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 $pq \leq p_1 q_1$; 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 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 $0 < k < n$.

Виконання роботи

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

```
Candidate: 907559527873907241026758414159930451858428348024270453211242167144939467535 00  failed the test
Candidate: 464048477297971957505371637285427755846826340976653760707578501991106390779 95  failed the test
Candidate: 740706151323575015327891005146415259212209840287842219931876679556486371298 19  failed the test
Candidate: 882273236905513919346208728823701148577716115001944800544551168005667494118 53  failed the test
Candidate: 414014406074086657652369244586788152369645907536436149251458090385830548210 90  failed the test
Candidate: 840583034438599362276176081823487187728352042999037821914783055079695519760 88  failed the test
Candidate: 111914432211790887984464881631726458117101107854563432984466381478431335338738  failed the test
Candidate: 673991636001116458896906945529720120112489421961101516040239596292688857887 01  failed the test
Candidate: 651552915260537262301073986099037165355947367214137236240967364023595496399 75  failed the test
Candidate: 715555615657768590649882689425539228788601812814533596317637345368797981989 21  failed the test
Candidate: 100621821206282637894512164642100124258007523512535661966908340522616506386485  failed the test
Candidate: 954313496164372511960808573716206080410718500270720216068778902854735591993 46  failed the test
Candidate: 225221883412004837706131968806972833702110455263031531553238305534078840111 76  failed the test
Candidate: 107282130121738267169741642072638402344923155533380080067648337334865533733834  failed the test
Candidate: 278851009623407784810766857584938271990595826341872740793221958205017228402 94  failed the test
Candidate: 364301119578819661498191923448751041633638928844082724491922584535534988764 92  failed the test
Candidate: 540442300541219484302298890255199435255999121874901475721611968132123920593 72  failed the test
Candidate: 247546321168938385948488538344588895602534250388382097340491181169146452507 76  failed the test
Candidate: 845170651261150830886988004827252677621502452974875346064893515063668805610 31  failed the test
Candidate: 412172216423181174119654574236924603548770744212248123722865336833713168950 78  failed the test
Candidate: 154178100220048218206940271055154431031868306305498482617504819303103111829 32  failed the test
Candidate: 854668744441449296953593088129087306491163838721380321869398030638448390942 82  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
68075061272619131072826986806764491962438235513048812009063909360 1 , e - 65537 )
Private key for user A: ( d - 129210029294645850142067310669542723283920252001339080538197450699641545744576539301236
61988290080676648718309084112075882606283248362908091377351740618633 , p - 882981814739830890616151113876454120800507
94429980294042048682722398363781963 , q - 263396708852653243159267927371648217433579503342946949576050796465936478 76227 )
Public key for user B: ( n1 - 295790595576532899174432222100139689007588928476234192357169572232847791747936538541922
29173033635898611508385342627103261132158281074999910881827228316 49 , e1 - 65537 )
Private key for user B: ( d1 - 151169724098454200298280861910402959299635069813738621523887844307292124094862713988003
03515991586608842888339208788747972817638142817651409628002693222575 3 , p1 - 6353282280304005686731824051973601493307
54798378802932951647411942570806584 79 , q1 - 4655713102714196835344492439503434998940200022841639531259785430354751 3037231 )
```

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

```
message M: 10
encrypted message C by A: 14796449748483110830594934743225449405365426647427020118142429861712604803766194792695139928474367779930234175247983155332454334359413364013314872522474 62
decrypted message M by A: 10
encrypted message C by B: 5735555044825383624171772492344834291851868810827267519969686188900054386147785164569184245057946552382180816892061276232207650254261480712875818206614 44
decrypted message M by B: 10
Signature for user A: (10, 150163796999369473833845271413801914065583863235532498649886387604617905351635237761676694850846936208611127070100533385416630957754013332539175455167609 6)
result of verifying signature A: True
Signature for user B: (10, 253181946131478150384020327095807115281153700776983557487948914920709105510045256036904214081195273681272438566841056695532816587290072054830486938132803)
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
850511646547195800999920641648763353901970171102671053538653881139884 56 , S1 - 225166177036281114240029438758949938597929657
509800892408049051830434649605138682561713457941339150698208265874656769559003242961713269854837879129566339 8 )
Authentication of user A: True
```

Перевірка коректності операцій за допомогою сервера

Для перевірки буде використовуватися наступний ресурс: https://www.tausquared.net/pages/ctf/rsa.html

Буде використано наступні ключі:

```
#Server check
publicKeyServer = (476004819700066579579161388571994358600875208523437619704550064693701581100162787778903905940885421698190957329556370087407091860508216684898552450071048 1,
                   65537)
privateKeyServer = (181767499547032458314358193240501256059104062872996168129851380426699051964727137710404999077411135326950013593341886871807172223449776218027431245225274 5,
                    571180808864600086126669370539553824949385676522879749032133054641440075667 99,
                    833369770679576131826609206156111733391850996629980847350216323634801022187 19)

publicKeyProgram = (868494438794429103553481684151180737116920389807385928573771305804120730604872485876557576919869054678259884797195046078700362225851657229403751876829889 7,
                    65537)
privateKeyProgram = (184984266462173060355265740407194896156599962178941669239685576967510280887333164255557772826368471709744103144902302666375128074473119178001561131375769 3,
                     102859931232048261086373034384133824342961802102105418558751187833277027117727,
                     844346703708305287564551140007390724142832486548167778152524632582485704857 11)
```

Зашифрування повідомлення на сервері, використовується відкритий ключ програми:

Calculate `n = p * q`.

`n`:
```
868494438794429103553481684151180737116920389807385928573771305804120730604872485876557
576919869054678259884797195046078700362225851657229403751876829889 7
```

Calculate n | Calculate p = n / q | Calculate q = n / p

Compute the Carmichael's totient function `tot(n) = λ(n) = lcm(p - 1, q - 1)`. (Note that Euler's totient function `tot(n) = ϕ(n) = (p - 1) * (q - 1)` could be used instead. See StackExchange.)

`tot(n)`:

Calculate λ(n) | Calculate φ(n)

Choose any number `e` where `1 < e < tot(n)` and `e` is coprime to `tot(n)`. Common choices are 3, 17, and 65537 (these are Fermat primes).

`e`:
```
65537
```

Check if coprime

Compute `d`, the modular multiplicative inverse of `e` (mod `tot(n)`).

`d`:

Calculate d

That's it for key generation! The public key is `(n, e)` and the private key is `(n, d)`

Encryption and decryption

Encryption is done with `c(m) = m^e mod n` where `c` is the ciphertext and `m` is the message. Note that both of these values must be integers `1 < m < n` and `1 < c < n`.

Decryption is done with `m(c) = c^d mod n`.

`m`:
```
7777777
```

`c`:
```
103137275358457462002573540102651873438461306662064137601872086055939053220835633 5
779579099834102000918344081192243693730426379155522514228428908608816857
```

Encrypt | Decrypt

Результат розшифрування у програмі, використовуючи секретний ключ програми:

```
Decryption by program: 7777777
```

Зашифруємо повідомлення у програмі, використаємо відкритий ключ сервера:

```
message = 999999999
```

```
Encryption by program: 284012827045579962638887744836026318303861516606796585170247392743536075270068781923190957194708749655420804327340300362214650678466486676656022306254863 4
```

Результат розшифрування на сервері, використовуючи секретний ключ сервера:

Calculate `n = p * q`.

`n`:
```
476004819700066579579161388571994358600875208523437619704550064693701581100162787778903
905940885421698190957329556370087407091860508216684898552450071048 1
```

Calculate n | Calculate p = n / q | Calculate q = n / p

Compute the Carmichael's totient function `tot(n) = λ(n) = lcm(p - 1, q - 1)`. (Note that Euler's totient function `tot(n) = ϕ(n) = (p - 1) * (q - 1)` could be used instead. See StackExchange.)

`tot(n)`:

Calculate λ(n) | Calculate φ(n)

Choose any number `e` where `1 < e < tot(n)` and `e` is coprime to `tot(n)`. Common choices are 3, 17, and 65537 (these are Fermat primes).

`e`:
```
65537
```

Check if coprime

Compute `d`, the modular multiplicative inverse of `e` (mod `tot(n)`).

`d`:
```
807172223449776218027
4312452252745
```

Calculate d

That's it for key generation! The public key is `(n, e)` and the private key is `(n, d)`

Encryption and decryption

Encryption is done with `c(m) = m^e mod n` where `c` is the ciphertext and `m` is the message. Note that both of these values must be integers `1 < m < n` and `1 < c < n`.

Decryption is done with `m(c) = c^d mod n`.

`m`:
```
999999999
```

`c`:
```
284012827045579962638887744836026318303861516606796585170247392743536075270068781 9
231909571947087496554208043273403003622146506784664866766560223062548634
```

Encrypt | Decrypt

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.