

**Національний технічний університет України «КПІ» імені  
Ігоря Сікорського  
Фізико-технічний інститут**

**Комп'ютерний практикум 4  
Криптографія**

Виконали:  
студенти ФБ-21  
Князян Кирило Андрійович  
Новіцький Олександр Костянтинівич

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

## **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 роботи**

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  $p_1, q_1$  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,  $p_1$  і  $q_1$  – 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  $(e, n)$ ,  $(e_1, n_1)$  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

1. За допомогою вбудованої у python `random.getrandbits` генеруємо числа довжини 256 бітів (можна і більше), які потім проходять перевірку спочатку діленням на набір простих чисел, а потім тест Міллера-Рабіна, що можна побачити на скриншотах, тобто генерується число заданої довжини біт, якщо воно проходить перевірку, то повертається, якщо ні, тоді виводиться повідомлення, що перевірка не пройдена, потім, коли дві пари простих чисел згенеровано, йде перевірка на умову  $pq \leq p_1q_1$ , якщо вона не виконується, тоді генерація йде по новому і так поки не буде отримано дві пари простих чисел, що відповідають умові:

```
Кандидат 65279287345951393021462762674795437823283891904020567927159202641254737266597 не є простим числом
Кандидат 91083694266842771501777442627422629019487149533018277081361908199946481898547 не є простим числом
Кандидат 63548573770013265920892136681016657760602118699836945432215493159814502612555 не є простим числом
Кандидат 75225428014574069932605596561308784454912396205059960286248972929680085284567 не є простим числом
Кандидат 81607475584229798392001519608466093617417425625049289518158833954398488079961 не є простим числом
Кандидат 108292665407053132861044172249297101536021883804273607418696487138284268120309 не є простим числом
Кандидат 62281032821678281258258293759473937419623764421008223697725356792336320272639 не є простим числом
Кандидат 65161415661441453802241431031264028162167719347412002040732608062700159837163 не є простим числом
Кандидат 102003647586939773648845338695376888422318876388385708565710557952058870606557 не є простим числом
Перевірка потенційних пар простих чисел: p=89131363530857972885396169923362023024570931226593825288274727763943628425897, q=926159486834575989238413470119
65330580915069033495911505148835094236425767147, p1=69376327261000217212500730245172426322318157870048569263569159860786835384721, q1=10960811032831863427
1109010694808874464582664165614484240905292571622473301371
Кандидат 65616487552215748698030316659220255516374784524583774652066768738495834828879 не є простим числом
Кандидат 90929431699861328461033540298534708651438577276095661935282247609598619402601 не є простим числом
Кандидат 104516961243605524632699617169054456557970174860720711059442474720043354785107 не є простим числом
Кандидат 103728826011011362235542653525795471785668033300414003064862555574804244770213 не є простим числом
```

```
Кандидат 92676042331496823084483593074044787757161159633894892974795632889618335397453 не є простим числом
Кандидат 95801847750405915000933416717921806719821166658709240454057118398784884942447 не є простим числом
Кандидат 84394291748741935291831286723496942248391035878722133469298630778529824853449 не є простим числом
Кандидат 6025196469364220455717837316488011611978432539052550017950247288643418416353 не є простим числом
Перевірка потенційних пар простих чисел: p=746911567215583158678647499536727785084036772179128495538363196126353906919071, q=586020860064517182129891926260
58453360369349566532240607459512413626383042199, p1=60773440811609281147606060930932466113431207945603379425783914451203703496481, q1=98231656441727840297
003107953561067945123221102230079265899044902449181087063
Пара простих чисел для абонента А: p=746911567215583158678647499536727785084036772179128495538363196126353906919071, q=586020860064517182129891926260584533
60369349566532240607459512413626383042199
Пара простих чисел для абонента В: p=60773440811609281147606060930932466113431207945603379425783914451203703496481, q=982316564417278402970031079535610679
45123221102230079265899044902449181087063
```

2. Тепер на основі простих чисел генеруємо ключі для RSA, тобто для абонентів A і B на основі їх пар простих чисел обраховується  $n = p * q$ ,  $\phi(n)$ ,  $e$  обрали 65537 - ту, яку використовують найчастіше, і відповідно обрахували  $d = e^{(-1) \bmod \phi(n)}$ . В кінці просто повертаємо відкритий ключ  $(n, e)$  і секретний -  $(d, p, q)$ :

```
Генерація ключів для абонентів A і B
Ключі абонента A:
Відкритий ключ (n, e): (4377057590118124776673482654119121930568896429595961678852741144032800121820083478502130745910531376814064530455313723018876922679
059163542019968270877129, 65537)
Секретний ключ (d, p, q): (1215533944796830354386947591512703040059110350163213185759492940192516627510003408329968259043767635324488813124601699307685566
4336332575975301857157433, 74691156721558315867864749953672785084036772179128495538363196126353906919071, 586020860064517182129891926260584533603693495665
32240607459512413626383042199)

Ключі абонента B:
Відкритий ключ (n, e): (5969875758587684469228839288246889530308826804802506817573817730088699211336508895811071846088510210105171492295216512990689163382
626892595262869575125303, 65537)
Секретний ключ (d, p, q): (2841605112976529548462603439230689822512227477239028337797202253537955229834100353095466200577194736121694202150759456754663797
350766114783895936870187073, 60773440811609281147606060930932466113431207945603379425783914451203703496481, 9823165644172784029700310795356106794512322110
2230079265899044902449181087063)
```

3. Тепер відповідно йде шифрування, розшифрування і створення та перевірка підпису: спочатку генерується повідомлення, потім воно шифрується, розшифровується і згенероване повідомлення порівнюється з розшифрованим для перевірки правильності роботи, після цього генерується цифровий підпис і функція Verify перевіряє його. Можна побачити усі кроки нижче і впевнитись в правильності роботи усіх функцій при роботі з абонентами A і B:

```
Перевірка шифрування, розшифрування та підпису для абонента A
Повідомлення: 39296874997903101294495579533739453570874157668439021292230880210655063828999465261579032802441327307250651784067982980846435068806250195980
14149962349920
Зашифроване повідомлення: 37092369893195881246300058662986920452535276486598713700476067632506107650006534864643448104788509102997159134349626811887929390
46320894134205578348325841
Розшифроване повідомлення: 3929687499790310129449557953373945357087415766843902129223088021065506382899946526157903280244132730725065178406798298084643506
880625019598014149962349920
Повідомлення розшифровано правильно
Цифровий підпис: 17899014582245076292929179032318471142599850335360387895238501501685951822339557474022113642496791705166125964729910763461822632873640552
08629696874291540
Підпис дійсний
```

```
Перевірка шифрування, розшифрування та підпису для абонента B
Повідомлення: 5195903976579411180035104315997207850448488038544899760099741856038291672734090632405232787047516857945497852199551120174655365528229686180
23023104141142
Зашифроване повідомлення: 41349737033653531810636738232320466862118112806173438797036329499142141051871854615758402643645771574369141061397179808675317349
85499016039127731381267267
Розшифроване повідомлення: 5195903976579411180035104315997207850448488038544899760099741856038291672734090632405232787047516857945497852199551120174655365
552822968618023023104141142
Повідомлення розшифровано правильно
Цифровий підпис: 35755676077989806110133809665588769038860881011645766061763459208935444847499539134901225758673336532762094301535609567603405638747495531
76929498261904613
Підпис дійсний
```

4. Тепер відповідно робота протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA: спочатку генерується ключ  $0 < k < n$ , потім за допомоги SendKey отримуємо шифровані ключ і підпис (відправник A шифрує ключ за допомогою відкритого ключа отримувача B, створює підпис своїм секретним ключем і шифрує цей підпис відкритим ключем отримувача, повертає шифровані ключ і підпис), ReceiveKey розшифровує ключ і підпис і перевіряє підпис,

після успішної перевірки повертає ключ (отримувач В розшифровує ключ та підпис своїм секретним ключем, верифікує підпис, використовуючи відкритий ключ відправника, та в разі успішної верифікації отримує ключ):

```
Перевірка протоколу розсилання ключів  
Ключ: 2274890395309839278556685385427322788692833476592796921703398399785680664241746801467286261536407813823313647506753408300516490370098149111602094062887246  
Зашифрований ключ: 1094705864935287108745072449986251911604883619391581388700193595784378221553751280883765805291193762168420484334982832772030079413994746263140485250234569  
Зашифрований підпис: 4699103444653660947960802018026648088391588310577134346464729585791120826580110474314862548963985371403135727065658055164439352386092896005151969719325691  
Отриманий ключ: 2274890395309839278556685385427322788692833476592796921703398399785680664241746801467286261536407813823313647506753408300516490370098149111602094062887246  
Ключ передано успішно  
PS: D:\3>
```

## Висновки:

У результаті виконання четвертої лабораторної роботи ми детально розібрались у роботі RSA. Ми навчились генерувати та перевіряти числа на простоту за допомоги ділення на прості числа разом із тестом Міллера-Рабіна, отримувати відкритий та секретний ключі на основі пари простих чисел, також шифрувати, розшифровувати та підписувати і перевіряти підпис, а в кінці використали розроблене раніше для реалізації роботи протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA.