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 №4

Виконали:
Студенти III курсу
Групи ФБ-95
Філюк Владислав
Яковенко Ірина
Перевірила:
Селюх П.В.

2021

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 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq ≤ p1q1 ; p і q – прості числа для побудови ключів абонента A, 1 p і q1 – 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) , (,) 1 n1 e та секретні d і d1 .

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.

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

Відкинуті значення::

Bad p (№1)::
109558319265192804954238703946844515498780598521534568464179164531919307436 61
Bad p (№2)::
217888759839610585818979719898856254970722594388860829180893484965196203403 23
Bad p (№3)::
254091531195930849675590094252063283410940176718124787162210898582984377822 56
Bad p (№4)::
630662661573437813080435299744953626657678535439406237657735432432037893424 96
Bad p (№5)::
537110038654855624505882640983899606148509074916799779386613803029565946270 79
Bad p (№6)::
104338591987445452642474630489436210651823109153961890448648378705887851846 93
Bad p (№7)::
563587687491190721476174779459024226516146832107597071997313855274060424453 69
Bad p (№8)::
523957039860050246565551597195545694351877661439739291390188038156238969048 56
Bad p (№9)::
976412230737377941359368610199236205160268871125087463200775856894709709863 27
Bad p (№10)::
625686396591586533510645496558922842533839316074150333584519831850731491623
Bad p (№11)::
610010591103600867230730939749200799881221709201248630877225358653905087316 70
Bad p (№12)::
539786594504303441874911063050326685160334379139521509260544446832734236183 61
Bad p (№13)::
624617354727524734922485101058072002933660951658826899371780076125085486089 1
Bad p (№14)::
160623022762303786209888460091849476099631670592947969898232677853857278930 20
Bad p (№15)::
586110389559622488640634302433083995958784863588391450571052162761632420401 68
Bad p (№16)::
695015374676731251534258877838801459202424035126591242162174885575127382250 48
Bad p (№17)::
885920056317702155850282763854263321743253632047400792403618658256345006364 64
Bad p (№18)::
362244875302144486212669526794689395164629735458785700594178482528319689981 91
Bad p (№19)::
599423962996863457475330463936859273820091059243999314445407578008058934625 5
Bad p (№20)::
556588054479191091248118937963712249106182885495625676591069666316754013046 33
Bad p (№21)::
895092822942443445511514500437583727291800418257854925599395977258228918051 70
Bad p (№22)::
147284129641058365816651551425405223353424352583959181940618580068727934215 50
Bad p (№23)::
705267649013307898652203972097487098234625223175550717753203276915770847923 63
Bad p (№24)::
100483946739346352390073878520999514017921844486353046228914148431039147442 361
Bad p (№25)::
570737269067040133541979609059465933386509490213006109892591754538594033277 4
Bad p (№638)::
111849344396969324602412538928001235974138418935089351089901854989303232638 65
Bad p (№639)::
270096210084557547459893483798881970703300643878882037277369203123097123002 85
Bad p (№640)::
379251962739050555861685914785836094873611991233789566790748588675676150868 97
Bad p (№641)::
744073005182769236307467944759911409297596871562150624547238529436582633728 96
Bad p (№642)::
195961392201053076275124111731580222761340915919920471717674573392034528360 14

A
p_A 110265230049730903399043606248343413611049047572414089071102446663406906795 91
q_A 140942107754806461697265837403750597863300000692773866566403591853332589134 39

B
p_B 900783624163144956344202302423715126895104821307141924982869100334634587613 23
q_B 114350009283407550801030291944489426924695193896358096199969136273218594957197

A
n_A
155410139352776964501470828136276837303167606108987829662486793923288663494076739572031549211837268062915598672537477815210693172506132565408291152923449
e_A
282310775807852886305841033215794097327194564566720776307854450668081047032361838445471910597673411524456737303574958651531438396696391340233944472342 3
d_A
643099389489482295309945395995952177806561767675562672576821545754031425132399515219155636313402878604275794269843437556646833588802871014575789974854 47

B
n_B
103004615785397123952016822194559122632733009033568543064517999611001282833210006720819379069817807677695521123630503350149515418115051033747739758240916 31
e_B
298720504394590759911160710769575368522924140897982369974714296020187733288766712333795560321122079144760344577538963785254904141480450974470210528704395
d_B
260420163368263327283334809259667679437323710568646021502592742599145954199272233024105861301241882303858864786018748417946353789400357704826979595757968 3

encoded data: (int) 48656C6C6F2C20776F726C6421 == (bytes) b'Hello, world!'

Message
657906118682181508616531496644347410714459381265412574539568704909313678872995969940004827414811942851406874504164123058484654480838648770636803054255360112490661600242889398727671768773168190634515274608736080894242335367112196735679637641584435644804148906185379346955361169311011099324552307083443650016 13
Sign
149430083466524558648642585346422612129825708337955525001093790492509656642671438631992748448994158565847188382746027478015654827635640205129420991562349
Tests::
start 256
conf 256
auth 256

Перевірка verify
{'verified': True}

Перевірка sign
{'signature':
'875F4CD0852EAC069657601E9B7BDF4E14ACDDCB94A70EDA8995289F4901426651019894 0A9ACB411159245FFEB2DD914386D0BD0BCFDC1419124CCEC9B22D15'}

Перевірка sendKey
{'key':
'1FEFB9CE09E0851BFD883ABD2768974318C4EC5A0DBFC28B05A87F53A3C9A7CDD2C69DCAFD80F61BCD4ED1C786EF67DD8A26E893237233E77799B847D510566B', 'signature':
'1BE94F1DBB99F096E245F53B0624AD4D551492BAC1A215B37B485F7AF62E339459F33E01898FDAB6C40FC89056F921F1C788428EBBA92CCF661BCAABDD8C922C'}

Перевірка за допомогою API сайту
Verify
{'verified': True}
Sign
{'signature':
'0756D2575EAC3EBA393DD100E13BC651AA56CCAF1C05D06079EDCB94BC23E437EDDA44C7F31589D4CFD9C17B9966B0E80DA77378E571867FD4A19EB2AEC66B78'}
SendKey
{'key':
'31D3A6AF4030619C96EA1B991009B5AFE3A518A12C5AF4E471A5B52C5796DFA1F7613619D637106E3F8AB7DF3FA04E6F9F6BFAE77469BCE2E107A667EFBD69', 'signature':
'0269DBC8CEBF31E50EF14BBD89E0808510EACB035E23393E5BA76DBB201329EC0D0C6AFDC7B794E8A65112814BD32673AC9670D2C78A42CF695AF3CC8C74FF2A'}
create_message
{'key': '48656C6C6F2C20776F726C6421', 'verified': True}
Encrypt
{'cipherText':
'4C45FDCE8A9C84B3AC9D1E9E7D12EA7B56B72F78AC26B753DA16E13E29F1C75065156EDBC2ADBFE1F23A9328901268D5D3177FBB1BC6C037FBBD78879CEC29F0'}
b'{"message":"48656C6C6F2C20776F726C6421"}'
Decrypt
{'message': '48656C6C6F2C20776F726C6421'}
text b'Hello, world!'

Висновки: *під час виконання даної роботи ми реалізували тест Міллера-Рабина, алгоритм асиметричного шифрування RSA, та на їх основі організували роботу протоколу конфіденційного розсилання ключів по відкритому каналу з підтвердженням справжності.*