КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №4

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

Роботу виконали:
студенти групи ФБ-14
Антонова Олександра і Веденкін Артем

Київ-2023

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.

Постановка задачі:

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 використовувати вбудований генератор псевдовипадкових чисел мови програмування Python. В якості тесту перевірки на простоту використати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч.
2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p1 , q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p1q1$; p і q – прості числа для побудови ключів абонента A, p1 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) , (e , n) та секретні d і d.
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().

1. Реалізуємо функцію пошуку випадкового простого числа, в якості тесту перевірки на простоту використаємо тест Міллера-Рабіна:

```
def is_prime(n, k=5):
    if n <= 1:
        return False
    elif n <= 3:
        return True
    elif n % 2 == 0:
        return False

    def miller_rabin_test(d, n):
        a = random.randint(2, n - 2)
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            return True
        while d != n - 1:
            x = (x * x) % n
            d *= 2
            if x == 1:
                return False
            if x == n - 1:
                return True
        return False
```

```
    d = n - 1
    while d % 2 == 0:
        d //= 2

    for _ in range(k):
        if not miller_rabin_test(d, n):
            return False

    return True

def generate_prime(bits=256):
    while True:
        candidate = random.getrandbits(bits)
        if candidate % 2 == 0:
            candidate += 1
        if is_prime(candidate):
            return candidate

random_prime = generate_prime()
```

```
Random number: 266969338192624445081953299385675599317209416763257063550846769431889792339 77
```

2. Реалізуємо функцію ***generate_prime_pair()***, що генерує дві пари простих чисел p, q і p1 , q1 довжини 256 біт при умові, щоб $pq \leq p1q1$.

```
Abonent A:
p = 96818864494404015589147040280097928210762184517367708708577760970546119896137
q = 62760884812165125112342416286330743907524966762407764992347522454126499023797

Abonent B:
p1 = 10199523194297379896629702092681282390345892177150333771911124239730434219510 7
q1 = 86480962284475753197877807301535859503515925154428322889830595410906097158733
```

3.

```
Abonent A:
p = 946143996265804788228486284981767032583258264006682881761962374747753179004156690058270838908279305701069024346
967967948336682092623040451174441465345066438791671969241927702090683736944000514055991045844785606844747719437 1111
423271056567833272432788575785150001372260495333588573375064466142281878137089333 9
q = 470646925574781703646748863456412943432993654806321074192619891991238926920429405794903994399687302686625676702
819544343444974267920865034459872182568225396016914774936353356486486202122557181378314482152962505351061205060706 9
998072760362149169523170477354820565803995011085312802286947987201920127054866313
Public Key (e, n): (445299762993538764298815738525883893759509296971141151759452515674948568144495119788043557554486
397532252602789277336496520240155336874792155981686541013593999563983772195746934901203999278160902939604884444013 9
179398363720595854080390993114258094036228395778618733241976601128694335345792507874934039295066947587521618731224 9
100560937509433195302434459577348702877346061683946917129100412595901026586103597952641292948263811508648049460836 6
258005772383842928886813084209209101712842050718626521989660783856185839304668105541195110688665637049493658094462 2
227631402556364542165496973961924995558010715308056027189107, 65537)
Private Key (d, p, q): (635977506083513562390239912202614285913149369005276588868650616707740451627702568189706118 2
385411002973653998621173834741130168514780170813128287624149522252100628471058366896725136479137641119543383153308 7
564318158729945294676201114785473083231764943433243631404292916993249074247811718843915793866179437961758638167946 8
302777808898964916806811392146873410677330297375388238304366265641074937644657313091233349631516765949039764558991 8
932088621413207622940897083980795269584255182098558336732231257024201725062670397526794729199600614467393839918652 2
349484478514377633210944861445067447711164011003978320972113153, 946143996265804788228486284981767032583258264006668
288176196237474775317900415669005827083890827930570106902434696796794833668209262304045117444146534506643879167196 9
241927702090683736944000514055991045844785606844747719437111142327105656783327243278857578515000137226049533388573
375064466142281878137089333 9, 470646925574781703646748863456412943432993654806321074192619891991238926920429405794 9
039943996873026866256767028195443434449742679208650344598721825682253960169147749363533564864862021225571813783144
215296250535106120506070699980727603621491695231704773548205658039950110853128022869479872019201270548663 13)
```

```
Abonent B:
p1 = 153627222895890706267274370531217008675294311575345232957958025833263894497149942685383826080212339414105353 31
481469198061524508286354126110300250307996106817851588329517968344236575174143893536506327785835108719574242017025 1
751218397660334160209342537285305581370693265412693292790482338359806140535505939539
q1 = 158549045304716967461162987178586529471821918907112406065560122407007552525604817677067799262787824240147831 28
501495479034699493024244646622868967830085203874921062709123459104257748114742224806948624537943893071432424432839 9
143442351472876188750065701580103950737127332692643386498502495466084332516632173887
Public Key (e1, n1): (243574495229584273833678280005718376554582636580135184237740708156165594113303642077622967192
617670336833621374638102102616229418105203763108432415238991619213499521631113843369054147219264988810029741847860 4
471505832854699272384507628459695190123682625739812172841001254818998289190104173999832836136292314604299318676525 5
219769442872506604910833388052164360226960048027708794029407639012509759662352665749926694331763010856042168295354 0
156351551784355375889982765221305370911956089280175724204469501509124627142362440675415160319323898714342993174287 4
379459615388153600141659683388155260579428581706680143395661809 3, 65537)
Private Key (d1, p1, q1): (781302628853368418531758365851383449338608142909745921089733305897266142705627227940825 7
955541463090805066433522502071198251941387929861769787244202746664966516909446159078406555466809349510344941705041 6
188142881113902484837731460273380654763395913844329618889325821335091330060935585043480514039216407734618544216742 0
031464230700242507375007181608566512652328003420558045359539995567445486733768030299204959975674497862879566198168 5
084404269469585031948887017541773556333817353757612681519838152432792336874035641737740244996397952966909059363575
847418613913196859555037791642167244507204475075111541385056161650068 1, 153627222895890706267274370531217008675294311
575345232957958025833263894497149942685383826080212339414105353314814691980615245082863541261103002503079961068178 5
158832951796834423657517414389353650632778583510871957424201702517512183976603341602093425372853055813706932654126 9
329279048233835980614053550593953 9, 158549045304716967461162987178586529471821918907112406065560122407007552525604 8
176770677992627878242401478312850149547903469493024244646622868967830085203874921062709123459104257748114742224806
948624537943893071432424432839914344235147287618750065701580103950737127332692643386498502495466084332516632173887
)
```

4.

```
Original Message: 368255758385057435686708261597316124476269156099985722307643112583924582150934437747410989062890 6
462853396616285697751761821947401644290160719589497007147777652366151895493030448983359948365050773097415833920992 1
414357799952998015240199524500229464741254847083522571695176102286866788694253088120082997091578529366257756981465 3
923798856282025945590362524175333248167178894981612309245910384966235778888006522997500191077240380281949432801488 0
537892957848779878346658168175788585452651252631201338838308607281318416048075984490459239387732074547056971586363 4
662358265687170724251728993102462953412102931136223491510

Encrypted Message A: 339947704875589015860962429888867594661810550369205744829606509631385403641843254372896589325 4
017733381841704038458243665638393556794216828007329939831927874345800902374062691484804374916057703934895464984957
969324592188376044519845421407263450035615394102170934146627616377950285380808953409619435976308541059031591476435
055501132344470398195049075059966964990960188128274463504224897608162304782484699132664992326958068673674595819388 1
271522800055261564790469086724205987743415062549941173247311377639010831631806928378194933522928833440475450011248 6
043040671378355574889031168631636091601462037655820356411289
Decrypted Message A: 368255758385057435686708261597316124476269156099985722307643112583924582150934437747410989062890 6
906462853396616285697751761821947401644290160719589497007147777652366151895493030448983359948365050773097415833920992 1
921414357790952998015240199524500229464741254847083522571695176102286866788694253088120082997091578529366257756981465 3
653923798856282025945590362524175333248167178894981612309245910384966235778888006522997500191077240380281949432801488 0
880537892957848779878346658168175788585452651252631201338838308607281318416048075984490459239387732074547056971586363 4
634662358265687170724251728993102462953412102931136223491510
```

```
Encrypted Message B: 816972651052109253134515465591802252225733744148139563135949404556464710398235699628401642517021928729725431240767639431022216645441234497393745546754748790685456580871127908648773519035275464514728507458985382373083434158465540261378131320912613848177314501954953854441718386260401244146100280408813164729782981805173897974485896263762062811092630876776358085169274666069809046323674473197225862127506516128195889074349821590669138546687645798663024213177949868695264787481855557194325525630406356742966588768817800364014587351739836518622825685083279132774458837921535144853951563044269346902059958307234520958147 0
Decrypted Message B: 368255758385057435686708261597316124476269156099985722307643112583924582150934437747410989062890646285339661628569775176182194740164429016071958949700714777765236615189549303044898335994836505077309741583392099214143577909529980152401995245002294647412548470835225716951761022868667886942530881200829970915785293662577569814653923798856282025945590362524175333248167178894981612309245910384966235778888006522997500191077240380281949432801488053789295784877987834665816817578858545265125263120133883830860728131841604807598449045923938773207454705697158636346623582656871707242517289931024629534121029311362234915 10
```

```
Signature A: 100366286021320050765230133001915129743259247305396525983700623167065863089652033012809313767766835346689299050037001039230039082211721938013877816109314398354518923989102545939340584141546865276290309522305925984015569248491608299146397346101522571564785709029486878884917056721125306442909964179827737432019806918404928932523110502208906441723532295112774680241576283213141432175952955216358908352653056225489799640141605626872580744305826140825323937434702713209105064010153339533581438852008679992872916065707795071477608002168244976895291458152438044702452019810172944496307987280208071784445449407307227386281
Verification A: True

Signature B: 235524869676589486491038668758871814184127165920700532430110696249285390956226180340451191511013328079855503576191312768521479311988945192308041114347638070274727302670902901833452424206335486537220952865835455681381869133981956593547813186646932056366404952590757654748868150470244180923040566440691900150596428708305509298121663692775811290563641948324681270680748401589251424286568990155949776517774542988583354317867912841149000089720556654304544405163596074189832877564679827766895557973441522585456251062043543532504642251725252711980679243979928803630557041609904975174022277238118446189289315293933862723900 4
Verification B: True
```

Висновки:

У ході виконання комп’ютерного практикуму ми практично ознайомилися із асиметричною криптосистемою типу RSA, створивши тест перевірки чисел на простоту алгоритму Міллера-Рабіна, реалізували засекречений зв'язок й створення електронного підпису, вивчили протокол розсилання ключів, перевірили свою роботу за допомогою вказаного вище сайту.