

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ НАЦІОНАЛЬНИЙ
ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

ФІЗИКО- ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

з дисципліни

Криптографія

З теми: «Вивчення криптосистеми RSA»

Перевірила:

Селюх П.В

Виконали студенти групи ФБ-94

Резніченко Н.І та Белоцький Д.О

КИЇВ - 2021

Тема:

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

Мета та основні завдання:

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.
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

<https://asym-crypt-study.herokuapp.com/>

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.

Хід роботи:

Створено 2 екземпляри класу algorithm_RSA: Alice та Bob.

Для абонента Alice обрані прості числа p_1 та q_1 :

| | |
|-------|--|
| p_1 | 16467468260425182790496074357379176173879865978900605915609094071589 |
| q_1 | 16696093071164533511076270266584223450576262035878328764812815977427 |

Для абонента Bob обрані прості числа p та q :

| | |
|-----|---|
| p | 94640622186351625232736059525167679160229114821267850089707437193 |
| q | 46121804063990424063746602946365258150763154795244002112742585573 |

Числа-кандидати, що не пройшли перевірку на простоту. В даному випадку наведені перші 10 чисел, що не пройшли тест для першої пари, інші можна переглянути в файлі logfile.log:

Numbers that did not fit the first pair simplicity test:

51085107210182213445045771590250575181148254608929660199964018587
10050202174581883894973415792029585791456123942924490435966195919
22292817676632673950623692368465588584482777809087362366744183447
75646780479099123429088340073966758432336703811855288201631341541
86889915852937570606839485632628924838127344867165392703496200643
85055616200423008731839624691803587334371661168920742456894809399
10589731500585050367717930316898854300564376469921115248823370499
71010302449628241624882210368915368515426323808054513722265928207
77407327669373348603569883201518036440278671598789831213000601857
15609618359397162217395255477785087621468025868212592258827874009

Після успішної генерації пар простих чисел, обчислимо параметри криптосистеми RSA для кожного абоненту:

Для абоненту Alice:

- Відкритий ключ:

| | |
|-----|--|
| e | 9902683726655200503204981257804278820388947805280403939004456230691 7967625438666472413558598513063431805129599192340400404569860375551 |
|-----|--|

| | |
|---|---|
| n | 274942382722506768273734747602466204676031019576358748926059598470674533197637119418615977851775820021474196882851837959844604146021503 |
|---|---|

- Закритий ключ

| | |
|---|--|
| d | 19710356443461088716267641143655822762642444216901913961031600486774912996812748903627600172760121954376695000330600335967993638595295 |
|---|--|

Для абоненту Bob:

- Відкритий ключ:

| | |
|---|--|
| e | 3407740561347002904885450963422327010221910671318852111898913915167461073348434624054079200196914514943224289535780666921013473097 |
| n | 4364996232973054681427172597994319627167571911734913776053527631779723011968605805887596810274216537771421273617542850266325416589 |

- Закритий ключ

| | |
|---|--|
| d | 3279944873201021158664412434759871802088116353009739170888523746707081995662123592608142487653007455540721789567380741045916756217 |
|---|--|

Alice генерує повідомлення M (відкритий текст):

4284479227899675198196668437406766606502585673764873451493376493869083003259087377796886578183040867818712542723983699598891986716

Alice шифрує повідомлення M відкритим ключем абонента Bob й отримує значення C (шифрований текст):

2973490351304740039262822107823965589052365472286870953949826840954115794165177349839224551379968585147528473487375575478708827703

Bob дешифрує повідомлення своїм закритим ключем й отримує значення M:

4284479227899675198196668437406766606502585673764873451493376493869083003259087377796886578183040867818712542723983699598891986716

Після виконання цих дій перевіримо чи співпадає значення відкритого тексту. В результаті отримали:

Verification was successful (M = M_decrypt)

Alice підписує повідомлення цифровим підписом S:

111405583227132653209839697114515483432784359373961617641747185378338398121763014892508893653579773159349164862768090053141202820132695

Bob перевіряє цифровий підпис Alice й отримуємо вивід:

Sign verification state: True

Кроки протоколу конфіденційного розсилання ключів з підтвердженням справжності:

1. Bob обирає в якості повідомлення число k :
130056696318745476658192209384614787793013802699838224198211693998529970607053847788180554562557636367212329490970048181994000129
2. Викликається процедура `SendKey()`, що в якості параметрів приймає згенероване повідомлення k , та відкритий ключ абонента Alice (`Alice.e`, `Alice.e`).
3. Шифруємо повідомлення k відкритим ключем абонента Alice, отримуємо значення k_1 . k підписується закритим ключем абонента Bob й отриманий цифровий підпис шифрується відкритим ключем абонента Alice. Таким чином отримаємо пару чисел (k_1, S_1) :

| | |
|-------|---|
| k_1 | 28191631914019266726124735673447835843158541757883708982756686839637267114058554903046558398452201422761518865986358685674798806741031 |
| S_1 | 239331399467496804826036899359055441952207490088965071144984793121419329077627146170482340312921373342865985506856385180580922336555990 |

4. Alice викликає процедуру `ReceiveKey()`, що в якості параметрів приймає k_1, S_1 , за допомогою свого секретного ключа d Alice дешифрує отримане повідомлення й отримує значення k, S :

| | |
|-----|--|
| k | 130056696318745476658192209384614787793013802699838224198211693998529970607053847788180554562557636367212329490970048181994000129 |
| S | 1755712809060305994746852593337975254983464619594822346776606313582163598445831669612069666506124334148809647424360869857875333250 |

5. Alice викликає процедуру `sign_check()`, що підтверджує справжність отриманого повідомлення. В результаті отримуємо:
Sign verification state: True

Перевіримо реалізовані операції за допомогою тестового середовища: <https://asym-crypt-study.herokuapp.com/>

Для абоненту Alice локально виконаємо кроки 1-3 з попереднього пункту.

| | |
|-------|---|
| k_1 | 68576351494159637567703872815874189805351450538775333721784059791815258242749789698090763721861710702626460045468889573770812957970669473596405300622172662487268820018058082946951249354137681308100247230838626428131704927940089343185722299885943794366868777838194996268904602127921463565633333448635531592579 |
| S_1 | 125697083840476393032363059193924519286649142378896115061107789502806048255254659321601110280105083617864971229693854761653427725497156091612542632159997954203717178496606143831737372691217756013392174510711692749806830032121556407463942036330263028487881514868877176264093750695041960591253143953732861780748 |

Оскільки сайт працює з 16-річною системою числення, то переведемо отримані значення у неї:

| | |
|-----------------------|--|
| k ₁ hex | 0x61a7ed64d7dcde27ef9c71a0bd65e8bf99896fcf0c65ef268871c808dec3407072bb05ab29d23b11f163c5a348d48c5d6157f41b5e6d6f95e492fe87a8e59c00003f9fe25c2cc644752953a3231f4a7d631cde7ac120cb1dc5b384dfeff4496df5d1704a86927c1aecbc856f9dcfd80327fad379435ca5727c3b95d044170383 |
| S ₁ hex | 0xb2ffa387afd1e1410bfaf54ae0ac681a7d17f91ca6cb187694164995c4a8eec2798628357489009756f37a51b2470a2ff6bb1bfb08da1150b89c691c1db0b051aac318dc8d443500481f87908b2aa88239be7635cc8b9b81b876a6fe3ea3ae1d82bc8d305e3dfc92a52223626ec94788bb2967b85b2461bf6a54a86e5085630c |

Згенеруємо ServerKey за допомогою сайту:

RSA Testing Environment

Server Key
Encryption
Decryption
Signature
Verification
Send Key
Receive Key

Get server key

Clear

Key size
1024
Get key

Modulus
E2DBC75C336AA7F896CF5107099DBBE08577875824A2628C8D282F0F66F620790F540F52953FA301FC0EF8

Public exponent
10001

Процедуру ReceiveKey виконаємо за допомогою сайту, якщо все виконується правильно, то отримаємо значення ключа та значення істини при верифікації:

RSA Testing Environment

Server Key
Encryption
Decryption
Signature
Verification
Send Key
Receive Key

Receive key

Clear

Key
61a7ed64d7dcde27ef9c71a0bd65e8bf99896fcf0c65ef268871c808dec3407072bb05ab29d23b11f163c5a348d48c5

Signature
b2ffa387afd1e1410bfaf54ae0ac681a7d17f91ca6cb187694164995c4a8eec2798628357489009756f37a51b2470a2

Modulus
1969e4099d65555c50f8b8b6504792a7dd919199eb3a5ccace72df8f4b128027596f14ff10045ef55930054203ceac4:

Public exponent
d8cf6078a254ca1837441cd1b102d76e24ed9d5dc054f508e385ae9b8ec1cbd28b803271810d1408c0716aeaa698c

Receive

Key
10596BFED73571D5CCDA961A72C6A6CDD95D4714C9C2B5885031E90135FF3CAECDB6E43738CE017457AI

Verification
true

Повні результати роботи програми можна переглянути в файлі output.txt

Висновки:

В ході лабораторної роботи ми ознайомилися з тестами перевірки чисел на простоту й реалізували тест Міллера-Рабіна мовою програмування Python. Також ознайомилися з методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомилися з системою захисту інформації на основі криптосхеми RSA, з використанням цієї системи реалізували систему засекреченого зв'язку й електронного підпису, вивчили протокол розсилання ключів.