

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Фізико-технічний інститут

КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №5

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

Варіант №11

Виконали:

ФБ-32 Пінькас Б. О.

ФБ-32 Драчук О. І.

Київ 2025

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.

Порядок 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 $pq \leq p_1q_1$; 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_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 і 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()`.

Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою

<http://asymcryptwebservice.appspot.com/?section=rsa>.

Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

Хід роботи

Функціонал криптосистеми RSA реалізовано в файлі [RSA.py](#), реалізовано взаємодію абонентів A , B через криптосистему RSA у скрипті [lab4.py](#), функції арифметики з модулем винесено в окремий файл `math_mod.py`, кожену операцію перевірено із взаємодією з тестовим середовищем в скрипті `test_site.py`, для генерації простих чисел для ключів, що використовувались в `test_site.py`, написано скрипт `gen_primes_128_256bit.py`

Спочатку, ми згенерували пари простих чисел розміру 256 біт для абонентів А і В. Генератором псевдовипадкових чисел ми будемо генерувати число розміром 256 біт, і додавати до цього числа 2, доки не натрапимо на просте число. Таким чином, ми маємо знайти 4 простих числа.

Для перевірки на простоту ми використовували тест пробних ділень, перед цим згенерувавши прості числа в діапазоні від 2 до 1000. Якщо число пройшло перевірку, тобто не поділилось ні на одне просте в нашому діапазоні, далі воно проходить 15 ітерацій тесту Міллера-Рабіна. Така кількість операцій дозволяє отримати похибку $e = 4^{-15} \approx 10^{-9}$, що надає нам можливість визначити чи є число простим з дуже високою точністю, достатньою в межах лабораторної роботи.

```
Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418295
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418297
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418299
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418301
Число не є псевдопростим за основою 52080445672404857254139589764539822766224323198678908826587275789866765191277

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418303
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418305
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418307
Число не є псевдопростим за основою 11128238920620391890086915759908788029065585693543315590662344855367187220488

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418345
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418347
Число не пройшло тест пробних ділень

Згенеровано число: 85118714922376686876690589359602778083694121122248847194791182892292958418349
Число 85118714922376686876690589359602778083694121122248847194791182892292958418349 пройшло тест Міллера-Рабіна з 15 ітераціями
Похибка: 9.3e-10
-----
```

І так 4 простих числа

```
Згенеровано число: 109489276012612362728828094736276370790343389609574255556605693378273622475021
Число не пройшло тест пробних ділень

Згенеровано число: 109489276012612362728828094736276370790343389609574255556605693378273622475023
Число 109489276012612362728828094736276370790343389609574255556605693378273622475023 пройшло тест Міллера-Рабіна з 15 ітераціями
Похибка: 9.3e-10
-----

p_A: 85118714922376686876690589359602778083694121122248847194791182892292958418349
q_A: 95825281436042647452583446918525962630133460360962565640749150060012716508367

p_B: 95295101854828751025315612106359523223115713763777408690547889358211080230707
q_B: 109489276012612362728828094736276370790343389609574255556605693378273622475023
```

Далі, за допомогою визначених двох простих, ми будуємо пару ключів pk і sk
 pk (public key) містить 2 значення: n , e
 sk (secret key) містить 3 значення: d , p , q

Ці значення визначаються з наступних формул:

$$n = pq$$

$\varphi(n) = (p - 1)(q - 1)$. Обирається випадкове число e ,
 $2 \leq e \leq \varphi(n) - 1$ таке, що $\gcd(e, \varphi(n)) = 1$

$$d: ed \equiv 1 \pmod{\varphi(n)} \Rightarrow d \equiv e^{-1} \pmod{\varphi(n)}$$

```
-----
n_A: 8156524812911029010322739549086486461233416922195583558388835639669207156223336877361514578466781776831202851109353345979532402821111329116482118144826083
e_A: 65537
d_A: 5271118534586886974902711264208919558304757723205968391114063137569768690778023456151707566836088939848840300387246738830165820890713711069301892786329065

n_B: 1043379170963335344619783219515525911491871682866041099345261661582594794411025504515481178130157531115825650331878602674514480772160420958634475185131261
e_B: 65537
d_B: 5978451047197028984426539198609137266404363419671110599548549154153372899579094857601166837854036345941714301206504903663327458996602820032954229497780545
-----
```

Далі, з використанням створених ключів, ми шифруємо і розшифровуємо повідомлення з підтвердженням цифрового підпису, підписуємо не відкрите повідомлення, а хеш $m = H(M)$:

```
-----
A send number to B
Sending message: '485761155218289551037555455729690861433047866318179098815004710058667614161097460708683339492060219216423575045407941878451013114136445746155103800749477'...
msg in int format: 485761155218289551037555455729690861433047866318179098815004710058667614161097460708683339492060219216423575045407941878451013114136445746155103800749477
Encrypted: 8387704430820395907281388848759378440680654023213907348227260878549761697205383335577107942858644611457542350667681625551700609699694472877318772088722117
Hash: 60462938933560590202979783287225093064717676233689393042949347493020136063298
Sign: 5622299977700571532069782697017322279412302019542542213086503157183108791811519091716537014598597753398962935304152697419929209373635032105167241443954205

A send text to B
Sending message: 'I'm A'...
msg in int format: 314194075713
Encrypted: 7807874261931791529074316363003132373350244556277344753085501646580069541288103515278887113085092881430998066779812625339477644492733393162159529475440138
Hash: 76807678959117770766090225879073126004192828600045539836684313845932811375757
Sign: 5566869287807407590707680544447117325252325983977095011323911874067284676458689826614971535682219870861565271758659717401625250394213262823463341519146001

B recv number from A
Decrypted: 485761155218289551037555455729690861433047866318179098815004710058667614161097460708683339492060219216423575045407941878451013114136445746155103800749477
Hash: 60462938933560590202979783287225093064717676233689393042949347493020136063298
Повідомлення і підпис співпадають
received message 485761155218289551037555455729690861433047866318179098815004710058667614161097460708683339492060219216423575045407941878451013114136445746155103800749477

nB recv text from A
Decrypted: I'm A
Hash: 76807678959117770766090225879073126004192828600045539836684313845932811375757
Повідомлення і підпис співпадають
received message I'm A
-----
```

```

B send number to A
Sending message: '4857611552182895510375554557296908614330478663181790988150047100586676141610974607086833394920602192164235750454079418784510
13114136445746155103800749477' ...
msg in int format: 485761155218289551037555455729690861433047866318179098815004710058667614161097460708683339492060219216423575045407941878451
013114136445746155103800749477
Encrypted: 83877044308203959072813888487593784406806540232139073482272608785497616972053833355771079428586446114575423506676816255517006096996
9447287731872088722117
Hash: 60462938933560590202979783287225093064717676233689393042949347493020136063298
Sign: 5622299977700571532069782697017322279412302019542542213086503157183108791811519091716537014598597753398962935304152697419929209373635032
105167241443954205

A recv number from B
Decrypted: 48576115521828955103755545572969086143304786631817909881500471005866761416109746070868333949206021921642357504540794187845101311413
6445746155103800749477
Hash: 60462938933560590202979783287225093064717676233689393042949347493020136063298
Повідомлення і підпис співпадають
received message 48576115521828955103755545572969086143304786631817909881500471005866761416109746070868333949206021921642357504540794187845101
3114136445746155103800749477

```

Для хешування $H(M)$ використано хеш-функцію sha-256.

Протокол обміну повідомленнями send/recv реалізовано наступним чином:

1. Абонент А шифрує ВТ публічним ключем абонента В.
2. Абонент А знаходить хеш ВТ і підписує хеш ВТ своїм секретним ключем
3. Абонент А пакетом надсилає ШТ, цифровий підпис, тип повідомлення (щоб абонент В знав як інтерпретувати отримане повідомлення)
4. Абонент В розпаковує отриманий пакет
5. Абонент В розшифровує ШТ своїм секретним ключем
6. Абонент В знаходить хеш розшифрованого повідомлення
7. Абонент В перевіряє цифровий підпис порівнюючи підписаний хеш з хешем розшифрованого повідомлення

Реалізовані процедури:

Зашифрування:

$C = M^e \bmod n$, де C - криптограма, M - відкрите повідомлення

Розшифрування:

$M = C^d \bmod n$.

Цифровий підпис:

$S = m^d \bmod n$, де $m = H(M)$

Перевірка цифрового підпису:

Для підтвердження цифрового підпису має виконуватись рівність:

$m = S^e \bmod n$, де $m = H(M)$

Потім, з використанням реалізованих функцій, ми організовуємо **протокол конфіденційного розсилання ключів з підтвердженням справжності**

Протокол реалізується наступним чином:

Абонент A формує повідомлення (k_1, S_1) і відправляє його B , де

$$k_1 = k^e \bmod n_1, \quad S_1 = S^e \bmod n_1, \quad S_1 = k^d \bmod n.$$

Абонент B за допомогою свого секретного ключа d_1 знаходить (конфіденційність):

$$k_1 = k^{d_1} \bmod n_1, \quad S = k_1^{d_1} \bmod n_1,$$

і за допомогою відкритого ключа e абонента A перевіряє підпис A (автентифікація):

$$k = S^e \bmod n.$$

```
A send key to B
Created key: 693558604587889260231114374398538457138133358030534948616442871911630183100657085938873212261095337397852012860378058796337536644
6039754787575356873571346
msg in int format: 693558604587889260231114374398538457138133358030534948616442871911630183100657085938873212261095337397852012860378058796337
5366446039754787575356873571346
Encrypted: 31027510886963103069814259920677502814281878510062708750908687845559793100755152287868182282950272081998017767162618593861462781595
05351857489579168534713
Hash: 113907040497693736169165512823090777261667053420650808869693321690786804114193
Sign: 143194415303530800276913781198517412689623355706889917799622492099540875523168290948932493830594152685314412854299206790113456721587963
917804527996092138
msg in int format: 14319441530353080027691378119851741268962335570688991779962249209954087552316829094893249383059415268531441285429920679011
3456721587963917804527996092138
Encrypted: 74782273227020687371127373717142296122018884687979181386260622596394006421732438968160631684399673741427803837035577454162152653928
95924483109071222712632
sender saved key: 6935586045878892602311143743985384571381333580305349486164428719116301831006570859388732122610953373978520128603780587963375
366446039754787575356873571346

Decrypted: 69355860458788926023111437439853845713813335803053494861644287191163018310065708593887321226109533739785201286037805879633753664460
39754787575356873571346
Decrypted: 1431944153035308002769137811985174126896233557068899177996224920995408755231682909489324938305941526853144128542992067901134567215
87963917804527996092138
Hash: 113907040497693736169165512823090777261667053420650808869693321690786804114193
Повідомлення і підпис співпадають
reciever recieved key 693558604587889260231114374398538457138133358030534948616442871911630183100657085938873212261095337397852012860378058796
3375366446039754787575356873571346
```

Важливий н'юанс, має виконуватись співвідношення $n_A \leq n_B$, інакше абонент A перегенерує пару ключів

```
B send key to A
Перегенерація пари ключів відправника для коректної роботи протоколу конфіденційного розсилання ключів

Згенеровано число: 94461635391431958431444980200213202188847787558304048179569732186360265032799
Число не є псевдопростим за основою 84307365890044747632664642281865903891624522016935926441711112283180299927606

Згенеровано число: 94461635391431958431444980200213202188847787558304048179569732186360265032801
Число не пройшло тест пробних ділень
```



```

Згенеровано число: 95478005450748136261921528190962381626519383703203686337733384267508073224573
Число 95478005450748136261921528190962381626519383703203686337733384267508073224573 прошло тест Міллера-Рабіна з 15 ітераціями
Похибка: 9.3e-10

-----

Created key: 399185872495333319878056658671083324518944358568270185390187699826404018539262834911511430950801412598851503818029115240416764809
5725068464669289325154554
msg in int format: 399185872495333319878056658671083324518944358568270185390187699826404018539262834911511430950801412598851503818029115240416
7648095725068464669289325154554
Encrypted: 44175002043411917092892393422404564893496464852051095348420412141806357401418065942675421407141108500937271437148055176613985849078
34204889592304986577703
Hash: 13073541620531670106887482067785334811641994210590487300989172845424290467187
Sign: 4022074098874373060318994718744118951282907713256703426769391337470879203106191947407778589870814816784627843204416803910175351932233688
17387279553272587
msg in int format: 402207409887437306031899471874411895128290771325670342676939133747087920310619194740777858987081481678462784320441680391017
535193223368817387279553272587
Encrypted: 37883969828634167421385421370944189627198037846524687679062797318041828982024302048567612599636109695159931866573581246368837599410
61026550116708111003774
sender saved key: 3991858724953333198780566586710833245189443585682701853901876998264040185392628349115114309508014125988515038180291152404167
648095725068464669289325154554

Decrypted: 39918587249533331987805665867108332451894435856827018539018769982640401853926283491151143095080141259885150381802911524041676480957
25068464669289325154554
Decrypted: 40220740988743730603189947187441189512829077132567034267693913374708792031061919474077785898708148167846278432044168039101753519322
3368817387279553272587
Hash: 13073541620531670106887482067785334811641994210590487300989172845424290467187
Повідомлення і підпис співпадають
receiver recieved key 399185872495333319878056658671083324518944358568270185390187699826404018539262834911511430950801412598851503818029115240
4167648095725068464669289325154554

```

Перевірка кожної операції шляхом взаємодії з тестовим середовищем: Створимо публічний ключ серверу: RSA Testing Environment

Server Key
Encryption
Decryption
Signature
Verification
Send Key
Receive Key

Get server key

Clear

Key size
256

Get key

Modulus
9ED649B66A1845081B7AB323900688BAE2502DF15D2AEAE9E39E24D053815AD

Public exponent
10001

Щоб ключі клієнта не перегенеровувались кожного разу, згенеруємо прості числа довжини 256 біт за допомогою `gen_primes_128_256bit.py`

```

# В якості клієнта будемо використовувати абонента C з pk_C, sk_C
print("CLIENT\n")
# Для фіксованого значення, обрав p, q, згенеровані gen_primes_128_256bit.py
p_C = 89952803302743973242684820091100452396769521585510083744087843288083864385013
q_C = 113185952448982343257461846858511666938310140333639326275953780654565069484579
small_primes = GenSmallPrimes()
if miller_rabin_test(p_C, 15, small_primes):
    print("q_C просте")
if miller_rabin_test(q_C, 15, small_primes):
    print("p_C просте")

pk_C, sk_C = GenKeyPair(p_C, q_C)

```


Публічні ключі сервера і клієнта в десятковому і шістнадцятковому представленні:

```
-----
SERVER

n_s hex: 9ED649B66A1845081B7AB323900688BAE2502DF15D2AEAE9E39E24D053815AD
e_s hex: 10001

n_s: 71844044093385359615915615497416937140209825701522898686751755302431710778797
e_s: 65537
-----

CLIENT

q_C простое
p_C простое
n_C: 10181393717277041233179350702014760892843692697973384129514762638857245511157148065416429769646052803675755936918654650635041509841857280799067830722214527
e_C: 65537

n_C hex: C2659DA77B71F1CDC0B826CF80E05F5A49176029DD2FD7C38073089B9CB705EA7B3E564246ACD85F5C614CAE101E853D40D5A9529251995D644AFB8C25A3BA7F
e_A hex: 10001
-----
```

Перевіримо операцію шифрування:

```
ШИФРУВАННЯ
msg hex: 77BD8C9
msg in int format: 125556937
Encrypted: 57788365163320323940464710795442031921015219537466225156556221725641538682349
Encrypted hex: 7FC30E3B05037EBC4FABFD77B1241B34BAB2A80566CFD784537A4B261D323DED
```

Decryption

✖ Clear

Ciphertext

7FC30E3B05037EBC4FABFD77B1241B34BAB2A80566CFD784537A4B261D323DE

Bytes

Decrypt

Message

077BD8C9

Розшифрування:

Encryption

✖ Clear

Modulus

C2659DA77B71F1CDC0B826CF80E05F5A49176029DD2FD7C38073089B9CB705EA7B3E564246ACD85F5C614CAE101E853D40D5A9529251995D644AFB8C25A3BA7F

Public exponent

10001

Message

I'm site

Text

Encrypt

Ciphertext

5D027A8CB9FDBE4CC6DD97DB47F1CB86F3C2CB8DB569F714B020DF79B20B8E6C2C4BE2C8F9830642C6E4C614CAE101E853D40D5A9529251995D644AFB8C25A3BA7F

```
ДЕШИФРУВАННЯ
Cipher hex: 5D027A8CB9FDBE4CC6DD97DB47F1CB86F3C2CB8DB569F714B020DF79B20B8E6C2C4BE2C8F9830642C6E46C30B5DFD28A37B084FA562A553E971BA80A7DBED0F0
Cipher: 487131233613900457164263234548734568054547401048903755978026350728758122565593351166578084904803982340616666663219120983093017837024108196740987136046320
Decrypted: I'm site
```

Перевірки цифрового підпису:

Sign

✖ Clear

Message

1234F

Bytes

Sign

Signature

34C616A3CDCDC0A7AB4A924050FE2772A2BA92D0244EDF2A6725FB2FF740BA5D

ПЕРЕВІРКА ПІДПИСУ
Sign hex: 34C616A3CDCDC0A7AB4A924050FE2772A2BA92D0244EDF2A6725FB2FF740BA5D
msg hex: 1234F
Sign: 23870260099720569915825432259600753299423708347047875504845729319400794798685
msg: 74575
Повідомлення і підпис співпадають
Повідомлення hex: 1234F

Створення цифрового підпису:

ПІДПИС
Sign: 7753347806355188080051208899098400115413264092226369684451014055228866093379055150845839639823796414006009818466948031503831881833864099058277289218498489
msg: 125556937
msg hex: 77BD8C9
Sign hex: 9409931BDE5C0DABA87A6C23E46BCC7F16640E2785F8E9306C5B28CF7F4710CCADB9928A6555178F00DA0723E55FB03515B67B939A5121ACF23659FAE3035FB9

Verify

✖ Clear

Message

77BD8C9

Bytes

Signature

9409931BDE5C0DABA87A6C23E46BCC7F16640E2785F8E9306C5B28CF7F4710CCADB9928A6555178F00DA

Modulus

C2659DA77B71F1CDC0B826CF80E05F5A49176029DD2FD7C38073089B9CB705EA7B3E564246ACD85F5C61

Public exponent

10001

Verify

Verification

true

Отримання ключа:

Send key

✖ Clear

Modulus

C2659DA77B71F1CDC0B826CF80E05F5A49176029DD2FD7C38073089B9CB705EA7B3E564246ACD85F5C61

Public exponent

10001

Send

Key

B4E8DD8A83359E474DBBCD14A61D126FAC38AAC090711664BEBF45A68BF4310C4CE5AB1DE699CF45CEE

Signature

04F72A65529C29281B5B80B1E647D00592DFF333D10EBAEF7133DB66165BBD643428BF186F12FBAF27EC6

ОТРИМАТИ КЛЮЧ
k1 hex: B4E8DD8A83359E474DBBCD14A61D126FAC38AAC090711664BEBF45A68BF4310C4CE5AB1DE699CF45CEE8B918C387FA5FDFC8B26A89B54CBBF9805FE9ED16A2A1
s1 hex: 04F72A65529C29281B5B80B1E647D00592DFF333D10EBAEF7133DB66165BBD643428BF186F12FBAF27EC6E31AFBC3DC79A6ACF5CEC39360BF39972CF3EFC3B36
Decrypted: 17318181518674080718
Decrypted: 21617540142335544930049271412742872347278022970069931048047909316237644572548
Повідомлення і підпис співпадають
Отримано ключ 17318181518674080718

Відправлення ключа:

Для наступної задачі створимо ще одного абонента з довжиною простих чисел - 128 біт (згенеруємо нові p , q за допомогою `gen_primes_128_256bit.py`), так як сайт працює з довжиною простих чисел значно менше за 256 біт, внаслідок чого регенерація ключів не дасть жодного результату

```
ВІДПРАВИТИ КЛЮЧ
n_D: 47840724541389047378460163765150132463267694378875689830441929041849549224451
e_D: 65537

n_D hex: 69C4E3F938FE3A6248AE1339C81D5D0C6363EBCF7C17312794AF5D90BEDE7A03
e_D hex: 10001

Created key: 30788947889112521897125467457260276810500308987509815045403651908928498729886
msg in int format: 30788947889112521897125467457260276810500308987509815045403651908928498729886
Encrypted: 2990637745147341644154707680872921910241569579317894911611583520295118053616
Sign: 15476151816769841289013114935624060268843450183937922811675101331452562210019
msg in int format: 15476151816769841289013114935624060268843450183937922811675101331452562210019
Encrypted: 2437995497744902782405179969736410991328710018610470329493575394716032720076
Saved key: 30788947889112521897125467457260276810500308987509815045403651908928498729886
key hex: 69CA4173012AA418E94D8A93DF2E7347BE64917241160D5D5D6E6E5EACA20F0
sign hex: 563DB46F4B1B0DDEB5AE412DFAF9DECD304A5E578D363C15517355173F85CCC
```

Receive key

✕ Clear

Key

69CA4173012AA418E94D8A93DF2E7347BE64917241160D5D5D6E6E5EACA20F0

Signature

563DB46F4B1B0DDEB5AE412DFAF9DECD304A5E578D363C15517355173F85CCC

Modulus

69C4E3F938FE3A6248AE1339C81D5D0C6363EBCF7C17312794AF5D90BEDE7A03

Public exponent

10001

Receive

Key

4411ED4CD5BC7149D5F363260E999A0CCAA7C5A451B7BAEED8B1059A89E839E

Verification

true

Ура, все працює (як і очікувалось, при тому, що ми перевіряли взаємодію абонентів А, В локально. Тож, нам треба було помучитись з сайтом, який використовує представлення Нех замість десяткового, а також має менші довжини ключів, через що ключі локально доводилось регенерувати з меншими простими для конфіденційного розсилання ключів від клієнта до сервера)

Висновки:

У ході даного комп'ютерного практикуму, було побудовано криптосистему RSA з наступними високорівневими процедурами: GenKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), RecvKey().

Для генерації пари ключів спочатку обирається пара простих чисел фіксованого розміру бітів, у нашому випадку ми брали прості числа розміром 256 бітів. Для знаходження простого числа такого розміру, ми спочатку генеруємо випадкове число генератором псевдовипадкових чисел в діапазоні 256-бітних чисел. Далі, ми перевіряємо число на простоту, спочатку число проходить тест пробних ділень, де ми намагаємось поділити наше число на прості числа від 2 до 1000. Якщо тест провалено - число складене, якщо ні - застосовуємо до нього ймовірнісний тест Міллера-Рабіна з 15 ітераціями. Така кількість ітерацій дозволяє знаходити прості числа з точністю $e = 4^{-15} \approx 10^{-9}$.

Потім, на основі пари простих чисел, процедурою GenKeyPair() генерується пара ключів: публічний і секретний ключі

Процедури Encrypt()/Decrypt() дозволяють зашифровувати/розшифровувати повідомлення.

Sign()/Verify() дозволяють підписати повідомлення і перевірити цифровий підпис.

SendKey()/RecvKey() дозволяють конфіденційно обмінюватись ключами з підтвердженням справжності.

Для стандартизації обміну зашифрованими повідомленнями з використанням цифрового підпису, було реалізовано процедури send(), recv().

Коректність реалізації високорівневих процедур було перевірено за допомогою тестового середовища, складність взаємодії з яким була обумовлена тим, що тестове середовище працює з шістнадцятковими числами, у той час як наша система працює з десятковими, через що для взаємодії доводилось конвертувати шістнадцяткові числа з сервера в десяткові на нашому клієнті, і навпаки.

Отже, побудована криптосистема підходить для демонстрації роботи RSA. Для практичного застосування, потрібно допрацьовувати та удосконалювати реалізовані процедури. Даний комп'ютерний практикум дозволив практично ознайомитись з механізмом роботи криптосистеми RSA.