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

Фізико-технічний інститут

### КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

Виконав: Маслюк В.О. ФБ-25

**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.

### Хід 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 i p1 q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq  $\mathfrak L$  p1q1; p i q прості числа для побудови ключів абонента A, p1 i q1 абонента B.

### Перевірка:

```
print(miller_rabin_test(7))
print(miller_rabin_test(10))
print(generate_prime_pair(256))

a = generate_prime_pair(256)
for i in a:
    print(i)
```

#### True

False

105126369585780747889117880311015803794117943355871915848105204284858108154283 80814331234829204505183036521717853929013580541449192486606709489393475599233 73301290402714979432872716254409627398514506144113351370461118155524538557631 73159555554228835361309872293408161505615986828897924202395053501814048303477 104130249885491664045800963685671483389587097696555116473512251663791189964311

Функція перевірки визначає, чи є число простим.

- 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 A і B тобто, створити та зберегти для подальшого використання відкриті ключі (e,n), (e1, n1) та секретні d i d1.
- 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 и 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.

### Шифрування:

```
Шифрування
Повідомлення для В:
363
Зона
Зашифрований текст від А для В:
1495366559122381983966863763487686796254469283577672505663848552472023714610316639447783033954225982487839481986223152658112694922553344979658720757252620
Повідомлення для А:
222
Зашифрований текст від В для А:
1771437772991886006047624588200557994699394939192342735276849520282801042255008323013703831570661122678937421385774195260510301330134346431850142950989802
```

#### Розшифрування:

```
Розшифрування
Розшифрований текст А від В:
222
Оригінальне повідомлення від В:
222
Розшифрований текст В від А:
363
Оригінальне повідомлення від А:
363
```

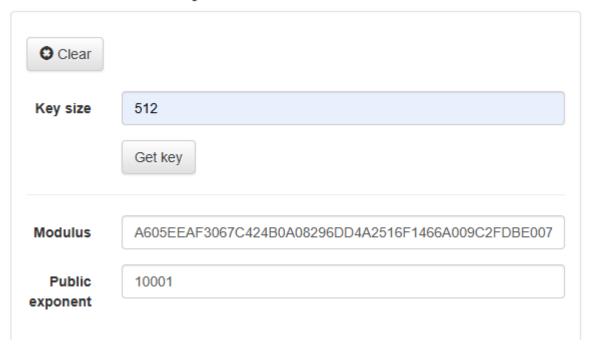
#### Обмін ключами:

```
Обмін ключами
А генерує k1 та S1
k1: 3834797388024111638996757271828738737050289114258190062252987709183788487961851634012952061291737791359036405367990928300856516639868276945775310150106870
S1: 712344076180648296875705674823350929837866693487494123365058515616666550325196737121900296859982078748104147594083254951139298529331981018893589288977661
А надкляає повідомлення (k1, S1) до В. В отримав повідомлення
В перевіряє підпис
Перевірено? True

В генерує k2 та S2
k2: 5315294053906389434204690938207410930271510152919838481094369213007214362073694202704018998986546911690579709336864315220103600549501633600058880982438707
S2: 1631380132565712771375592424311096794138980519336036595424394710492980137146481153432005630700465723372869772286972822428031114104263180025419979321524781
В надкляає повідомлення (k2, S2) до А. А отримав повідомлення
А перевіряє підпис
Перевірено? Тrue
```

Згідно з методичкою, розмір ключа повинен бути як мінімум 256 біт. У коді ми генеруємо прості числа р та q розміром 256 біт кожне, що дає модуль п розміром близько 511-512 біт. Тому в полі Кеу size на сервері слід вводити значення 512.

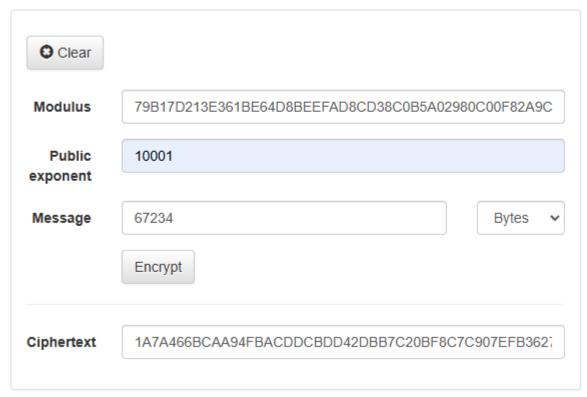
### Get server key





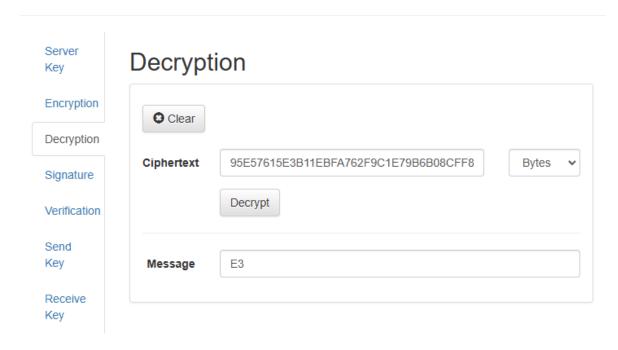
## Encryption

Повідомлення від сервера:

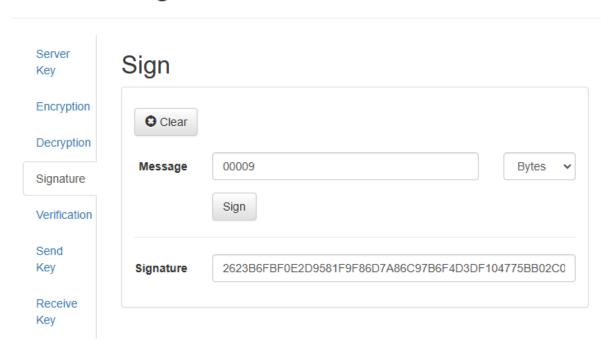


### Public\_A: e: 10001 n: 79B17D213E361BE64D8BEEFAD8CD38C0B5A02980C00F82A9CD0CBD35B9894E7CF4F621E2259E14058BC26B490BAA165C5C3 79785EFDB308480B08C8AA84F7AA1 Secret\_A: d: 2490328750755291618206276869301151941540174381469346221622428693511295369751823280138925075925936329 809670362807849143002678896382805883788265703186484637 Введіть експоненту сервера: 10001 Введіть модуль сервера: A605EEAF3067C424B0A08296DD4A2516F1466A009C2FDBE00721145DB6A1E7FE52C8AEC6500A2D6E C7CB771C3477753578FF864F44530FB15BA624AC2836520F Зашифроване повідомлення від сервера: 1A7A466BCAA94FBACDDCBDD42DBB7C20BF8C7C907EFB3627379278ADF2B56380A7 DF62599EC8A7BC1B9AE602B8A63BA395456AA5F61070FA86C67AFE985CA09F Розшифроване повідомлення А: 67234 Повідомлення для сервера: Зашифрований текст від А для сервера: 95E57615E3B11EBFA762F9C1E79B6B08CFF88E898BE8A1D0AF39235E9B8868269D5A34A59A181268572CE007EDDD6CFED82BF49 D8B7B0136EABEE9FF22363834 Використання функції перевірки

# **RSA Testing Environment**



# **RSA Testing Environment**

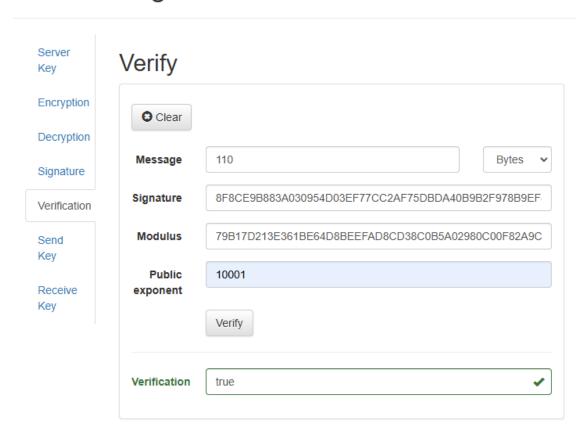


Використання функції перевірки
Повідомлення від сервера: 00009
Підпис від сервера: 262386FBF0E2D9581F9F86D7A86C97B6F4D3DF104775BB02C029A87B3308889947EDCEBC6A012261DF2E
10347DCB470B99C6B3C2E72DD2610EA73601855F8C80
Перевірка? True

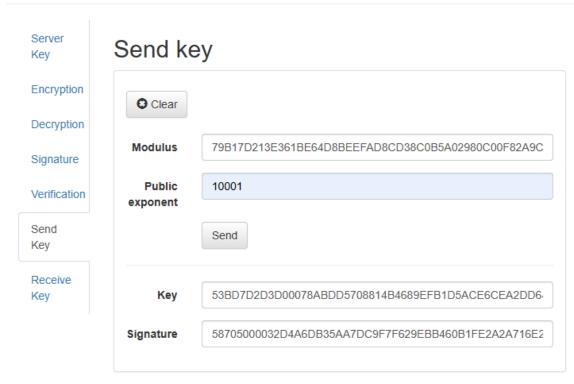
Використання функції підпису
к: 110
SA: 8F8CE9B883A030954D03EF77CC2AF75DBDA40B9B2F978B9EF467C1823DD0678ED937DAC417B9537256EA21C422E43D3BF8F
99F9940F8175A664EC92962F5875
Модуль: 79B17D213E361BE64D8BEEFAD8CD38C0B5A02980C00F82A9CD0CBD35B9894E7CF4F621E2259E14058BC26B490BAA165
C5C379785EFDB308480B08C8AA84F7AA1
Показник: 10001

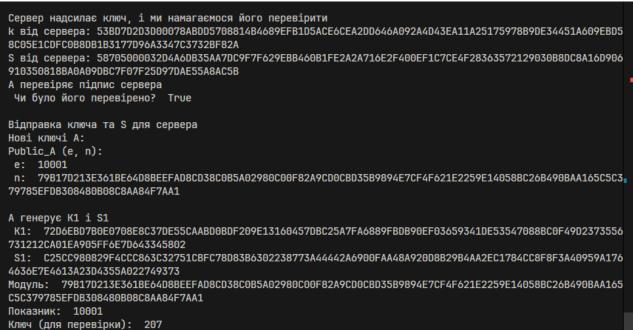
Сервер надсилає ключ, і ми намагаємося його перевірити
к від сервера:

## **RSA Testing Environment**

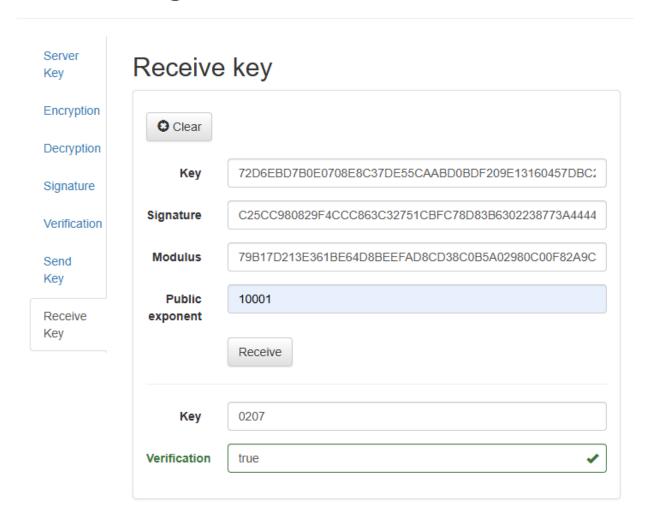


### **RSA Testing Environment**





# **RSA Testing Environment**



**Висновок**: Під час виконання лабораторної роботи я ознайомився з тестами перевірки чисел на простоту та методами генерації ключів для асиметричної криптосистеми RSA. Я також практично вивчив систему захисту інформації на основі криптосхеми RSA, організував схеми засекреченого зв'язку та електронного підпису з використанням цієї системи, вивчив протокол розсилки ключів та зміцнив навички, отримані у попередніх лабораторних роботах.