Міністерство освіти і науки України

НТУУ«Київський політехнічний інститут»

Фізико-технічний інститут

# «КРИПТОГРАФІЯ» КОМП’ЮТЕРНИЙ ПРАКТИКУМ №4

## Вивчення криптосистеми RSA та алгоритму електронного підпису;

## ознайомлення з методами генерації параметрів для асиметричних криптосистем

Виконав:

Студент групи ФБ-81

Шмалій Григорій

Перевірив:

Чорний О. М.

**Хід роботи**

**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 pq ≤ p1q1 ; p і q – прості числа для побудови ключів абонента A, 1 p і q1 – 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) , ( e1,n1 ) та секретні d і d1

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.

Основні моменти створення роботи:

1. З попередніх лабораторних робіт було взято функції пошуку НСД та оберненого елементу за модулем - gcd, Converse відповідно. Вони потрібні тоді, коли відкритий ключ е генерується випадково, а не за формулою $2^{16} + 1$.

2. Для швидшого піднесення в степінь була створена функція Gorner, яка шукає степінь числа за модулем.

3. Була створена функція Miller_rabin - яка перевіряє чи число є простим

4. Функція find_prime() шукає просте число генеруючи випадкове число та перевіряє чи воно просте

5. Функція generatekeys() - генерує ключі заданої довжини, використовуючи попередні функції

6. Функції encrypt, decrypt, signature - для шифрування, розшифрування та пошуку підпису

7. Було створено клас Body, який має функцію GENOPENKEY для генерації ключів заданої довжини, CREATEMESSAGE яка створює повідомлення та VERIFY для перевірки повідомлення та підпису

8. Було створено клас Hacker, який має функцію GENOPENKEY для генерації ключів заданої довжини, CREATEMESSAGE для підміни повідомлення.

**Виконання програми**

Створюються два об'єкти класу Body та один об'єкт класу Hacker

Генеруються випадки передачі повідомлення між об'єктом А та В. Об'єкт А створює повідомлення, шифрує його, підписує, та надсилає об'єкту В. Об'єкт В в свою чергу розкодовує повідомлення за допомогою свого закритого ключа, та верифікує його, розкодовуючи підпис та звіряючи з розкодованим повідмленням, тобто перевіряє підпис. В один з цих випадків об'єкт класу Hacker перехоплює повідомлення та змінює його на своє, в результаті чого об'єкт В повідомляє про помилку перевірки підписів.

Приклад виконання: Для прикладу я взяв 5 тестових випадків, в одному з яких бере участь об'єкт класу Hacker

Випадок 1

```
---------__PERSON A__----------
Key e:  0x10001
Key n:  0x1d7694251bfb2713714e1dbbbe9627436f96993badb0b1c0ee4c3098421d7b829
Secret message:  0xfec
Encrypted message:  0xf3208f9182082f1484d9cd0d6623cb90bee3ed93879f087f2026587d3a770b05
Signature:  0x14ade276edef45d75766118277d707274c21a424c89d7a71cd0269c68023e7460


---------__PERSON B__----------


Verified!!!
Key e:  0x10001
Key n:  0x1eb78dbb29367c4b640529402c015653db6ecdd90cf1ffc378a8569ffb248a291
Encrypted message:  0xf3208f9182082f1484d9cd0d6623cb90bee3ed93879f087f2026587d3a770b05
Decrypted message:  0xfec
Signature:  0x14ade276edef45d75766118277d707274c21a424c89d7a71cd0269c68023e7460
```

Випадок 2

```
---------__PERSON A__----------
dfsdfs 4425
Key e:  0x10001
Key n:  0x1e1206b111276bc8176ee8baa04bc0cbc3f2612e330e5ac515d4399d8ed403e35
Secret message:  0xd0d
Encrypted message:  0x800212af517de59824a87326948426ea30af940379144992231b5292aa1a796d
Signature:  0x14001433b8f0e677324eb99d7c06e0ece68583a10618ad626da1de0e67c81f2d7


---------__PERSON B__----------


Verified!!!
Key e:  0x10001
Key n:  0x1f3d3d0d960d851599ee89aa8c49fff412bc4877e055080a7984ff33d06e86b1d
Encrypted message:  0x800212af517de59824a87326948426ea30af940379144992231b5292aa1a796d
Decrypted message:  0xd0d
Signature:  0x14001433b8f0e677324eb99d7c06e0ece68583a10618ad626da1de0e67c81f2d7
```

Випадок 3 (підміна повідомлення)

```
---------__PERSON A__----------
dfsdfs 3880
Key e:  0x10001
Key n:  0x1e1206b111276bc8176ee8baa04bc0cbc3f2612e330e5ac515d4399d8ed403e35
Secret message:  0x165a
Encrypted message:  0x1ce9e40aad751746e8dd3bbca1af0a48be3fe2e7aea3b1e18f4d38255903dd1c2
Signature:  0x1ba697c40001e873a89f7d93c5414b4f3eab81c75dda913f17f42f26d76d5fa63


--------------__HACKER__---------------
Key e:  0x10001
Key n:  0x1e309aae3eec1ba6001e19d9e7f8519f8c3b8addd7bf36e818167254273b21ff1
Secret message:  0x106a
Signature:  0x137b521ab5bb3ded62610855ee90933b8180a7835f95da17400e284d65f68b0a5
Encrypted message hacker:  0xe6d0c92ddc553106f17fa38511b0a52a216db110aa50be1ffbe0edadec246fe7


---------__PERSON B__----------


Oops, something went wrong
Key e:  0x10001
Key n:  0x1f3d3d0d960d851599ee89aa8c49fff412bc4877e055080a7984ff33d06e86b1d
Encrypted message:  0xe6d0c92ddc553106f17fa38511b0a52a216db110aa50be1ffbe0edadec246fe7
Decrypted message:  0x449d927baeead8f25259487a80abdcda160215f121d85d03104e34be503c7960
Signature:  0xcc6028e8d3d2a0b9e8b6c018bb9ac3e72abaacef6fe0ff8533bdf10b7bb119d3
```

Випадок 4

```
---------__PERSON A__----------
dfsdfs 3751
Key e:  0x10001
Key n:  0x1e1206b111276bc8176ee8baa04bc0cbc3f2612e330e5ac515d4399d8ed403e35
Secret message:  0xcb8
Encrypted message:  0x5d147859515f85e7bf3719e39db8d0415201309b239fa4c62a1be03c2e9fcb94
Signature:  0x1a4024f39e375c421870dfc5e38b27a0d988599c95057844c6ba85edac95d9d13


---------__PERSON B__----------


Verified!!!
Key e:  0x10001
Key n:  0x1f3d3d0d960d851599ee89aa8c49fff412bc4877e055080a7984ff33d06e86b1d
Encrypted message:  0x5d147859515f85e7bf3719e39db8d0415201309b239fa4c62a1be03c2e9fcb94
Decrypted message:  0xcb8
Signature:  0x1a4024f39e375c421870dfc5e38b27a0d988599c95057844c6ba85edac95d9d13
```

Випадок 5

```
---------__PERSON A__----------
dfsdfs 3934
Key e:  0x10001
Key n:  0x1e1206b111276bc8176ee8baa04bc0cbc3f2612e330e5ac515d4399d8ed403e35
Secret message:  0x270a
Encrypted message:  0x19ad2ad9c32488c1637f4e56b57050b5632c4d2197eedd159159072dd3614f99d
Signature:  0x22ec23daf6195d4cd9eada235e755a1ed972834bd610bc6c4916924e67b73fd1


---------__PERSON B__----------


Verified!!!
Key e:  0x10001
Key n:  0x1f3d3d0d960d851599ee89aa8c49fff412bc4877e055080a7984ff33d06e86b1d
Encrypted message:  0x19ad2ad9c32488c1637f4e56b57050b5632c4d2197eedd159159072dd3614f99d
Decrypted message:  0x270a
Signature:  0x22ec23daf6195d4cd9eada235e755a1ed972834bd610bc6c4916924e67b73fd1
```

**Наступним блоком йде з'єднання з тестовим середовищем:**

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

Яке далі називатиму 'сервер'.

Сервер генерує відкриті ключі, та повідомляє їх об'єкту А. А генерує повідомлення , кодує його, формує підпис та відправляє знову на сервер. Сервер в свою чергу розкодовує повідомлення, звіряє підпис та повідомляє чи підпис верифіковано.

**Приклад виконання**

```
--------------------Connect with server----------------------
Server keys:  n:  0x9df7c83cf119287fd41ce9536104ce9486f85bb3d24f9c5ead30deb79bb581aaf77c36a8fa1
e:  0x10001


---------__PERSON A__----------
dfsdfs 3355
Key e:  0x10001
Key n:  0x1e1206b111276bc8176ee8baa04bc0cbc3f2612e330e5ac515d4399d8ed403e35
Secret message:  0x2575
Encrypted message:  0x94ec8651868335e6bba0bcfd6b81c6d4e246d0dd8a989797c373b40677692b5af837d14317e
Signature:  0x141e122236c2b581557a2b429ec6e1cd89e7f3b8d2221aef536832a6d921a77b9
Decrypted message: 2575
Server said: True

Process finished with exit code 0
```

**Висновок**

В ході лабораторної роботи була створена тестова модель криптосистеми RSA, тести на псевдопростість числа, алгоритм Горнера для пошуку великих степенів за модулем. Протестована робота з готовим тестовим середовищем, посилання на яке надане вище. Згенеровані тестові випадки з можливою підміною повідмолення. Покращив навички

програмування на мові python, ознайомився з модулем requests та удосконалив навички роботи з сервісом контролю версій.