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

КРИПТОГРАФІЯ КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

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

Виконали: Вісловух Владислав Ісаченко Федір Варіант 9 Група: ФБ-06

Мета та основні завдання 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.

Хід 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.

Для пошуку випадкового простого числа ми вибрали пошук з заданого інтервалу(min=2**256,max=2*min-2)

Для генерування рандомних чисел використовували randint з random.

В якості тесту на перевірку був вибран тест Міллера-Рабіна, також використовувалися попередня пробні ділення.

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.

Згенерував пару де pq<=p1q1 для позбавлення подальших проблем.

p = 132405041945393801550531400419169696591181392528771732715668415064594306698 603

q = 152759408146090984034884488540376560975618711399302167507989714588627982065 373

p1 = 163514984698926303865492995117947175457118976699021058292165551453816147731 421

q1 = 125717298368234364773928873183920154123633426595585713332882910114617404988 461

- 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 е та секретні d i d1.
- 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 М 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.

Приклад роботи коду:

е для абонента A : 8843264818144953645539071619234246432063995337631352667609212725640758238497231617440085573610331602073510877478342892116910545014518652473353050147481
п для абонента A : 35706694380567503317153091132795547705487530106068704185126024752993059660681546455389079476278802728528658155337665723884560331202813990045246748185564009

d для абонента B : 238822668445165840973360190301624388884949998354538955009874459484339063955806369824999947341600893519257324967022602206157664181918959864737274342627126461

e для абонента B : 374990840695713163521246834954347599120274545814655520399124047254604053058850982749992773797966223370515315154437487572418825758118734493982451919259

d для абонента B : 51306878033286451336755954356200621334366519672465281882163802066691307572191549979370811746830955348185199766532221501424767263940886222831622697

Message : 2581255178848869861233139982967522211355822206128061102471126578396730734869014414117308716923700455299781243609041812284743398166211159962614936725071405

Зашивроване повідомлення : 27679241308364851534040719514547572529043773128628248038948526122439062627797683282150941346193067613906058202573542646087069397443131989570100044448691163

Розшифроване повідомлення : 25812551788488069861233139982967522211355822206128061102471126578396730734869014414117308716923700455299781243609041812284743398106211159962614936725071405

Ключ к : 270283625245158762717240575888972088291617218996589105143484092769218473505214548344089967700932797527539579540880668843475581076495287276559762598223682

Ключ к 1 : 27028362524515876271724057588897208829161721899658910514348409276921847350521454834408996770093279752753957954080662843475581076495287276559762598223682

Ключ к 1 : 2702836252451587627172405758889720882916172189965891051434840927692184735052145483440989677009327975275395795408066283475581076495287276559762598223682

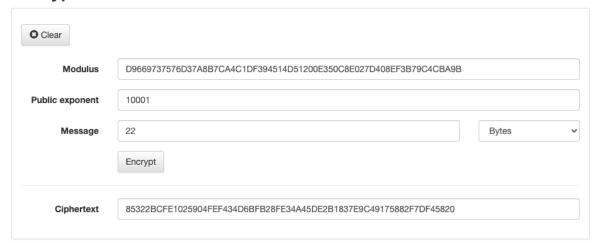
Kлюч к 1 : 270283625245158762717240575888972088291617218996589105143484092769218473505214548344098967700932797527539579540806628333398296752229135582220612

Порівняння роботи сайту та мого коду.

Get server key



Encryption



```
# mod D9669737576D37A8B7CA4C1DF394514D51200E350C8E027D408EF3B79C4CBA9B

ACLE_n = 98333150198878732362049457536571285314575200836982636537534492909076445182619

# public exp (e) 10001

ACLE_e = 65537

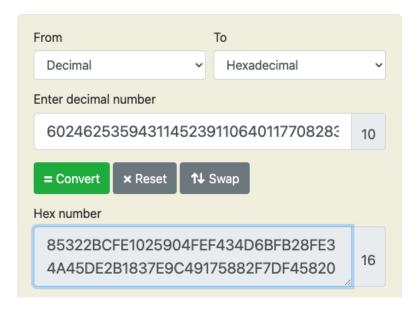
# message 22

ACLE_message = 34

ACLE_encrypt = encrypt(ACLE_message_ACLE_e_ACLE_n)

# encrypt 60246253594311452391106401177082839448753614472186264481298290795565785831456
```

Decimal to Hexadecimal converter



Як ми бачимо все шифрування працює коректно

Sign



Verify



```
# signature 35EADA246FE0A0C28D1EB49E92AE11A982E8890E2543ACB3F4B516C0253958F7

ACLE_sign = 24387528751115011838962269802416685358045221117918323190434208470255678347511

ACLE_verify = check_sign(ACLE_message, ACLE_sign, ACLE_e, ACLE_n)

print(ACLE_verify)
```

/Users/qqakkashi/study/crypto/lab4/venv/bin/python /Users/qqakkashi/study/crypto/lab4/main.py ['Verification ok', 34]

Як ми бачимо написаний код працює відповідно, так отримує такі ж результати як на сайті.

Висновки

По ходу роботи ми ознайомилися та використали на практиці тест Міллера-Рабіна для перевірки текстів на простоту. Також дізналися про методи генерації ключів для криптосистеми RSA. Практично по-працювали з системою RSA організував секретний зв'язок та обмін даними за електронним підписом. Перевірили правильність нашої системи завдяки онлайн ресурсу.