

Лабораторна робота №4

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

Виконали:

Анучін Максим ФБ-11

Ступак Ярослав ФБ-11

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

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 $pq < p_1q_1$; p і q – прості числа для побудови ключів абонента А, p_1 і 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) , (e_1, n_1) 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 и 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 $0 < k < n$.

Хід роботи

Написав функцію яка приймає 2 аргументи: мінімальну і максимальну величину простого числа яке буде згенеровано. У випадку якщо аргумент наданий тільки один, він сприймається як мінімальна довжина в бітах

Далі генеруємо дві пари простих чисел для абонентів А В мінімальною довжиною в 256 біт:

ap = 139672102951664142186453343489092249579032609880927332775446586938425219739863

aq = 159691178035788586299454291475781380222731465646510389662936615379262121803767

bp = 129380935509241031518642830120924231012205656097411878296711164423826926040303

bq = 123462647380084836374109519510234993652684361136623104776725248243497930500851

Після чого генеруємо пари з відкритих і закритих ключів. В якості випадкового числа е використовуємо значення $2^{16}+1$

an =

2230440265908719103622612668789086389391442645373941704245491696831953178173716829777
9266383365877953479989676959689630439144309083379865913733224073463921

ad =

1997516372842523312768695627954252520752429058164590390882717307202920913171921278625
0629190776209642595511555027558608732162977108926842009568825204183861

bn =

1597371281848292241526116722068168984113301722053125819539813566313767352304196124973
6304393686351825014698943990712551959302200121588252059016807001797853

bd =

1730280411041247939727772496446576989825644891413268869938071090721490827167448738503
182836663918717012516840930377837742446891135058313262331735870564373

Далі шифруємо повідомлення А «885555555588» і В «606060606060»

Після шифрування отримуємо наступні числа

Для А:

790965332463256612762853559341702348090406712609313622778327184386
513707715847276645605588816605863843490985154954304825855970511447
3047560364334304050690

Для В:

111276624131531332163186105043439275856276787171942973165670886186
669504318959560751124624832404870137274884498373339357647476301998
67269606345144728422709

При дешифруванні отримуємо назад наші повідомлення

A's decrypted message: 88555555588

B's decrypted message: 606060606060

Далі перевіряємо функції підпису і верифікації, повідомлення залишаємо тими самими:

Підписане повідомлення А:

88555555588,
208760337503257297737550356992758887455890323679329602516863082847
772874855592446096371518020193991451739648492586004457957966064912
19897982193266316378346

Підписане повідомлення В:

606060606060,
773298523119152168382321486649078444503728033739637919260109742411
926466218398631983289091759824872489426417017038795062682434801739
9914389686631204952278

Далі для перевірки верифікуємо повідомлення А з правильним ключем, і повідомлення В з неправильним:

A's key verification: Message is verified

B's key verification: Fake sign

Для перевірки скористаємося сайтом <http://asymcryptwebservice.appspot.com/>

Переведемо значення отримане при шифруванні в hex:

9705948199d514c7723a1d9a8cff3f5d0c37e3fd5dc214dceb3741c509611e49544b4
633c2bf8daa82e1ed01edf9fe8fd535270a17e1d6ad7a413a9b0df9be02

Після шифрування на сайті отримали такий самий hex

Encryption

Modulus

1a9dda631e80ec25d43b443d99722f45f53d83245e8dd22120346fc1e409ce9ce9e7e9dfb604870412cc2e43bc52ed

Public exponent

10001

Message

ce2f396d04

Bytes

Ciphertext

9705948199D514C7723A1D9A8CFF3F5D0C37E3FD5DC214DCEB3741C509611E49544B4633C2BF8DAA82E1f

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Verify

Message

ce2f396d04

Bytes

Signature

18e97ed6a8f51d5f2f27d7ce4b9492e1519eebb7d4c386ed21ed8e1e89915f5c95dce4285e1288babcdde6500a0862

Modulus

1a9dda631e80ec25d43b443d99722f45f53d83245e8dd22120346fc1e409ce9ce9e7e9dfb604870412cc2e43bc52ed

Public exponent

10001

Verification

true

Додаємо функції для розсилання ключів по відкритих каналах

Функція приймає на вхід пару ключів абонента А, відкритий ключ абонента В і повідомлення, та повертає повідомлення з ключа та підпису

k1=279564076713784024553646056941679656125754470832344179250862574
550637301886436188355418380369103432841421496542234269538280777123
86136841365013298059985876

S1=246056597640108374433044755808342721736409218821464108894504965
899501821686842960874666899867054452703862696749077921279291624271
97822077599784746469753274

Абонент В отримує повідомлення, і за допомогою свого секретного ключа знаходить

k=885555555588

S=2908106630803468343834865881459075330077433986318976569769383059
235899190502953392046520266245959422208369440894703464629812870881
094712842572804098224011

Після чого використовуючи відкритий ключ абонента А, перевіряє підпис А

Message is verified

Висновок: Під час виконання лабораторної роботи ми ознайомились з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомились з системою захисту інформації на основі криптосхеми RSA, організували з використанням цієї системи схеми засекреченого зв'язку й електронного підпису, вивчили протокол розсилання ключів і закріпили навички набуті у попередніх лабораторних роботах.