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

студенти групи ФБ-93

Бурячок А.А.

Данілін Д.Д.

Перевірила:

Селюх П.В.

Київ - 2021

**Мета**: 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.

**Завдання**:

* 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.
* За допомогою цієї функції згенерувати дві пари простих чисел і довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб , де - прості числа для побудови ключів абонента А, - абонента В.
* Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p, q) та відкритий ключ (e, n) . За допомогою цієї функції побудувати схеми RSA для абонентів А і B – тобто, створити та зберегти для подальшого використання відкриті ключі (), () та секретні і .
* Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А 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 А і B, перевірити правильність розшифрування. Скласти для А і B повідомлення з цифровим підписом і перевірити його.
* 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 .

**Хід роботи**:

**Частина 1**

Пишемо функцію, пошуку випадкового простого числа заданої довжини з використанням генератора псевдовипадкових чисел та тестів Міллера-Рабіна. У функції пошуку випадкового простого числа ми використовували постулат Бертрана, який пришвидшує пошук.

**Частина 2**

За допомогою цієї функції генеруємо дві пари простих чисел та довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб , де - прості числа для побудови ключів абонента А, - абонента В.

**Частина 3**

Пишемо функцію генерації ключових пар для RSA. Після генерування функція повертає секретний ключ (d, p, q) та відкритий ключ (e, n).

**Частина 4**

Реалізуємо функції шифрування, розшифрування та створення повідомлення з цифровим підписом. На вхід ці функції приймають лише ті змінні, які необхідні.

**Перевірка**

Перевірка викликається у функції check().

modulus = "8924ADDA6B60D6B731404DEE5E431A38FAC4394EB313614AB6C834A88A2009E5E87D03138F27B28FF9BFFA69CC06D3D2EC2C513375F2725BB978C55463C95EBD"

exponent = "10001"

n = 8923656186321824081488135504643688162548557149446412287814673826161489761208234503251945575864814519792032321401499497921783851153183411834517642465402893

e = 2823530029678610103498835374423876674948499549170405759747701428808961205060874889721846963083436577371845653879481281075390143374156453411628257844977759

d = 4961441460831825012373226525627678025794513010830758327806336480230027620606576128573641064597210030770586440798368714061641497820932899289200128431851999

'''n, e, d = generate\_keys()

while n < hex\_to\_int(modulus):

n, e, d = generate\_keys()'''

print(f"dec n = {n}")

print(f"dec e = {e}")

print(f"dec d = {d}")

print(f"hex n = {int\_to\_hex(n)}")

print(f"hex e = {int\_to\_hex(e)}")

print(f"hex d = {int\_to\_hex(d)}")

m = encode("some message")

def part1():

#генерируем ключи на сервере, шифруем у себя, расшифровуем на сервере

print("\nCheck part1")

c = encrypt(m, hex\_to\_int(exponent), hex\_to\_int(modulus))

print(f"hex cipher = {int\_to\_hex(c)}")

def part2():

#генерируем ключи у себя, шифруем на сервере, расшифровуем у себя

print("\nCheck part2")

c= "1A82B9887A65073DF4E09FB4D84085C4FECFE185A7965CA7DB6E941E654D1BF33BA10BCF165F6C64493BB6C89675F41A7658349B262A30C2D61CEB4A8F5BDF03"

dm = decrypt(hex\_to\_int(c), d, n)

print(f"decrypted message = {decode(dm)}")

def part3():

#генерируем цифровую подпись у себя, проверяем на сервере

print("\nCheck part3")

s = sign(m, d, n)

print(f"hex signature = {int\_to\_hex(s)}")

def part4():

#генерируем цифровую подпись на сервере, проверяем у себя

print("\nCheck part4")

s = "2C205280A13C9E4F36356DF777163E41527EA8DF0D270BFFAA62B4C82102DAA10C30D019FC9808AC806F2007242030CD2C2BD92B00C286940E222E8A3907952C"

print(verify(m, hex\_to\_int(s), hex\_to\_int(exponent), hex\_to\_int(modulus)))

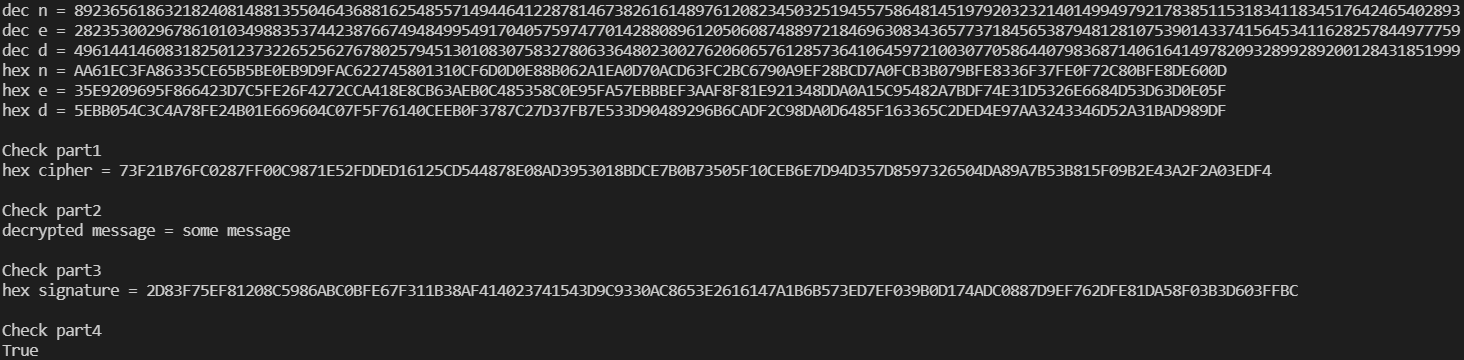
part1()

part2()

part3()

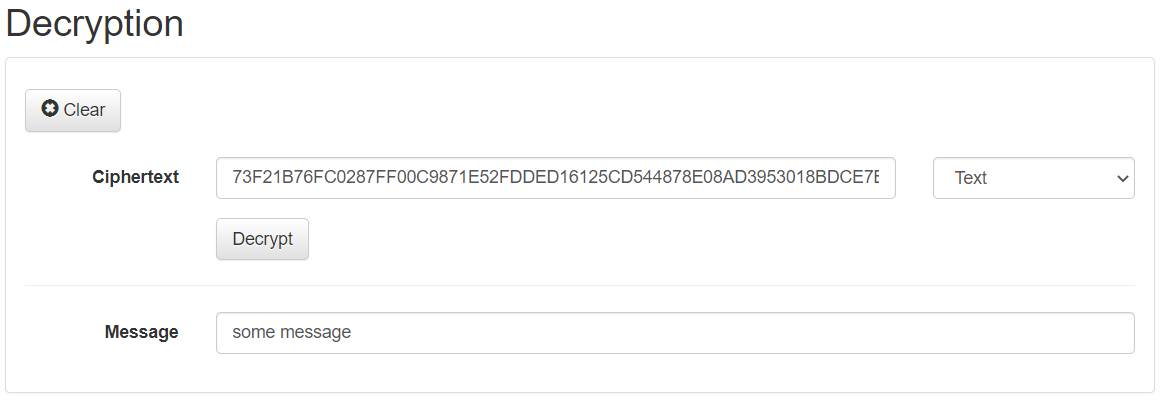
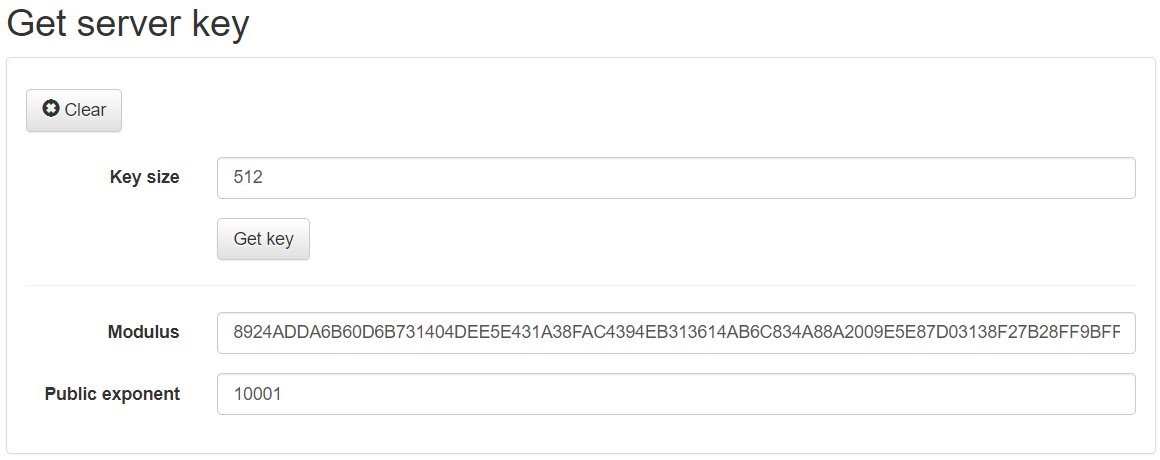
part4()

Результат виводу наведено нижче.



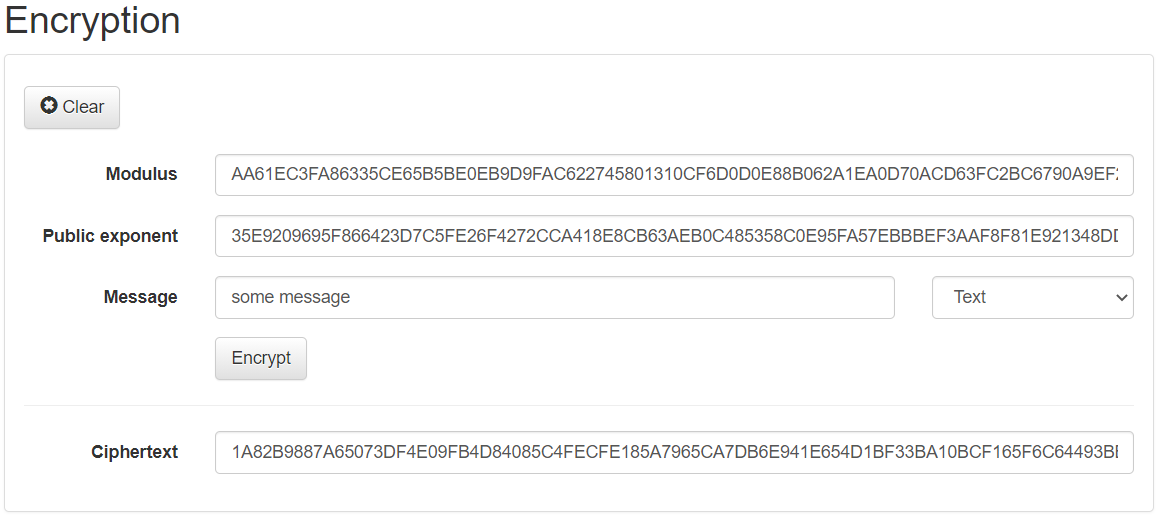
**Частина 1**

Генеруємо ключі на сервері, шифруємо у себе, розшифровуємо на сервері.



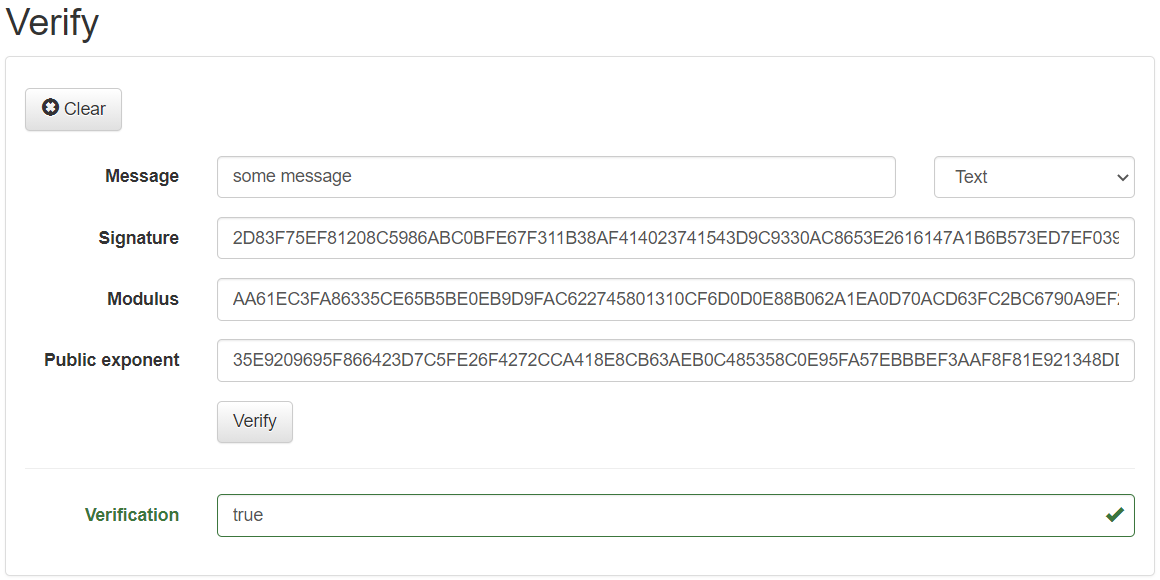
**Частина 2**

Генеруємо ключі у себе, шифруємо на сервері, розшифровуємо у себе.



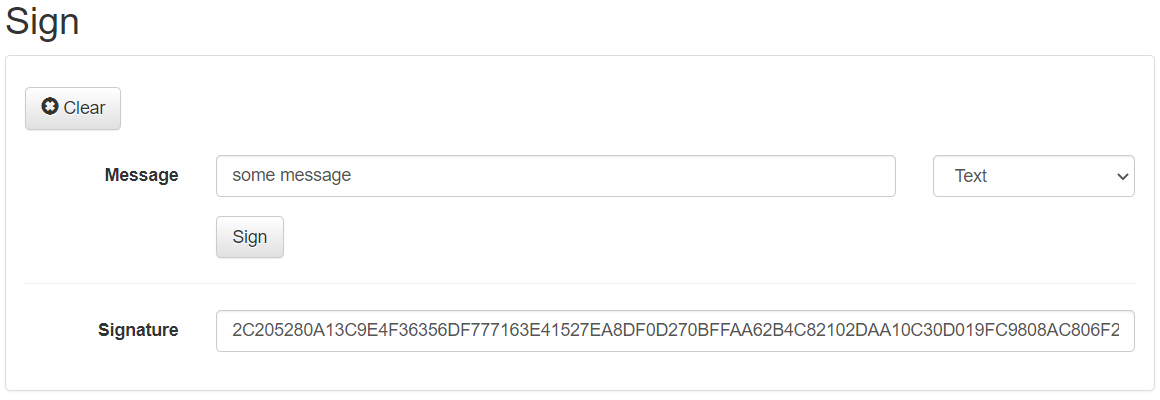
**Частина 3**

Генеруємо цифровий підпис у себе, перевіряємо на сервері.



**Частина 4**

Генеруємо цифровий підпис на сервері, перевіряємо у себе.



**Демонстрація**

Демонстрація викликається у функції demo().

sender = Abonent()

receiver = Abonent()

receiver.generate\_key\_receiver()

sender.generate\_key\_sender(receiver.n)

print("sender:")

sender.print\_keys\_dec()

sender.print\_keys\_hex()

print("receiver")

receiver.print\_keys\_dec()

receiver.print\_keys\_hex()

msg = encode("hello world")

cipher = sender.encrypt(msg, receiver.e, receiver.n)

print(decode(receiver.decrypt(cipher)))

k = encode("hello this is a private message")

ks = sender.send\_key(k, receiver.e, receiver.n)

res = receiver.receive\_key(ks, sender.e, sender.n)

print(decode(res))

Результат виводу наведено нижче.



**Висновки**: Під час виконання комп’ютерного практикуму ми ознайомилися з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA та практично ознайомилися з системою захисту інформації на основі криптосхеми RSA, організації з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчили протокол розсилання ключів.