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

КРИПТОГРАФІЯ КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

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

Виконав: Андреєв Д.Ю. Група: ФБ-06

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.

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 i p1, q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq < p1 q1; p i q прості числа для побудови ключів абонента A, p1 i 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 i 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.
 - 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().

Хід 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.

Для виконання цього завдання було реалізовано функцію miller_rabin_test, яка приймає ціле число та перевіряє його на простоту. Для реалізації було використано методичні вказівки щодо цього алгоритму. Далі на основі цієї функції було реалізовано пошук випадкового простого числа get_rand_prime. Для генерації псевдовипадкових чисел було використано модуль random мови програмування Python.

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q i p1, q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq < p1q1; p i q – прості числа для побудови ключів абонента A, p1 i q1 – абонента B.

Для цього було реалізовано функцію get_keys, яка приймає на вхід інтервал, а на вихід повертає 4 простих числа, перевірених за допомогою теста Міллера-Рабіна. Також перевірено, що pq < p1q1.

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.

Для досягнення такого результату було розроблено функцію generate_keys, яка приймає на вхід р та q а повертає секретний ключ (d, p, q) та відкритий ключ (n, e). Алгоритм роботи функції було взято з методичних вказівок до лабораторної роботи.

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.

Було реалізовано функції encrypt_rsa, decrypt_rsa, create_sign_rsa, verify_sign_rsa. Всі формули для роботи функцій було взято з методичних вказівок до лабораторної роботи.

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.

Для реалізації завдання було написано функції send_message та receive_message. Алгоритми роботи знову ж таки було взято з методичних вказівок.

Тепер можемо перевірити роботу функцій. А також виконати перевірку за допомогою запропонованого у методичці онлайн інструмента.

Для початку просто виконаємо наші операції та переконаємося, що все працює правильно.

```
For A:

p = 59006266589550610188633398537400837066651515902412333339026797195552851837421

q = 71226222163457968750014058549345924140810658457433425846309581115813744970583

n = 4202793453143559121780826580107024940823451860482144595124114301651566733008753226926078320393509250127165136984292720642771164771850978223712255464686443

e = 33427414346347536717924080112764070241982179143313334847762222185845465628933550140800218559610789972393960273572776514970178026375128269194059175415501

d = 31359653105807951278220018931814706607598796194829212153813125186780232814598270973820618423552252737367489914057976898844955132635625967857443425923998541

For B:

p = 590062665895506101886333398537400837080051515902412333939026707195552851837421

q1 = 7122622216345796875001446585493459241400180554575743425846508958111581137444070833

n = 704090133923710222404770469790085242109213053605590950165412642412925559027104333279060085721144088884745325182205660124380032724574044395337562215373613

n = 2643049675849061091087143065590838010262458591119161376681841804607464822555062710443327906005721144088884265519496091201390234162405949239224615666462977503

n = 777291736171464606928673131037370947100805107377073371097313107073312906511389117504725220953855176416665778572770999000912013902341624051089547275795

Msg: 133558221924020863394316097556505240267054577144318751232327644319117090414584155065071857961306733254732256367422972080237955347227719

Encrypted msg: 2448545382455566843827670458771443187512323276443191170904145841550657808189365529962581871588613062325473226367422972080237955347227719

Encrypted msg: 24485453824555668450870914356451474738638776125641018365457228

Encrypted msg: 24485459070913653657796367714431875123232764431911709041458415576657808701904895983749080237505547227719

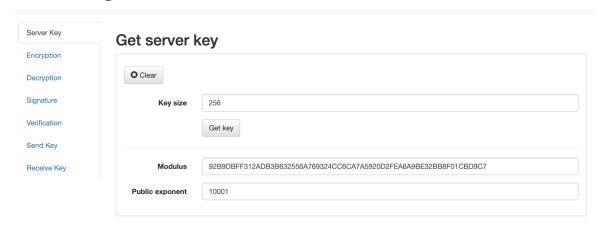
Encrypted msg: 13555821924020863394316697566364082709540971365590142313087366085599625818715886130623253473226367422972080237955347227719

Encrypted msg: 13555822192402086339431669756636482679608697136549147386388776125641018367476228

Enc
```

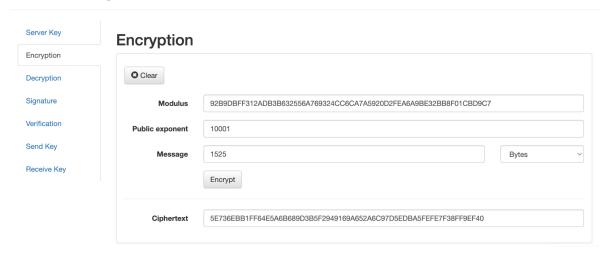
Тепер перевіримо роботу функцій за допомогою сайту. Спочатку отримуємо ключ.

RSA Testing Environment



Тепер перевіримо шифрування. В якості повідомлення виберемо "1525".

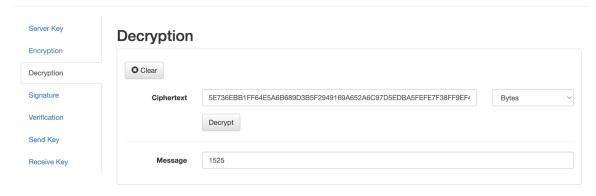
RSA Testing Environment



Ciphertext: 0x5e736ebb1ff64e5a6b689d3b5f2949169a652a6c97d5edba5fefe7f38ff9ef40

Далі перевіряємо розшифрування.

RSA Testing Environment



Бачимо, що ми отримали початкове повідомлення.

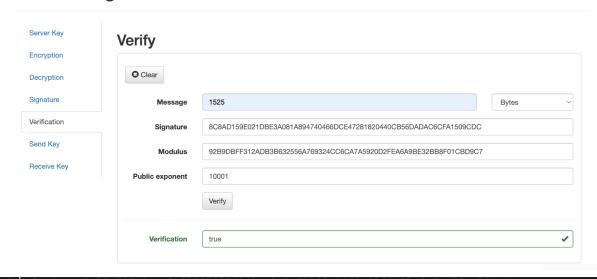
Тепер перевіряємо цифровий підпис. Спочатку створення.

RSA Testing Environment



А тепер перевіряємо його.

RSA Testing Environment



Sign is: 63569068584622326108610496896038483234608129002822606241375539641871542295772 Verification success

Код, який використовувався для тестування на сайті наведено в файлі таіп.ру.

Висновки

В ході виконання комп'ютерного практикуму відбулося ознаймолення з криптосхемою RSA, використано її для створення секертного зв'язку. Також було реалізовано тест Міллера-Рабіна, за допомогою якого було згенеровано ключі ключі для криптосистеми RSA. Було також перевірено коректність реалізації функцій за допомогою тестового онлайн середовища.