

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

ФІЗИКО- ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

Криптографія

**З теми: « Вивчення криптосистеми RSA та алгоритму електронного підпису;
ознайомлення з методами генерації параметрів для асиметричних криптосистем »**

Перевірила:

Селюх П.В

Виконали студенти групи ФБ-92

Ханас Максим Любомирович

Гуманков Денис Максимович

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.

Завдання:

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 $1 < p, q$ довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $p \nmid q$; $p \nmid q$ – прості числа для побудови ключів абонента А, $1 < p < q$ – абонента В.
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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , $(,)$ і n і та секретні d і d .
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.
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 \square \square п. \square

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().

Код програми знаходиться у файлі **main.py**.

Хід роботи:

Спочатку ми реалізували функцію пошуку простих чисел з заданого інтервалу, після цього за допомогою неї згенерували дві пари p та q . Далі реалізували функцію генерації пар ключів зі знайдених раніше p та q . Далі нами було реалізовано функції шифрування та дешифрування, також було зроблено функцію, яка відповідає за створення цифрового підпису, та ще одну, за допомогою якої цей підпис можна підтвердити. Після цього були створені два абоненти, на яких і було перевірено функціонування програми.

Результати:

Параметри абонента А:

```
[*]AliceInformation:
e: 2371241860842832846237252199174291145655634115582019060701231848762274067416560289206851779476868449828908633174562250902439553556046734731367536992607025
n: 31000657054408346018507910388312642006067545113860073258187341670769847422718743615896715866750294741760591814601565839020897775031814032449507262206607667
d: 11397712946077397413957559591409177940691665019327945186542477218588510746322983284551061797379452418666999288957083802871288075981246812880249758461160041
p: 18557736774379943336221454552447743980709087926737610785971633165427792712423
q: 1670497245612510455028533198790024378782309795410495068817752853923530383829
my_message: 233294038008052013117340569759310111827525701707077988727418068714042553127795743544934369322026130545685613644063548558864905033784600677303860571118889499
signature: 28943001804418477041291980097019038941430157939751956343321031229203125726759035109272793230959672238457504791977526652583981387893481314407991278739988343
encrypted message: 701244130828392092870924433931199418259165309570831389525423498526937853833174352671128554452285159235353175329889251071908272572174765952300387397692760
```

Параметри абонента В:

```
[*]BobInformation:
e: 34563625838417361237246839043906977510351668464483916885456517863820914369839578641446826389309165209977199785267340734864936063800345684932184569218509
n: 3630540905462063677472322194817315981093761149872492093694838205900589760116177266131679619677892114133880236553973675785061670278923403793220390346027931
d: 3479752622863410811967082683842551173730643162825061626731246240650909640091604163530529004997512692447055898842722056745133359877123167880126831340006237
p: 157510628708166142818293016648591957630255515825353723163609987125958166105687
q: 230494998695864750419332645957543530413759959374846988208670003305065140142013
my_message: 3410982612908335542305809687261218944849054363657818025360392522357528478422511138406574699381816669519216695415267039918238050711941216240441002421735040
signature: 15045578701858444999877913181648315906751736591983880892836036788372940547384737547861461361693477712454761406126883856091083273863488610882930825836784830
encrypted message: 24881672726295824731516056845201660097159978486784816291791892460010606775835902952416077604978276486944648347139191128409867143830314463835626808809243210
```

Приклад Комунікації:

```

1) Alice send message to Bob, and Bob verifies it
2) Bob send message to Alice, and Alice verifies it
3) Try yo intercept message using Dude as interface
4) Validate the message integrity(Teacher's request)
1
[*]Bob Verifying message :
Decrypted message: 23329403800852013117340569759310111827525701707077988727418608714042553127795743544934369322026130545685613644063548558866905033784600677303860571118889499
Decrypted signature: 23329403800852013117340569759310111827525701707077988727418608714042553127795743544934369322026130545685613644063548558866905033784600677303860571118889499
Bob receive: 23329403800852013117340569759310111827525701707077988727418608714042553127795743544934369322026130545685613644063548558866905033784600677303860571118889499
[*]Bob Sent public public keys
[*]Bob Sent public public keys
[*]Alice Sent public public keys
[*]Alice Sent public public keys

```

Перевірка роботи програми на прикладі вхідного тексту А (абонент А відправляє абоненту В текст, який було зашифровано відкритим ключем абонента В) за допомогою сайту <https://www.dcode.fr/rsa-cipher>

//Це приклад виконання, додаткової частини коду яка провіряє тільки цілісність без шифрування

```

[*]AliceInformation:
e: 33173840777384708064050955449108962309419060478846083520626520338203461038902166280051018494718924118627284098599560793488440698457858675747087094183974667
n: 4305647684727491248425888576387089335893578311081108344370372334621430156911905862498987977359764961958628941996440641451160254071165452168597641452942780
_d: 19040667866178689740103345156479578772220715452203212929815944884151738409498508285938704543491454870483534746336716009492758441531126313046376920107863
_p: 22099965623685256659113689136876824954939639743264707136208475234007082774711
_q: 194822238268928998116434412502066855353968640450692621756635566233694832871211
_my_message: 955878909550887962065433807448443409392591827720701964418286100482976687017607020123260222616567903365595838921603715280676910312952293931999367660247663
signature: 20938052092454021426999693854598227095265165668134407590679282040197202594138732533730923812031650834829895032679212355536180737869348578742849314190894643
encrypted message: 47867774634549005872035007708940850425759352468951173452813970109665956754301157402524997154279621292522458053204102551189763206712187422230996453433012519
-----
[*]BobInformation:
e: 1948999140888438230959286989137152363535083281512156971230318640116424074159160921794665806193531328683731401555225840388511013473948483109491175167904429
n: 49019887405971024791285181934728495978270977086278195038483321155729021042785512691354610606379957573171421675440934239351578552476011221272824236485949169
_d: 23563062159344940030772736116277692121158619166477880010298934604670183240671698711459864514854002422092175186217588226370759359669717875525614182167318649
_p: 22470329167640373860900522982141048122983973069711817953900205310822945952571
_q: 218153846524381308061487060105838771155716416163512215457369804770716241535939
_my_message: 1994503101876030474272451547931416064812480880479234482382963296174022915458165922402788043874657367938630201384570193115950381922102284068052667551239329
signature: 1835571831222844171694426241559913072150117957155996905626721258203089559244624727633976588472599590274592829917290639570129265202646155978548732782306833
encrypted message: 1935605004981641188231444982845716091452541956536740656366581304274830254858917011825268888666519692356783355441912488375118457567617167671547232648549800
-----
1) Alice send message to Bob, and Bob verifies it
2) Bob send message to Alice, and Alice verifies it
3) Try yo intercept message using Dude as interface
4) Validate the message integrity(Teacher's request)
4
[*]Alice Sent public public keys
[*]Alice Sent public public keys

[*]AliceSending message without encryption !!!!
Message:
_my_message: 955878909550887962065433807448443409392591827720701964418286100482976687017607020123260222616567903365595838921603715280676910312952293931999367660247663
Dude receive: 955878909550887962065433807448443409392591827720701964418286100482976687017607020123260222616567903365595838921603715280676910312952293931999367660247663

```

← → ↻ https://asym-crypt-study.herokuapp.com ☆

Asym Crypto Lab Environment RSA Rabin Zero Knowledge Protocol Documentation

RSA Testing Environment

[Server Key](#)
[Encryption](#)
[Decryption](#)
[Signature](#)
[Verification](#)
[Send Key](#)
[Receive Key](#)

Verify

Clear

Message

5062265847032929181702945988251712299178919527409692173006527986639

Bytes

Signature

3091298419144674521812012013250303645718797953743895719585074065814980290609842304581060089

Modulus

0643188717745267416055920487555222762442915285665522057489210394283022016288344514389947537

Public exponent

7237225881971825733281351751703742943611605589611212986540542570879175952112767587320313511

Verify

Verification

true

✓

Висновки: під час виконання даної лабораторної роботи ми ознайомилися з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA. Також, ми ознайомилися з криптосистемою RSA та реалізували засекречений зв'язок з використанням цієї системи.