

КРИПТОГРАФІЯ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №5

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

Мета

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 $1 < p, q$ довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p_1q_1$; p і q – прості числа для побудови ключів абонента А, $1 < p < q_1$ – абонента В.
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 А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , $(,)$ і n_1 e та секретні d і d_1 .
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. 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 А і В, перевірити правильність розшифрування. Скласти для А і В 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. Написали алгоритм генерації простих чисел на основі тесту Міллера-Рабіна
2. Написали функції генерації пари ключів, шифрування розшифрування, верифікації, відправки та отримання ключів

Результати роботи програми

```
Згенеровано повідомлення M=2334137417967725729855083369190414077067757951271287258509066539971

A зашифрував повідомлення відкритим ключем B, і отримав C=798644666486311869067162675704970541
B розшифрував повідомлення своїм закритим ключем і отримав M=233413741796772572985508336919041
B зашифрував повідомлення відкритим ключем A, і отримав C=636946658832122551241418209061372534
A розшифрував повідомлення своїм закритим ключем і отримав M=233413741796772572985508336919041

A підписав повідомлення, і отримав S=289255088984449910430268380685547853766152187498324889889
B перевіряв підпис з результатом True
B підписав повідомлення, і отримав S=194150930795040708781680353397114478375648796775489798386
A перевіряв підпис з результатом True

A згенерував певний ключ k=2032452553517134347289522865208033821419260088857646748276402143497
A сформував повідомлення (k1, S1)=(32048323094433966634698376948027583969528795776151516736997
B отримав повідомлення, після чого знайшов і перевіряв ключ k=20324525535171343472895228652080
```

Перевірка коректності операцій шифрування з сервером:

```
pr, pub = generate_key_pair(p, q, 13)
pr, pub
(d=97, p=19, q=11, n=209, e=13)
```

Encryption

✖ Clear

Modulus

d1

Public exponent

d

Message

15


Encrypt

Ciphertext

62

```
hex(decrypt(98, pr))
'0x15'
```

Get server key



Key size

128

Get key

Modulus


9D5ACD006BE870182AAEF3515F8F5D5B

Public exponent

10001

```
int("9D5ACD006BE870182AAEF3515F8F5D5B", 16)
209160259982864571316780480937895877979
int("10001", 16)
65537
pub = PublicKey(209160259982864571316780480937895877979, 65537)
hex(encrypt(55, pub))
'0x513156e1cc6269548de078674d2a6d7f'
```

Decryption



Ciphertext

513156e1cc6269548de078674d2a6d7f

Decrypt

Message

37

```
int("37", 16)
55
```

Sign

✖ Clear

Message

123

Sign

Signature

81A6572D0B619F0E86C2A6AEBBD7904F

```
verify((int("123", 16), int("81A6572D0B619F0E86C2A6AEBBD7904F", 16)), pub)  
True
```

```
sign(150, pr)  
(150, 138)  
hex(150), hex(138)  
(0x96, 0x8a)
```

Verify

✖ Clear

Message

96

Signature

8a

Modulus

d1

Public exponent

d

Verify

Verification

true

```

send_key(5050, pr, pub)
(64175798577819004000730523428347984471, 70606774045179535361525975323676258772)
hex(64175798577819004000730523428347984471)
'0x3047cf28f3380bc84c6753d1fc483a57'
hex(70606774045179535361525975323676258772)
'0x351e5ebb021cd2e3f0d296c64fbc31d4'

```

Receive key

Key	3047cf28f3380bc84c6753d1fc483a57
Signature	351e5ebb021cd2e3f0d296c64fbc31d4
Modulus	d1
Public exponent	d
<input type="button" value="Receive"/>	
Key	13BA
Verification	false

```

int("13BA", 16)
5050

```

Send key

Modulus	36340751c299bbd134d3091dd335fed408f51631b1e61ea0dc007c750d8e664eed62834cf3925a633a6fd8a3ebb4
Public exponent	10001
<input type="button" value="Send"/>	
Key	10221F8AD29E0CBF619919267B7D710CF00D33352183485CF8316E7705E07A0648E8683EF51118F5B2D7F3F
Signature	2D59B54DA8C95CEE0EC31F04039C1F83CA8B9C12365CB385E357C5D5AD15BEDB91636796B4000CB0E0

```

send_key((int("10221F8AD29E0CBF619919267B7D710CF00D33352183485CF8316E7705E07A0648E8683EF51118F5B2D7F3F", 16), int("2D59B54DA8C95CEE0EC31F04039C1F83CA8B9C12365CB385E357C5D5AD15BEDB91636796B4000CB0E0", 16)), pr, th_pub)
6166671626116485822

```

Висновок: ми ознайомилися з тестами перевірки чисел на простоту та використали тест Міллера-Рабіна. Навчилися генерувати ключі для асиметричної криптосистеми RSA.