

**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**

Виконала:  
Студентка 3 курсу  
Гузенко Г. С.

Київ – 2021

**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.

**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  $p \leq q$ ;  $p$  і  $q$  – 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  $(e, n)$ ,  $(p, q)$  і  $n$  і 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

В якості функції пошуку випадкового числа я обрала функції з бібліотеки random: randint(), getrandbits(), аби згенерувати число розміром 256 за завданням. Перевірка на простоту, я почала з ділення на відомі перші 16 простих чисел, якщо випадкове число ділилось на них відповідно воно не було простим. Далі за завданням я реалізувала тест Міллера-Рабіна, в якому спочатку знайшла розклад, а потім перевіряла простоту числа за основою випадкового числа x. Для пришвидшення роботи алгоритму, я також зробила функцію піднесення до степеня по модулю за схемою Горнера. Вкінці функція random\_number() обов'язково поверне сильно просте число.

У функції get\_keys() підбираються 4 випадкових простих числа, такі що добуток перших двох, менший за добуток наступних двох. У build\_rsa() заповнюється просто словники, по суті збереження даних в глобальних змінних. Далі реалізовані усі необхідні функції: шифрування і розшифрування, а також шифрування і розшифрування з аутентифікацією.

### Дані отримані під час роботи коду:

Частина кандидатів які не підійшли:

Posibble:

109297875822414814348084232951008841716105662358627595749529275077601836144065

Posibble:

46722840711211259511013336924004760338761190672639453774877670467580702826574

Posibble:

96560375696149842241586990645261717027881610329234429923905965865318218605928

Posibble:

27662551304222064343718660245194097547696447748906626246809424577602295148398

Posibble:

79832697958791951973644360655825810839588335920242670302420496543153316520882

Posibble:

49781221167390007609657894563207369044763927724657943445525968943668648628090

Posibble:

45857193600740887205887490547214766200506456957589475071112702482433174017316

Posibble:

41239789166448552833517572263322925943878746467539330831073669524200273742184

Posibble:

10619759447575499754781147557068282811382343721054708050661497047056221829583

Posibble:

101643900455959022770538996118296358706922574700926962758550471460683194871688

Posibble:

41591442324983450237475078533743293284848907900257388039182821387744214648368

Posibble:

63424668668166169917476575381547736852106139409697714400737381798585152787168

Secret A:

754169202318114348295540999292418073339956311196874883581328063942  
394191094573945948166800159042739610798878441095991631237938660607  
51884267824460895495

Open A: {'e':

350693554977020710026674916257562785659316283605297496011739746291  
411245554700757109688735478892177324468430404000755397669710047668  
986118163299612605771,

'n':

583723030498605961098618749678881556909728390746374066062011702029  
529680792349666848221455646514990635882672984169994957192028043106  
263313959953722216137}

Secret B:

254300151681480049268812310495309380156339847026616753329615656414  
788857603751538663006060682340839922949328147600601300172303526425  
5505886787420929698971

Open B: {'e':

166565755026037223064225936481308012043791616367798717489269193721  
599270939771913382657143168012659045697069937919893694568250992985  
7834207484909768970247,

'n':

292812323913472889977057985370726873162931198918225537441477818041  
350072221218199081054292116390621489184134483029055868771626408752  
3257710482033829154993}

```
its cool
Secret A: 75416920231811434829554099929241807333995631119687488358132806394239419189457394594816680015904273961079887844109599163123793866060751884267824460895495
Open A: {'e': 350693554977020710026674916257562785659316283605297496011739746291411245554700757109688735478892177324468430404000755397669710047668986118163299612605771, 'n': 5837
Secret B: 254300151681480049268812310495309380156339847026616753329615656414788857603751538663006060682340839922949328147600601300172303526425505886787420929698971
Open B: {'e': 1665657550260372230642259364813080120437916163677987174892691937215992709397719133826571431680126590456970699379198936945682509929857834207484909768970247, 'n': 292
Plaintext: 149306126317636225209864496435952615457657406618929399541987905015815843347072348064578012054530895417774493043341523107218622425357089175779465893019175
Ciphertext: 1778754936278857047952751841636991232130846489254553993916855989011082878287461937340467310652302231806076854980383689208537398682993290552571609141881580
ok
```

Випадкове к =

149306126317636225209864496435952615457657406618929399541987905015  
815843347072348064578012054530895417774493043341523107218622425357  
089175779465893019175

Надпис its cool означає, що А пройшов аутентифікацію, а надпис ок, що повідомлення розшифровано правильно.