

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

ФІЗИКО- ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

Лабораторна робота №4

з дисципліни Криптографія

з теми:

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

Виконала:

Бородай Ю.

Групи ФБ-96

Мета та основні завдання 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 $1 < p, q$ 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 А, $1 < p$ і q_1 – абонента В.

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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , (d, n) і d_1 .

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 і 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 `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.

Тест простоти Міллера-Рабіна:

```
def millerrabin_test(n):
```

```
    for prime in [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41]:
```

```
        if n % prime == 0:
```

```
            return False
```

```
    if n == 3 or n == 2: # числа прості
```

```
        return True
```

```
    if n < 2 or n % 2 == 0: # парні тому не прості
```

```
        return False
```

```
    else:
```

```
        s = 0 # встановлюємо лічильник
```

```
        d = n - 1 # для розкладу
```

```
        while d % 2 == 0: # пока d \ на 2, збільшуємо счетчик
```

```
            s += 1
```

```
            d = d // 2
```

```
        for i in range(6):
```

```
            a = random.randrange(2, n - 2)
```

```
            x = pow(a, d, n)
```

```
            if x == 1 or x == n - 1:
```

```
                continue
```

```
            for i in range(1, s):
```

```
                x = pow(x, 2, n)
```

```
                if x == 1:
```

```
                    return False
```

```
                if x == n - 1:
```

```
                    break
```

```
            if x != n - 1:
```

return False

return True

Частина варіантів ключів не пройшовших тест на простоту:

157679868220163380271067298285262327966643469870872099528739683673083882050507 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050509 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050511 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050513 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050515 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050517 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050519 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050521 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050523 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050525 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050527 Is not a prime number.

157679868220163380271067298285262327966643469870872099528739683673083882050529 Is not a prime number.

....

157679868220163380271067298285262327966643469870872099528739683673083882051079 prime


121109653385964961222211875386845537491656264703991151153008036834481440668375 Is not a prime number.

...

121109653385964961222211875386845537491656264703991151153008036834481440668699 prime

```
шт з підписом
84520234029848051540374593190689204892096880436499471446385132374762882227983637832885677612803870318946555
46823942878225908790921432285068244523409496131
верифікований вт
123456789
True
-----перевірка на сайті-----
Key:
32
0x33a6945ccff5e5b00daf398794f4bbc289c3a4b7a441d12070d8edf2e3058477bcf27b9819a5e029189bb9aff20573f1e8281ce12
f28df96999a8c567875dfb0
Signature: 0xbd3be9b40ea6183fba5e51ee5c6a3c90921ac4861acd43060dd3fba1e99304e485181d0c7d3186127a40dd7ac92f7
a3b4467df8b5cf572ae2c67f22a2ce1661c
0x83b16a603bd205bc4d3b4de5930816fb1985c16aacca56cf3ae45a1de94b7c200b596b97aa33f9ebc71100617e24028d09cb27454
74bc2894dbc4fef27e0507f
```

Get server key

 Clear

Key size

256

Get key

Modulus

C637B443C75377FAE9A1E2EED1E1364D83063E1053EA664641E3C899339

Public exponent

10001

Encryption

✖ Clear

Modulus

C637B443C75377FAE9A1E2EED1E1364D83063E1053EA664641I

Public exponent

1001

Message

1234

Bytes ▼

Encrypt

Ciphertext

9B3BD4ABECB9EEBB77D4B6AFF0AB9F9FFE5EEA9F83F6E049I

Sign

✖ Clear

Message

1234


Bytes ▼

Sign

Signature

88D6EC267F775ACF88A5605B9463CCD3A509E1A8EECE161267ABA287802

Verify

 Clear

Message

1234

Bytes

Signature

88D6EC267F775ACF88A5605B9463CCD3A509E1A8EECE161267ABA2878

Modulus

C637B443C75377FAE9A1E2EED1E1364D83063E1053EA664641E3C89933

Public exponent

10001

Verify

Verification

true

Висновок

Ознайомилась з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомилась з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчила протокол розсилання ключів.