НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

з лабораторної роботи № 4

із дисципліни «Криптографічні методи захисту інформації»

на тему

Шифрування з відкритим ключем на основі алгоритму RSA

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-01 | ст. викладач Бай Ю. П. |
| Боженко А. О. |  |

Київ — 2022

ЗМІСТ

[Постановка задачі 2](#_Toc103458663)

[Основні теоретичні відомості з асиметричних криптосистем 3](#_Toc103458664)

[Математичне підґрунтя алгоритму RSA 4](#_Toc103458665)

[Опис алгоритму RSA 5](#_Toc103458666)

[Генерація ключів 5](#_Toc103458667)

[Контрольний приклад 1 6](#_Toc103458668)

[Шифрування і розшифрування за алгоритмом RSA 8](#_Toc103458669)

[Контрольні питання 9](#_Toc103458670)

[Список літератури 10](#_Toc103458671)

[Додаток 1 11](#_Toc103458672)

[Додаток 2 13](#_Toc103458673)

Мета роботи: розробити асиметричну криптосистему на основі алгоритму шифрування RSA.

# Постановка задачі

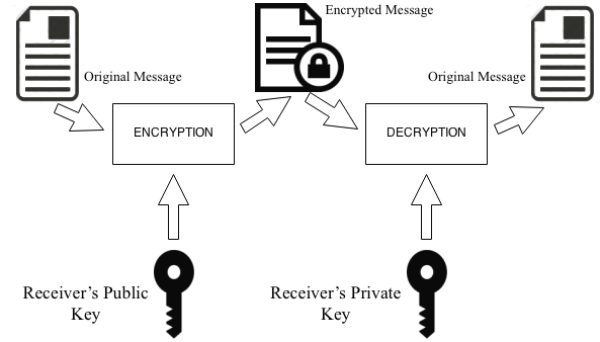
1. Скласти програму, яка дозволяє виконувати шифрування та розшифрування за алгоритмом RSA. Перевірити роботу програми на контрольних прикладах. Навести скріншоти детального покрокового виконання алгоритму.

2. Виконати дії ОДЕРЖУВАЧА і розшифрувати задане повідомлення, користуючись алгоритмом RSA. Необхідні результати занести до [Таблиця RSA](https://docs.google.com/spreadsheets/d/1IeqT1byTw2PgTOQXvXaozfia_Dh4pSM5RaWOLqZCtEM/edit?usp=sharing).

УВАГА! Числа n в стовпчику D мають бути унікальними.

# Основні теоретичні відомості з асиметричних криптосистем

Асиметричні криптосистеми — ефективні системи криптографічного захисту даних, які також називають [криптосистемами](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) з відкритим ключем. В таких системах для зашифровування даних використовують один [ключ](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F)), а для розшифровування — інший (звідси і назва — асиметричні). Перший ключ є відкритим, тому може бути опублікованим для використання усіма користувачами системи, які шифрують дані. Розшифровування даних за допомогою відкритого ключа неможливе. Для розшифровування даних секретний ключ (закритий). Ключі пов’язані математично. Отримувач генерує закритий та відкритий ключ, передає відкритий ключ відправнику повідомлення (мал. 1). Переваги асиметричного шифрування: безпека шифрування (дані захищені від атак зловмисника-посередника (MiTM) та інша перевага: аутентифікація, тобто ключ шифрування належить лише людині, що шифрує.



Мал. 1 – схема асиметричного шифрування

Один з типів асиметричного шифрування: RSA (абревіатура від прізвищ [Rivest](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%A0%D1%96%D0%B2%D0%B5%D1%81%D1%82), [Shamir](https://uk.wikipedia.org/wiki/%D0%90%D0%B4%D1%96_%D0%A8%D0%B0%D0%BC%D1%96%D1%80) та [Adleman](https://uk.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BE%D0%BD%D0%B0%D1%80%D0%B4_%D0%90%D0%B4%D0%BB%D0%B5%D0%BC%D0%B0%D0%BD)) — криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі [факторизації](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D1%8F) великих цілих чисел. Алгоритм RSA складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів.

# Математичне підґрунтя алгоритму RSA

В основі алгоритму RSA полягає складність задачі факторизації великих чисел. Факторизація числа – це розклад числа на прості множники. Просте число – число, щ ділиться лише на себе та на 1. Взаємно прості числа — це цілі числа, які не мають спільних дільників, окрім 1, або, інакше кажучи, якщо їх найбільший спільний дільник дорівнює 1.

Теорема 1. Мала теорема Ферма

Якщо р – просте число, то xp-1 mod p = 1 для будь-якого х, взаємно простого з p.

Визначення. Функцією Ейлера φ(n) називається число додатних цілих чисел, менших вiд n і взаємно-простих з n.

Зауваження. Якщо n ‒ просте число, то φ(n) = n ‒ 1

Теорема 2.

Якщо n = p⋅ q, де p і q – прості числа, p ≠ q, то φ(n)=(p – 1)⋅ (q – 1)

Теорема 3.

Якщо n = p⋅ q (p і q – прості числа, p≠ q), x – взаємно просте з p і q, то xφ(n)mod n = 1

Наслідок 1.

Якщо n = p⋅ q (p і q – прості числа, p≠ q), e – взаємно просте з φ(n), то відображення Ee,n: x→xemod n буде взаємно однозначним.

Наслідок 2.

Якщо n = p⋅ q (p і q – прості числа, p≠ q), e – взаємно просте з φ(n), то існує таке ціле d, що e∙d mod φ(n) = 1.

# Опис алгоритму RSA

Алгоритм використовує два [ключі](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F)) — відкритий (public) і секретний (private). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

### Генерація ключів

Для того, щоб згенерувати пари ключів виконуються такі дії:

1. Вибираються два великі [прості числа](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) {\displaystyle p\,}p і {\displaystyle q\,}q приблизно 512 біт завдовжки кожне
2. Обчислюється їх добуток {\displaystyle n=pq\,}n = p \* q
3. Обчислюється [функція Ейлера](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_%D0%95%D0%B9%D0%BB%D0%B5%D1%80%D0%B0) {\displaystyle \varphi (n)=(p-1)(q-1)}
4. Вибирається ціле число {\displaystyle e\,}e  таке, що {\displaystyle 1<e<\varphi (n)} та {\displaystyle e\,}e [взаємно просте](https://uk.wikipedia.org/wiki/%D0%92%D0%B7%D0%B0%D1%94%D0%BC%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%96_%D1%87%D0%B8%D1%81%D0%BB%D0%B0) з {\displaystyle \varphi (n)}
5. За допомогою [розширеного алгоритму Евкліда](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D1%96%D0%B4%D0%B0) знаходиться число {\displaystyle d\,}d таке, що {\displaystyle ed\equiv 1{\pmod {\varphi (n)}}}(e \* d) mod = 1

**Шифрування**

Одиницю повідомлення – m, шифрують за фомулою  ,

де n = p \* q, e – відкритий ключ, с –защифрована одиниця повідомлення.

**Розшифрування**

, де d – закритий ключ.

Наведений вище варіант шифрування називається RSA з підручника і є цілком уразливим. В жодному разі його не можна використовувати в криптосистемах.

# Контрольний приклад 1

1. Контрольний приклад 1 ([RSA-encryption](https://brilliant.org/wiki/rsa-encryption/))

p = 11, q = 17, e = 3

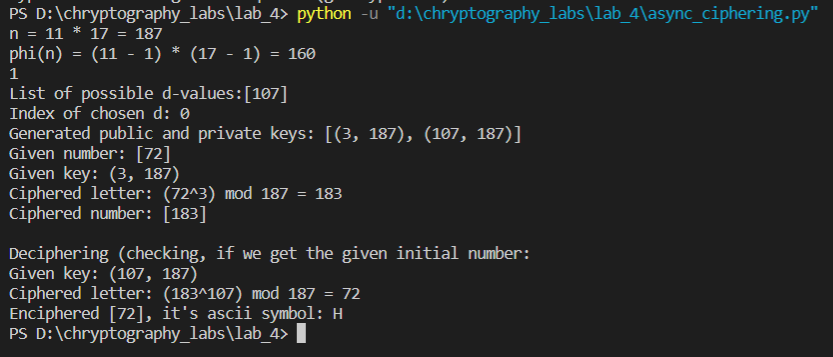
public key {e, n} = {3, 187}

private key {d, n} = {107, 187}

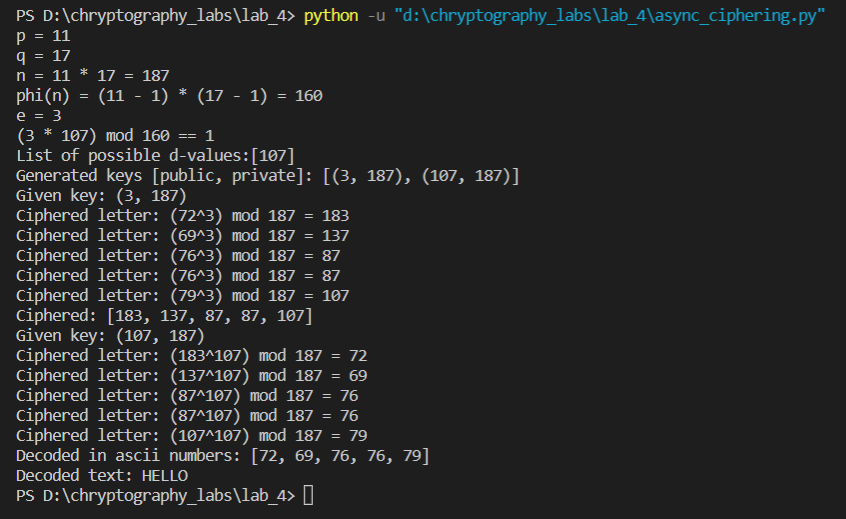
M = 72

C = 183

M’ = 72, text = chr (72) = {H}



Мал 2 – скриншот до першого прикладу (шифрування певних літер)



Мал 3 – скриншот до першого прикладу (шифрування всього слова)

Контрольний приклад 2 ([RSA uk.wiki](https://uk.wikipedia.org/wiki/RSA))

p = 3557, q = 2579, e = 3

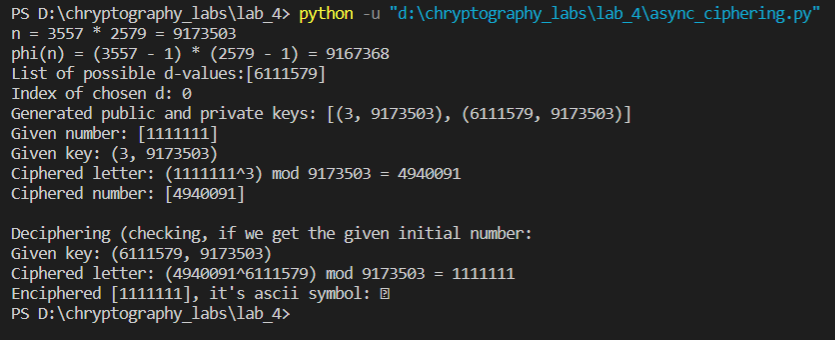
public key {e, n} = {3, 9173503}

private key {d, n} = {6111579, 9173503}

M = 1111111

C = 4051753

M’ = 1111111



Мал 4 – скриншот до другого прикладу

# Шифрування і розшифрування за алгоритмом RSA

Таблиця 1.

|  |  |  |
| --- | --- | --- |
| Крок | Опис кроку | Результат |
| 1 | Обрати два довільних простих числа p і q  p ≠ q; 1 < p, q < 200 | p = 41  q = 3 |
| 2 | Обчислити добуток n = p ∙ q. Увага! n > 90 та має бути унікальним в стовпчику D в [Таблиця RSA](https://docs.google.com/spreadsheets/d/1IeqT1byTw2PgTOQXvXaozfia_Dh4pSM5RaWOLqZCtEM/edit?usp=sharing) | 123 |
| 3 | Обчислити функцію Ейлера φ(n) = (p – 1) ∙ (q – 1) | 80 |
| 4 | Обрати відкриту експоненту e : 1 < e < φ(n),  e – взаємно просте з φ(n) | 17 |
| 5 | Обчислити секретну експоненту d таку, що  (e∙d) mod φ(n) = 1 | 33 |
| 6 | Зберегти закритий ключ {d, n} | {33, 123} |
| 7 | Опублікувати відкритий ключ {е, n} | {17, 123} |
| 8 | Одержати від відправника / викладача зашифроване повідомлення C. Дії відправника:  1) обрати текст для шифрування M; 2) символи тексту замінити цілими числами mi згідно з таблицею ASCII; 3) виконати шифрування за формулою ci = (mi)e mod n . | [19, 35, 36, 12, 95, 79, 82, 71] |
| 9 | Розшифрувати задане повідомлення C:  1) обчислити mi = (ci)d mod n; 2) поставити у відповідність знайденим цілим числам mi літери англійського алфавіту, записати одержане слово | [67, 65, 84, 69, 71, 79, 82, 89]  CATEGORY |

В процесі шифрування використовується наступне перетворення літер англійської абетки в коди ASCII: ord(‘A’) = 65, chr(65) = ‘A’ .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | … | Z |
| 🡫 | 🡫 | 🡫 | 🡫 |  |  |
| 65 | 66 | 67 | 68 | … | 90 |

# Контрольні питання

1. В чому полягає принципова відмінність асиметричних криптосистем від симетричних?

Відповідь: у симетричних криптосистемх для шифрування та розшифрування використовується однаковий ключ, тоді як в асиметричних – ключі різні (відкритий та закритий).

1. Що таке одностороння (однонаправлена) функція з секретом? Наведіть приклади односторонніх функцій.

Відповідь: одностороння функція – це функція, значення якої за данимим аргументами знайти легко, а от за значенням знайти аргументи дуже важко або неможливо. Приклад односторонньої функції – знаходження добутку чисел, отримавши числа, легко знайти добудок, а от якщо дане число та треба знайти його множники – то варіантів можливих аргументів багато.

1. Складність якої математичної задачі полягає в основі алгоритму RSA?

Відповідь: складність факторизації великих чисел.

1. Як визначається і обчислюється функція Ейлера?

Функція Ейлера – це кількість чисел, що не більші за дане число та є взаємнопрості з ним (мають спільний дільник лише одиницю).

1. Як пов’язані між собою відкритий і закритий ключі в алгоритмі RSA?

Відкритий (e) та закритий (d) ключі пов’язані математично рівністю e∙d mod φ(n) = 1, lt φ(n)- значення функції Ейлера, отримане з обрахованих вхідних даних.

# Список літератури

1. Тарнавський Ю.А. Технології захисту інформації [Електронний ресурс] / Ю. А. Тарнавський. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 162 с.
2. Шнайер Б. Прикладная криптография: Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М.: Диалектика, 2003. – 610 с.
3. Гулак Г.М. Основи криптографічного захисту інформації: підручник / Г.М. Гулак, В.А. Мухачов, В.О. Хорошко, Ю.Є. Яремчук / – Вінниця: ВНТУ, 2011. – 199 с.
4. Столлингс В. Криптография и защита сетей: принципы и практика, 2-е изд.: Пер. с англ. – М.: «Вильямс», 2001. – 672 с.
5. Саймон Сингх. Книга шифров. Пер. с англ. А. Галыгина. — М.: АСТ: Астрель, 2007. — 448 с.
6. Асиметричне шифрування:

<https://uk.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D1%96_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8_%D1%88%D0%B8%D1%84%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

1. RSA:

https://uk.wikipedia.org/wiki/RSA

# Додаток 1

**Текст програми, що реалізує шифрування/розшифрування  
 за алгоритмом RSA**

import random

def inter\_plains(a : int, b : int):

    """a, b - ints;

    function defines, if a and b

    are mutually plained to each other,

    i. e. their GCD is 1."""

    if (a / b == int(a / b)) or (b / a == int(b / a)):

            return False

    return True

def is\_plain(a : int):

    """a - int number;

    function defines, if a is plain number"""

    if a == 1:

        return False

    divisors\_numb = 0

    for i in range(1, a):

        if a / i == int(a / i):

            divisors\_numb += 1

        if divisors\_numb > 1:

            return False

    return True

def find\_d(e\_const : int, phi : int, n : int):

    """e\_const, phi - constants, passed to Ferma's formula;

    phi - phi(n), passed to Ferma's formula: (e\_const \* d) % phi = 1

    n - number of d's to find.

    function returns number d, that satisfies Ferma's formula."""

    d\_list = []

    i = 0

    while len(d\_list) < n:

        i += 1

        if i == e\_const:

            continue

        if (e\_const \* i) % phi == 1:

            d\_list.append(i)

    log\_message(f"List of possible d-values:{d\_list}")

    random.seed(n % 32)

    d\_index = random.randint(0, len(d\_list) - 1)

    return d\_list[d\_index]

def chars\_to\_ints(text : str):

    """test - string;

    function returns list of ascii char's numbers"""

    if text == "":

        return []

    return [ord(text[0])] + chars\_to\_ints(text[1:])

def ints\_to\_chars(coded\_chars : int):

    """coded\_chars - list of ints, that represent ascii encoded text.

    function returns decode string."""

    if coded\_chars == []:

        return ''

    return chr(coded\_chars[0]) + ints\_to\_chars(coded\_chars[1:])

def generate\_async\_keys(p : int, q : int, e\_const : int, d\_list\_len):

    """p, q, e\_const - ints; d\_list\_len - int-number of generated d-keys.

    function returns list of two tuples with mathematically

    dependant keys [(key\_1, n), (key\_2, n)]"""

    n = p \* q

    log\_message(f"n = {p} \* {q} = {n}")

    if p == q or n <= 90 or not (is\_plain(p) and is\_plain(q)):

        return []

    phi\_of\_n = (p - 1) \* (q - 1)

    log\_message(f"phi(n) = ({p} - 1) \* ({q} - 1) = {phi\_of\_n}")

    if e\_const >= phi\_of\_n or not inter\_plains(phi\_of\_n, e\_const):

        return []

    d\_const = find\_d(e\_const, phi\_of\_n, d\_list\_len)

    return [(e\_const, n), (d\_const, n)]

def cipher(text, key):

    if key == []:

        return []

    log\_message(f"Given key: {key}")

    ciphered\_text = text[:]

    for i in range(len(ciphered\_text)):

        log\_message(f"Ciphered letter: ({ciphered\_text[i]}^{key[0]}) mod {key[1]} = {(ciphered\_text[i] \*\* key[0]) % key[1]}")

        ciphered\_text[i] = (ciphered\_text[i] \*\* key[0]) % key[1]

    return ciphered\_text

def log\_message(text):

    """text - string to be printed in terminal."""

    print(text)

if \_\_name\_\_ == "\_\_main\_\_":

    e\_c = 17

    p\_c = 41

    q\_c = 3

    # d = 33

    # n = 123

    t = [19, 35, 36, 12, 95, 79, 82, 71]

    keys = generate\_async\_keys(p\_c, q\_c, e\_c, 1)

    print(f"Given encoded text in ascii numbers: {t}")

    decoded = cipher(t, (33, 123))

    print(f"Decoded in ascii numbers: {decoded}")

    print(f"Decoded text: {ints\_to\_chars(decoded)}")

# Додаток 2

**Скріншоти виконання кроків 1-9 Таблиці 1.**

