



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Криптографія
КОМП'ЮТЕРНИЙ ПРАКТИКУМ
Робота №4

Перевірив:
Чорний О.М.

Виконала:
Студентка групи ФБ-81
Ренькас І.О.

Київ – 2020

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

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 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 A, $1 < p < q_1$ – 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 $0 < k < n$.

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

Програма та всі її елементи створені так, аби працювати згідно RSA алгоритму, що описаний в методичних вказівках до Комп'ютерного практикуму 4

Так як, після написання алгоритму, його потрібно було перевіряти за допомогою спеціального сайту, при запуску програми, спершу потрібно вибрати буде це одинарна генерація ключів з цифровим підписом користувача чи програма створювати ключі та підписи для 2 локальних користувачів і надсилати між ними повідомлення.

Також я вирішила створити 2 різні структури: `PrivateKeyStruct` та `PublicKeyStruct` – які об'єднуються в одну структуру `KeyPairStruct`, оскільки є певне обмеження на те, що функції мають приймати тільки

ті дані, якими буде оперувати, і тому передавати повністю KeyPairStruct буде порушенням цього обмеження.

Для перевірки натуральних чисел отриманих у програмі випадковим чином я вирішила використовувати Імовірнісний тест Міллера-Рабіна

Значення ключів для Боба:

p: 86085501926049617575387245938523746390000378206105937095201241240199718066859

q: 93570027779562786518005959613709557147413306810372342437571890146236073416703

n:

805502280663806848014106802325792826012667558001550206822077314913167433506962601516676
4257104146855585047541390605825660593994834738327829554722321345877

e:

498054187540082244845380849928591439686530256937272914846604018700513657667946024525073
4402042274978402570887223978988443785564105195709297923461434846071

d:

639495177573841488344261890677691031091701647899807809164355044359968405060293172955171
6129317170884584939904324716688787794533853982800362867931390153067

Значення ключів для Аліси:

p: 74573951365814344726051911897242794614702467025477004418815656932773156367329

q: 78478519701951626169660907817769613639357812999296783047942627292167423171643

n:

585245331151444343418659914567302073727624030438555789504783898234676723405418467280524
5117698079525038455382534298247322676181497722375613538201524451547

e:

312132604582848566800513635032051965462870538250642378817021033801989682193907264318699
3807132282018788404893321012745365496928976864930530126375795406281

d:

473267574580364635260396454628914846606923020000748780365934594544190220307249955519085
2680611287968296902752585597337526445684000331902246276371820929401

Числа, що не пройшли перевірку на простоту, і причини, чому вони не підходять:

86721907984277411307925183766794190804519889094945378666775679949402187472613 – не було
вивлено чи є простим число чи ні, але ітерації закінчились

86721907984277411307925183766794190804519889094945378666775679949402187472615 – подільне
на 3

86721907984277411307925183766794190804519889094945378666775679949402187472617 – подільне
на 7

86721907984277411307925183766794190804519889094945378666775679949402187472619 – подільне
на 19

86721907984277411307925183766794190804519889094945378666775679949402187472621 - подільне
на 3

86721907984277411307925183766794190804519889094945378666775679949402187472623 – подільне
на 17

86721907984277411307925183766794190804519889094945378666775679949402187472625 – подільне
на 5

86721907984277411307925183766794190804519889094945378666775679949402187472627 – подільне на 3

86721907984277411307925183766794190804519889094945378666775679949402187472629 – подільне на 13

86721907984277411307925183766794190804519889094945378666775679949402187472631 – подільне на 7

86721907984277411307925183766794190804519889094945378666775679949402187472633 – подільне на 3

86721907984277411307925183766794190804519889094945378666775679949402187472635 – подільне на 5

86721907984277411307925183766794190804519889094945378666775679949402187472637 - не було вивлено чи є простим число чи ні, але ітерації закінчились

Після того як ключі згенеровані відбувається виконання кроків протоколу конфіденційного розсилання ключів по відкритих каналах зв'язку з підтвердженням справжності відправника:

1. Користувач, що хоче відправити повідомлення (у нашому випадку Аліса), хоче надіслати деяке повідомлення k

2. Аліса, використовуючи відкритий ключ Боба, зашифровує повідомлення k за формулою $k_1 = k^{e_1} \bmod n_1$

3. Аліса, використовуючи свій секретний ключ, створює свій цифровий підпис S за формулою $S = k^d \bmod n$ і, використовуючи відкритий ключ Боба, зашифровує свій цифровий підпис за формулою $S_1 = S^{e_1} \bmod n_1$

4. Аліса надсилає Бобу повідомлення (k_1, S_1)

5. Боб отримує повідомлення (k_1, S_1) від Аліси

6. Боб за допомогою свого секретного ключа розшифровує k і S за формулами $k = k_1^{d_1} \bmod n_1$
 $S = S_1^{d_1} \bmod n_1$

7. Боб, використовуючи відкритий ключ Аліси, проводить перевірку цифрового ключа Аліси $k = S^e \bmod n$

Результат проведення цього в коді програми:

Sended message from A

k1:
cb4ba68579f3051ee92778cdb07519a265f16f1a6aea8b7ea12bfc6c56b4456d2e346edcbcec5336c0fb7629eeda0eae02b37097c48aa2208445435c75ca4

S1:
35fc3bb130573efd1c68ac36aab789bb5991169cadd6eeb7875aab3a341a447f85e43fd766a8a5ca6b5caf158cf51e35754d3852e3be9eb23b673652853e428e

Recieved message to B

k:
3b98b951bec9dfd303c274a49d32718c7344f76658f9ad49732d41583623442714a7fad71e51783de7cf520923b02be4029ad8cf1f73622719dc790390a1b5c8

S:
2bf1463ff1035e09437c9fc4a03445b8ab54373380d8b62e47916c3d3c3eaf4e99423eddd04c63ae0b3f45664b
3345efbbae16a4f52c551a90d2d4a8c290fee8

Data is decrypted properly!

Sign VERIFIED!

Перевірка коректності роботи програми за допомогою сайту:

Key size	<input type="text" value="512"/>
	<input type="button" value="Get key"/>
Modulus	<input type="text" value="C81F36597541D2F167A0A8FFD35B05FA26CF078F4F0E120CC3"/>
Public exponent	<input type="text" value="10001"/>

```
RSA Class created
Choose testing type:
1. Local test
2. Test with site
2
Enter public exponent: 10001
Enter modulus: C81F36597541D2F167A0A8FFD35B05FA26CF078F4F0E120CC33A222ACC2C8D24942C409A437FB2B53D92B7C40B7CB4C461837A404
9EF7FFCC1BE9FC8D9C0FC8F
```

.....пропускаємо момент, коли програма генерує і виводить всі значення згенерованих чисел, що не є простими

RSA Data:

p: 103344331161703100868926609237382370066108685405314426604537690706055305991171

q: 70844033811680764555199826518091201564611425306041798169568233825691442193913

n:
732132929106522854393750056645636367685597249031033515527583889963830837924141195448922
8127733429071224670489713553936084091591253701024327524809247942123

e:
402962893167891891908378752934726195533120892856352155772301827503948744395865853902560
2446572127546705272857693353624633305498041617981680858777585442877

d:
106637272401359623118066133061479776222483174023594747801265674696611415198103954410551
6433667564613180939554383523308652154649939696074247514502973327093

Data for A

k:
4cf06a7a3c6b91a26fe47348f1d29ba38479edbb924f30ee1a9ca2040ddf5c79ad02a2af38f38327dd8d056fc73f
57cdb8daadb0223342c9cc0c7fa3c703e03c

S:
707d38e42e2d1627c8537495dc89e466aa16b02c64a578431d710facb203393bb84625ccd268e146472785284
5cc0577653eeb5769ec69686333ca1fac27849

Sended message from A

k1:
6dd6cad7e465d863172e1ab7b9b26dd9e6b49a95c20ccff63c06186bed107d1195bc6686c7a12ab758e75205ff
799d4bfc079e1561b087b115c102778d946ba8


S1:
bb1c6e1d6faecc71ba7e8990efb3cda3d3f87fa3edc1a6ee7ac0426670ef1d7261a3c7c83091191fa599613b83c26d147e344473957b1680d35a2a51c5c65a91

Public key A: n =
8bc9e981440458b8ebb0e94ab425660558c1196d9b3a781775934965753168f401f7ed9266d1a02ec2f4abd6d78cd90c631332f0e41ddb7d09dba7bee8d9e5eb

e =
4cf06a7a3c6b91a26fe47348f1d29ba38479edbb924f30ee1a9ca2040ddf5c79ad02a2af38f38327dd8d056fc73f57cdb8daadb0223342c9cc0c7fa3c703e03d

Вводимо дані на сайті, аби переконатись, що все правильно

Verify

 Clear

Message

4cf06a7a3c6b91a26fe47348f1d29ba38479edbb924f30ee1a9ca2040ddf5c79ad02a2af38f38327dd8d056fc73f57cdb8daadb0223342c9cc0c7fa3c703e03d

Bytes

Signature

707d38e42e2d1627c8537495dc89e466aa16b02c64a578431d710facb203393bb84

Modulus

8bc9e981440458b8ebb0e94ab425660558c1196d9b3a781775934965753168f401f7ed9266d1a02ec2f4abd6d78cd90c631332f0e41ddb7d09dba7bee8d9e5eb

Public exponent

4cf06a7a3c6b91a26fe47348f1d29ba38479edbb924f30ee1a9ca2040ddf5c79ad02a2af38f38327dd8d056fc73f57cdb8daadb0223342c9cc0c7fa3c703e03d

Verify

Verification

true

Receive key

Clear

Key

6dd6cad7e465d863172e1ab7b9b26dd9e6b49a95c20ccff63c06186bed107d1195bc6686c7a12ab758e75205ff799d

Signature

bb1c6e1d6faecc71ba7e8990efb3cda3d3f87fa3edc1a6ee7ac0426670ef1d7261a3c7c83091191fa599613b83c26d1

Modulus

8bc9e981440458b8ebb0e94ab425660558c1196d9b3a781775934965753168f401f7ed9266d1a02ec2f4abd6d78cd

Public exponent

4cf06a7a3c6b91a26fe47348f1d29ba38479edbb924f30ee1a9ca2040ddf5c79ad02a2af38f38327dd8d056fc73f57cd

Receive

Key

4CF06A7A3C6B91A26FE47348F1D29BA38479EDBB924F30EE1A9CA2040DDF5C79AD02A2AF38F38327DD8E

Verification

true

✓

Висновки: під час виконання даної роботи я дізналась різницю між симетричною і асиметричною криптографією, вивчила основні пункти алгоритму асиметричного шифрування RSA та . імовірнісний тест Міллера-Рабіна на перевірку натуральних чисел на простоту та створила прмітивний код програми, яка в певній мірі демонструє роботу алгоритму RSA.