Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

Криптографія

Лабораторна робота №4

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

Виконала:
Студентка 3 курсу
Гузенко Г. С.

Київ – 2021

Мета: 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.

Завдання:

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 $0 < k < n$.

Хід роботи

В якості функції пошуку випадкового числа я обрала функції з бібліотеки random: randint(), getrandbits(), аби згенерувати число розміром 256 за завданням. Перевірка на простоту, я почала з ділення на відомі перші 16 простих чисел, якщо випадкове число ділилось на них відповідно воно не було простим. Далі за завданям я реалізувала тест Міллера-Рабіна, в якому спочатку знайшла розклад, а потім перевіряла простоту числа за основою випадкового числа x. Для пришвидшення роботи алгоритму, я також зробила функцію піднесення до степеня по модулю за схемою Горнера. Вкінці функція random_number() обов'язково поверне сильно просте число.

У функції get_keys() підбираються 4 випадкових простих числа, такі що добуток перших двух, менший за добуток наступних двух. У build_rsa() заповнюється просто словники, по суті збереження даних в глобальних змінних. Далі реалізовані усі необхідні функції: шифрування і розшифруваннння, а також шифрування і розшифруваннння з аутентифікацією.

Дані отримані під час роботи коду:

Частина кандидатів які не підійшли:

Posibble:
109297875822414814348084232951008841716105662358627595749529275077601836144065
Posibble:
467228407112112595110133369240047603387611906726394537748776704675807028265 74
Posibble:
965603756961498422415869906452617170278816103292344299239059658653182186059 28
Posibble:
276625513042220643437186602451940975476964477489066262468094245776022951483 98
Posibble:
798326979587919519736443606558258108395883359202426703024204965431533165208 82
Posibble:
497812211673900076096578945632073690447639277246579434455259689436686486280 90
Posibble:
458571936007408872058874905472147662005064569575894750711127024824331740173 16
Posibble:
412397891664485528335175722633229259438787464675393308310736695242002737421 84
Posibble:
106197594475754997547811475570682828113823437210547080506614970470562218295 83
Posibble:
101643900455959022770538996118296358706922574700926962758550471460683194871 688
Posibble:
415914423249834502374750785337432932848489079002573880391828213877442146483 68

Posibble:
6342466866816616991747657538154773685210613940969771440073738179858515278716 8
Secret A:
7541692023181143482955409992924180733399563111968748835813280639423941918945739459481668001590427396107988784410959916312379386606075188426782446089549 5
Open A: {'e':
3506935549770207100266749162575627856593162836052974960117397462914112455547007571096887354788921773244684304040007553976697100476689861181632996126057 71,
'n':
5837230304986059610986187496788815569097283907463740660620117020295296807923496668482214556465149906358826729841699949571920280431062633139599537222161 37}
Secret B:
2543001516814800492688123104953093801563398470266167533296156564147888576037515386630060606823408399229493281476006013001723035264255505886787420929698971
Open B: {'e':
1665657550260372230642259364813080120437916163677987174892691937215992709397711933826571431680126590456970699379198936945682509929857834207484909768970247, 'n':
2928123239134728899770579853707268731629311989182255374414778180413500722121819908105429211639062148918413448302905586877162640875232577104820338291549 93}

```
its cool
Secret A:  754169202318114348295540999292418073339956311196874883581328063942394191094573945948166800159042739610798878441095991631237938660607518842678244608954 95
Open A:  {'e': 350693554977020710026674916257562785659316283605297496011739746291411245554700757109688735478892177324468430404000755397669710047668986118163299612605771, 'n': 5837
Secret B:  2543001516814800492688123104953093801563398470266167533296156564147888576037515386630060606823408399229493281476006013001723035264255505886787420929698971
Open B:  {'e': 1665657550260372230642259364813080120437916163677987174892691937215992709397711933826571431680126590456970699379198936945682509929857834207484909768970247, 'n': 292
Plaintext:  1493061263176362252098644964359526154576574066189293995419879050158433470723480645780120545308954177744930433415231072186224253570891757794658930191 75
Ciphertext:  1778754936278057047952751841636991232130846489254553993916855989011082078287461937340467310652302231806076854980383689200537398682993290552571609141881580
ok
```

Випадкове к =
1493061263176362252098644964359526154576574066189293995419879050158158433470723480645780120545308954177744930433415231072186224253570891757794658930191 75
Надпис its cool означає, що А пройшов аутентифікацію, а надпис ок, що повідомлення розшифровано правильно.

```
main
"C:\Users\Гузенко Галина\AppData\Local\Microsoft\WindowsApps\python3.9.exe" E:/Уче6ка/git_crypto/fb-labs-2021/cp_4/huzenko_fb-91_cp4/main.py
Plaintext:  2209200963432161652838048107682084753803790251867808089351849118285208376172233243471555704819488529663793998804661921920506588078112084085425310550047674
Ciphertext:  2889428180530185705586567165554817871297957773939945192563600057343715976898817644257455530104405314956650324314094650960818963401581200317865570785937158628777153248624
Signature  6345618009318882307154031460278232875567008948750811141406167540516538516772871796834954524282569281379638605297991881416446381790347559671432962747511284454703092914055
```

Get server key

Clear

Key size 1024

Get key

Modulus 869694C8547C4E01C310FD5CD6815B2D5C1D4EC67C6380556ECBF45213BB61A660AB4C879DE622348A09

Public exponent 10001

Receive key

Clear

Key 29259632FC97F8D3CB4E13607755A0F092F1734D103AC4793F9169068C387896D71182F9E1FD603B1220A5

Signature 5A5D566CCDDB75C8CF706AA22829214DF21BE8B2C9539E2B9CADCCC6C9D8B207818E1BAF2A4F3DD430

Modulus B252DC1DC60A009C1255421893CCD50D0E9A724A38B669AF47669126D823DD48F742A1831811CE4ECC95

Public exponent 6B2278614DAEA2F9272437206D6C51296F9715E101C02E2DA462C245B915D23808FC9E8D7648BC2CEFAA0

Receive

Key 0437D5BE71A9B8FFFECBF6C825DD797381CF3F67B6E2B85A86A16EA820F73B4C485997213563BF5B685B4

Verification true ✔