Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

Комп’ютерний практикум №4

З дисципліни «Криптографія»

Виконали:

Студенти групи ФБ-33

Рудий А.О., Шкуропінський М.М.

Київ – 2025

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 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.

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

Для початку перенесли функцію extended_euclid з минулої лабораторної роботи, щоб мати змогу організувати перевірку на простоту використовуючи тест Мілера-Рабіна функція is_prime. Також за допомогою функції generate_prime ми генеруємо випадково за допомогою бібліотеки random самі наші числа, які будуть проходити саму перевірку.

Відповідно пройшли такі кандидати

Для A

Public Key:

n	882198440510916379689012535366541733803681162699088980340838479066819768647193922519507591708718687423744291665612272024717585388085624444101877127724889 3
e	675066040946906095769231025664284860013279944200108786534035198488058844799444392984325094368796752031929482951538093097630717031429321590975209068518884 9

Private Key:

p	920794295971678757155713347586559917092627464093357816572925856195034308337 53
q	958084171861606008673160099417217440860938266408295199819574321693622636133 81
d	323907365139195617860523731334242359535052444333592689745550556750096370878726949935920979484211470546125476072589612371861820299550982288032404376406020 9

Не пройшли для p (всього 87 штук):

1	6323311589204294846619783637542242610878535999362960471996658728880944924282 5
2	1068756702901165533032847233738456076459648517883777047375926753552121601023 04
3	9500277631926861186630453026176551733181891206026164492763434368601854124820 3
4	6305440707051169863551251239302373229922692618695471923171363854731463868659
…	

Не пройшли для q (всього 28 штук):

1	1126595160169954538372477141633969874005331219066934151106043027320820548025 54

2	852179736657750799128057696503555064709172935679264871817253345512844605464 13
3	950363286593451657115277083541804258438852883668774266261604272832800942088 6
4	110538676592221092168454322340411673586548136408296381478612035679501493319175
…	

Для B

Public Key:

n	107049504795685373270257899056356014615811646538221455311141302629465835017633910942248390675702800728563752431489152424710116952576752537499030001108628 23
e	809081059899496202998820589350033306775035764051463322815370503275707847804095547956218656875757704979272875512588634547791939352757586252911880586439495 3

Private Key:

p_1	997022533332446993033200826165954226128485359037739785939560710610895214312 93
q_1	107369192988931984265079827131552491314686614847208993990925874325463691020211
d	789848806209997598309015528390683865763078695065276848129667946026595234688006898162746144350089531605629647725023181424042936186704741458384035029483429 7

Не пройшли для p_1 (всього 156 штук):

1	936162099678449097171793478824402423806975614143239927587999790645821148554 56
2	111176362895677448959418727168443413931884140889203299184610540320866076986599
3	901625497292150253370966971766854460921899346955566373636808590994702577749 33
4	599616346358951455513081131209986408305448926311613816580091011322764686810 81
…	

Не пройшли для q_1 (всього 245 штук):

1	982448865501768965875243768942549586101903120001418665334382960584456109525 94

2	937597799763449929952063830284208136494596435485693213975139108475269090641 16
3	656972656627821143270376640966057357065615459317035035339650670729136083555 27
4	657846737161624705770451327008037991559030420180868224936975129103147090163 60
…	

ВТ:
(288301002783057748244205459754732521628935040893161905385985807638258587054229753286752313576744930434093162913927659730382299628069050869759813897618363485870542297532867523135767449304340931629139276597303822996280690508697598138976183634)

```
Початкове повідомлення
Message = 288301002783057748244205459754732521628935040893161905385985807638258587054229753286752313576744930434093162913927659730382299628069050869759813897618363485870542297532867523135767449304340931629139276597303822996280690508697598138976183634
```

ШТ:
(103762797183609662944940919069711688016691263058561094016495290054915648078977621363102831818855947590996168930380753094755682236141855645176645637782457945645177664563778245794)

```
Шифрування повідомлення (Encrypt(M, e_B, n_B)):
Encrypted message: 103762797183609662944940919069711688016691263058561094016495290054915648078977621363102831818855947590996168930380753094755682236141855645176645637782457945645177664563778245794
```

Підпис до шифрування:
(267586230172557470755021276677265400766015102480688536456344590078598272210691306309662680653865981258599000136624993699797102581929242375703796622518127275037966225181272)

```
Підпис (S = M^d mod n_A):
Signature (до шифрування): 267586230172557470755021276677265400766015102480688536456344590078598272210691306309662680653865981258599000136624993699797102581929242375703796622518127275037966225181272
```

Підпис після шифрування:
(433781031503393467365225227460031032764608971578421947339699403980398914539746796688772520536241446351521702010102007435528315397451977576532073397831952475037966225181272)

```
Шифрування підпису (Encrypt(S, e_B, n_B)):
Encrypted signature: 433781031503393467365225227460031032764608971578421947339699403980398914539746796688772520536241446351521702010102007435528315397451977576532073397831952475037966225181272
```

Кроки конфіденційного розсилання ключів з підтвердженням справнжності

Абонент А генерує випадкове повідомлення k (message), яке він хоче передати.

```
message = random.randint(1, min(n_A, n_B) - 1)
```

Після чого, він створює підпис. Абонент А підписує повідомлення k своїм секретним ключем.

```
def Sign(message, d, n):
    return pow(message, d, n)
```

```
Підпис (S = M^d mod n_A):
Signature (до шифрування): 267586230172557470755021276677265400766015102480688536456344590078598272210691306309662680653865981258599000136624993699797102581929242375703796622518127
```

Після чого, абонент А шифрує і повідомлення k отримуємо k_1 і підпис S отримуємо S_1, використовуючи відкритий ключ абонента В (e_1, n_1)

```
def Encrypt(text, e, n):
    return pow(text, e, n)
```

```
Шифрування підпису (Encrypt(S, e_B, n_B)):
Encrypted signature: 433781031503393467365225227460031032764608971578421947339699403980398914539746796688772520536241446351521702010102007435528315397451977576532073397831952
```

Після чого йде відправка, абонент А відправляє абоненту В пару зашифрованих значень (k_1, S_1)

```
def SendKey(message, sender_public, sender_private, receiver_public):
    signature = Sign(message, sender_private, sender_public[0])
    encrypted_signature = Encrypt(signature, receiver_public[1], receiver_public[0])
    encrypted_message = Encrypt(message, receiver_public[1], receiver_public[0])
    return encrypted_message, encrypted_signature
```

Дії отримувача (Абонент В)

Абонент В отримує пару (k_1, S_1) і виконує зворотні дії

```
def ReceiveKey(encrypted_message, encrypted_signature, sender_public, receiver_private, receiver_public):
    message = Decrypt(encrypted_message, receiver_private, receiver_public[0])
    signature = Decrypt(encrypted_signature, receiver_private, receiver_public[0])
    return VerifySignature(message, signature, sender_public[1], sender_public[0])
```

Абонент В використовує свій секретний ключ d_1 , щоб розшифрувати обидві частини повідомлення

Після чого йде перевірка підпису, абонент В, отримавши розшифровані k та S, повинен переконатися, що k справді прийшло від А. Він використовує відкритий ключ абонента А (e, n), і перевіряє підпис.

```
def VerifySignature(message, signature, e, n):
    return message == pow(signature, e, n)
```

Якщо воно рівне, вертає True отже перевірка підпису пройшла успішно, і повідомлення не спотворено

```
В приймає повідомлення і перевіряє підпис
Розшифроване повідомлення: 288301002783057748244205459754732521628935040893161905385985807638258587054229753286752313576744930434093162913927659730382299628069050869759813897618363
Перевірка підпису: True
Обмін завершено
```

Висновок:

У ході виконання цієї лабораторної роботи ми успішно реалізували ключові компоненти асиметричної криптосистеми RSA. Ми практично ознайомились з тестами перевірки чисел на простоту, зокрема реалізувавши тест Міллера-Рабіна та функцію генерації великих простих чисел. На основі цього було створено функції для генерації ключових пар RSA (відкритого та секретного

ключів). Також ми реалізували базові криптографічні операції: шифрування, розшифрування, створення цифрового підпису та його перевірку . На завершення, всі ці компоненти були об'єднані для реалізації захищеного протоколу конфіденційного розсилання ключів з підтвердженням справжності відправника, що продемонструвало комплексне застосування алгоритму RSA на практиці