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

Криптографія

Комп’ютерний практикум №4

Підготували:
студенти групи ФБ-02
Єсаф'єв Євгеній
Сапегін Валентин

Київ – 2022

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.

Постановка задачі:

- 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 p, q і p1, q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p1q1$; p і q – прості числа для побудови ключів абонента A, p1, q1 – абонента B.
- 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) , (e1, n1) та секретні d і d1 .
- 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.
- 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 $0 < k < n$.

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

Хід роботи:

Генеруємо p, q, p1, q1 (при чому pq <= p1q1):

```
q: 753775827103931247002107207576287265237041179940946462381959832027572957363 71
p: 736636776313784359417666174751443228483139031243451138827455433887359291773 89
q1: 847445056780313824204523585366786249781861593778621238371564069861082887713 11
p1: 823187793258394887470964588049556446834901898159897350388078096486740729201 69
```

Для юзера Lil та Big:

```
Generated message: 1072612107

Encrypted message from Lil to Big:
4935889147305667527866495121051670451462661847894969378350261214865957065492974611150625978238015733431450452205796743194338636473515406189078313 7
Signature:
4187098019499311638462909263566418270640442610112832928827232835805209319986446822128235199574649426072460615756233038680341983044840924727031696 4


Big decrypted message: 1072612107
Signature verify: True
```

Пересилання:

```
Generated key: 1050099265

Success
```

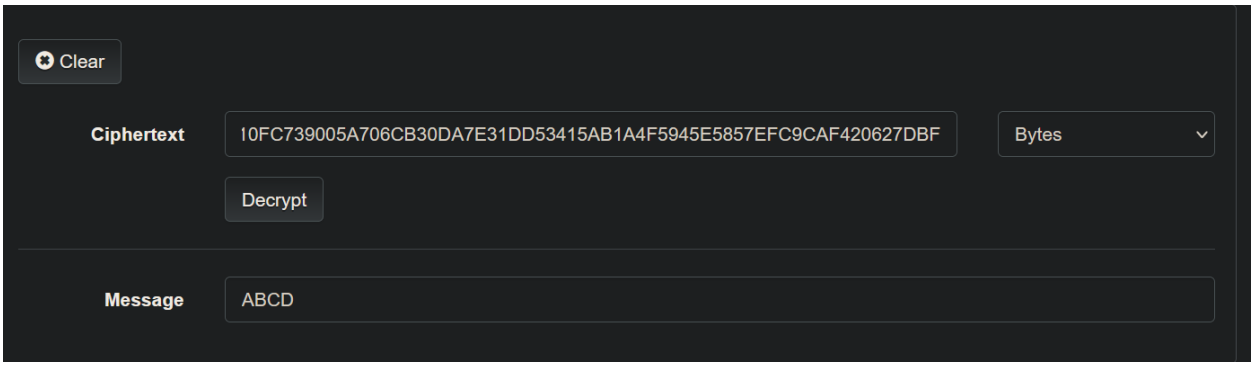
Все працює, переконаємось в цьому перевіркою на сайті:
Зашифровуємо нашим кодом:

```
    # Server modulus here
    n_hex = 'A5F15327E0D32F463F7E98381FC735B66BE14229F9A86A5D0D0E204DB2AA04AD'
    e_hex = '10001'

    mes = 'ABCD'

    me.hex_encrypt(mes, e_hex, n_hex)
```

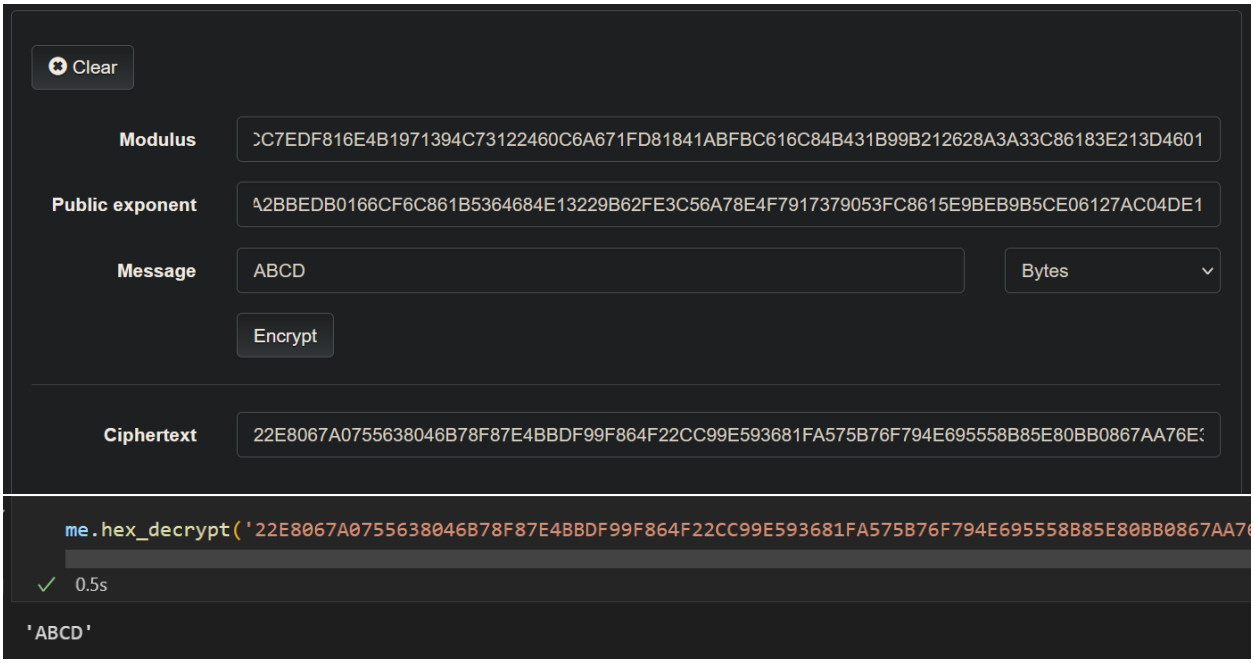
✓ 0.8s

```
'1710FC739005A706CB30DA7E31DD53415AB1A4F5945E5857EFC9CAF420627DBF'
```

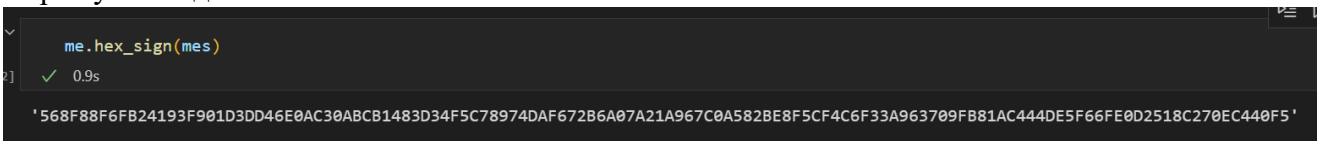
Розшифровуємо на сайті:



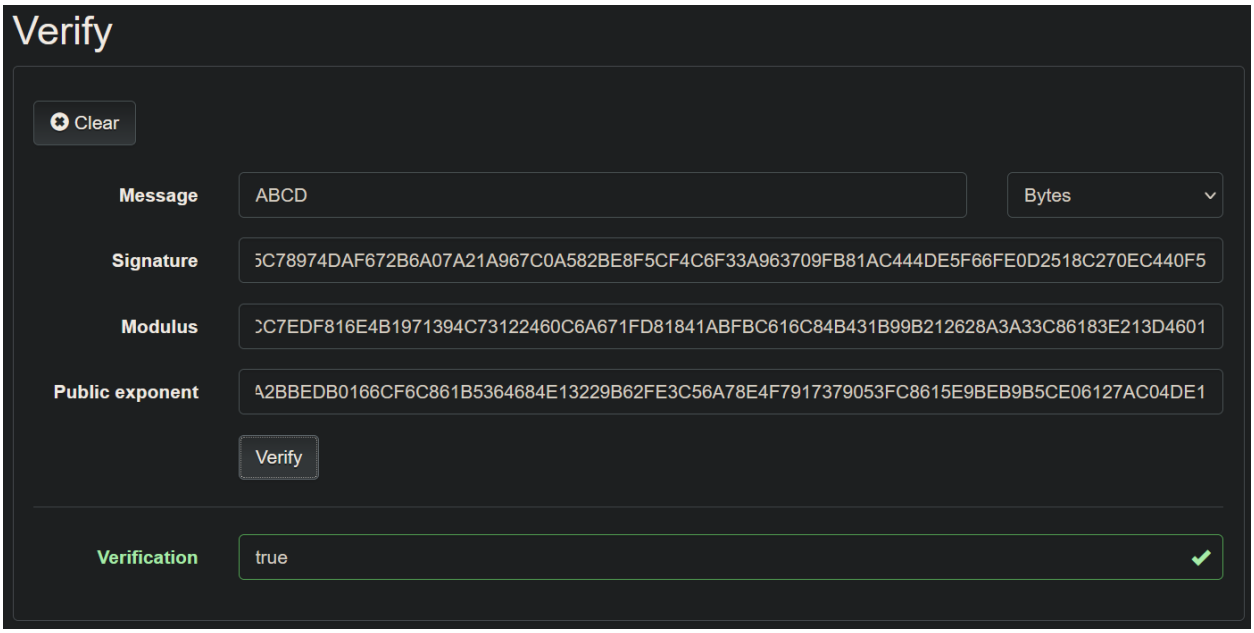
Робимо навпаки:



Отримуємо підпис:



Та перевіряємо:



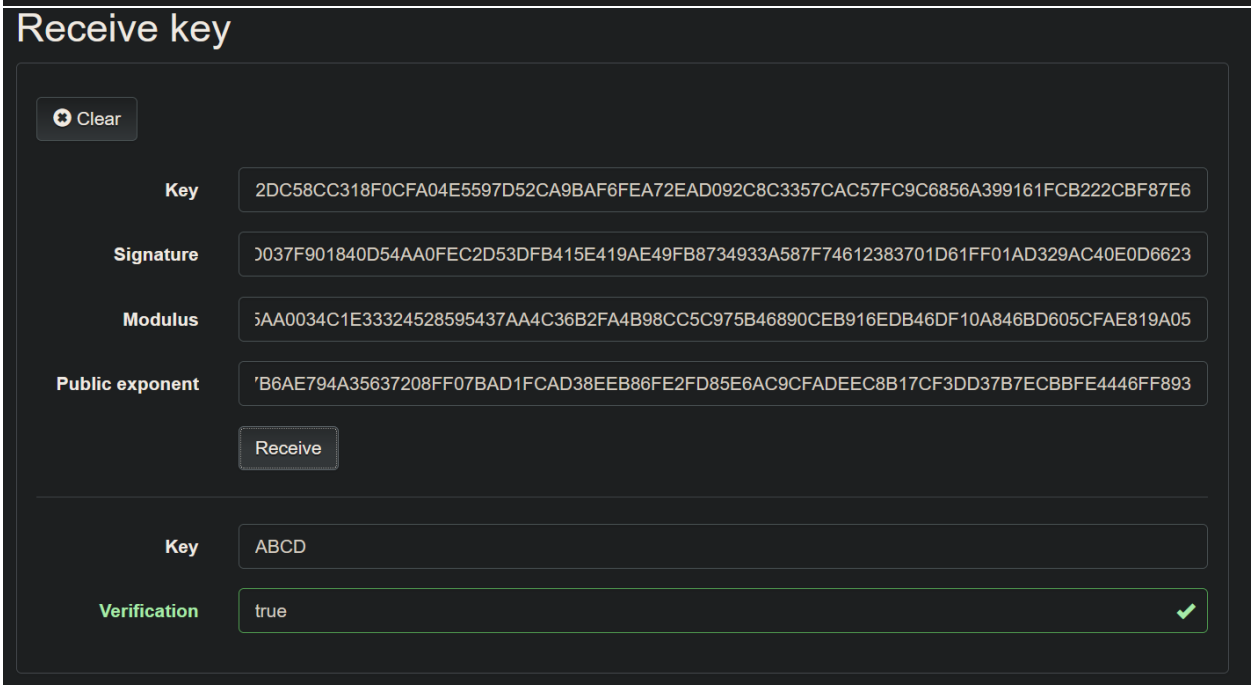
Та навпаки:

Отримуємо згенерований з код ключ на сайті:

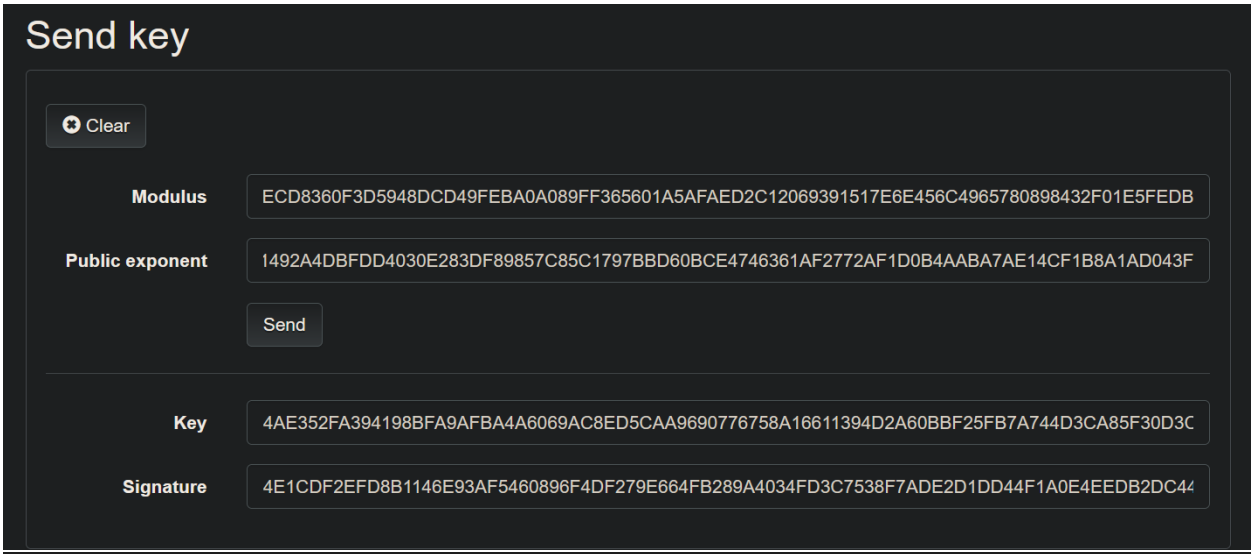
```
    # Server modulus here
    n_hex = '839439A8E3FC1E38EDF9EA6E3E80C246A955202BF4450800B08EE6A83856CD56157EBF04D941AF421B6F6CBA5FF4C8D88AFF5C6C10C5E5C74E39A39A4A9EE
    e_hex = '10001'

    key = 'ABCD'

    me.hex_regenerate_keys(n_hex, range_min, range_max)
    me.print_public()
    me.hex_send_key(key, e_hex, n_hex)
```

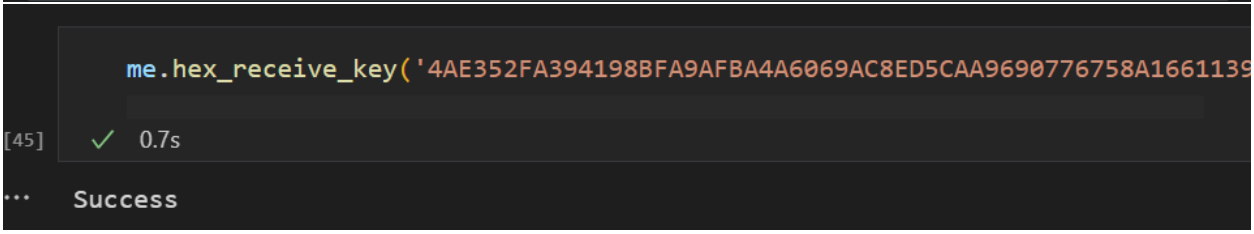
✓ 0.2s

```
Modulus:
69B6B7102D96F1EBAF1FF91C4A93D1610EB9A6D8F785AA0034C1E33324528595437AA4C36B2FA4B98CC5C975B46890CEB916EDB46DF10A846BD605CFAE819A05

E:
3E9DF0E16111DDA7E9EA595A90F74D744E383B55971A87B6AE794A35637208FF07BAD1FCAD38EEB86FE2FD85E6AC9CFADEEC8B17CF3DD37B7ECBBFE4446FF893

('7EAC94FFE663B486EB48FE200954847BF194849F6A6142DC58CC318F0CFA04E5597D52CA9BAF6FEA72EAD092C8C3357CAC57FC9C6856A399161FCB222CBF87E6',
 '21898545AA1BBB49D2D53357616AB87F17A8F75112D037F901840D54AA0FEC2D53DFB415E419AE49FB8734933A587F74612383701D61FF01AD329AC40E0D6623')
```



І навпаки:





Все збігається.

Висновок:

Під час виконання практикума ми розібрали перевірку простоти чисел за допомогою тесту Міллера-Рабіна. Реалізували принцип роботи криптосистеми типу RSA, навчились шифрувати, розшифровувати, використовувати цифрові підписи, яки генерували з використанням даної системи, організували захищений зв’язок.