Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

Комп’ютерний практикум №4

Вивчення криптосистеми RSA та алгоритму електронного підпису;
ознайомлення з методами генерації параметрів
для асиметричних криптосистем

Виконав:

студент 3 курсу ФТІ

групи ФБ-83
Самчук Тарас

Перевірив:

Чорний О. М.

Київ 2020

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 і рекомендації щодо 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 1 1 p , q 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) , (,) 1 n1 e та секретні d і d1 .
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.

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 Encrypt(), 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().

Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою http://asymcryptwebservice.appspot.com/?section=rsa. 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.

Порядок виконання програмного коду:

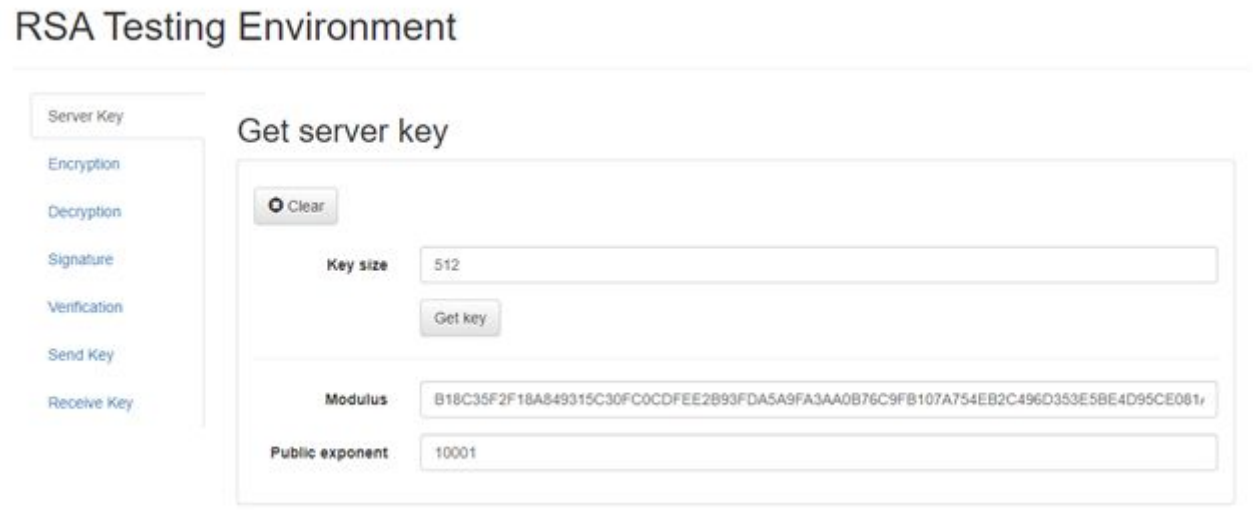
1. Генерується пара простих чисел та відкритий ключ і секретний ключ.
2. Аліса формує повідомлення з підписом.
3. Боб приймає повідомлення, перевіряє підпис та розшифровує повідомлення.

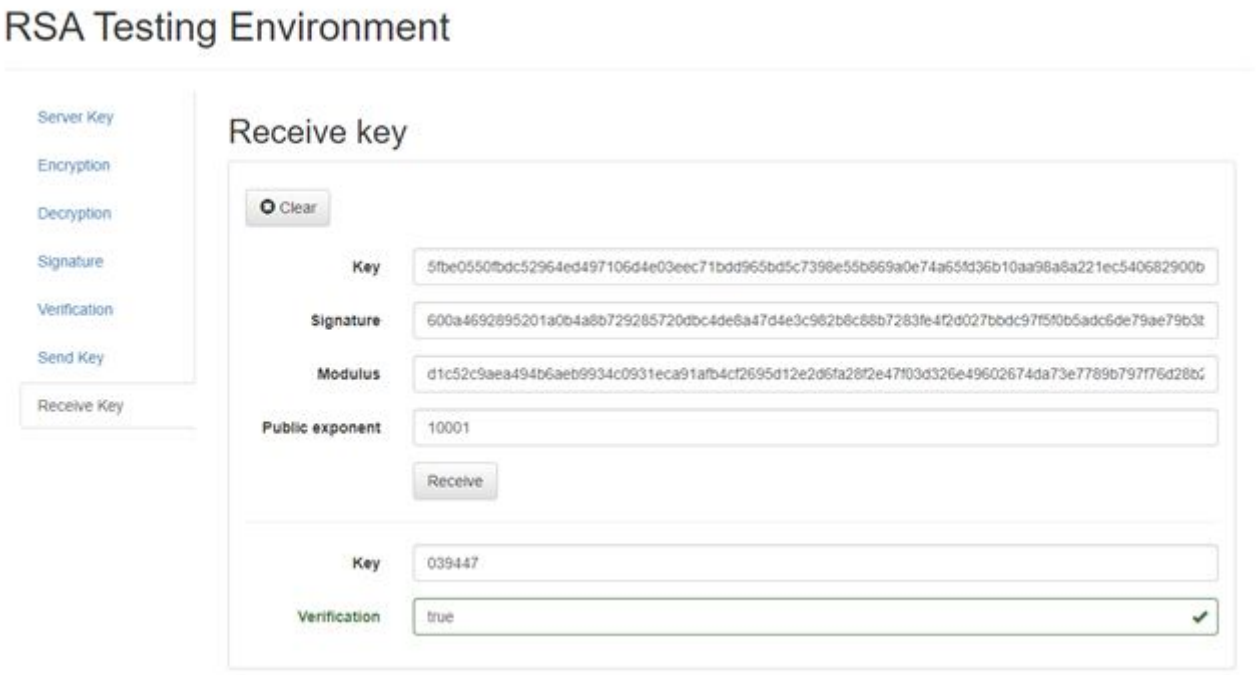
Згенеровані не прості числа:
0xf837eb6218bcdc7dd51d292635c709e6f42299bcdcfd9fea23ae263d03eae4ef isn't prime
0x8d8f9575aac0b997693a62a9ca69373e79a705e11ceab295c03a1bd0285018c9 isn't prime
0x92fdcb932512d3f6d1274efbbcfd1296026de77baa0d5e51ac9b8c3f3b57344f isn't prime
0xc6996b78372998308b09b90ab7bbfaa29ba9ab7581abeeeed92b20cc975a19fd isn't prime
0xc8d1972f13f699ce8898ad367ebbe6fb4806efcce5d4281f481128961d8f3af5 isn't prime
0xd433003e80a5de9fb4cceabdf343627c82b80f625bd8954735ad63a5e5bedf03 isn't prime
0xc26a14c51748f45118b8db60fc1d595d51752ad459a91f61a3adc4a7ac8ba37e isn't prime
0xec008bc61956ca42ee77c8b7edf0e52ac52c083f17d2e280e9b38b5848eacd19 isn't prime

text = 234567
p1 = 1108597992863897194784298604471409808594870284080375601357978756188228676480 71.
q1 = 1147254091264797703148462415887667772922659171776139409978789673320745452416 01
n1 =
1271843582881049065145065941621235301619554744473811945388457119319516244191295629139048839855963347962628355733360829990037295883090877986862109933660167 1
e = 65537
d1 =
2140536600573412147115381744065524112617863013495604888480504451850750594844094476237671743986455550984519455800803907913259746400467207628868308394014 73

p = 9365144258898719762586031381670965952614895215367525814846940337953555422234 01.
q = 1023573366342345272460975329714443602783576587729775927006208822429812615189 53
n =
9585912235362650903429926106062037586500534840207325977433061623293364639854906087817967804930099361714839753176638183322660468299299595462826932121619153
d =
2009272264174080831597676044356229463139262509726231547858430619637471200355282548114122317766913481650600802868874440651679012204027718534862093980830273

Робота з сайтом





Висновок: під час виконання лабораторної роботи №4 я ознайомився з асиметричною криптосистемою RSA, генерацією ключів для криптосистеми, а також генерацією простих чисел. У ході роботи були реалізовані функції GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey().