Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

Лабораторна робота №4

З предмету «Криптографія»

На тему

«криптосистеми RSA та алгоритму електронного підпису»

Виконали:

Студенти групи ФБ-83,84

Мельниченко А.

Іванченков М.

Перевірив:

Чорний О.М

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.*

Генерацію числа з перевіркою на простоту з використанням тесту Міллера – Рабіна реалізовано методом *random_s()*. Допоміжні *is_Prime()* та *trial_composite()*.

```
Тестування числа  0x1472c16f9541a901a3fcf2901f111f5ce54194a3eb99f87717f82b4b8c41d7f2
Тест не пройдено
Тестування числа  0x560e3b2482f81002fccd8b7e7e027048baa193781ab41024311166d910000e11
Тест не пройдено
Тестування числа  0x33518748737a76b54f7e127120fe5dd19c36f4e7d1566906110e6ef815c8f5cc
Тест не пройдено
Тестування числа  0xb53dbf76c14530bff56a549e4d2855b341b57ee4f641525ebd5b97a491605dcb
Тест не пройдено
Тестування числа  0x3c55c7dcded9548c5d62b751c9dabd23b4f304bd3e340747a5d0f3cf63cd81f
Тест не пройдено
Тестування числа  0x409e7a4af058719dbebcf0f64bf4b25320473755c988ba6fc47f40d81c90d13b
Тест не пройдено
Тестування числа  0x28930455f0de610b460c6f7c0b0bd5937d35da02bc750c43cdb21b682bcc991c
Тест не пройдено
Тестування числа  0x49a5bbefd3140f6e4e5dcba8806dc0013fba338c51c13987e2db28ad1e13b113
Тест не пройдено
Тестування числа  0x5b06f3bb001e518bb901b6aa75cf117e6edeef08f2c8b46b7c5a98999fe2218d
Тест пройдено
0x5b06f3bb001e518bb901b6aa75cf117e6edeef08f2c8b46b7c5a98999fe2218d
```

Метод генерує прості та використовує тест Мілера-Рабіна. Повертається перше число, яке пройшло тест. Метод *is_Prime()* виконує повторні перевірки не більше аніж $log_2\ n$ разів, де n – число, що перевіряється.

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*

$$p \cdot q < p_1 \cdot q_1 \quad (*)$$

p, q, p_1, q_1 *- прості числа для побудови ключів абонента А та абонента В.*

Реалізовано функцією *pair_generator()*.

Для генераціє чисел функція використовує *random_s()*, описаний в попередньому пункті. Додатково перевіряється умова (*), а *check_pairs()* забезпечує відсутність повторів в парах.

В результаті роботи методу згенеровано наступні пари ключів

```
p =  0x7b98a28b92cd8330addaec48b756e028ff43a1d2411bb868dfbd0c13cd5f54ad
q =  0x54efb50c057246cdefe116e81ef669788e98c5afffa2a93a6027cb8b0d60248d
p' =  0x67c38bc71da218c073ba1015f99bb24fac971ea49cd676d39ee707c023e7ab09
q' =  0xabc98d98e6f34b1f70b335e9851681071782c833780644acf671220ec5e69179
```

Разом з консольним виводом ключі записано у файл для можливості використати у подальших операціях генерації ключів та шифрування.

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, d_1).

Реалізовано функцією *gen_key(p, q)*. Сгенеровані ключі для А та відкритий ключ В показано на рисунку

```
Ключ A
p =  10151155452580470316663456337267153951534622651119168881463627826241829982103457745290022086369161320574745948818817629491703881780968907516
q =  55904162975388984499907687419732774357744724867052568337377553530218841330861
d =  384178052696581116925600572119853335183886762258375134545081129619248224759l7
e=  65537
n =  21477152469517248293619994123691808772008153560382133619075470585240534979892288462873055373931182343591304437268677752657552005811441561078
Ключ B
e' =  65537
n' =  364682589299009936381615596862387983792540426843233310061067697646233358539893672148511823574453011307464720069739814990303518626794385594
```

В якості e використовується рекомендоване $2^{16} + 1$

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 и B, перевірити правильність розшифрування. Скласти для A і B повідомлення з цифровим підписом і перевірити його.*

Згідно вказівок до завдання шифрування та дешифрування реалізовано функціями *encrypt(M, key)* та *decrypt(M, key)*. За генерування цифрового підпису відповідає функція *sign()*. Перевірка правильності підпису здійснюється методом *verify()*. У

якості параметрів методу передаються повідомлення, згенерований підпис та відкриті ключі абонента А.

В якості повідомлення генерується випадкове число M з діапазону (0, 255). Такий діапазон дозволяє зашифрувати усі символи базового алфавіту.

```
Initial message  74
Зашифроване повідомлення 0x127061e5bc6e08fea65e281722f2e8973741f408cf802893a14af8c0c672792d206da594524c2d73fe55985223179ae826eba3888c02d5a35bc7160
Підпис  0x1626249c2d8463ebc0e7ce2e9dfecaf5ff81716531943b2b02e63eb22bcd6800a30ec1a05a33f2bd1b026b6dcf7549ae195d009a6e7ccce0b9d7dceec3e85d67
Підпис перевірено  True
Розшифроване повідомлення  74
```

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* $0 < k < n$.

sign(M, key) – створення підпису;

verify(M, S, key) – верифікація підпису;

sendKey(k, A_keys, B_key) – відправка ключа: k – сгенероване повідомлення, A_keys – ключі абонента A, B_key – відкритий ключ абонента B;

receiveKey(k, A_key, B_keys) –приймання ключа: k – сгенероване повідомлення, A_keys – ключі абонента A, B_key – відкритий ключ абонента B.

Знимок екрану виконання програми для відправлення та отримання повідомлення

```
Initial message  172

Відправка повідомлення
Перевірка n < n1:  True
k =  0xab93d8379e37ecfda5e9d49bbbf549094bb3486ee0ef63ee8c863eca462b1a4cbc7aa6420fd7d4e39e136df7070ef858f3add60cbdd9a0de17efd10e6e637ad
S =  0xc6a44f086d869005dac39d21be7dafdf591da80c5c1e06099b04bd146997b4f9dde505ddb9d9ad890dae9568dce18a73548bacac7c099979ce331ec0f4d8573
k1 =  0x546d485a0d6fbe2dd13257b0ec91956d90418d4b6b84e02bd2eeef6b3208e8343b8130bdbda78689b3f05ebba85d30a7b019daf6f1486e3cd34eae029ad834a
S1 =  0x32c4957fd10f41eb2ef1b27170ff11ca4dd5e4f95e566f53a3d5a052bab1cfc91cfc67a6108b6b1c63fadab368049b9376ed01238388cb7d89db3605eb98847c

Отримання повідомлення
k =  0xab93d8379e37ecfda5e9d49bbbf549094bb3486ee0ef63ee8c863eca462b1a4cbc7aa6420fd7d4e39e136df7070ef858f3add60cbdd9a0de17efd10e6e637ad
S =  0xc6a44f086d869005dac39d21be7dafdf591da80c5c1e06099b04bd146997b4f9dde505ddb9d9ad890dae9568dce18a73548bacac7c099979ce331ec0f4d8573
Авторизацію пройдено  True
```

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().