

Криптографія

Комп'ютерний практикум №4

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

Виконав студент групи ФБ-11

Пташник Юрій

Мета роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричних криптосистем 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 і p_1, q_1 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 A , p_1 і q_1 – 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) , (e_1, n_1) та секретні d і d_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 і 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 $0 < k < n$.

Виконання роботи

Приклад кандидатів для p , q , p_1 і q_1 , які не пройшли перевірку простоти:

```
Candidate: 90755952787390724102675841415993045185842834802427045321124216714493946753500 failed the test
Candidate: 4640484772979195750537163728542775584682634097665376070757850199110639077795 failed the test
Candidate: 74070615132357501532789100514641525921220984028784221993187667955648637129819 failed the test
Candidate: 88227323690551391934620872882370114857771611500194480054455116800566749411853 failed the test
Candidate: 41401440607408665765236924458678815236964590753643614925145809038583054821090 failed the test
Candidate: 84058303443859936227617608182348718772835204299903782191478305507969551976088 failed the test
Candidate: 111914432211790887984464881631726458117101107854563432984466381478431335338738 failed the test
Candidate: 67399163600111645889690694552972012011248942196110151604023959629268885788701 failed the test
Candidate: 65155291526053726230107398609903716535594736721413723624096736402359549639975 failed the test
Candidate: 71555561565776859064988268942553922878860181281453359631763734536879798198921 failed the test
Candidate: 100621821206282637894512164642100124258007523512535661966908340522616506386485 failed the test
Candidate: 95431349616437251196080857371620608041071850027072021606877890285473559199346 failed the test
Candidate: 22522188341200483770613196880697283370211045526303153155323830553407884011176 failed the test
Candidate: 1072821301217382671697416420726384023449231555333080080067648337334865533733834 failed the test
Candidate: 27885100962340778481076685758493827199059582634187274079322195820501722840294 failed the test
Candidate: 36430111957881966149819192344875104163363892884408272449192258453553498876492 failed the test
Candidate: 54044230054121948430229889025519943525599912187490147572161196813212392059372 failed the test
Candidate: 24754632116893838594848853834458889560253425038838209734049118116914645250776 failed the test
Candidate: 84517065126115083088698800482725267762150245297487534606489351506366880561031 failed the test
Candidate: 41217221642318117411965457423692460354877074421224812372286533683371316895078 failed the test
Candidate: 15417810022004821820694027105515443103186830630549848261750481930310311182932 failed the test
Candidate: 85466874444144929695359308812908730649116383872138032186939803063844839094282 failed the test
```

Значення обраних p , q , p_1 і q_1 та параметрів криптосистеми RSA для абонентів A і B :

```
Public key for user A: ( n - 232574503979214641053574988721500177310032451398290560868768094658127107483184016825200
6807506127261913107282698680676449196243823055130488120090639093601 , e - 65537 )
Private key for user A: ( d - 12921002929464585014206731066954272328392025200133908053819745069964154574457653930123
61988290080676648718309084112075882606283248362908091377351740618633 , p - 88298181473983089061615111387645412080050
794429980294042048682722398363781963 , q - 2633967088526532431592679273716482174335795033429469495760507964659364787
6227 )
Public key for user B: ( n1 - 29579059557653289917443222210013968900758892847623419235716957223284779174793653854192
22917303363589861150838534262710326113215828107499991088182722831649 , e1 - 65537 )
Private key for user B: ( d1 - 1511697240984542002982808619104029592996350698137386215238878443072921240948627139880
035159915866088428883392087887479728176381428176514096280026932225753 , p1 - 635328228030400568673182405197360149330
75479837880293295164741194257080658479 , q1 - 4655713102714196835344492439503434998940200022841639531259785430354751
3037231 )
```

Значення прикладів ВТ, ШТ, цифрового підпису для A і B :

```
message M: 10
encrypted message C by A: 147964497484831108305949347432254494053654266474270201181424298617126048037661947926951399
2847436777993023417524798315533245433435941336401331487252247462
decrypted message M by A: 10
encrypted message C by B: 57355504482538362417177249234483429185186881082726751996968618890005438614778516456918424
505794655238218081689206127623220765025426148071287581820661444
decrypted message M by B: 10
Signature for user A: (10, 1501637969993694738338452714138019140655838632355324986498863876046179053516352377616766948508469
362086111270701005333854166309577540133325391754551676096)
result of verifying signature A: True
Signature for user B: (10, 2531819461314781503840203270958071152811537007769835574879489149207091055100452560369042140811952
73681272438566841056695532816587290072054830486938132803)
result of verifying signature B: True
```

Опис кроків протоколу конфіденційного розсилання ключів з підтвердженням справжності:

1. Абонент A перевіряє справедливість нерівності $n_1 \geq n$, де n_1 – відкритий ключ абонента B . Якщо нерівність не справджується, абоненту A необхідно згенерувати іншу пару ключів.

2. Абонент A формує зашифроване повідомлення k_1 шляхом шифрування повідомлення k за допомогою відкритого ключа абонента B , формує цифровий підпис S за допомогою власного секретного ключа та формує зашифрований підпис S_1 шляхом шифрування отриманого підпису S за допомогою відкритого ключа абонента B .
3. Абонент A надсилає повідомлення (k_1, S_1) абоненту B .
4. Абонент B , отримавши повідомлення від абонента A , дешифрує повідомлення k_1 за допомогою власного секретного ключа, отримуючи повідомлення k . Дешифрує зашифрований підпис S_1 за допомогою власного секретного ключа, отримуючи підпис S .
5. Абонент B автентифікує абонента A шляхом порівняння дешифрованого повідомлення k та результату перевірки підпису. У разі співпадіння значень k та результату перевірки підпису, повідомлення k не є спотвореним, а підпис S є вірним.

Чисельні значення характеристик протоколу конфіденційного розсилання ключів:

```
Sended values by user A to user B: ( k1 - 9327169108039559087347656536644047884432275327070449207966240514272758288779823026
85051164654719580099992064164876335390197017110267105353865388113988456 , S1 - 225166177036281114240029438758949938597929657
5098008924080490518304346496051386825617134579413391506982082658746567695590032429617132698548378791295663398 )
Authentication of user A: True
```

Перевірка коректності операцій за допомогою сервера

Для перевірки буде використовуватися наступний ресурс:

<https://asymcryptwebservice.appspot.com/?section=rsa>

Згенеруємо відкритий ключ сервера:

RSA Testing Environment

Server
Key

Encryption

Decryption

Signature

Verification

Send
Key

Receive
Key

Get server key

✖ Clear

Key size

256

Get key

Modulus

ABB56B149D2B8AC5BA1C9764C4431C7A6CAFC0D6FFE1145B4

Public exponent

10001

За допомогою написаних функцій згенеруємо відкритий та секретний ключ для програми:

```
pProgram = GenerateRandomPrime(length = 128)
qProgram = GenerateRandomPrime(length = 128)

publicKeyProgram, privateKeyProgram = GenerateKeyPair(pProgram, qProgram)

print(publicKeyProgram)
print(privateKeyProgram)

publicKeyProgram = (41887016183231955725709937040856737739607424282165092723873588768948795089469,
65537)

privateKeyProgram = (25178738766474249596630040583556634927380026189639794459545454434999512306773,
209439695060123648592801721058537048411,
199995593820968326575863902654491115079)
```

Зашифруємо повідомлення на сервері за допомогою відкритого ключа програми:

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Encryption

Clear

Modulus

5C9B36049B9DB6D579AF093810DCF5B93F2C5B44C5CB760232

Public exponent

10001

Message

76ADF1

Bytes ▼

Encrypt

Ciphertext

345309F52AD379F345916874A9071CB08EA2480421BC2E61E44

Розшифруємо у програмі за допомогою секретного ключа програми:

```
Decrypted by program: 76ADF1
```

Зашифруємо повідомлення у програмі за допомогою відкритого ключа сервера:

```
hex of 999999999 : 3B9AC9FF
Encrypted by program: 103ECFB0256ED672847CFEE1D5E96B70095F767045540ABBBEC024FE73BD3BD5
```

Розшифруємо на сервері:

RSA Testing Environment

[Server Key](#)
[Encryption](#)
[Decryption](#)
[Signature](#)
[Verification](#)
[Send Key](#)
[Receive Key](#)

Decryption

✖ Clear

Ciphertext

103ECFB0256ED672847CFEE1D5E96B70095F

Bytes ▼

Decrypt

Message

3B9AC9FF

Створимо цифровий підпис на сервері:

RSA Testing Environment

[Server Key](#)
[Encryption](#)
[Decryption](#)
[Signature](#)
[Verification](#)
[Send Key](#)
[Receive Key](#)

Sign

✖ Clear

Message

D903

Bytes ▼

Sign

Signature

6BFD032DDE2250D28AAC82B841FBE0474A75DCA42A057FBB0

Перевіримо цифровий підпис у програмі:

```
Result of verifying by program: True
```

Створимо цифровий підпис у програмі:

```
hex of 22 : 16  
Sign by program: 32FC9110FC8772F5DD9152F8A2836D04C114ACF68A800D4D8717673D375F3C66
```

Перевіримо цифровий підпис на сервері:

RSA Testing Environment

Server
Key

Encryption

Decryption

Signature

Verification

Send
Key

Receive
Key

Verify

✖ Clear

Message

16

Bytes



Signature

32FC9110FC8772F5DD9152F8A2836D04C114ACF68A800D4D87

Modulus

5C9B36049B9DB6D579AF093810DCF5B93F2C5B44C5CB760232

Public
exponent

10001

Verify

Verification

true



Надішлемо ключ з сервера:

RSA Testing Environment

Server
Key

Encryption

Decryption

Signature

Verification

Send
Key

Receive
Key

Send key

✖ Clear

Modulus

5C9B36049B9DB6D579AF093810DCF5B93F2C5B44C5CB760232

Public
exponent

10001

Send

Key

2FC51C5D61B20DAF21D639DDDF0F9381295774B4DDF9ED5DC

Signature

37CEF8F0892B53D9EDB79C7B7810134B7415D8140EAA323722

Автентифікація сервера у програмі:

```
Result of receiving key: True
```

Відправимо ключ з програми:

```
hex of 565655 : 8A197  
kProgram: 86406FF7F77691F980F983A58FD979996AEC117CF4D0AEE5ABC0F7BEB20AB637  
sProgram: 62E33115BDD18E7916A695DFC6701C2C341FB20CE4030997403A47A79E82796B
```

Отримаємо ключ на сервері:

RSA Testing Environment

Server
Key

Encryption

Decryption

Signature

Verification

Send
Key

Receive
Key

Receive key

✖ Clear

Key

86406FF7F77691F980F983A58FD979996AEC117CF4D0AEE5AB

Signature

62E33115BDD18E7916A695DFC6701C2C341FB20CE403099740

Modulus

5C9B36049B9DB6D579AF093810DCF5B93F2C5B44C5CB76023

Public
exponent

10001

Receive

Key

08A197

Verification

true

✓

Висновки

У ході виконання даного комп'ютерного практикуму було отримано навички використання асиметричної криптосистеми RSA. Були опановані методи генерації ключів, перевірки простоти чисел. Також були засвоєні необхідні кроки для організації протоколу конфіденційного розсилання ключів з підтвердженням справжності.