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
Приходько Андрій
Шахова Катерина

Київ – 2021

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 p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \le p_1 q_1$; p і q – прості числа для побудови ключів абонента A, p_1 і q_1 – абонента B.

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), (e_1, n_1) та секретні d і d_1.

4. 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.

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().

Виконання :

Згенерували p, q, p1, q1 :

p	19967815403168467153943041671329090616699958539135548704435086221568526834487 9
q	20517410021067320520005877961810001827918314886902412237950210905021646700862 3
p1	22625215338101326070247918131991793457616846140755924340922162079806387276709 9
q1	18691417246373479939802931796263344408970579114613675098650494574309424032543 9

Кандидати, що не пройшли перевірку зберігаються у файлі not_passed_test.txt

Далі за допомогою цих чисел змогли знегерувати публічні та приватні ключі для так званих Аліси та Боба :

Аліса	
e	31811810397541411518390776294835057193314687426271862377448009668698747444709905182745849045723700219230766301737381579326438243281828096105520784648772369
n	40968785585179110685184299376332342832605494307488528671728507393914766727141639329418216226852424329933127753248984922917069362009463494923614991430891617
d	71957404807263339412578133436097567584765200658101185509756286955677167733840831322908733733142337001761931099545631462161463393864045095880013163352497 45
p	19967815403168467153943041671329090616699958539135548704435086221568526834487 9
q	20517410021067320520005877961810001827918314886902412237950210905021646700862 3

Боб	
e	13437496771523038521314953327205306261853919451770944010713823 61635266673526050706642638391337053264766868458323636862212275 82211491145794945871686054413 55
n	42289734017350091106740379689670948367559293508426883343057712 11104435137799270227171673604013086794484751058302247689527643 36132031388607802370992119314 61
d	33386278783561399406060190743965534352984733098170016997745157 82148608763060739087225647645161314427622211994332101825539646 03144185063145787626394422570 07
p1	22625215338101326070247918131991793457616846140755924340922162 0798063872767099
q1	18691417246373479939802931796263344408970579114613675098650494 5743094240325439

Далі створимо повідомлення 12345678901234567890

Та зашифруємо його публічним ключем Аліси

```
# частина для Аліси
encrypted_alice = encrypt(msg, alice_public)
print("Encrypted message by Alice:", encrypted_alice)
decrypted_alice = decrypt(encrypted_alice, alice_private)
print("Decrypted message by Alice:", decrypted_alice)
```

encrypted	34809127595258564360052074283669617365666364615674580122629562131457791124798581492901324261639596537012831131738274105259646412131230237743403814828437 43
decrypted	12345678901234567890

Також Аліса підписала це повідомлення :

```
alice_sign = sign(msg, alice_private)
print("Signature for Alice:", alice_sign)
print("Sign check: ", verify(msg, alice_sign, alice_public))
print("\n")
```

Alice signature	33704223421761149896047225290754086219858531711683438981494191898442750097269380688795467421458213059284350870889194743528024130562444744168658895313904082

```
Signature for Alice:
 337042234217611498960472252907540862198585317116834389814941918984427 5
 009726938068879546742145821305928435087088919474352802413056244474416 8
 658895313904082
Sign check:  True
```

Зробимо таке саме для Боба :

```python
# частина Боба
encrypted_bob = encrypt(msg, bob_public)
print("Encrypted message by Bob:", encrypted_bob)
decrypted_bob = decrypt(encrypted_bob, bob_private)
print("Decrypted message by Bob:", decrypted_bob)
```

encrypted	3546139497461550635964916833858247923579028902177941773564580251507195995939549404053086486829169989560051514910142784369454239053408853624235697528008843 7
decrypted	12345678901234567890

```python
bob_sign = sign(msg, bob_private)
print("Signature for Bob:", bob_sign)
print("Sign check: ", verify(msg, bob_sign, bob_public))
print("\n")
```

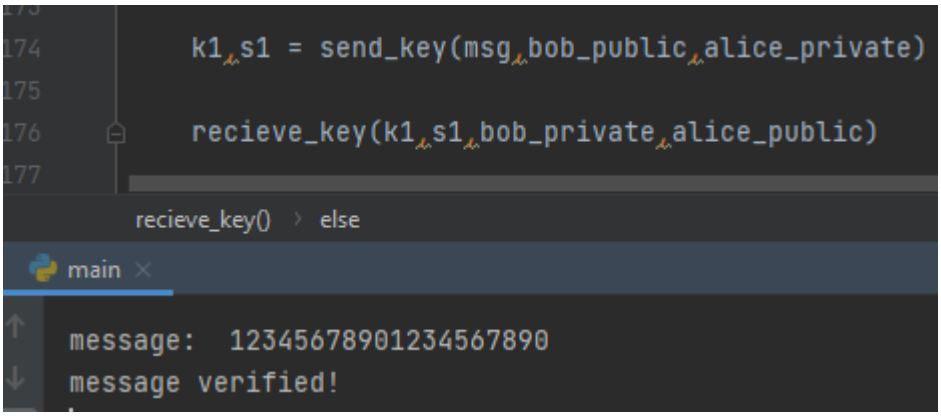
Bob signature	3740488175318051803207255442432789465615260966259862798375343125331512 5343125331512305439310091736428813148411061632944965217847997117363123368665678786571940042667092 6

```
Signature for Bob:
 3740488175318051803207255442432789465615260966259862798375343125331512
 3054393100917364288131484110616329449652178479971173631233686656787865
 719400426670926
Sign check:  True
```

Протокол конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника

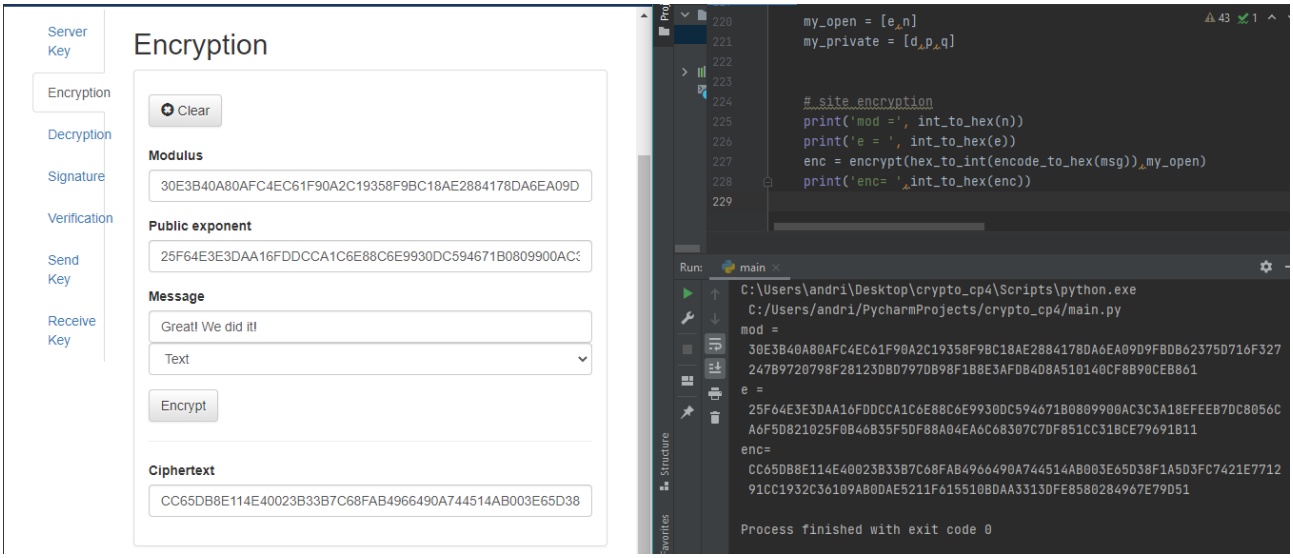
Нехай Аліса має свої ключі та відкритий ключ Боба. Тоді вона формує повідомлення (k1,s1) та відправляє його Бобу.

В свою чергу за допомогою приватного ключа Боб розшифровує повідомлення та може перевірити підпис Аліси за її публічний ключем.

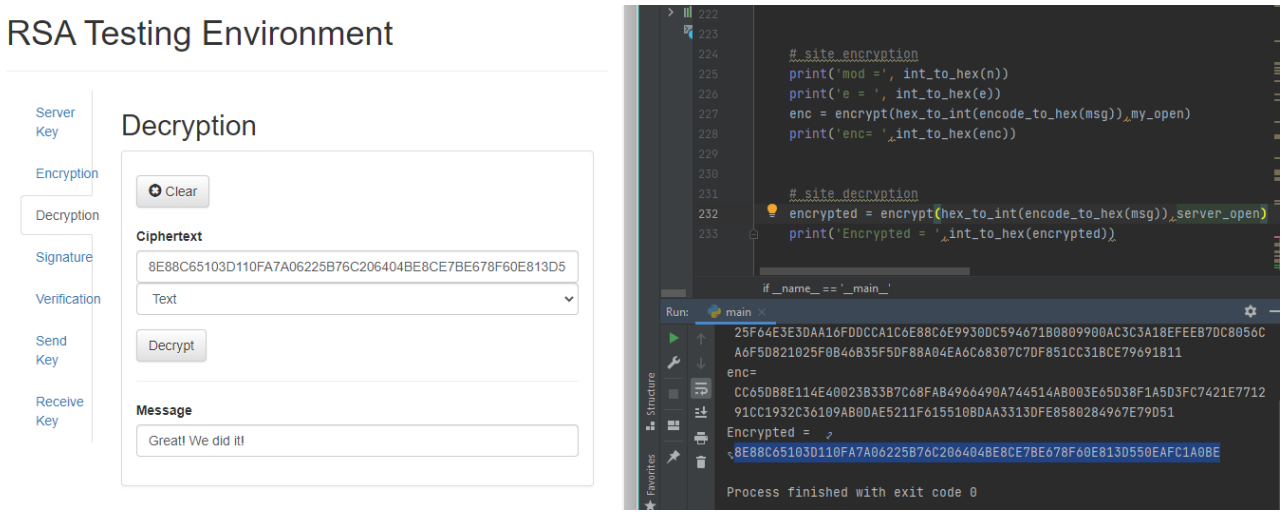


Перевірка з сайтом

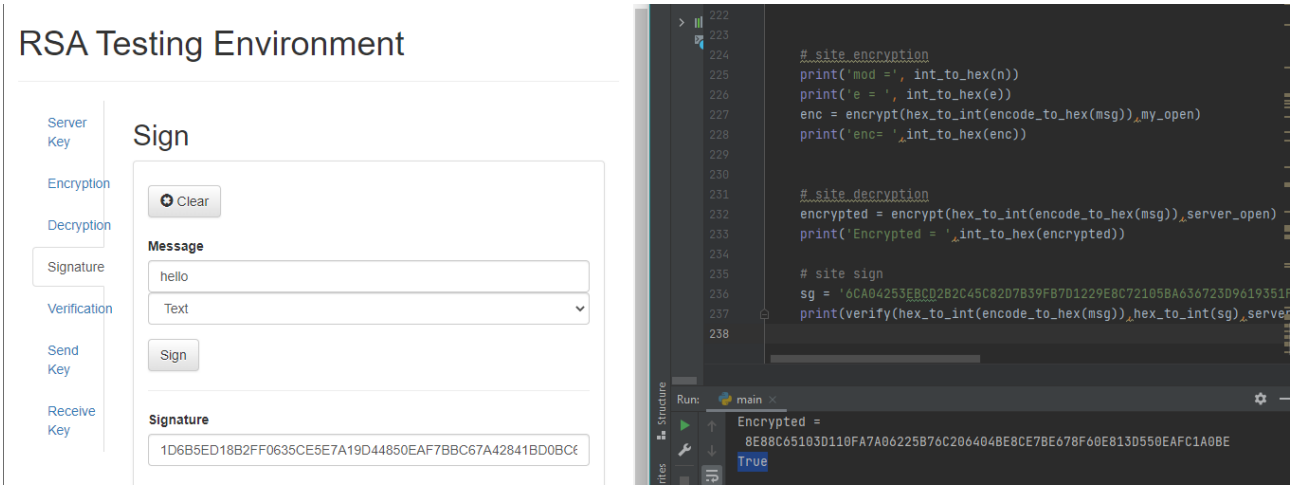
1) Згенерували ключі у себе. Закинули на сайт. Шифроване повідомлення однакове



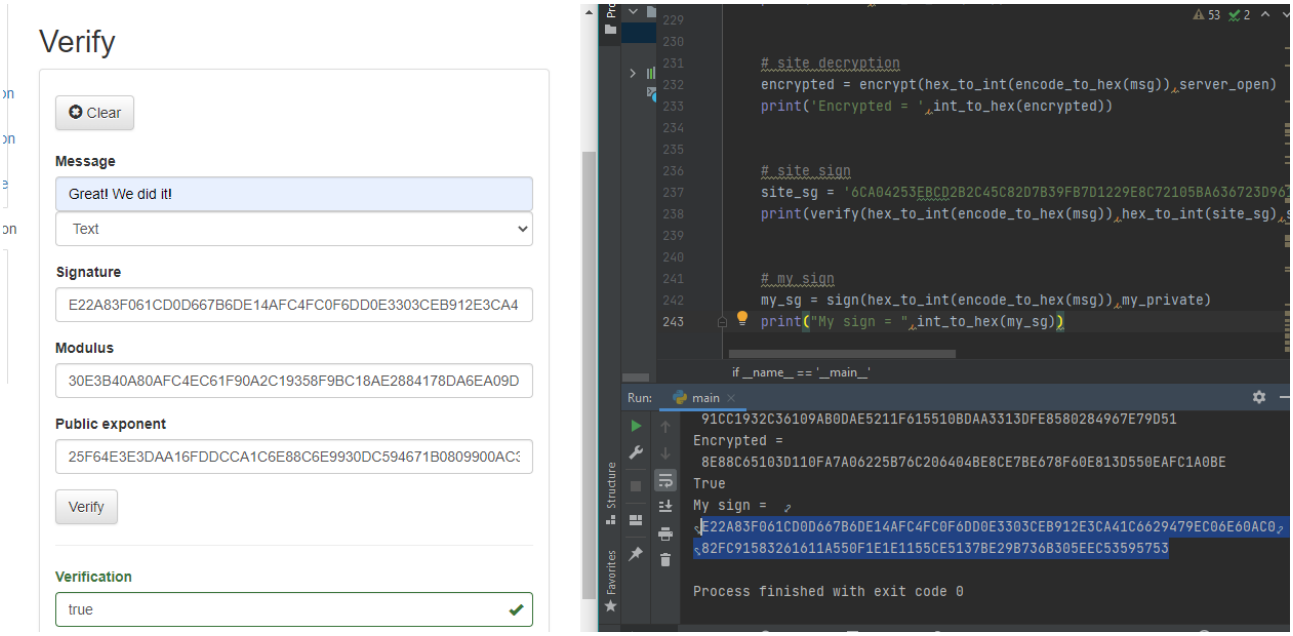
2) Зашифрували у себе ключем з серверу. Розшифрували на ньому



3) Підписали на сервері. Перевірили у себе



4) Підписали у себе. Перевірили на сервері



Висновок: виконуючи дану ЛР ми дізналися про принцип роботи асиметричної криптографії на прикладі алгоритму RSA. Ми змогли зробити функцію перевірки чисел на простоту та за допомогою цього могли генерувати великі прості числа для ключів. Дізналися про цифрові підписи RSA та протокол розсилання ключів.