Комп’ютерний практикум №4

Вивчення криптосистеми RSA та алгоритму електронного підпису, ознайомлення з методами генерації параметрів для асиметричних криптосистем

Роботу виконали:

Біла Анастасія і Лета Яна,

студенти 3 курсу ФТІ НТУУ «КПІ»,

спеціальність «Кібербезпека», група ФБ-02

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:

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 використовувати вбудований генератор псевдовипадкових чисел мови програмування Python. В якості тесту перевірки на простоту використати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч.

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 $pq<=p_1q_1$; 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 , n) та секретні d і d .

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().

- – значення вибраних чисел p, q, p_1 , q_1 із зазначенням кандидатів, що не пройшли
  тест перевірки простоти, і параметрів криптосистеми RSA для абонентів A і B;

- – чисельні значення прикладів ВТ, ШТ, цифрового підпису для A і B;

- – опис кроків протоколу конфіденційного розсилання ключів з підтвердженням
  справжності, чисельні значення характеристик на кожному кроці;

- – висновки.

Хід роботи:

1.Реалізуємо функцію ***test(p)*** для перевірки на простоту числа, користуючись тестом Міллера-Рабіна із попередніми пробними діленнями із методичних вказівок.

2.Реалізуємо функцію ***key_combination()***, що генерує дві пари простих чисел p, q і p_1 , q_1 довжини 256 біт при умові, щоб $pq<=p_1q_1$.

```
Combination for A:
p:  69870608079018661601180799246745312521562105437529825401318468267276894639211
q:  67553771274572988924531841453353976879984408352672647655010173602171063373843

Combination for B:
p1:  82357176973860233037153705037657086267562250834061897690283334577624425056349
q1:  11542646595301312762698909088788340194531524710937620744152783801357341826080 3
```

3.Реалізуємо функцію ***key_combination_rsa(p, q)***, за допомогою якої обчислюємо відкритий і секретний ключі: (n, e) та (d, p, q).

A's keys:

d=39955908213880096524532977414362819396019624021696774908645718090985006628748270761920112343093829095341845925239439868577239546830393386937797959646370 27

p=6987060807901866160118079924674531252156210543752982540131846826727689463921 1

q=675537712745729889245318414533539768799844083526726476550101736021710633738 43

n=472002307698535826876335701626939643795361899523927151999469383289200744062735795654308789714983745550558069751767290348150851359236558617928547389955787 3

e=596958022316664084558435054822137830537842589959146103708035209869712756652830579169178496959762115932425343479340310478628077420218258622261109943796243

B's keys:

d1=642586829902436771525273236358380401750119407317836611929543826788159801242020829734225076247248845326444361065498170322052648758804610720076075003037127 7

p1=8235717697386023303715370503765708626756225083406189769028333457762442505634 9

q1=115426465953013127626989090887883401945315247109376207441527838013573418260803

n1=950619788395955491394285530076916073295433194042226342563098797041990592623915198513213005117960119961150248653565333941066580134071641960968145885298824 7

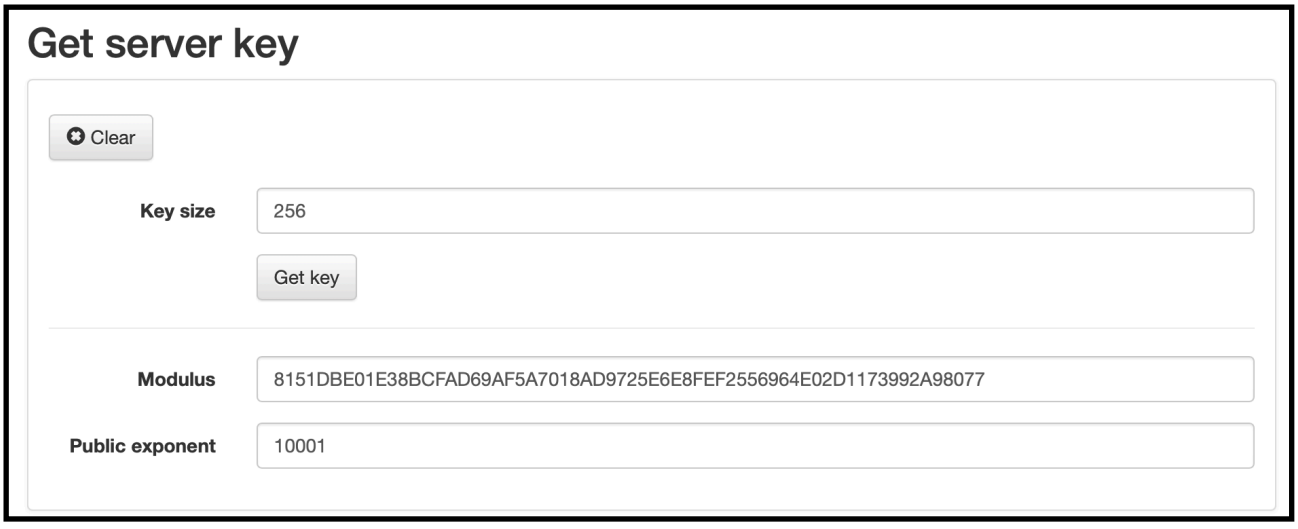
e1=720798306012652698478587788317297630059225490836587790723658591914171624564374340183821805642220991174363184008740982541972933698007710667528758706712 3877

4.Далі реалізуємо функції шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B; організуємо роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA.

Отримаємо такі результати з перевіркою правильності зашифрування і розшифрування повідомлення та генерації ключа k:

```
Combination for A:
p:  6987060807901866160118079924674531252156210543752982540131846826727689463921 1
q:  6755377127457298892453184145335397687998440835267264765501017360217106337384 3

Combination for B:
p1:  8235717697386023303715370503765708626756225083406189769028333457762442505634 9
q1:  1154264659530131276269890908878834019453152471093762074415278380135734182608 03

A's keys:
d =  3995590821388009652453297741436281939601962402169677490864571809098500662874827076192011234309382909534184592523943986857723954683039338693779795964637027
p =  6987060807901866160118079924674531252156210543752982540131846826727689463921 1
q =  6755377127457298892453184145335397687998440835267264765501017360217106337384 3
n =  4720023076985358268763357016269396437953618995239271519994693832892007440627357956543087897149837455505580697517672903481508513592365586179285473899557873
e =  5969580223166640845584350548221378305378425899591461037080352098697127566528305791691784969597621159324253434793403104786280774202182586222611099437962 43

B's keys:
d1 =  6425868299024367715252732363583804017501194073178366119295438267881598012420208297342250762472488453264443610654981703220526487588046107200760750030371277
p1 =  8235717697386023303715370503765708626756225083406189769028333457762442505634 9
q1 =  1154264659530131276269890908878834019453152471093762074415278380135734182608 03
n1 =  9506197883959554913942855300769160732954331940422263425630987970419905926239151985132130051179601199611502486535653339410665801340716419609681458852988247
e1 =  7207983060126526984785877883172976300592254908365877907236585919141716245643743401838218056422209911743631840087409825419729336980077106675287587067123877

Our message:  2784704690640402619022374552253494066605330815550767380906811954006796310183032731046058315458317933860399114350207311954094331388303365013346976268
Encrypting:  1903784994169403881274655940144812578925739350022350751319269258005853049537730230661035390166154492156245919833268626791618045800449881323957970867 0
Decrypting:  2784704690640402619022374552253494066605330815550767380906811954006796310183032731046058315458317933860399114350207311954094331388303365013346976268 7
Phi:  4720023076985358268763357016269396437953618995239271519994693832892007440627220532163734305499311742864880598228271356967718311119309257537416025941544820

Checking message: Correct.
Checking key: Correct.

Process finished with exit code 0
```

5.Перевіримо роботу функцій за допомогою сайту, розташованого за таким посиланням: http://asymcryptwebservice.appspot.com/?section=rsa .



Message: **A91**

Encryption

Clear

Modulus	8151DBE01E38BCFAD69AF5A7018AD9725E6E8FEF2556964E02D1173992A98077
Public exponent	10001
Message	A91 Bytes
	Encrypt
Ciphertext	798CA917DA51ECA0E0405FE727738C3DF944A7B21223A316DECD71945CE27C1A

```
"""Checking...."""
n_hex = '8151DBE01E38BCFAD69AF5A7018AD9725E6E8FEF2556964E02D1173992A98077'
e_hex = '10001'
check_n = int(n_hex, 16)
check_e = int(e_hex, 16)
my_message_hex = 'A91'
my_message = 2705
print(to_encrypt(my_message, check_e, check_n))
# output:
# 549783803058473690473032505724511774123638432793054693860386782807419478661 38
```

From: Decimal — To: Hexadecimal

Enter decimal number

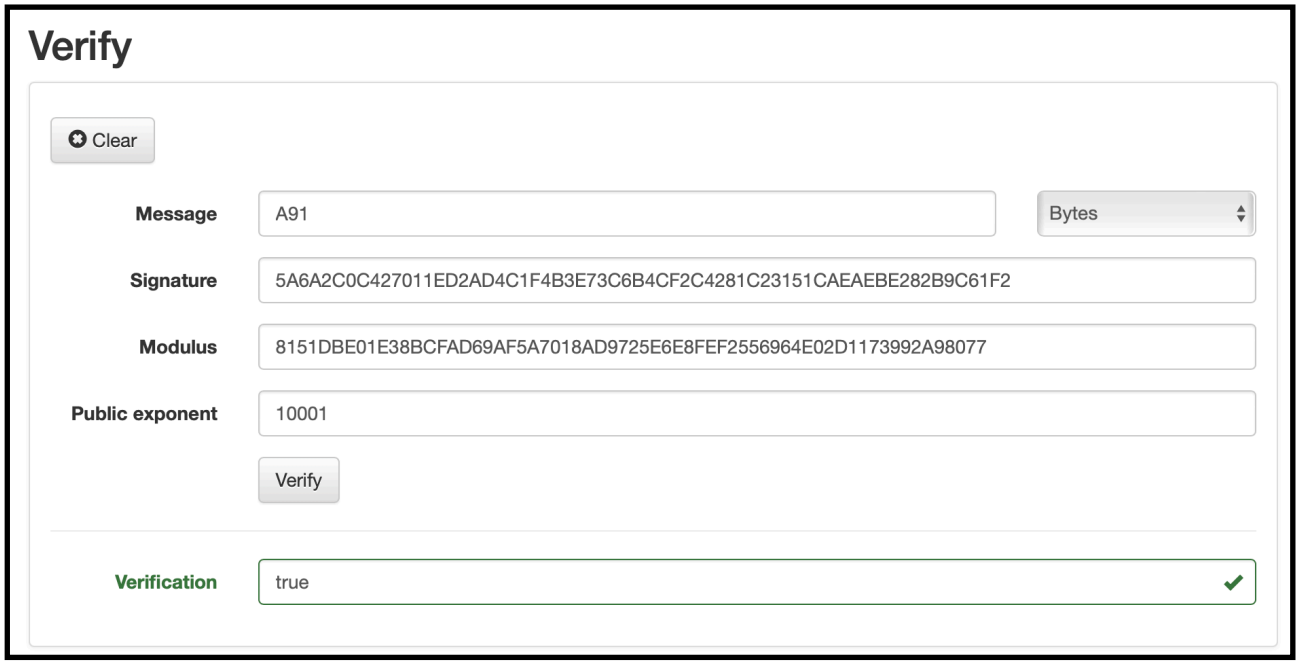
549783803058473690473032505724511 (10)

= Convert × Reset ⇅ Swap

Hex number

798CA917DA51ECA0E0405FE727738C3D
F944A7B21223A316DECD71945CE27C1A (16)

Значення співпадають, отже, функція працює вірно.



```
signed_message = int('5A6A2C0C427011ED2AD4C1F4B3E73C6B4CF2C4281C23151CAEAEBE282B9C61F2', 16)
print(to_authenticate(signed_message, check_e, check_n))
```

```
2705


Process finished with exit code 0
```

Функція повернула вірне значення **k**, тобто все працює відмінно.

Висновки:

У ході виконання комп’ютерного практикуму ми практично ознайомилися із асиметричною криптосистемою типу RSA, створивши тест перевірки чисел на простоту алгоритму Міллера-Рабіна, реалізували засекречений зв'язок й створення електронного підпису, вивчили протокол розсилання ключів, перевірили свою роботу за допомогою вказаного вище сайту.