



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

Криптографія

Лабораторна №4

Виконали:
Студенти групи ФБ-82
Козачок Вячеслав
Кузнєцов Ілля
Перевірів:
Чорний. О.

Київ – 2020

Оглавление

Мета та основні завдання роботи.....	3
Порядок виконання роботи.....	3
Хід роботи	4
Перевірка шифрування	6
Перевірка підпису	7
Перевірка відправки секрету	8
Висновки	9

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 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 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 А, p_1 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) , (e, n) та секретні d 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 M 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>

Хід роботи

Для генерації використовували простий перебір чисел від n до $n*2-1$.
Генерується послідовність біт заданого розміру після чого алгоритм перевіряє це число на простоту, якщо число складене, то додати 2 та перевірити знову, і так поки не знайдеться число просте.

```
[*] Generating prime number 'p' ...
len(bits_str)=256
bits_str='100101010110010010011010101100000011011001111101000010000000101111011001010
0100100011000110010000100100000011000001111101010111010100010110001111110001000100000110
100011000110000010011001111110000001101110010000001001110010001000100000110111011'
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883bb is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883bd is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883bf is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883c1 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883c3 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883c5 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883c7 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883c9 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883cb is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883cd is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883cf is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883d1 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883d3 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883d5 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883d7 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883d9 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883db is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883dd is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883df is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883e1 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883e3 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883e5 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883e7 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883e9 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883eb is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883ed is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883ef is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883f1 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883f3 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883f5 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883f7 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883f9 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883fb is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883fd is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c8883ff is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c888401 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c888403 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c888405 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c888407 is not prime
[*] Number 0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c888409 is not prime
```

[illegible]

```

bits_str='11111001111111001100000101101010011001001011110001001000110000110011101110110001
100101110011110000111010000100010111100100100000001011111110100110000011000010001101110100
101001000111100110101010110100011110111111111110001100001101000110000010100000100011'
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2823 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2825 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2827 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2829 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c282b is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c282d is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c282f is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2831 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2833 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2835 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2837 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c2839 is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c283b is not prime
[*] Number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c283d is not prime
[*] Prime number 0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c283f
[!] Generated prime 'q'
[+] p=0x955924d581b3f4202f6524632120307d5ea2c7f11068c6099f81bc809c888487
[+] q=0xf9fe60b5325e24619dd8cb9e1d08bc9017fa60c2374a479aad1efff1868c283f
[+] f=0x91d81bb15a3829be6b95f9c3492d4adbc81bf12965a271e79c2e7d321ea3169420aa5a70059398bbb08
54611cd2e4f02bb0b9cb33370a808bd5f0bc7db0d0874
[+] n=0x91d81bb15a3829be6b95f9c3492d4adbc81bf12965a271e79c2e7d321ea31695b001dffab9a5b13d7dc
336130b573c1031a8c5667b23b5ad09ffc839fe21b539
[+] e=0x10001

```

Перевірка шифрування

Шифруємо на нашій стороні повідомлення ключем який видав нам сервер

```

def test_with_server_key():
    server_modulus_str =
'0x9F7C062AF66FE39A065A57AA6826EE45CE350953725EAEB1A6EFCC44E534B8AB'
    e = 0x10001
    message = 0x1111
    server_modulus = int(server_modulus_str, 16)
    print(hex(encrypt(message, e, server_modulus))[2:])

>>> test_with_server_key()
5508eb21d413d05421164b6804006b7e05fd2612c15916b36f92fa8d6a07d3c6

```

Перевіряємо за допомогою сайту чи коректно розшифрується повідомлення.

Decryption

Ciphertext
5508eb21d413d05421164b6804006b7e05fd2612c15916b36f92fa8d6a07d3c6

Bytes

Message
1111

Перевірка підпису

Sign

Message

1234

Bytes

Signature

1D2F016E33204AAEB83516A8A314B43EA8A507E6976F7267F5DCE55DB537DC7D

```
def test_verification():
    server_modulus_str =
'0x9F7C062AF66FE39A065A57AA6826EE45CE350953725EAEB1A6EFCC44E534B8AB'
    signed_message_str =
'0x1D2F016E33204AAEB83516A8A314B43EA8A507E6976F7267F5DCE55DB537DC7D'
    e = 0x10001
    message = 0x1234
    server_modulus = int(server_modulus_str, 16)
    signed_message = int(signed_message_str, 16)
    print(verify(message, signed_message, e, server_modulus))

>>> test_verification()
True
```

Скрипт показав True, а отже верифікація успішна.

Перевірка відправки секрету

```
def test_send_key():
    server_modulus_str =
'0x9F7C062AF66FE39A065A57AA6826EE45CE350953725EAEB1A6EFCC44E534B8AB'
    e, d, n = gen_key_pair(64)
    e1 = 0x10001
    message = 0x1234
    server_modulus = int(server_modulus_str, 16)
    k1, s1 = send_key(message, d, n, e1, server_modulus)
    print (hex(k1) + '\n' + hex(s1))

>>> test_send_key()
[*] Generating prime number 'p' ...
[!] Generated prime 'p'
[*] Generating prime number 'q' ...
[!] Generated prime 'q'
[+] p=0x9e103a1385b70f05
[+] q=0xe5b983a02f05fea3
[+] f=0x8dd70ef2a39d598b84cec5db45d17888
[+] n=0x8dd70ef2a39d598d0898838efa8e862f
[+] e=0x10001
[+] d=0x3aa535c7fb003f1ae1a180cb25b5f349
0x95cd106dd757bbb9b07e894395edad7003031a1fc9e999ff42bf440e9014830f
0x7f0af04345febde93f4df7a537adc094a35bfc268d4f58707226c693b13f8b2
```


Receive key

Clear

Key

95cd106dd757bbb9b07e894395edad7003031a1fc9e999ff42bf440e9014830f

Signature

7f0af04345febde93f4df7a537adc094a35bfc268d4f58707226c693b13f8b2

Modulus

8dd70ef2a39d598d0898838efa8e862f

Public exponent

10001

Receive

Key

1234

Verification

true

✓

Наша програма успішно обмінялася секретом з сайтом, що свідчить про правильність роботи скрипта.

Висновки

Ми засвоїли основні принципи роботи протоколу RSA та навчились програмувати його самотужки. Навчилися генерувати великі прості числа за кількістю біт в них, робити перевірку на простоту за допомогою алгоритму Міллера-Рабіна.