Криптографія

Комп'ютерний практикум №4

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

Мета та основні завдання роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної кринтосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі кринтосхеми 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 винадкових 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 згеперувати дві пари простих чисел p,q і p_1,q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \le p_1q_1$; 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 повиппа повертати та/або зберігати секретпий 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, n) та секретні d і d_1 .
- 4. Паписати 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 Епстурт(), яка шифрус повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Відновідно, програмний код повинен містити сім високорівневих процедур: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey().

Хід роботи

Реалізували функції Gen_random_number() та Miller-Rabin() для генерації р та q, p1 та q1, та перевірки їх на простоту.

```
p and q find for abonent A and abonent B

For abonent A

p: 75356006373634469236464298193733705001533680488307266386443146550822716295419

q: 58616912969763481869134267146597644182140870626675757140257663035120581124901

For abonent B

p1: 84358982512063654863119490327146880431876654997287705732114648571902080605403

q1: 1123108893757370961322261637182987184805573649184224612970224235262164672300731
```

Також написали функцію GenerateKeyPair() для пошуку n, phi(n), e, d Де n = p*q, phi(n) — функція Ейлера, e - відкритий ключ, d - таємний ключ

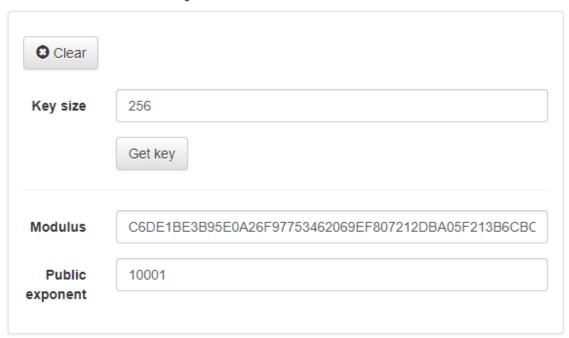
Далі шифруємо наше повідомлення

Для прикладу взяли текст «Hooray, we are done it!»

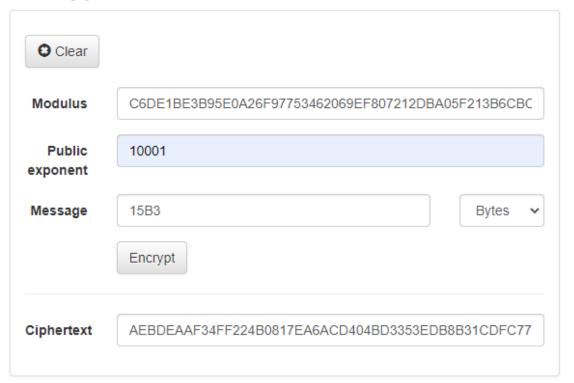


RSA Testing Environment

Get server key

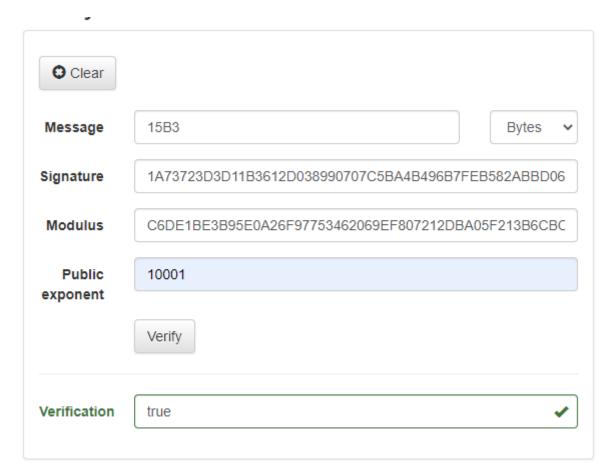


Encryption



Sign





```
Moduls = int("C6DE1BE3B95E0A26F97753462069EF807212DBA05F213B6CBCDC581004616259", 16)
Public_exponent = int("10001", 16)
Message = 5555
Signature = int("1A73723D3D11B3612D038990707C5BA4B496B7FEB582ABBD069018441B056E08", 16)
Verification = Verify(Message, Signature, [Moduls, Public_exponent])
print('Message: ', Message)
print('Signature: ', Signature)
print('Modulus: ', Moduls)
print('Public exponent: ', Public_exponent)
print('Verification: ', Verification)
```

Message: 5555

Signature: 11964109925686052554112432221627228390040191129793772659849706290902615813640
Modulus: 89950376554448262637310497633216635243299057667427333919242081280819183313497

Public exponent: 65537 Verification: True

Мета та основні завдання роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної кринтосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі кринтосхеми RSA, організація з використанням цісї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Висновок:

В ході цієї лабораторної роботи ми ознайомилися з тестами чисел на простоту і методами генерації ключів для асиметричної криптосистеми RSA, та практично закріпили знання. Навчилися створювати електронний підпис.