Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

КРИПТОГРАФІЯ

Лабораторна робота №4

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

Варіант 19

Виконали:
Студенти 3 курсу
Снігур А.Ю. та Носова Є. О.

Київ – 2021

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.

Постановка задачі:

Реалізувати програму для методами генерації ключів для асиметричної криптосистеми типу RSA, організації методами генерації ключів для асиметричної криптосистеми типу RSA

Порядок виконання:

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 і довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб ; *p* і *q* – прості числа для побудови ключів абонента *A*, і – абонента *B*. q p, 1 1 , q p 1 1 q p pq 1 p 1 q

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ та відкритий ключ . За допомогою цієї функції побудувати схеми RSA для абонентів *A* і *B* – тобто, створити та зберегти для подальшого використання відкриті ключі , та секретні і .) „ (qpd) ,(en) ,(ne) , (1 1 n e d 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 . nk   0

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:

Було використано функцію inverse_of_el для знаходження оберненого та нсд

gcd = el * v + u * mod. Також написано Імовірнісний тест Міллера-Рабіна (miller_rabin). el_to_power - Схема Горнера швидкого піднесення до степеня. prime_check - Імовірністний тест Ферма. generate_prime - генерація простих чисел вказаної довжини в бітах. generate_keys - генерація ключів вказаної довжини в бітах. to_decrypt - Функція розшифрування. to_encrypt - Функція шифрування. to_sign - Функція цифрового підпису. to_verify - Функція перевірки цифрового підпису. to_send - Функція відправлення (формує повідомлення що буде відправлене). to_receive - Функція отримання (оброблює отримане повіжомлення, розшифровуючи його та перевіряючи його піддліність)

```
Keys of Alice:
n is: 1bb3de7735c8ddcf50d1d5a6931ca08b9ff443f005bcdb03c348f6481f7b4010fc8dee63a936dde72931c6c34dc48fe0069e30fc811e3dfb1a9b41da2ca123a620d741254c3723a7bd2a98d51
e is: 10001
fi(n) is: 1bb3de7735c8ddcf50d1d5a6931ca08b9ff443f005bcdb03c348f6481f7b4010fc8dee63a936dde72931c6c34dc48fe0069e30fc811e3dfb1a9b41da2ca123a5f62553c7f82b463172d74
d is: 10e21531e7e88c2e1e1e542c59b9f7aa4088b8a2de8c539a3e790807ba8ffde29c58cfb1935e29ed08b11a66cacd81b5d1b2b53e1e5ed4fd6520bc08806c907243ca49fe3424037a7d77f1ce2
p is: 18e12361fe51141d6250d8482b8c4d9333173d6b2d1462822dfe26d18f5dc17de52ea57fde551764c681710033b1efecc0e6ec6ff577620dae3e6e86ed71bff91
q is: 11d0c9fb55bac958e8027697584a3307ca419b365aec08f3f73d2bb25d1f2d878e6edac152bf1541f8a2ee5a74f6214958496 3e86fb9f22af6b7df1a171f216d7
Keys of Bob:
n is: 2f398fea758abd6c102a49ab369cd91b51a5496cd9915f2e203bb41b62532cc564b770038a277157c8a949494a2cb59097d0489a568bd3da6747295acd9473703a2adc05580dd5f3ecbe03cee
e is: 10001
fi(n) is: 2f398fea758abd6c102a49ab369cd91b51a5496cd9915f2e203bb41b62532cc564b770038a277157c8a949494a2cb59097d0489a568bd3da6747295acd94737002f77de50e136881bbd23
d is: 25492d77d0965a38980acb5f831f13cb9e0b028e0eb9d88c72a6b0ed6e147da4dc8078d052a411b95d954b67b248da91ef4b6407778daf1721b3edd2e3bdf1e6de70c9d5afe762fc0b6f51f32
p is: 191d4151ab5fb18f9799f9cab84c4e78de4f402954c0a495e1a399cad9a69c949e84e97602690dea0a19ff22a47227f11da8d007d99cf466f02b14e54838b33d9
q is: 1e161cce9e9abbe29951ce51841f78f9b817fd331e1133f3200b0186eb63de23c45a37c48cdaeaee9cbf5e7cf64f8a11677923d5130744f8ad0461b5f789051d5
open text Alice wants to send: 59088860
Encrypted text with Bob's public key: 4443454410053509802024673920538270036791235539579607485081459419072441439501522364600544397651022551703152863076929966338
The text Bob received after decryption with his private key: 59088860
```

```
open text Alice wants to send: 16214248
encrypted msg is:2860d88f80b8c256db030701417124a63b4edf875bf43cbe74afe6af5e2a253d8af1b9df941e4724dcbedb4cf8b299370e5f35ea356a497bea9ae7c6f906c475e1d1efb789887
signature is: 213874290458f198b6be2e643181ace2e7922d0a56f224a5b71bc182f7a8d6fb2fb15f8b35b472657190f9a4776bdee89ea291af589592778193d1a4bf68d870a13b0e7b9f2416fd
Signature is valid
encrypted msg is: 16214248



Server public key:{'modulus': '14E7394F4086BEE0BD993E70C27CC91314274859C0FABB5A6C78F2E7E35A76BF6F9AB6E99C6C424449071C3E22C7AFA9F24B1719DFBD68195D418E815BCD16E
open text Alice wants to send: 40823062
msg server got after verification: 40823062
verification result: False

Process finished with exit code 0
```

Висновок:

В ході лабораторної роботи була створена тестова модель криптосистеми RSA, тести на псевдопрості 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.