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

Криптографія

Лабораторна робота №4

Виконала:
Студентка 3 курсу
Групи ФБ-25 Ляшенко Аліна

Біла Церква -2024

Тема: вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем.

Мета: 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.

```python
def miller_rabin_test(n, k=5):
    if n == 2 or n == 3:
        return True
    if n <= 1 or n % 2 == 0:
        return False
```

- Бачу, що із 5 випадкових чисел, усі 5 були простими.

```
Prime number: 1074358558556777665541273253168275384026723922768873784215653565974758750587 47 is prime.
Prime number: 1050301994005353332729850048342008951592563705852123524096960864501313721063 13 is prime.
Prime number: 6420551559172091381395344386959832434910695357092896403589702956405945019459 1 is prime.
Prime number: 6420551559172091381395344386959832434910695357092896403589702956405945019459 1 is prime.
Prime number: 9200908266723172033160799051607275522054798152343462234364729698683317834553 3 is prime.
```

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 $pq \leq p_1 q_1$; p і q – прості числа для побудови ключів абонента A, p_1 і q_1 - абонента B.

- У більшості випадків, умова не виконувалася, але були результати.

```python
def generate_key_pair(bit_length=256):
    min_value = 2**(bit_length - 1)
    max_value = 2**bit_length - 1
    p = generate_random_prime(min_value, max_value, bit_length)
    q = generate_random_prime(min_value, max_value, bit_length)
    return p, q

p1, q1 = generate_key_pair(256)
p2, q2 = generate_key_pair(256)


if p1 * q1 >= p2 * q2:
    print(f"Pair for subscriber A: p1 = {p1}, q1 = {q1}")
    print(f"Pair for subscriber B: p2 = {p2}, q2 = {q2}")
else:
    print("Condition pq <= p1q1 not executed!")
```

```
Pair for subscriber A: p1 = 92383395892155802757199276821998459471320422882901971521787832103946438292877, q1 =
99593847833332920491015182050363777121689974893724530726583669525867094239897
Pair for subscriber B: p2 = 59790860726053557365594488112382216161762959739121430025098410052107723299561, q2 =
95286354407673933878955999078397046150883683152813749487051781758960561197533
```

```
Condition pq <= p1q1 not executed!
```

```
Condition pq <= p1q1 not executed!
```

```
Pair for subscriber A: p1 = 78719172356248048403103401433082070832829017858088132165297770305510415391051, q1 =
98010324305914249530342849980536804127055183026999553564141158962718320806727
Pair for subscriber B: p2 = 61849256598863556209438633180327577925815896202017397044313670033522651235297, q2 =
70102872014219734507558674594991876612993239427677509845332789671168884106237
```

```
Pair for subscriber A: p1 = 10951552165933420090087442552624799061349740184551109686509855393670524006652 7, q1 =
10863668105555070602517607402841260838842180585765329562638928190068204314106 1
Pair for subscriber B: p2 = 10763460987346685788797227508898958183792920327467846850229049650276456622314 3, q2 =
88472695422706012000551604875531906454205102820762101050643481905324849602687
```

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 і d1.

```
Public key of A: (65537,
861611448083701981288413652604781949118037131975104978379740250975454256909588977533329901836885331209342897924346101873057212914793718684187544706495 19)
Private key of A:
(411105024361465446311071530844431871292873050595259665134725081221332294941212729464907425347079981728307821038374482123905232565303151677845247223607587 3,
109747423166700941225950884246561728778826877947641821364790140381770061757849, 785085811787085464054991918783985214143840642083611250141454987613561545930 31)
Public key of B: (65537,
778767917009576492085528163426646366186465061865428319601391908714853655016792747170894097340803279095982754420016384667466055893097261686841681861103439 7)
Private key of B:
(380049960933782213808557719316942043879666632187039289955465117665245043996544184907602857645068270780460162060484191532032858373991132728980447469133315 7,
816588204093877643861419924586619880110691591516324450633955965210793491920 43, 953684994597407620410265620230294285013187921754558120334542521691389477410 79)
```

```
Public key of A: (65537,
110578130802749112696373275538273520858795666358673938464049182247928193590144525272816721563180322570123930738403070485366410491442080927858628248732645 49)
Private key of A:
(937088790554020319826033439144326575060904910797624527878118853909651638586282546690702629043454870145292685631110581701677955079495698741471672074530465,
108509218729988874363524794703299126161201086998586092922047832739701016319817, 101906669402816785410242739656908042780825936315220910719783255580334596271997)
Public key of B: (65537,
736799222556704323722529379036505428537693514019606725107688490596974457000229422047595723107914981572615672114210095228248068490276424107828657413033814 1)
Private key of B:
(819577693888700202929221534885045786966108567252534144992149334948493796104726601183472070890050857459256657790311759190318891126371143792955387754784 01,
777688956962310807324442806067936912161736338776403353061741215926083181214 69, 947421479963758665568714033958010306293929070489749173914481968973612463346 89)
```

```
Public key of A: (65537,
72912172854200713146684462303239824070433342284218719097523721189745461303092303917164488957013240075418600778837016824293210004770812630176747796847792 09)
Private key of A:
(19678510054550898181768386853528632805252375884206794686924936759452183034454827767740038766732113243116051323560667827846353937438550262133309532627271 69,
72327500357024847521895130557159612362936410191469600120370083297792570806473, 10080836817847954148582106068862840139591792343472084806973209408579059772723 3)
Public key of B: (65537,
72590329992802934933668314093598841871844872459682048587449082475428775841174949970329993850575240918027064838605946762452664467576412301156086553884377 13)
Private key of B:
(32127731079714389286753789746233852744781304767772975591307934239929918550575569227045975164120007030440478460625585940009936412630518873624147576874710 97,
80152250402401951544394168964588132104864188627002136664009073455230304940607, 90565554464616247630043238176713097244463678954818424444381558233685217131759)
```

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 M і знайти криптограму для абонентів А и В, перевірити правильність розшифрування. Скласти для А і В повідомлення з цифровим підписом і перевірити його.

- Вивід:

```
Encrypted message for A:
[310731740547979045457466055604648121211030079694165600941659136101445176199418248757810528053634613
021295351148754703583372547260194462501828676573017477 5,
851069482084149251423261982860053408491652743911962397450657384395742000743705405335154417301755201 6
476588957372197039724404037982648093825273125485924425,
188673351681649165607989797080822922138859022966000980159105914165394594597510723884474105151370857 5
669469398151516951884360149792260785396886724274136671,
188673351681649165607989797080822922138859022966000980159105914165394594597510723884474105151370857 5
669469398151516951884360149792260785396886724274136671,
975944368934535558872438746215505294188646368082888594624224870702168973428336452885128722834903705 3
719156023225342924927377928539588130669302945700072303,
168579320531692505368520500317543009255314155783556110635435024254535202608339524799049561918586308 7
858199117405426611527591114060429329471819164335721502,
999426521523129004463085449664236121725122826449881595157740540907411282961651713029554328288753875 8
197882008126300268491307269105362608138610763893545965,
371475560520657125477972620473607232174483052816494206402268139797426295383815151534268725210310738 9
197673285834083599401081592492390349645102341448043994,
819422168650890142505697673758541904108647646186424063943743681856596193688710881914563146840990536 1
192007895258704445015389864333694719039402486822074303,
716128246492007480249542493780383249679661038997439714308905283194878802633211970285140269944900284 3
456716245035052541884388284623148389546099840535292 43,
176505891203118601351310098273699606955367765870026490912662666244058372582207876733818202335758986 1
346114482995946443317006733407127242429575963339174341,
999426521523129004463085449664236121725122826449881595157740540907411282961651713029554328288753875 8
197882008126300268491307269105362608138610763893545965,
716128246492007480249542493780383249679661038997439714308905283194878802633211970285140269944900284 3
456716245035052541884388284623148389546099840535292 43,
176505891203118601351310098273699606955367765870026490912662666244058372582207876733818202335758986 1
346114482995946443317006733407127242429575963339174341,
999426521523129004463085449664236121725122826449881595157740540907411282961651713029554328288753875 8
197882008126300268491307269105362608138610763893545965,
287778769633459500099890940978590553370686922695026892650316273648598752075247378659385580181930706
829459897297029995204707792710118657031224652504470822,
999426521523129004463085449664236121725122826449881595157740540907411282961651713029554328288753875 8
197882008126300268491307269105362608138610763893545965,
176505891203118601351310098273699606955367765870026490912662666244058372582207876733818202335758986 1
```

```
346114482995946443317006733407127242429575963339174341,
851069482084149251423261982860053408491652743911962397450657384395742000743705405335154417301755201647658895737219703972440403798264809382527312548592442 5,
165990553879730891605751798682913769185891169413006034445147798833277950970357543690151556616983107111894364747614900836251791836426701600346915235234912 6,
877000251016571945291264424321399465072365367820579144917721573201616999319313223556672978775973845252260194398554762373017087373943337646449999740635709 0,
851069482084149251423261982860053408491652743911962397450657384395742000743705405335154417301755201647658895737219703972440403798264809382527312548592442 5,
371475560520657125477972620473607232174483052816494206402268139797426295383815151534268725210310738919767328583408359940108159249239034964510234144804399 4,
999426521523129004463085449664236121725122826449881595157740540907411282961651713029554328288753875819788200812630026849130726910536260813861076386354596 5,
695248292145006244077641649111918997554236163427354749562511056647303837680274707407827570282260197110112517526956608023655202736363708553941085908289768 9,
851069482084149251423261982860053408491652743911962397450657384395742000743705405335154417301755201647658895737219703972440403798264809382527312548592442 5,
176505891203118601351310098273699606955367765870026490912662666244058372582207876733818202335758986134611448299594644331700673340712724242957596333917434 1,
176505891203118601351310098273699606955367765870026490912662666244058372582207876733818202335758986134611448299594644331700673340712724242957596333917434 1,
287778769633459500099890940978590550337068692269502689265031627364859875207524737865938558018193070682945989729702999520470779271011865703122465250447082 2,
913493078133371275748008458752935320874140913806903717665999420997914237304370106658735538042904226661961668907950310904211828298753668719345123564963399 ,
851069482084149251423261982860053408491652743911962397450657384395742000743705405335154417301755201647658895737219703972440403798264809382527312548592442 5,
197837420889439788288365925173328704909052835245230680902471593731286938556225182775700219698774997975164381923565868807391053199032766525313100844533726 02]
Decrypted message for A: Hello, this is a secret message.
Digital signature from A:
547905057609113071481141277329811140147093557967588934542110202148863043625940305076303121146914007941761632953969422776991672919143994414987011081346 4418
Signature valid for A: False
Encrypted message for B:
[149163620050566589199019491146982651529378674110580598479906478765536432625112109916810099804460213993307482744305145615661909940625848320532757648527129 2,
795862440772035036075279779457355202493021354463393322013250918691385842659023429530207017835531230617802402313239266351536296226814099741518647004140210 4,
925472297296504702071366610654640116178827783927395130963894186875852108268735624785772397347996070414108314668612805964499438840193987552651985718359327 2,
925472297296504702071366610654640116178827783927395130963894186875852108268735624785772397347996070414108314668612805964499438840193987552651985718359327 2,
940040254838719713666249890901061874795199680984123466426120785908714869554463727077799212031977115797991520574980837028440405791722718066857852547344821 9,
381030336028509773189136253392436297670772623897613911999668119094747828486267699950550744505243251595893204875055194345073553143109338434187940757142330 0,
603715191304380769658620414232806999573058248952640015180410577901644245765723926192943937110886237573859021197704918421136548819539637894715922463929259 7,
787659832263656231364408292964268336692968126024785638131979202208336821638493615780581929982278010517635587852571666407592364806393299201833926544091735 7,
772017962962781870075437308019228198283625828174748018209898655381743007361832977032067628939841723311704151905530528450579316925242657893813228607716358 4,
863116155643637537187425686250391967994240194496523365406297558383920601252536392109494379642140585687316704062649661150042384591725244449146717467069932 3,
782687001768182395569847772861503021740895226368225905485650136791907849687874649236305477443683504629039919594426090460624844203820064020989978532984712 9,
603715191304380769658620414232806999573058248952640015180410577901644245765723926192943937110886237573859021197704918421136548819539637894715922463929259 7,
863116155643637537187425686250391967994240194496523365406297558383920601252536392109494379642140585687316704062649661150042384591725244449146717467069932 3,
782687001768182395569847772861503021740895226368225905485650136791907849687874649236305477443683504629039919594426090460624844203820064020989978532984712 9,
```

```
603715191304380769658620414232806999573058248952640015180410577901644245765723926192943937110886237573859021197704918421136548819539637894715922463929259 7,
797984506380119458720745777170502523769430180214984687142923083318028390356297339017281763880495413142471546907787237059503587959689182459599257518787898 5,
603715191304380769658620414232806999573058248952640015180410577901644245765723926192943937110886237573859021197704918421136548819539637894715922463929259 7,
782687001768182395569847772861503021740895226368225905485650136791907849687874649236305477443683504629039919594426090460624844203820064020989978532984712 9,
958624407720350360752797794573552024930213544633933220132509186913858426590234295302070178355312306178024023132392663515362962268140997415186470041402104 ,
106828378347507334556879436660219496500095952213006145610402109983282792883653423286641283021989934553988565086674279938263418797282757442617280884296658 40,
110875616045534899621118270658001020489386245929992048900187202043202522094714300027824643307001089116459658090057690648284600978155333312516826513704028 93,
958624407720350360752797794573552024930213544633933220132509186913858426590234295302070178355312306178024023132392663515362962268140997415186470041402104 ,
787659832263656231364408292964268336692968126024785638131979202208336821638493615780581929982278010517635587852571666407592364806393299201833926544091735 7,
603715191304380769658620414232806999573058248952640015180410577901644245765723926192943937110886237573859021197704918421136548819539637894715922463929259 7,
749117852227153448602471219770261493737683383347740458464595917267768773123127610519179902482907911425430752927136149200781303746596410585802341295918011 7,
958624407720350360752797794573552024930213544633933220132509186913858426590234295302070178355312306178024023132392663515362962268140997415186470041402104 ,
782687001768182395569847772861503021740895226368225905485650136791907849687874649236305477443683504629039919594426090460624844203820064020989978532984712 9,
782687001768182395569847772861503021740895226368225905485650136791907849687874649236305477443683504629039919594426090460624844203820064020989978532984712 9,
797984506380119458720745777170502523769430180214984687142923083318028390356297339017281763880495413142471546907787237059503587959689182459599257518787898 5,
961183650179287646740710195824323676059414369896141562195391218471293096652103650154489720318214668023851450469782382514863838382058898594447088268824309 ,
958624407720350360752797794573552024930213544633933220132509186913858426590234295302070178355312306178024023132392663515362962268140997415186470041402104 ,
819995607812658838985190433120573361604604010646645061690561749156283116653967439558833035921661958188827366127186569067540370565505419365271432875579840 1]
Decrypted message for B: Hello, this is a secret message.
Digital signature from B:
241583227368047411941993567996152480652280996569312126768798743449998182227580316139844576544304538502611892233065967355097987784406396261346382868940 4989
Signature valid for B: False
```

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().

```
Signature for key:
597360442052957483600352006120098468191696327122904616683186674929983862237094709964640764774163173791839989753058883728836937666998028183108274472629715
Received key: 14131071892593132373351279129044766903036232749626091424039629523179816150488
Signature valid: True
```

Висновок.

Отже, виконуючи лабораторну роботу я писала функцію, яка дозволяє генерувати прості числа, створювала пари ключів RSA, а також виконувала шифрування, розшифрування та перевірку цифрових підписів.

Для реалізації першого завданні я використовувала тест Міллера-Рабіна, що дозволяє з високою ймовірністю визначити простоту числа. Далі була реалізована функція генерації пари ключів RSA. Це завдання включало в себе обчислення відкритого та секретного ключів, де відкритий ключ використовується для шифрування повідомлень, а секретний — для їх розшифрування. Важливим моментом у цій частині роботи було створення цифрових підписів.