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:

У нашій лабораторній роботі ми реалізували усі потрібні функції, в тому числі знаходження найбільшого спільного дільника за допомогою розширеного алгоритму Евкліда, пошук оберненого елемента, генерація простих чисел, перевірка натуральних чисел на простоту, а також генерація пар ключів.

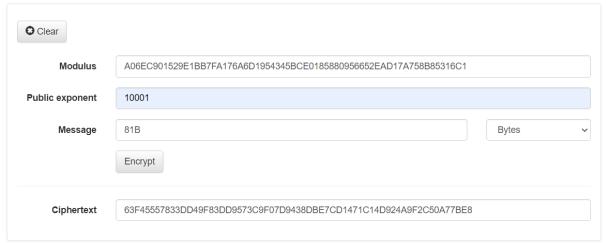
Key matched!
Combination for A:
COMMITTALION TOP A.
p:64709924902482481683585900353439746859601848767884193665958032205562123611081
q:74194228603361459949839616500487342076193305108876599314690588062448335409633
Combination for B:
p:103423437029672750986014051833240808562850301701532863441175374961371406718777
q:73056933397168414133778195400591784688622263557426935576633684249340038932929
d:2624257008309435993402634392418164683148006260911546455002213226204542853156730320624066095275565074634790839837617683865319275639862545955523902114619727
n:4801102961121137774444109272106850521301399496280363856333228505870939425730341868873071552673701212463192105184531136379710738299882170665225381312943273
e:2290216086898604392457012323474105388402021064892510991551309338198899481711295440674798036409537928304673579164378453805527571156535254572853442600612783
d: 4552114707129981384093126910577967315674218906178827161925888013299685130472679717438238701426696925712049215592397321217207313615484063794059421209556803
$\tt n:7555799150783043650050710737928099358305404862248327994442371862751612633800006893191050828814916770752219867995540675075053521850629955178577592267907833$
e: 6442684737046093098688215380726610093800990081935757296460335520649484060716878676482371382688899855641204202389667375874811656116612780967495821719274347
Manager 1/

Get server key



Message(hex) = '81B'

Encryption

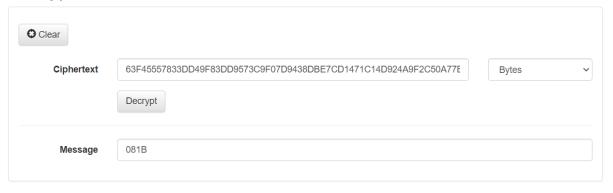


Decimal to Hexadecimal converter

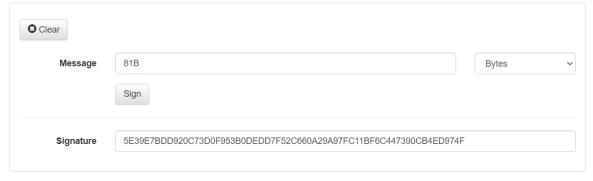


Значення співпали.

Decryption

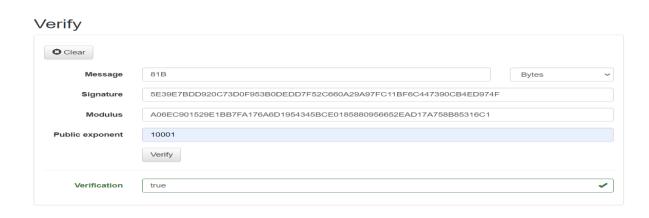


Sign



print(Client.authentication(test_message, sign_int, server_e_int, server_n_int)) #True

45210671701325481290643043386336312396560172791927300126915100669227072715752 True



Висновки:

В ході даної лабораторної ми реалізували систему захисту інформації на основі криптосхеми RSA, написали функцію перевірки на простоту за алгоритмом Міллера-Рабіна, також реалізували функцію генерації ключових пар для RSA та написали функції шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і В. На нашу думку, ця система стане у нагоді у нашому повсякденному житті.