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 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq p1q1 ; p і q – прості числа для побудови ключів абонента А, 1 p і q1 – 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) , (,) 1 n1 e та секретні d і d1.
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 М 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 < 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().

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 A:

```
[*]AliceInformation:
e: 23712418608428328462372521991742911456556341155820190607012310487622740674165602092068517794768684498289086331745622509024395535560467347313675369926070 25
n: 31000657054608346018507910388312642006067545113860073258187341670769847422718743615896715866750296741760591814601565839020897775031814032449507262206607667
_d: 11397712946077397413957559591609177940691665019327945106542477218588510746322983284551061797379452418666999288957083802871208075981246812880249758461160041
_p: 18557736774379943334622145455244774398070908792673761078597163316542779271 2423
_q: 16704977245612510455020533319879002437878230979541049506881775285392353038382 9
_my_message: 233294038008520131173405697593101118275257017070779887274186087140425531277957435449343693220261305456856136440635485588669050337846006773038605711188894 99
signature: 28943001804418477041291980097019038941430157939751956343321031229203125726759035109272793230959672238457504791977526652583981387893481314407991278739988343
encrypted message: 7012441308283920928709244339311994182591653095708313895254234985269378538331743526711285544522851592353531753298892510719082725772174765952300387397692760
```

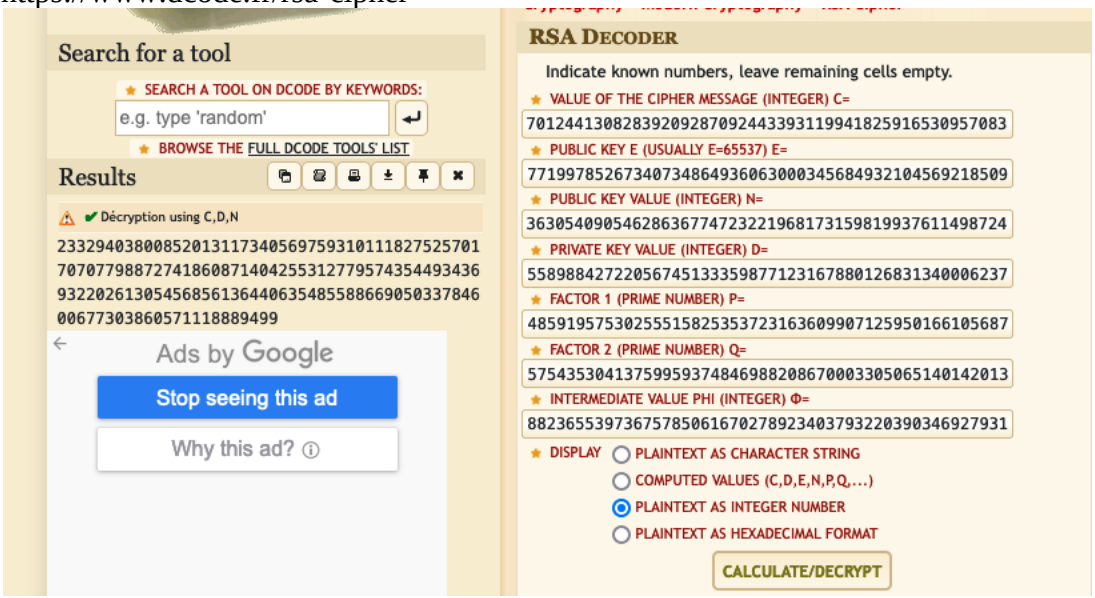
Параметри абонента B:

```
[*]BobInformation:
e: 34563625838417336123776248639043906977510351668464483916885456517863820914369839578641446826389309165289977199785267340734864936063000345684932104569218509
n: 36305409054628636774723221968173159819937611498724920936940382059005897691116177266131679619677892114133888236553973675785061670278923403793220390346927931
_d: 34797552622863410811967082683842551173730663162825061626731246240650999640091604163530529004997512692447055898842722056745133359877123167880126831340062 37
_p: 15751062070816614281829301664859195753025551582535372316360990712595016610568 7
_q: 23049499069586475041933264595754353041375995937484698820867000330506514014201 3
_my_message: 34109826129883355423058096872612189648490543636578180253603925222357528470422511138406574699381816669519216695415267039918238050711941216240441002421735040
signature: 15045578701858444999877913181648315906751736591983880892836036788372940547384737547861461361693477712454761606126883856091083273863408610882930825836784830
encrypted message: 24881672726295024731516056845201660097159978486784816291791892460010606775835902952416077604978276486944648347139191128409867143830314463835626808809243210
```

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: 233294038008520131173405697593101118275257017070779887274186087140425531277957435449343693220261305456856136440635485588669050337846006773038605711188894 99
Decrypted signature: 2332940380085201311734056975931011182752570170707798872741860871404255312779574354493436932202613054568561364406354855886690503378460067730386057111888949 99
Bob receive: 233294038008520131173405697593101118275257017070779887274186087140425531277957435449343693220261305456856136440635485588669050337846006773038605711188894 99
[*]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 A (абонент A відправляє абоненту B текст, який було зашифровано відкритим ключем абонента B) за допомогою сайту https://www.dcode.fr/rsa-cipher



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

```
[*]AliceInformation:
e: 331730407773847980640509554491089623094190604788460835206265203382034610389021662800510184947189241186272840985995607034884406984578586757670870941839746 67
n: 430556476847274912484258885763870893358935783110011083443703723346214301559119058624989879773597649619586289419964406414511602540711654521685976414529427 09
_d: 190406678661786887401033451564795787722207154522032129298159448841517384094985082859838704543491454870483533474633671600949275844153112631304637692010786 3
_p: 22099965623685256665911368913607682495493963974326470713620847523400708277471 1
_q: 19482223826892899811643441250206685535396866045069262175663556623369483287121 9
_my_message: 955878909550887962065433807448443409392591827720701964418286100482976687017607020123260222616567903365595838921603715280676910312952293931999367660247663
signature: 209380520924540214269996938545982270952651656681344075906792820401972025941387325337309238120316508348298950326792123555361807378693485787428493141908946 43
encrypted message: 4786777463454900587203500770894085042575935246895117345281397010966595675430115740252499715427962129252245805320410255118976320671218742223099645343301251 9
----------------------------------
----------------------------------
[*]BobInformation:
e: 194899914808843823095928698913715236353508328151215697123031864011642407415916092179466580619353132868373140155522584038851101347394848310949117516790442 9
n: 490198874059710247912851819347284959782709770862781950384833211557290210427855126913546106063799575731714216754409342393515785524760112212728242364859491 69
_d: 235630621593449400307727361162776921211586191664778800102989346046701832406716987114598645148540024220921751862175882263707593596697178755256141821673186 49
_p: 22470329167674037386090052298214104812298397306971181795390020531082294595257 1
_q: 21815384652438183086148706010583877115571641615351221545736980477071621415359 39
_my_message: 199450310187603047427245154793141606481248088047923448238296329617402291545816592240278804438746573679386302013845701931159503819221022840680520675513923 29
signature: 183557183122284417169442624155991307215011795715599690562672125820308955924462472763339765884725995902745928299172906395701292652026461559785487327823068 33
encrypted message: 193560500498164118823144498284571609145254195653674065636658130427483025485981701182526888886665196923567833554419124883751184575676171676715472326485498 00
----------------------------------
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
```

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.