

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

**КРИПТОГРАФІЯ**

**Лабораторна робота №4**

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

Виконали:  
Студенти 3 курсу  
Снігур А.Ю. та  
Носова Є. О.

Київ – 2021

## 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:

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  $pq \leq p_1q_1$ ;  $p$  і  $q$  – прості числа для побудови ключів абонента А,  $1 < p < q_1$  – 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  $(e, n)$ ,  $(d_1, n_1)$  та секретні  $d$  і  $d_1$ .

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 і 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  $0 < k < n$ .

## Хід роботи:

Було використано функцію `inverse_of_e1` для знаходження оберненого та `gcd = e1 * v + u * mod`. Також написано ймовірнісний тест Міллера-Рабіна (`miller_rabin`). `e1_to_power` - Схема Горнера швидкого піднесення до степеня. `prime_check` - ймовірнісний тест Ферма. `generate_prime` - генерація простих чисел вказаної довжини в бітах. `generate_keys` - генерація ключів вказаної довжини в бітах. `to_decrypt` - Функція розшифрування. `to_encrypt` - Функція шифрування. `to_sign` - Функція цифрового підпису. `to_verify` - Функція перевірки цифрового підпису. `to_send` - Функція відправлення (формує повідомлення що буде відправлене). `to_receive` - Функція отримання (оброблює отримане повідомлення, розшифровуючи його та перевіряючи його піддліність)

 file.txt – Блокнот

Файл Правка Формат Вид Справка

[illegible]

pA =

$$q_A =$$

pB =

14518632409863869313587841901644566493139169926468052760798319528014395219539670622  
69581689681245087717611692957489714455651519841491581440167726815447789947467077156  
30346610132448259921433983960050548246987354134106455130484378241744926431259210580  
740951249215391250086181297498903232437958602089780439285799

$$qB =$$

99169060431374865547298700327975833262092832495039733014879947851356734137625277952  
24814292931301057010089993725312320630212519831525363039266291728681970047379287189  
76441945519044904052069299286606069310106325679357796097906042698158112623731827694  
83909647199630072357762795511579436363924944753539440737159

A

N =

68198848589764796062030085971290340158509485287824881704709609186346844726519716971  
82432028982772570623067963897772032692853013313838775641095336283963168762043005965  
33096555286703670022002775043000029419415576919585149191322422320378561100055922894  
98932865845183862955521130519943730935083989790295138749393004167584881968594496906  
59528952382157476299923546588297496938443326401101565615521842486966231992376229836  
44640351349301814060262841370583530976381624006268441390198746718267730480616944924  
45849503997256914127862963477318893610876160875791195325577675891560875492708480929  
06571657439764636396652763192000471

e = 65537

d =

54675003490468470480722687138830977345137050315205259771535889287880885130171848717  
77197022060574234907100018175272599748465602217410060133341319616334419487106544035  
04057648524019981485665315829749066111135905319619807492886024553077040126310912077  
53840299439580776409448643035969964555293161230631504112665309426750901985800436925  
80707907497576932852999553296101644460118237876336462964990948228612234249321550497  
24333502510959920820273161822477476342863104398965756737142864848391722839491149940  
41076724475440452548950407259740225920996083205487068503618031047265188577882216152  
21300318956769453868709316147305073

B

N =

14397991348347077515270314507994254939430237353077546904801242582688530699355453417  
01055562072157452822727622673371237220626959601923205310595713420470837240890744244  
69929348568148885126341790380166752349651154951206823616114020235982690968761323339  
68114468049878381384228973492164198576034175559344710674790618041816946450554749264  
06625088463545471838028756804118255996687395956025738405941239423681761559236121821  
40592455065850684894293858892170334992224413425545329506157667087443220262037306417  
01784283897858390257876089601323813574349411533568995102893773366234444595660881750  
191092251392908170531437256040305041

e = 65537

d =

62757376879759924363521644908275308085472963398998001874140149170252005425605060089  
76967695523635472684748656375409732280151628543958112654298663284735330671725132382  
33059458198840510325324562308290499659754778573351560708899263326225117142016870244  
49266505075426986682619719666320169754946405698802356704771627435895510618311735941  
48730183346491362224652432972206056112837436565490079299344053683617953321541586090  
80529272480101051053672824396334893268645247635816591311025324956381417794329826947  
38395857449594593224950461124834118652736629054576134269316722859375032854807484988  
47945493467397351023115345230245785

Чисельні значення ВТ та ШТ та цифровий підпис А та В:

A want to send 412811

A encrypt this message

73326482484555629308481013535054049431042227124843864145489075505693391480964099827  
28822589517611507834942971059952548259491044165393996758322512537369071937515789053  
74867478689118823861873720685974743127446060933835974857948758365152849259506128346  
85615575131300915832442489915206613861436118158802038278334954090412037154539051210  
76120208097707020836040933729235295707775665315932286160221049985231475615722852257  
47525916423884117131029166656069883482908324638049502081902807336281518054142513954  
41632249548607865179770937904213748862513030468134500640219983955829283831260603558  
98384216699831304687980157226224551

B decrypt this message 412811

```
A want to send 412811
A encrypt this message 73326482484555629308481013535054049431042227124843864145489075505693391480964099827288225895176115078349429710599525482
B decrypt this message 412811
Messages is equals!
```

A signature is

58468987388546872264806819021581018828115523966958407979819979429724457059691193216  
27189428744413273857824853125172638211934680904573278691823550091669368862004484781  
84569571861971472939217585554739526098183238599531794236310629914008948266321227357  
48912816141328621505286955622554145576263693516357385934332490245845020030344784542  
57243560007924733763993179438068377051926982192775030051231926923619920612197080607  
39265027242152928481818555230783549568203351130566291832479289059534157031623401101  
50328437491401134155717324135973723821425316902098276265850391267611168175083554357  
83203743081656732755642447192031672

```
A want to sign her message
A signature is 584689873885468722648068190215810188281155239669584079798199794297244570596911932162718942874441327385782485312517263821193
B check signature
Signature is valid
```

## Get server key

✖ Clear

Key size

1024

Get key

Modulus

9BC7D5627F19D0989F188DAA369C479A7955FA85D9AFAE246C5B5A04883DA126A2476971DBD4EFE86E9EI

Public exponent

10001

```
Message = 3552927
C2 153417080100951831356407954828237743297793623096123136663341362800469642959190014721384407097803081920059094321205490730632079678316580
C2_Signature 75994131219269235676074942360441828029123655762605335443179524350541532418581174291507041302909224682144131945882215478630017
A_n: 3927954580670263833907611499425167294560910762497218619166192784323620899278798552484540311606791831601075728248289311880789873940924
A_e: 65537
```

## Receive key

✖ Clear

Key

15D8E9CB0C0484DFB035F7270C098A32C2CA9FA41ED59683222594024D2CD99ADBF1B266F03C9928F2564

Signature

6C381FC9E64D0C961BD2A6A1B1425E6716ABD8F826A95C05372D37EACA7EAD375E231CC251D3D4F13ED

Modulus

37EF991AF3258E10FFBB62745F08D3C9CB7C4DB2334C9C39BD5A44E0EB0268DD06C36842B15E81789774

Public exponent

10001

Receive

Key

36369F

Verification

true

✓

## Hexadecimal to Decimal converter

From

Hexadecimal

To

Decimal

Enter hex number

36369F

16

= Convert

✖ Reset

↔ Swap

Decimal number

3552927

10

## Висновок:

В ході лабораторної роботи була створена тестова модель криптосистеми RSA, тести на псевдопрості числа, алгоритм Горнера для пошуку великих степенів за модулем. Протестована робота з готовим тестовим середовищем, посилення на яке надане вище. Згенеровані тестові випадки з можливою підміною повідомлення. Покращили навички програмування на мові python, ознайомились з модулем requests та удосконалили навички роботи з сервісом контролю версій.