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:
студентки 3 курсу ФТІ
групи ФБ-83
Ракович Поліна,
Троцька Аліна

Перевірив:
Чорний О. М.

Київ-2020

Мета та основні завдання 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 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб 1 1 pq p q ; p і q – прості числа для побудови ключів абонента A, 1 p і 1 q – 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) , (,) 1 1 e n та секретні d і 1 d .

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 и 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.

Хід роботи

1. Спочатку написали функцію для генерації випадкових простих чисел заданої довжини `random_prime`, перевірка на простоту проходила за допомогою теста Міллера-Рабіна. Приклад чисел що провалили тест:
```
p провалило тест на прстоту p = 233829617906543994985113185320248648847
p провалило тест на прстоту p = 225736922569165813722500579710437962735
p провалило тест на прстоту p = 236908796711614580307085297504455339327
p провалило тест на прстоту p = 328044385143832165679749976375122168725
p провалило тест на прстоту p = 322200249529083415049265233372726964099
```
2. Далі реалізація классу RSA з зазначеним набором функцій . Основні складнощі виникли при тестах шифрування , бо значення e (10001), отримане з сайту, було в16й системі числення і передавалось у функції без переведення в десяткову.
3. Покрокова робота протокола записана у відповідні файли (`SendKey.txt` та `RecieveKey.txt`)

Простокол SendKey :

p користувача p = 29521930688662580418949433757293899775 9

q користувача q = 21519142025681664531726031706193271345 7

e користувача e = 65537

n користувача n =
635286619361660178396892408923078949610381082986746785443114893885489121428 63

d користувача d =
152848000581269476676720013181853134525169100707677270967984658633192942845 45

Отримані Modulus та publicExponent: n1 =
815022363680489193352802797341231429827378813156883357799840292758733352413 17 ,
e1 =65537

Операція SendKey :

k = 0x1607e72

Шифрування повідомлення 0x1607e72

Зашифроване повідомлення
0x57a6bfd8adfda2676131468cce943b88f4399c497817615f7f38277348160b02

k1 = 0x57a6bfd8adfda2676131468cce943b88f4399c497817615f7f38277348160b02

Створення підпису для повідомлення 0x1607e72

Підпис s = 0x1607e72

s = 0x529ec25738a487b8ae925e40e79364b542d540f598e7afb0a889fd7381346efc

Створення підпису для повідомлення
0x529ec25738a487b8ae925e40e79364b542d540f598e7afb0a889fd7381346efc

Підпис s = 0x529ec25738a487b8ae925e40e79364b542d540f598e7afb0a889fd7381346efc

s1 = 0x49a45704af28d179a3c9777ccce18b7f9780838ebe4f7986b1c7f5c5ce885824

Перевірка на сайті :

{"key":"01607E72","verified":true}

Простокол RecieveKey :

p користувача p = 21269040092198005815327434712638517028 9

q користувача q = 19364888036137615723446414797974144146 7

e користувача e = 65537

n користувача n =
411872580021536454095591295354641750777039454957166943977780644509971209739 63

d користувача d =
689167144134789318341128544922563046672330972957987070674721629663092061241 7

Отримані Modulus та publicExponent: n1 =
849908483299062696498268176581098851945615091203460438043653575780690644298 79 ,
e1 =65537

```
Отримані Key та Signature: k1 =
627810536422173095149441012849735578201389015205429983184658153083723512765 5 ,
s1 =
288875730010080882711242820884805345106585219501942255773084408471440970255 57

Операція ReciveKey :

k1 = 0xde1481e50fc45ee0428cbaca01ae45a32fce21dec1090471fd62345f42dd567

s1 = 0x3fddc97289a76c5d365d35fe63158c5929996b3c10567e4892d1e70a4c160e15

Розшифрування повідомлення
0xde1481e50fc45ee0428cbaca01ae45a32fce21dec1090471fd62345f42dd567

Розшифроване повідомлення 0x432c3b33e142c4b6

Створення підпису для повідомлення
0x3fddc97289a76c5d365d35fe63158c5929996b3c10567e4892d1e70a4c160e15

Підпис s = 0x3fddc97289a76c5d365d35fe63158c5929996b3c10567e4892d1e70a4c160e15

k = 0x432c3b33e142c4b6

s = 0x4e86234975c4dd91efe21021ceb633dde00e1efe1e3bb68506d072478b45f280

Результат верифікації True
```

Висновок

В цьому лабараторному практикумі ми ознайомились з тестом числа на простоту (тест Міллера- Рабінсона) і способами підбору простих чисел великої довжини. Реалізувала систему захисту інформації на основі криптосхеми RSA, а також протокол конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника.