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*

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

Київ — 2022

ЗМІСТ

[Постановка завдань 2](#_Toc104234981)

[Математичне підґрунтя і опис алгоритму Діффі-Хеллмана 3](#_Toc104234982)

[Математичне підґрунтя і опис алгоритму Ель-Гамаля 4](#_Toc104234983)

[Контрольний приклад до алгоритма Діффі-Хеллмана 5](#_Toc104234984)

[Завдання 1. Обмін ключами за алгоритмом Діффі-Хеллмана 5](#_Toc104234985)

[Контрольні приклади до алгоритма Ель-Гамаля 6](#_Toc104234986)

[Завдання 2. Шифрування і розшифрування за алгоритмом Ель-Гамаля 8](#_Toc104234987)

[Список літератури 9](#_Toc104234988)

[Додаток 1 10](#_Toc104234989)

[Додаток 2 12](#_Toc104234990)

***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

**Протокол Діффі-Геллмана** (англ. *Diffie–Hellman key exchange (D–H) -* це метод обміну криптографічними ключами. Цей алгоритм почасти був натхненний попередніми розробками, зробленими Ральфом Меркле. Застосування протоколу - один з перших практичних прикладів узгодження ключа, що дозволяє двом учасникам, що не мають жодних попередніх даних один про одного, отримати спільний секретний ключ із використанням незахищеного каналу зв'язку.

Основу даного алгоритму складає математична задача дискретногго логарифмування.

ах = b => x = loga(b)

ах (mod p) = b => x = ?

Маємо односторонню функцію: F(x) = gх (mod p)

Схема алгоритму Діффі-Хеллмана

Одна сторона (Аліса) публікує відкриті індентифікатори: (p, g) та обирає секретний ключ (а. .Інша сторона (Боб) отримує відкриті ідентифікатори та обирає свій секретний ключ (b). Далі сторони (Аліса та Боб в даному випадку) на основі ідентифікаторів та власних таємних ключів обраховують компоненти А = ga (mod p) та B = gb (mod p) та обмінюються нимим з одне-одним. Спільний ключ К, сторони генерують наступним чином. Аліса: K = Ba (mod p), Боб: K = Ab (mod p).

Доказ того, що згенеровані К в обох сторін однакові

**K = Ab mod p = (ga mod p)b mod p = gab mod p = (gb mod p)a mod p = Ba mod p**

# Математичне підґрунтя і опис алгоритму Ель-Гамаля

Схема Ель-Гамаля – розроблена Ель-Гамалем, американським криптографом, родом з Єгипту, криптосистема з відкритим ключем, яка заснована на складності обчислення дискретних логарифмів. Ель-Гамаль розробив один з варіантів алгоритму Діффі-Геллмана. Він удосконалив систему Діффі-Геллмана й отримав два алгоритми, які призначено для шифрування та для автентифікації.

Для того, щоб зашифрувати деяке число *M*, наприклад 8-байтний ключ, одержувач і відправник виконують наступні дії:

ОДЕРЖУВАЧ:

1. Генерує велике випадкове просте число *p* та ціле число *g* (*g* – генератор, основа степеня, маленьке число) .

2. Обирає закритий ключ – випадкове ціле число *x*: 1 < *x* < *p −* 1 .

3. Обчислює *y = gx* mod *p* .

4. Публікує відкритий ключ (*p; g; y*).

ВІДПРАВНИК:

5. Отримує відкритий ключ (*p; g; y*).

6. Обирає текст для шифрування *M (М – число)*: 1 < *M <* *p −* 1 .

7. Обирає випадкове ціле число *k*: *k –* взаємно простез *p −* 1, *k* < *p −* 1 .

8. Обчислює *a = gk* mod *p* та *b = (yk* *∙ M*) mod *p* .

9. Надсилає одержувачу шифротекст (*a; b*).

ОДЕРЖУВАЧ:

10. Використовуючи свій секретний ключ *x*, розшифровує отриманий шифротекст, користуючись формулою:

*M* ′ *=* (*a* ***p−1− x*** ∙ *b*) mod *p*

***Зауваження:*** число *k* – випадкова величина для кожної сесії, завдяки чому алгоритм Ель-Гамаля отримав назву ***шифру багатозначної заміни***, або ***схеми ймовірнісного шифрування***. При використанні різних значень *k* одержуються різні шифротексти для одного й того самого відкритого тексту *М*, що в цілому підвищує криптостійкість алгоритму.

# Контрольний приклад до алгоритма Діффі-Хеллмана

Приклад з ресурсу: [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)

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

Alice’s private key {a} = {6}

Bob’s private key {b} = {15}

А = g^a mod p = 5^6 mod 23 = 8

B = g^b mod p = 5^15 mod 23 = 19

K (обраховане Алісою) = B^a mod p = 19^6 mod 23 = 2

K (обраховане Бобом) = A^b mod p = 8^15 mod 23 = 2

Отже, К = 2

# 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)* | {7,29} |  |  |  |
| 2 | Обирає секретний ключ ***a*** | **10** |  | Обирає секретний ключ ***b*** | **4** |
| 3 | Обчислюєі публікує *A=g****a*** *mod p* | A=24 |  | Обчислюєі публікує *B = g****b*** *mod p* | B=23 |
| 4 | Обчислює *K=B****a*** *mod p* | **16** |  | Обчислює *K=A****b*** *mod p* | **16** |

# Контрольні приклади до алгоритма Ель-Гамаля

2.*а*. Приклад з ресурсу: [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)

*Таблиця 2 (Контрольний приклад 1).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Alice (ВІДПРАВНИК)*** |  |  | ***Bob (ОДЕРЖУВАЧ)*** |  |
|  |  |  | **1** | ***Обирає прості числа g, p (g - генератор, p - модуль)*** | g = 2 p = 11 |
|  |  |  | **2** | ***Обирає секретний ключ x*** | 8 |
|  |  |  | **3** | ***Обчислює y=gx mod p*** | y = 3 |
|  |  |  | **4** | ***Публікує відкритий ключ {p, g, y}*** | {11, 2, 3} |
| 5 | ***Одержує відкритий ключ {p, g, y}*** | {11, 2, 3} |  |  |  |
| 6 | ***Обирає текст для шифрування M*** | **5** |  |  |  |
| 7 | ***Обирає випадкове ціле число k: k < p - 2*** | **9** |  |  |  |
| 8 | ***Обчислює a = g^k mod p та b = (y^k ∙ M) mod p*** | **a = 6 b = 9** |  |  |  |
| 9 | ***Надсилає одержувачу шифротекст (a; b).*** | **(6, 9)** |  |  |  |
|  |  |  | **10** | ***Використовуючи секретний ключ x, розшифровує отриманий шифротекст:  M′ = (a^(p-1-x) ∙b) mod p*** | M′ = 5 |

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

*Таблиця 3 (Контрольний приклад 2).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ВІДПРАВНИК |  |  | ОДЕРЖУВАЧ |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 2 *p* = 2357 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **1751** |
|  |  |  | 3 | Обчислює *y=g****x*** *mod p* | *y* = 1185 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | 2357, 2, 1185 |
| 5 | Одержує відкритий ключ{*p, g, y*} | 2357, 2, 1185 |  |  |  |
| 6 | Обирає текст для шифрування *M* | 2035 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p - 2* | 1520 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | *a* = 1430 *b* = 697 |  |  |  |
| 9 | Надсилає одержувачу шифротекст *(a; b).* | 1430, 697 |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифротекст:  M′ = (*a****^***(*p-1-x*) ∙*b*) mod *p* | M′ = 2035 |

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

*Таблиця 4 (Шифрування 1).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ВІДПРАВНИК |  |  | ОДЕРЖУВАЧ |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 2 *p* = 1423 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **839** |
|  |  |  | 3 | Обчислює *y=g****x*** *mod p* | *y* = 1423 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | {4153, 2, 1423} |
| 5 | Одержує відкритий ключ{*p, g, y*} | {4153, 2, 1423} |  |  |  |
| 6 | Обирає текст для шифрування *M* | 1009 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p - 2* | 11 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | a = 2048, b = 405 |  |  |  |
| 9 | Надсилає одержувачу шифротекст *(a; b).* | (2048, 405) |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифротекст:  M′ = (*a****^***(*p-1-x*) ∙*b*) mod *p* | M′ = 1009 |

*Таблиця 5 (Шифрування 2).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Alice*** (ВІДПРАВНИК) |  |  | ***Bob*** (ОДЕРЖУВАЧ) |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 2 *p* = 1423 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **839** |
|  |  |  | 3 | Обчислює *y=g****x*** *mod p* | *y* = 1423 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | {4153, 2, 1423} |
| 5 | Одержує відкритий ключ{*p, g, y*} | {4153, 2, 1423} |  |  |  |
| 6 | Обирає текст для шифрування *M* | 1009 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p - 2* | 5 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | a = 32, b = 2630 |  |  |  |
| 9 | Надсилає одержувачу шифротекст *(a; b).* | (32, 2630) |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифротекст:  M′ = (*a****^***(*p-1-x*) ∙*b*) mod *p* | M′ = 1009 |

# 

# 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.
6. [Діффі\_Хеллман\_Вікіпедія](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#:~:text=Diffie%E2%80%93Hellman%20key%20exchange%20(D,%D0%B2%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D0%BD%D1%8F%D0%BC%20%D0%BD%D0%B5%D0%B7%D0%B0%D1%85%D0%B8%D1%89%D0%B5%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB%D1%83%20%D0%B7%).
7. [Презентація\_з\_лекції\_про\_шифри\_діффі\_геллмана\_та\_ельгамаля](https://slack-files.com/files-pri-safe/T030QG8TVQB-F03FXBC2SRG/_____________________________________________________-____________________________-____________.pdf?c=1653329557-c579f4746719f601)
8. [Ель\_Гамаль\_Вікіпедія](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)

# Додаток 1

**Текст програм**

from math import gcd

from random import randint

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 log\_message(text):

    print(text)

def diffie\_hellman\_key\_generate(g, p, a, b):

    """g, p - int, known to both communicators,

    a - int, known to one communicator,

    b - int, known to another communicator;

    the function finds mutaual secret key for communicators"""

    log\_message(f"Chosen public keys: g - {g}, p - {p}")

    log\_message(f"Chosen secret number of A-communicator: {a}")

    log\_message(f"Chosen secret number of B-communicator: {b}")

    a\_n = pow(g, a) % p

    b\_n = pow(g, b) % p

    log\_message(f"A = g^a mod p = {g}^{a} mod {p} = {a\_n}")

    log\_message(f"B = g^b mod p = {g}^{b} mod {p} = {b\_n}")

    k\_a = pow(b\_n, a) % p

    k\_b = pow(a\_n, b) % p

    log\_message(f"K calculated by A communicator: K = B^a mod p = {b\_n}^{a} mod {p} = {k\_a}")

    log\_message(f"K calculated by B communicator: K = A^b mod p = {a\_n}^{b} mod {p} = {k\_b}")

    print(f"K(by A communicator) == K(by B communicator): {k\_a == k\_b} ")

    return k\_a

def el\_gamal\_generate\_keys(p\_lower\_bound, n, g\_upper\_bound = 10):

    """p\_lower\_bound, g\_upper\_bound, n - int;

    the function returns [private\_key, secret\_key],

    where public key is tuple with three elements,

    private key is a number"""

    # Choose p

    p\_list = []

    i = p\_lower\_bound

    while i < n:

        if is\_plain(i):

            p\_list.append(i)

        i += 1

    ind\_p = randint(0, n - 1)

    p = p\_list[ind\_p]

    log\_message(f"Chosen index of p\_list: {ind\_p}, chosen p: {p}")

    # Choose g

    g\_list = []

    i = 1

    if g\_upper\_bound > p - 1:

        g\_upper\_bound = p - 1

        if g\_upper\_bound < 1:

            return []

    while i < g\_upper\_bound:

        if is\_plain(i):

            g\_list.append(i)

        i += 1

    ind\_g = randint(0, len(g\_list) - 1)

    g = g\_list[ind\_g]

    log\_message(f"Chosen index of g\_list: {ind\_g}, chosen p: {g}")

    # Choose x, let upper bound for possible x set will be g's upper bound

    x\_list = list(range(1, g\_upper\_bound))

    ind\_x = randint(0, len(x\_list) - 1)

    x = x\_list[ind\_x]

    log\_message(f"Chosen index of x\_list: {ind\_x}, chosen x: {x}")

    # Calculate y

    y = pow(g, x) % p

    return [(g, p, y), x]

def elgamal\_cipher(secret\_info, public\_key, n = 5):

    """secret\_info, n - int; public\_key - tuple with ints - (g, p, y);

    the function cipherss secret\_info and returns pair [a, b]"""

    p = public\_key[0]

    g = public\_key[1]

    y = public\_key[2]

    log\_message(f"Ciphering \nM (secret\_info): {secret\_info},  p: {p}, g: {g}, y: {y}")

    k\_list = []

    i = 1

    while len(k\_list) < n:

        if gcd(i, p - 1) == 1:

            k\_list.append(i)

        i += 1

    ind\_rand = randint(0, n - 1)

    k = k\_list[ind\_rand]

    # k = 9

    # log\_message(f"Chosen k: {k}")

    log\_message(f"Chosen index of k\_list: {ind\_rand}, chosen k: {k}")

    a = pow(g, k) % p

    log\_message(f"a = g^k mod p = {g}^{k} mod {p} = {a}")

    b = (pow(y, k) \* secret\_info) % p

    log\_message(f"b = (y^k \* M) mod p = ({y}^{k} \* {secret\_info}) mod {p} = {b}")

    return [a, b]

def elgamal\_encipher(ciphered\_pair, x, p):

    """keypair - [int, int]"""

    a = ciphered\_pair[0]

    b = ciphered\_pair[1]

    log\_message(f"given ciphered pair [a, b]: [{a}, {b}]; secret key: {x}, p: {p}")

    result = (b \* pow(a, p - 1 - x)) % p

    log\_message(f"(b \* a^(p - 1 - x)) mod p = ({b} \* {a}^({p} - 1 - {x})) mod p = {result}")

    return result

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

    # example 1 of diffie-hellman key exchange algorythm

    # p\_val = 23

    # g\_val = 5

    # a\_val = 6

    # b\_val = 15

    # key = diffie\_hellman\_key\_generate(g\_val, p\_val, a\_val, b\_val)

    # log\_message(f"Generated mutaual key: {key}")

    # example 1.b to elgamal cipher

    m\_val = 1009

    p\_val = 4153

    g\_val = 2

    y\_val = 1423

    log\_message(f"Let's cipher info: {m\_val}")

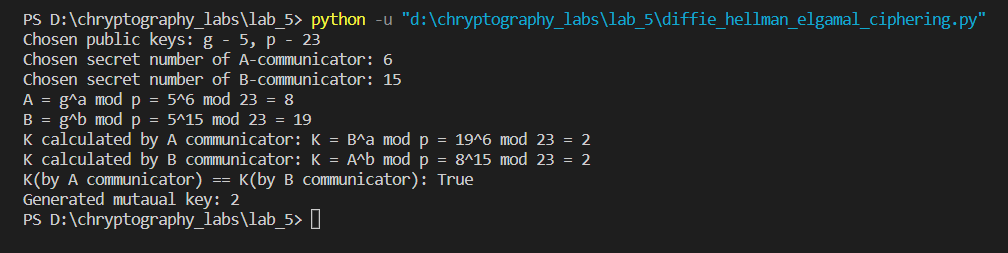
    ciphered = elgamal\_cipher(m\_val, (p\_val, g\_val, y\_val))

    log\_message(f"Ciphered info: {ciphered}")

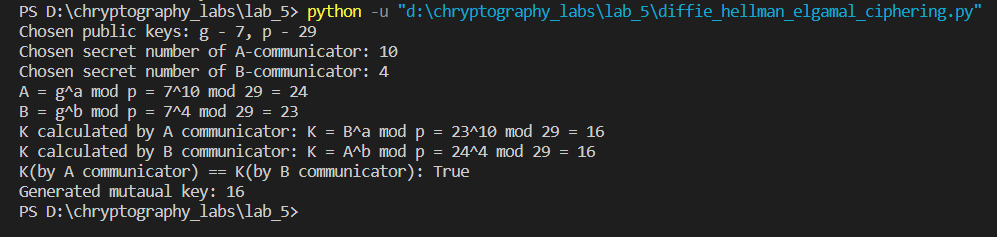
# Додаток 2

**Скріншоти виконання обчислень**

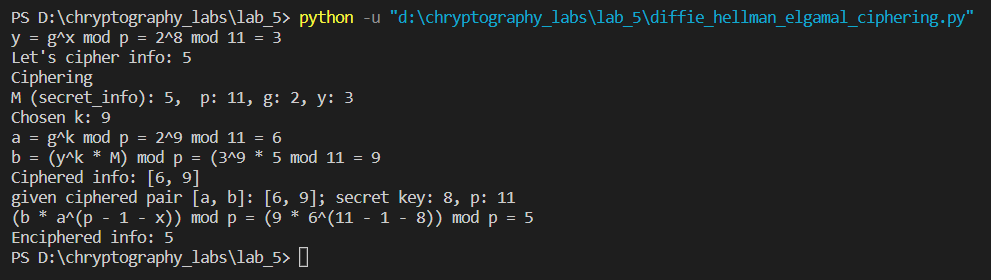
Завдання 1. Контрольний приклад



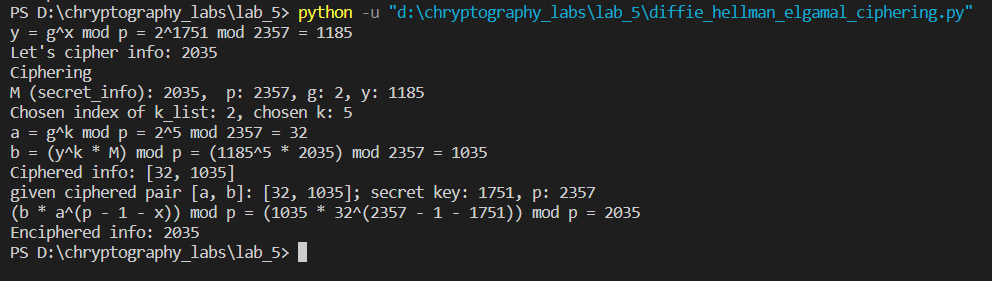
Завдання 1. Обмін ключами



Завдання 2.Контрольний приклад 1 – шифрування за алгоритмом Ель-Гамаля



Завдання 2. Контрольний приклад 2­­ – шифрування за алгоритмом Ель-Гамаля



Завдання 2 – шифрування ДДММ за алгоритмом Ель-Гамаля

