



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий Фізико-технічний інститут Кафедра
інформаційної безпеки

КРИПТОГРАФІЯ

Комп'ютерний практикум №4

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

Виконали:

Студенти ФБ-33

Дохоян Юлія

Терещенко Микола

Київ – 2025

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 $1 < p, q$ довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $p \cdot q \leq p_1 \cdot q_1$; p і q – прості числа для побудови ключів абонента А, $1 < p$ 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) , $(,)$ і n і 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 і 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 $0 < k < n$.

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()`.

Для кожного A та B генеруємо два простих числа p та q за допомогою функції `generate_prime()`.

Просте число перевіряється функцією `is_prime()`:

Якщо число не проходить перевірку, воно додається до лічильника невдалих кандидатів.

У результаті для кожного абонента отримано два простих числа p , q та кількість кандидатів, які не пройшли перевірку простоти.

```
Кількість кандидатів, що не пройшли перевірку простоти: 78
Кількість кандидатів, що не пройшли перевірку простоти: 47
Кількість кандидатів, що не пройшли перевірку простоти: 17
Кількість кандидатів, що не пройшли перевірку простоти: 114

=== ВИБРАНІ ПРОСТІ ЧИСЛА ===
A: p = 76070130507590958416480387068517507731821467091777594525186509686748893495433, q = 66272738589144604870767175141011058795204060432907321293417820158632085422619
B: p1 = 81655170586010940439382684886810038011652556651938204590862652656284763274073, q1 = 1115663475004179287672569900512431969040962773791464519623616213094486463726
09
```

Генерація ключів RSA

1. Обчислюємо модуль $n = p \cdot q$.
2. Обчислюємо функцію Ейлера $\varphi(n) = (p - 1)(q - 1)$.
3. Генеруємо відкритий ключ e та перевіряємо, щоб $\gcd(e, \varphi(n)) = 1$, використовуючи розширений алгоритм Евкліда `extended_euclid()`.
4. Обчислюємо приватний ключ d як обернений елемент до e за модулем $\varphi(n)$.
5. У результаті для кожного абонента отримуємо: відкритий ключ: приватний ключ:

```
=== ПАРАМЕТРИ RSA ===
A:
nA = 3766457626522231638992080889376955095250822019858460483322593049073948268359194593242552103321335623072384046011289115608381174278567530216722689660128667 (модуль, modulus)
eA = 3119266731932881423977513509265615602636315719935101818316398163095030356395440718065425379737665036363036549527780236876086254984194145755616991398779893 (відкритий ключ, public exponent)
dA = 2383344076324988863960205044970855550985950420120588680915876186589269887326149087242817320484612973731387848200677663988614329604923200606098528976334557 (приватний ключ, private exponent)
B:
nB = 9176904572402414596233174240385700297783357499654239590705480307113369609068132015733576398877177416774976940058471790432415154302119124817043993294768303 (модуль, modulus)
eB = 633436946852250944437736057359687361592223283778717542838621785288919842467582742209840916334125239755071300301443285402121799161559796697815284749916617 (відкритий ключ, public exponent)
dB = 7727911079739721689395032001429432263295179528506096438374684083301102514290108320409606127100718478962687301060048703495961340444812623013264385844525753 (приватний ключ, private exponent)
```

Далі обираємо відкритий текст випадково

```
=== ПОЧАТКОВЕ ПОВІДОМЛЕННЯ ===
M = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184 (випадкове повідомлення, 0 < M < min(nA, nB))
```

Шифрування та підписування повідомлення

1. Абонент A формує цифровий підпис:

$$\text{Signature} = M^d \bmod n_A$$

за допомогою функції `Sign()`.

2. А шифрує повідомлення та підпис для В за допомогою відкритого ключа В.

```
=== ШИФРУВАННЯ ТА ПІДПИС ===
Encrypted M = 8779718917324347258802418752568099319999563971092651172229531934702907169503872554380386637183415081806313314775782285037572881507469021145301605859223734
(шифротекст, ciphertext)
Encrypted Signature = 19347970389444885858065150197975527312747107167884137964920523308420265318475164475165413511396953443057789261385417523629589012133642938269339882
40256313 (шифрування підпису, encrypted signature)
Signature (A) = 23123727413981078734562230361060640793327623541217696272468848810979652000185834571465280858407764837752007925538220747367449081944079324988737270812373
02 (цифровий підпис, signature)
```

Розшифрування та перевірка підпису

```
Decrypted M = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184
(розшифроване повідомлення, plaintext)
Signature valid = True (перевірка підпису, True/False)
```

А обирає ключ K .

А шифрує та підписує його для В, відправляє зашифроване повідомлення і підпис.

В розшифровує отримане повідомлення і перевіряє підпис.

Результат: В отримав ключ K

```
A обирає K = 748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184
A шифрує та підписує K для B:
  Encrypted message = 87797189173243472588024187525680993199995639710926511722295319347029071695038725543803866371834150818063133147757822850375728815074690211453016058
59223734
  Encrypted signature = 193479703894448858580651501979755273127471071678841379649205233084202653184751644751654135113969534430577892613854175236295890121336429382693398
8240256313
B розшифровує та перевіряє підпис:
  Decrypted message = 74888624540498831749898612953400742248683128628828100431731391229481131161002605198000246873935669407016591554213139248779423914170844736873809523
4507184
  Signature valid = True
```

Перевіримо на сторонньому сервері

Search for a tool

SEARCH A TOOL ON DCODE

e.g. type 'caesar'

BROWSE THE FULL DCODE TOOLS' LIST

Results

Decryption using C,D,N

748886245404988317498986129534007422486831286288281004317313912294811311610026051980002468739356694070165915542131392487794239141708447368738095234507184

Copilot+PC

ProArt

ProArt P16 H7606

Неперевершений ноутбук для творчості

Найшвидші й найрозумніші ПК з Windows.

Купуй вже зараз

RSA Cipher - dCode

Tag(s) : Modern Cryptography, Arithmetic

RSA CIPHER

Cryptography > Modern Cryptography > RSA Cipher

MAZDA CX-5 від 1 274 000 грн.

MAZDA CX-5 лімітована пропозиція на автомобілі 2025 р.

Mazda

Докладніше >

RSA DECODER

Indicate known numbers, leave remaining cells empty.

VALUE OF THE CIPHER MESSAGE (INTEGER) C=

8779718917324347258802418752568099319999563971092...

PUBLIC KEY E (USUALLY E=65537) E=

65537

PUBLIC KEY VALUE (INTEGER) N=

9176904572402414596233174240385700297783357499654...

PRIVATE KEY VALUE (INTEGER) D=

7727911079739721689395032001429432263295179528506...

FACTOR 1 (PRIME NUMBER) P=

FACTOR 2 (PRIME NUMBER) Q=

INTERMEDIATE VALUE PHI (INTEGER) Φ=

DISPLAY

PLAINTEXT AS CHARACTER STRING

COMPUTED VALUES (C,D,E,N,P,Q,...)

PLAINTEXT AS INTEGER NUMBER

PLAINTEXT AS HEXADECIMAL FORMAT

Вибрати м

Технології

Summa

RSA Dec

RSA Cer

Comple

What is 1

(Definition)

How to e

cipher?

How to c

cipher?

How to g

How to r

ciphertext?

What are

attacks?

How to c

the private

Why usin

e=65537 for

How to c

into plainte

What is i

When wa

Similar

Multiplic

Prime Fa

Тепер протестуємо на сайті RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Verify

✖ Clear

Message

370176EA3B37E8DE93DA19AFB9A3D272FBBB3136292EB3C12ABF25B34A9ED

Bytes

Signature

1B4670C210A56DB06ED9E533494BCCA63412BBD35AD5A33A5D1006EAB87A70D1B23D792F698A640AA0BC

Modulus

7A86A5CAC22E9B59D66452E230486B112BDD99D1CF1E432D85ED38911F2CE3811D506B4B91143E05DF485

Public exponent

7388599E81F05519A34558F9E3550324072F3C1B65BB7D261664F4C8621EA79E82C542A49EE43E825951287

Verify

Verification

true

✓

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Encryption

✖ Clear

Modulus

DE7AFF460EC3CB5CB95B6C97286277934BCF8CA29D7FC83287ED924CF164D141105B6ACBB4F10BF EADB

Public exponent

281792D816654C39DF08454A05EE650005B88E75534D2DFA61D64456E1AC0D52B7089E335B303E8E85AEA7

Message

7DC7DAB12ABF9AB5E7BC73A874FCD8BE62ED478584792518F6A68771EE0C6

Bytes

Encrypt

Ciphertext

370176EA3B37E8DE93DA19AFB9A3D272FBBB3136292EB3C12ABF25B34A9ED6A10FE49A8A0640D449F5301

Зашифруємо локально, а розшифрую на сервері

```
Модуль (HEX): AE69FEE0621279E4158FC5DB95DB1DFF6C09370D680BD843AE9612DA7DC0C3D5
Початкове повідомлення M (HEX): 79449127C865A1F8C0A4F5DE22D7E9B3E0EE4A88545256716343EABBBBC58C8B7
Зашифрований текст C (HEX): 93D382DED3BCEB9FDA084D057E85277E2176CA92E6E16EEEFBB299F460D6104A
```

Decryption

✖ Clear

Ciphertext

93D382DED3BCEB9FDA084D057E85277E2176CA92E6E16EEEFBB299F460D61

Bytes

Decrypt

Message

79449127C865A1F8C0A4F5DE22D7E9B3E0EE4A88545256716343EABBBBC58C8B7

Висновок: лабораторна робота успішно продемонструвала теоретичні та практичні аспекти роботи алгоритму RSA. Було перевірено всі ключові етапи: генерацію ключів, шифрування, розшифрування, створення цифрового підпису та його перевірку. Реалізований протокол конфіденційної розсилки ключів показав ефективність асиметричного шифрування для забезпечення безпеки переданої інформації.