

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

Фізико-технічний інститут

КРИПТОГРАФІЯ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

Варіант 3

Виконали: студенти групи ФБ-12

Куцаєнко Дмитро та Федірко Ярослав

Київ – 2023

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 і рекомендації щодо 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 A , p_1 і q_1 – абонента B .

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 і B – 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 A і 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 і B , перевірити правильність розшифрування.

Скласти для A і B 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(). Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою

<http://asymcryptwebservice.appspot.com/?section=rsa>.

Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

Виконання роботи:

Згенерували p,q,p1,q1.

```
For A:
p= 108511168536341681422308437507991743096698030525277531815853537124123364799027
q= 94049532708379816158231266820069167544008383720079216703247209373205796937427
For B:
p1= 77018137063231381172754018773083212436142639784989286854141556598191700916921
q1= 84081655118444202162950348291605321744070144415902331175961910389842999824383
For A:
hex_p= 0xfe7251988643e9c2c12b6931b2791c0fa2a75224620f1f9f7a8e2151807b233
hex_q= 0xcfee262be978fbb3cea745cdad6e67e21702e24181c25e0fad6c81c8fdeded3
For B:
hex_p1= 0xaa46b885182091016c0513cfc005379ca2fe0446434294c8d0d437233cc93ab9
hex_q1= 0xb9e487c340cc4d74be75d4e45875357a608017af4180ccebc16495fe07e4cfff
```

Згенерували публічні та приватні ключі для Аліси та Боба.

```
Public key A:
e 30929107483983382814997466748316965549199058952253685745482847258725231115954880004843116799920694015131610274952461934539027517457413357286499535204251
n 10205424694483181748749772167974713866113232052357913389696366219310028075680350407009585189659304516589985070146991433894288148800728573317845053949483529
Private key A:
d 4683212194899462907985710976148196316598589729643889412130723597879588505953510244176917819494305318678378406928457132920482983745822518903637527979430895
p 108511168536341681422308437507991743096698030525277531815853537124123364799027
q 94049532708379816158231266820069167544008383720079216703247209373205796937427

Public key A:
hex_e 0x3b0dd533e271a830d5935a2b5ae1a4feddc8c50fb5dc38c39c92ab2d8e3123e4efa2fd9d7bca810389ac538e22191b860e747237a52f48e7c13125107ccaa9b
hex_n 0xc2db13a99925a25909b8b468dcdb3b766feb8d74b3e461baad2b7d4f271a11049d0fedfdfff8772c14c169ac603754e171dcc91bd3f7f98036491b04801a1a09
Private key A:
hex_d 0x596b107aa67184e0301d523d62c071d6e0d3eef7c22b19148fbd098a666dd9a4be1538bd98a57335c5dae665141adf42d8468c64f83781e189d99d8db00513ef
hex_p 0xfe7251988643e9c2c12b6931b2791c0fa2a75224620f1f9f7a8e2151807b233
hex_q 0xcfee262be978fbb3cea745cdad6e67e21702e24181c25e0fad6c81c8fdeded3

Public key B:
e 160070853793375957444457062756570935543142321727511912276407832136303292955073030013791974387115167180472567971895700096645671357575373868028076407606409
n 6475812438415685973472195549930730055291711194695202563222456575272836727511516861883477133849301430867826818349855596175838741786239994560601463073084743
Private key B:
d 4458607405974696128235238551892276545608775105938289426701361834096934000474665834556062832359744319436187619717832457686982681856245840653345794056064329
p 77018137063231381172754018773083212436142639784989286854141556598191700916921
q 84081655118444202162950348291605321744070144415902331175961910389842999824383

Public key B:
hex_e 0x1e9019b0a87208520b108b6fe7295592dd631da85a588ddaa8f03860b95154bb548023b7e4b8b2a17fa1623273247ef5ac8dc38c618493c54615d5a886b4c89
hex_n 0x7ba51calaf3f54e6f5ebf386edfaf0d1d8875ebf7e62db7ee3a452e8ea52ad8a961d04ecd6d5af94c44f9f6259f4aa6b2c91cec624725e71861dfea19eb11547
Private key B:
hex_d 0x552138127d57a50520a9c3d749c4306282715e96752791266073abff5f065fb5f5592fe628ef2c2967bab73f9a81fffa7301464ece4b1f29376e86bea11750d49
hex_p 0xaa46b885182091016c0513cfc005379ca2fe0446434294c8d0d437233cc93ab9
hex_q 0xb9e487c340cc4d74be75d4e45875357a608017af4180ccebc16495fe07e4cfff
```

Далі організовуємо роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Вивід наших значень повідомлення, зашифрованого повідомлення, зашифрованого повідомлення в шістнадцятковій системі числення, сигнатури, дешифрованого повідомлення, що збігається з початковим як це і треба.

```
Hello
Encrypted: 3523886715441909863585145216703218158463160314090342261385042904476833792872493969122617745589860657816533621921524545902915378441383487146265523856613520
Encrypted hex: 0x43486643927ed897214186ab778df378a17833ac7141ca331e4d80f4c061f05bc9ca6278f15cbf44c98d02501b6ff83e084119a85ce8753208f2c25b39f82090
Signature: 2230432356571484781605273711357547995088968066494497987580073574740382676411264846513469227937447284531981040132934881693355647436143288853722371874030527
Message is verified
Decrypted: Hello
```

Далі здійснюємо перевірку з сайтом.

Наші значення зашифрованого тексту збігаються, тобто перевірка є успішною.

Encryption

Modulus

7bcfd39830d88dcc709aaccfc89d8abcf7bd7acb9a8b3b8dd606ac45f06c45f386feb070c021ac9c6b89911d2990e0c

Public exponent

511832724eb5c7a8eae1b059c8b0d1ac61dc47d603a1370465d55ae025c958490c19f2bf51d74f9ba5fbb3c2ea1433

Message

hello

Text

Ciphertext

474213D977D99A3648C94EEDFEC3B3B6CA4CE3ED17F5707C293599F6229CC8B768049D067E30346EF7BD

```
Encrypted: 3732090329893463502408968651695460880109912356955021475144171093092200351412329220083567041521251878745364449031047763717414414460383143195035049174295238
Encrypted hex: 0x474213d977d99a3648c94eedfec3b3b6ca4ce3ed17f5707c293599f6229cc8b768049d067e30346ef7bd7b3b6fcd285d84c93d270dffba3b8917c3c9704e2c6
Signature: 3272181324095595039715401663981689128193821057001779664111352552075628413733005189626097134574895496255303767505017022876015779390000469144521248481446168
Message is verified
```

Висновок:

Завдяки цій лабораторній роботі ми на практиці засвоїли принцип роботи асиметричної криптографії типу RSA. Ми реалізували функції перевірки на простоту чисел, що допомогло нам для подальшої генерації ключів. Більш детально ознайомились з системою захисту інформації на основі криптосхеми RSA та змогли реалізувати організацію з використанням цієї системи засекреченого зв'язку й електронного підпису.