НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

з лабораторної роботи № 5

із дисципліни «Криптографічні методи захисту інформації»

на тему

*Криптографічні алгоритми Діффі-Хеллмана та Ель-Гамаля*

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-01 | *ст. викладач Бай Ю. П.* |
| *Дюбакін Р. С.* |  |

Київ — 2022

ЗМІСТ

[Постановка завдань 2](#_Toc103760756)

[Математичне підґрунтя і опис алгоритму Діффі-Хеллмана 3](#_Toc103760757)

[Математичне підґрунтя і опис алгоритму Ель-Гамаля 3](#_Toc103760758)

[Контрольний приклад до алгоритма Діффі-Хеллмана 3](#_Toc103760759)

[Завдання 1. Обмін ключами за алгоритмом Діффі-Хеллмана 4](#_Toc103760760)

[Контрольні приклади до алгоритма Ель-Гамаля 4](#_Toc103760761)

[Завдання 2. Шифрування і розшифрування за алгоритмом Ель-Гамаля 5](#_Toc103760762)

[Список літератури 7](#_Toc103760763)

[Додаток 1 8](#_Toc103760764)

***Мета роботи:*** розробити асиметричні криптосистеми на основі алгоритмів Діффі-Хеллмана та Ель-Гамаля.

# Постановка завдань

1. Скласти програму, яка дозволяє здійснити обмін ключами за алгоритмом Діффі-Хеллмана. Перевірити роботу програми на контрольному прикладі. В якості відправника здійснити обмін ключами з одержувачем, згенерувати спільний ключ. Навести скріншоти детального виконання алгоритму для контрольного прикладу та власного завдання.

1.*а*. Контрольний приклад ([Протокол Діффі-Хеллмана](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D1%96%D1%84%D1%84%D1%96_%E2%80%94_%D0%93%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0))

*g* = 5, *p* = 23, *a* = 6, *b* = 15,

*public key* {*g, p*}= {5*,* 23}

*Alice’s private key* {*a*}= {6}

*Bob’s private key* {*b*}= {15}

***K* = 2**

1.*б*. Виконати дії відправника та одержувачА, згенерувати спільний ключ *К,* заповнити *Таблицю 1*. Результати також записати в гугл-таблицю [Завдання ЛР 5](https://docs.google.com/spreadsheets/d/162sc672AWdm0WRwFi2c_4W_nY0QfqQcwDUWSAkf0Vu8/edit?usp=sharing).

2. Скласти програму, яка дозволяє виконувати шифрування та розшифрування за алгоритмом Ель-Гамаля. Перевірити роботу програми на контрольних прикладах. Навести скріншоти детального виконання алгоритму для контрольних прикладів (*а, б*) та власного завдання (*в*).

2.*а*. Контрольний приклад 1 ([Схема Ель-Гамаля](https://uk.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%95%D0%BB%D1%8C-%D0%93%D0%B0%D0%BC%D0%B0%D0%BB%D1%8F))

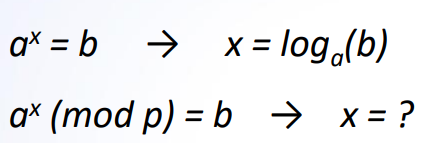
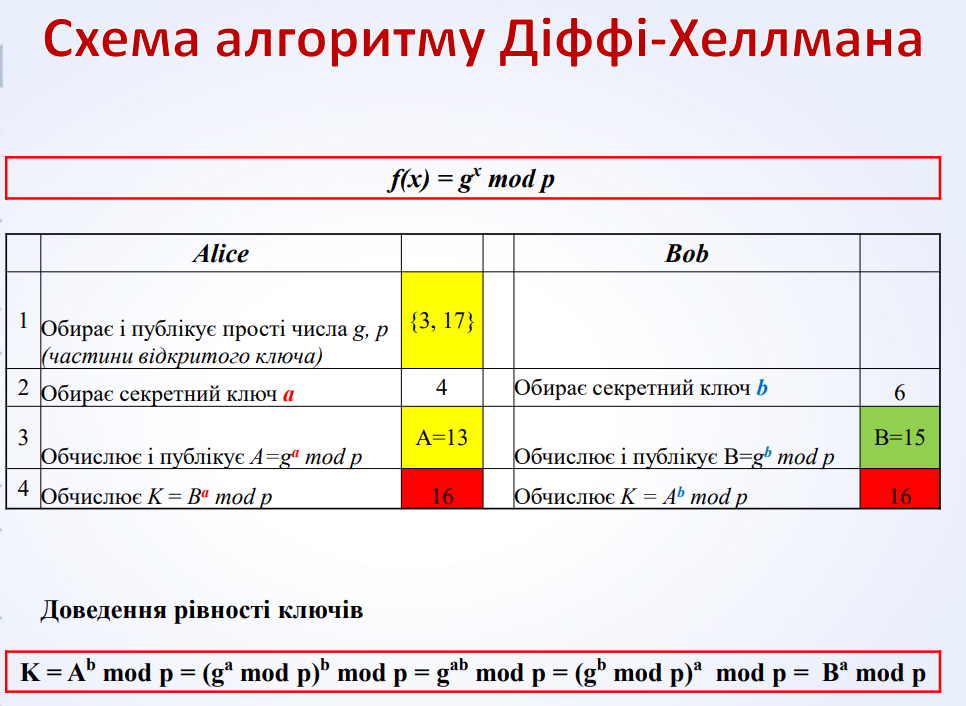
2.*б*. Контрольний приклад 2 ([ElGamal encryption](https://drive.google.com/file/d/11ul0MvFWCs2ZNP_qk2fL1_P-bvHLDgtH/view?usp=sharing))

2.*в*. Виконати дії відправникА: використовуючи заданий відкритий ключ, зашифрувати День свого народження, записаний у форматі *“ddmm”,* або інше число в діапазоні від 101 до 3112. Результати шифрування **для двох різних випадкових чисел *k*1 i *k*2** записати в [Завдання ЛР 5](https://docs.google.com/spreadsheets/d/162sc672AWdm0WRwFi2c_4W_nY0QfqQcwDUWSAkf0Vu8/edit?usp=sharing). Розшифрувати одержані криптотексти. Заповнити *Таблиці 2, 3*.

*Увага! Як визначити М ?*

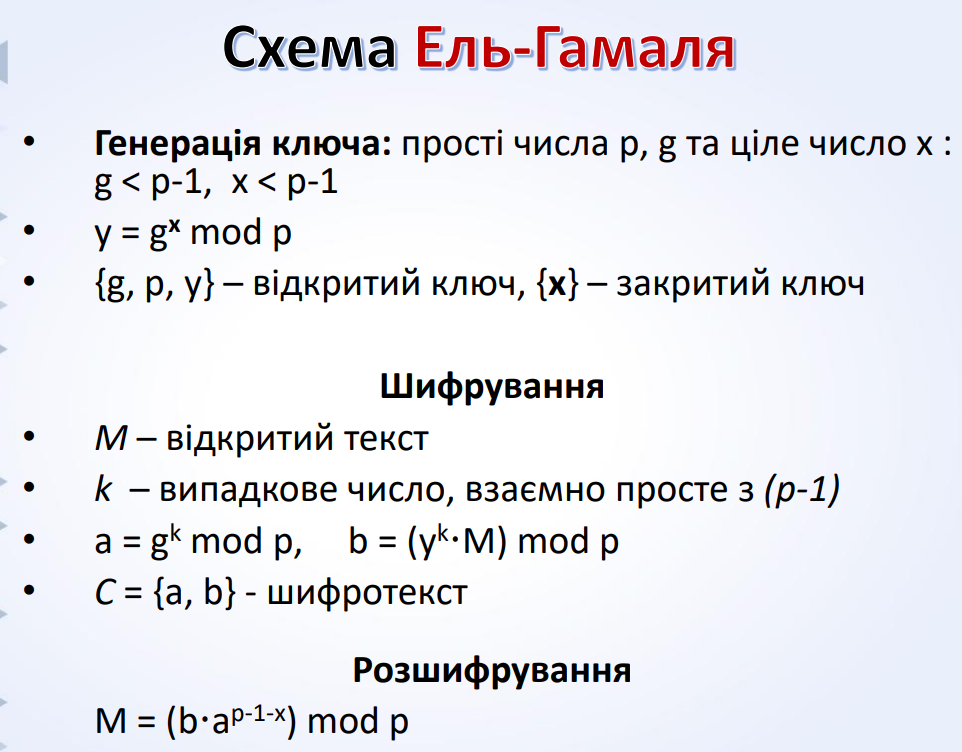
*01 січня* → *M* = 101 … 4 *липня* → *M* = 407 … *31 грудня* → *M* = 3112 .

# Математичне підґрунтя і опис алгоритму Діффі-Хеллмана

В основі алгоритму Діффі-Хеллмана полягає складність задачі дискретного логарифмування.

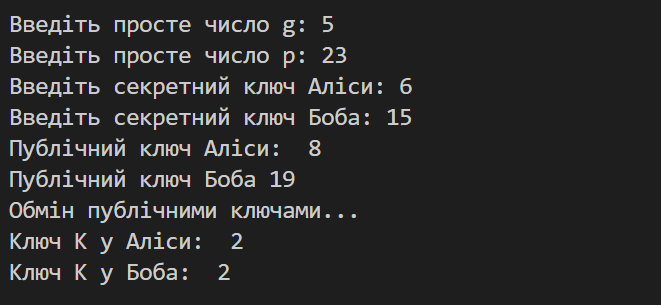
# Математичне підґрунтя і опис алгоритму Ель-Гамаля

В основі алгоритму Ель-Гамаля полягає складність задачі дискретного логарифмування.



# Контрольний приклад до алгоритма Діффі-Хеллмана

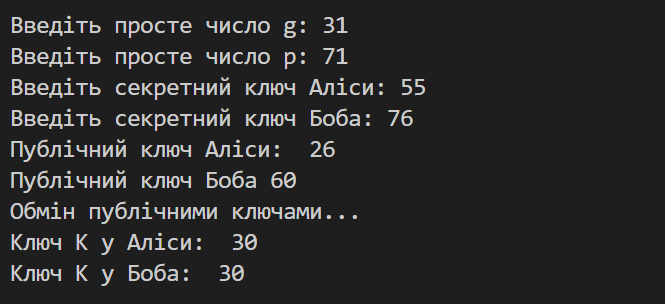
[Протокол Діффі-Хеллмана](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D1%96%D1%84%D1%84%D1%96_%E2%80%94_%D0%93%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0)

Виконати приклад. Додати скріншот, що містить усі проміжні результати.

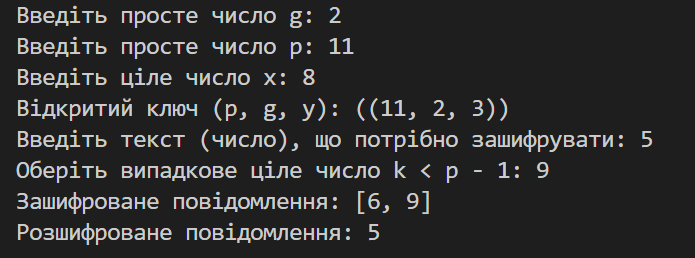
# Завдання 1. Обмін ключами за алгоритмом Діффі-Хеллмана

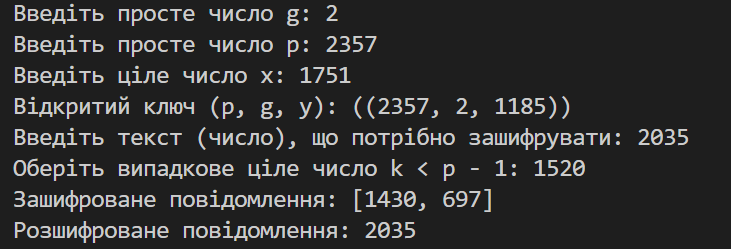
*Таблиця 1.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***f(x) = gx mod p*** | | | | | |
|  |  |  |  |  |  |
|  | ***Alice (Відправник)*** |  |  | ***Bob (Одержувач)*** |  |
| 1 | Обирає і публікує прості числа *g, p (частини відкритого ключа)* | {31,71} |  |  |  |
| 2 | Обирає секретний ключ ***a*** | **55** |  | Обирає секретний ключ ***b*** | **76** |
| 3 | Обчислюєі публікує *A=g****a*** *mod p* | A=26 |  | Обчислюєі публікує *B = g****b*** *mod p* | B=60 |
| 4 | Обчислює *K=B****a*** *mod p* | **30** |  | Обчислює *K=A****b*** *mod p* | **30** |



# Контрольні приклади до алгоритма Ель-Гамаля

2.*а*.

2.*б*.

# Завдання 2. Шифрування і розшифрування за алгоритмом Ель-Гамаля

*Таблиця 2 (Шифрування 1).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Alice*** (ВІДПРАВНИК) |  |  | ***Bob*** (ОДЕРЖУВАЧ) |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 3 *p* = 4211 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **?** |
|  |  |  | 3 | Обчислює *y=g****x*** *mod p* | *y* = 2395 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | 4211, 3, 2395 |
| 5 | Одержує відкритий ключ{*p, g, y*} | 4211, 3, 2395 |  |  |  |
| 6 | Обирає текст для шифрування *M* | 702 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p - 2* | 3208 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | *a* = 228 *b* = 1483 |  |  |  |
| 9 | Надсилає одержувачу шифротекст *(a; b).* | 228, 1483 |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифротекст:  M′ = (*a****^***(*p-1-x*) ∙*b*) mod *p* | M′ = 702 |

*Таблиця 3 (Шифрування 2).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Alice*** (ВІДПРАВНИК) |  |  | ***Bob*** (ОДЕРЖУВАЧ) |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 3 *p* = 4211 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **?** |
|  |  |  | 3 | Обчислює *y=g****x*** *mod p* | *y* = 2395 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | 4211, 3, 2395 |
| 5 | Одержує відкритий ключ{*p, g, y*} | 4211, 3, 2395 |  |  |  |
| 6 | Обирає текст для шифрування *M* | 702 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p - 2* | 1057 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | *a* = 3902 *b* = 927 |  |  |  |
| 9 | Надсилає одержувачу шифротекст *(a; b).* | 3902, 927 |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифротекст:  M′ = (*a****^***(*p-1-x*) ∙*b*) mod *p* | M′ = 702 |

# 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. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. – М.: Гелиос АРВ, 2001. – 480 с.
4. Столлингс В. Криптография и защита сетей: принципы и практика, 2-е изд.: Пер. с англ. – М.: «Вильямс», 2001. – 672 с.
5. Menezes A.J., Van Oorschot P.C., Vanstone S.A. Handbook of Applied Cryptography. – CRC Press, Inc., 1997. – 795 p.

# Додаток 1

**Скріншоти виконання обчислень**

def isPrimeNumber(number):

    for num in range(2, number):

        if number % num != 0:

            continue

        else:

            return False

    return True

def publicKeyDiffiHellman(g, p, private\_key):

    public\_key = (g \*\* private\_key) % p

    return public\_key

def commonKeyDiffiHellman(p, private\_key, public\_key):

    common\_key = (public\_key \*\* private\_key) % p

    return common\_key

def main():

    try:

        g = int(input("Введіть просте число g: "))

        p = int(input("Введіть просте число p: "))

        a = int(input("Введіть секретний ключ Аліси: "))

        b = int(input("Введіть секретний ключ Боба: "))

    except:

        print("Значення не є числом!")

    if (not isPrimeNumber(g) or not isPrimeNumber(p)):

        print("Числа не прості!")

        exit()

    A = publicKeyDiffiHellman(g, p, a)

    print("Публічний ключ Аліси: ", A)

    B = publicKeyDiffiHellman(g, p, b)

    print("Публічний ключ Боба", B)

    print("Обмін публічними ключами...")

    Ka = commonKeyDiffiHellman(p, a, B)

    print("Ключ К у Аліси: ", Ka)

    Kb = commonKeyDiffiHellman(p, b, A)

    print("Ключ К у Боба: ", Kb)

main()

def isPrimeNumber(number):

    for num in range(2, number):

        if number % num != 0:

            continue

        else:

            return False

    return True

def generate\_key(p, g, x):

    if (g < p-1 and x < p-1):

        y = (g \*\* x) % p

        return y

    else:

        print("Введіть правильні значення!")

        return None

def encrypt(plain\_text, g, p, y, k):

    a = (g \*\* k) % p

    b = ((y \*\* k) \* plain\_text) % p

    return [a, b]

def decrypt(cypher\_text, p, x):

    return (cypher\_text[1] \* (cypher\_text[0] \*\* (p - 1 - x))) % p

def main():

    try:

        g = int(input("Введіть просте число g: "))

        p = int(input("Введіть просте число p: "))

        if (not isPrimeNumber(g) or not isPrimeNumber(p)):

            print("Числа не прості!")

            exit()

        x = int(input("Введіть ціле число х: "))

        # k = int(input("Введіть рандомне взаємно просте число з р-1: "))

    except:

        print("Значення не є числом!")

    y = generate\_key(p, g, x)

    print(f"Відкритий ключ (p, g, y): ({p, g, y})")

    plain\_text = int(input("Введіть текст (число), що потрібно зашифрувати: "))

    k = int(input("Оберіть випадкове ціле число k < p - 1: "))

    cypher\_text = encrypt(plain\_text, g, p, y, k)

    print(f"Зашифроване повідомлення: {cypher\_text}")

    print(f"Розшифроване повідомлення: {decrypt(cypher\_text, p, x)}")

main()

print(encrypt(702, 3, 4211, 2395, 3208))

print(encrypt(702, 3, 4211, 2395, 1057))