Міністерство освіти і науки україни
Національний технічний університет україни
"київський політехнічний інститут імені ігоря сікорського"
Фізико-технічний інститут

# Криптографія
## Комп’ютерний практикум №4

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

Виконали:

Студент гр. ФБ-11 Падик Володимир

Та

Студент гр. ФБ-11 Ахунов Михайло

## Мета та основні завдання роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключ для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захис інформації на основі криптосхеми RSA, організація з використанням цієї систем засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів

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.**

На цьому етапі було розроблено інтерфейс для генерації простих чисел шляхом генерування випадкових значення та проведення тестів чи є це число простим. Було написано функцію для проведення тесту Міллера-Рабіна попередньо перевіривши пробними діленнями на декілька перших простих чисел.
Ось приклад виконання коду:

Candidate:
207485233626585534865251284731690847536685355897757979266326190998295916384 6
failed the test
Candidate:
103453682096013391350351115890902750869523248630699464414424703048482957764681
failed the test
Candidate:
829632242636293493518826226784697437031445816216321129190687292207921019706 92
failed the test
Candidate:
105859173764934330644976275768237325077229882825385223950405747170349377573967
passed the test
Candidate:
847660164628986413239597728387699654354711032275259517388671705478157268845 72
failed the test
Candidate:
922866999805521702970086513827690491246260673237022362604320427814324012140 60
failed the test
Candidate:
107670865394889930775257991148688467125177750037201207245912979436681900707831
passed the test

2. **За допомогою цієї функції згенерувати дві пари простих чисел $p, q$ і $p1, q1$ довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \le p1q1$ $p$ $q$ – прості числа для побудови ключів абонента $A$ $p1$ і $q1$ – абонента $B$.**

Було створено інтерфейс для перебору простими числами поки не буде досягнуто умови завдання $pq \le p1q1$

**p**=177856350073880850029877117928264318705336780947659223459391177272954654204 11,
**q**=996125901793036555882950034373390232777850032261394074564435818429828184101 39
**p1**=400149005832806669039121061653840597175044731344010099827856931643502975154 23

**q2**=10767086539488993077525799114868846712517775003720120724591297943668190070783
1

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 $A$ ,$B$ – тобто, створити та зберегти для подальшого використання відкриті ключі $(e, n)$ , $(e_1, n_1)$ та секретні $d$ ,$d_1$ .**

Було розроблено програмний інтерфейс для створення ключових пар для RSA.
Відкриті та секретні ключі було збереженно

**Public A key**: (
**n**=17716731710696256560715995621499976751259828453188705003645512732993108891157166626941856149411403851204220006298823332792018610614780960658543902599471 29,
**e** = 65537)
**Private A key**: (
**d**=59143179633888139682216847308722781232595713398364662415087130563257890400950731202302147552957281020523958939153197081606481147287183204949210067310799 3,
**p**=17785635007388085002987711792826431870533678094765922345939117727295465420411,
**q**=99612590179303655588295003437339023277785003226139407456443581842982818410139)

**Public B key**: (
**n**=43084389744923152646316215210465553397579149107234305319938276984146775261437688708447739172702527602468828956033875256561800580656485379324303843393775 13,
**e**= 65537)
**Private B key**: (
**d**=27270498426826101562086901130056793659129625452283343047159308571591261903884781726267391727340540102016328778989156231638319844592238242851555790701338 33,
**p**= 40014900583280666903912106165384059717504473134401009982785693164350297515423,
**q**=10767086539488993077525799114868846712517775003720120724591297943668190070783 1
)

4. **Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів $A$ ,$B$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 $A$ ,$B$, перевірити правильність розшифрування. Скласти для $A$ ,$B$ повідомлення з цифровим підписом і перевірити його.**

На цьому етапі було створено функції **Encrypt**(msg,pubKey) **Decrypt**(msg,pubKey,privKey) **Sign**(Msg, pubKey,privKey) **Verify**(msg,signed,pubKey). Кожна функція ізольована і приймає лише необхідні їй аргументи.
Було теж написано код для демонстрації роботи цих функцій ,
а також було створено зручний інтерфейс для проведення тестів у тестувальному серидовищі. Тести доводять правильність виконання функцій
Ось приклад роботи :

Generated message:
272110099899695986673150985486277168216441933323981288663468055509957119829538721733245640987018827744840554356594445344466462699172690210633071908757 7
Encrypted message for A:
141884564090467297674709893074904693869668218538063813418412289192168798799095194154057214851085486959704518568025646390747419019423056008630179366853209 8
Encrypted message for B:
231581203673433024791493885580412054723955088915007036714735348672019622105586973694924687385805340962644319315241341964778334972655474098198595383745410 0

Signed message for A:
132924117056906313644955653320780153654509318438036439061683017751106699647909365851197751422142131654091351986049754790238670170314917724686395871783837 6

Signed message for B:
341413627491639541314136719371097751029740879174234299499232911073776573432086581058762835823518692662984382799304413290592184732745093644265994706664923 9

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 $0 < k < n$**

На цьому етапі було створено функції SendKey() та RecieveKey().

Спочатку генеруємо $0 < k < n$

SendKey підписує ключ приватним ключем надсилаючого

Потім зашифровує оригінал та підпис публічним ключем одержувача і надсилає їх відкритим каналом звязку

RecieveKey отримує повідомлення та розшифровує ключ та підпис приватним ключем одержувача . Потім перевіряє чи отримане значення ключа було підписане саме надсилаючим.

Також було проведено тест на правильність виконання функції у тесту вальному середовищі .

Для коректної перевірки потрібно обирати довжину відкритого ключа сервера таким чином щоб ключ одержувача був не меншим ніж ключ насдсилаючого .

**Висновки:**

Після виконання лабораторного практикуму було здобуто навички розроблення асиметричних криптосистем типу RSA . Здобуто практичні навички роботи з тестами перевірки чисел на простоту і методами генерації ключів. Збудовано систему захищеного зв’язку.