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-ХХ | *ст. викладач Бай Ю. П.* |
| *Верзун П.В.* |  |

Київ — 2020

ЗМІСТ

[Постановка завдань 2](#_Toc58071954)

[Математичне підґрунтя і опис алгоритму Діффі-Хеллмана 3](#_Toc58071955)

[Математичне підґрунтя і опис алгоритму Ель-Гамаля 4](#_Toc58071956)

[Контрольний приклад до алгоритма Діффі-Хеллмана 5](#_Toc58071957)

[Завдання 1.б. Обмін ключами за алгоритмом Діффі-Хеллмана 8](#_Toc58071958)

[Контрольні приклади до алгоритма Ель-Гамаля 9](#_Toc58071959)

[Завдання 2.в. Шифрування і розшифрування за алгоритмом Ель-Гамаля 10](#_Toc58071960)

[Список літератури 11](#_Toc58071961)

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

# Математичне підґрунтя і опис алгоритму Діффі-Хеллмана

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

**Приклад**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***f(x) = gx mod p*** | | | | | |
|  |  |  |  |  |  |
|  | ***Alice*** |  |  | ***Bob*** |  |
| 1 | Обирає і публікує прості числа *g, p (частини відкритого ключа)* | {3, 17} |  |  |  |
| 2 | Обирає секретний ключ ***a*** | 4 |  | Обирає секретний ключ ***b*** | 6 |
| 3 | Обчислює і публікує *A=g****a*** *mod p* | A= |  | Обчислює і публікує B=*g****b*** *mod p* | B= |
| 4 | Обчислює *K* = *B****a*** *mod p* | K= |  | Обчислює *K = A****b*** *mod p* | K= |
|  |  |  |  |  |  |
| **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

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

**Приклад**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Alice*** (ВІДПРАВНИК) |  |  | ***Bob*** (ОДЕРЖУВАЧ) |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 3 *p* = 17 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **4** |
|  |  |  | 3 | Обчислює *y=g****x****mod p* | *y* = 13 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | 17, 3, 13 |
| 5 | Одержує відкритий ключ{*p, g, y*} | 17, 3, 13 |  |  |  |
| 6 | Обирає текст для шифрування *M* (наприклад, *М* = 12) | *M* = 12 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p - 2* | 7<17-2 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p*  та *b = (y^k ∙ M)* mod *p* | *a* = 11 *b* = 14 |  |  |  |
| 9 | Надсилає одержувачу шифро-текст *(a; b).* | 11, 14 |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифро-текст: M′ = (*a****^***(*p1x*) ∙*b*) mod *p* | M′ = 12 |

# Контрольний приклад до алгоритма Діффі-Хеллмана

[Протокол Діффі-Хеллмана](https://uk.wikipedia.org/wiki/Протокол_Діффі_—_Геллмана)

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

Єва — криптоаналітик, прослуховувач. Вона читає листування Боба і Аліси, але не може змінити вмісту повідомлень.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Аліса** | | | знає | не знає | | p = 23 | b = ? | | g = 5 |  | | a = 6 |  | | A = 56 mod 23 = 8 |  | | B = 5b mod 23 = 19 |  | | s = 196 mod 23 = 2 |  | | s = 8b mod 23 = 2 |  | | s = 196 mod 23 = 8b mod 23 |  | | s = 2 |  | | |  |  | | --- | --- | | **Боб** | | | знає | не знає | | p = 23 | a = ? | | g = 5 |  | | b = 15 |  | | B = 515 mod 23 = 19 |  | | A = 5a mod 23 = 8 |  | | s = 815 mod 23 = 2 |  | | s = 19a mod 23 = 2 |  | | s = 815 mod 23 = 19a mod 23 |  | | s = 2 |  | | |  |  | | --- | --- | | **Єва** | | | знає | не знає | | p = 23 | a = ? | | g = 5 | b = ? | |  | s = ? | | A = 5a mod 23 = 8 |  | | B = 5b mod 23 = 19 |  | | s = 19a mod 23 |  | | s = 8b mod 23 |  | | s = 19a mod 23 = 8b mod 23 |  | |

* s = секретний ключ. s = 2
* g = відкрите просте число. g = 5
* p = відкрите просте число. p = 23
* a = секретний ключ Аліси. a = 6
* A = відкритий ключ Аліси. A = ga mod p = 8
* b = секретний ключ Боба. b = 15
* B = відкритий ключ Боба. B = gb mod p = 19

Для того, щоб Єві знайти секретний ключ s потрібно просто перебрати всі можливі випадки. 19a mod 23 = 8b mod 23

|  |  |  |  |
| --- | --- | --- | --- |
| a | 19a mod 23 | b | 8b mod 23 |
| 1 | 19 | 1 | 8 |
| 2 | 16 | 2 | 18 |
| 3 | 5 | 3 | 6 |
| 4 | 3 | 4 | 2 |
| 5 | 11 | 5 | 16 |
| 6 | 2 | 6 | 13 |
| 7 | 15 | 7 | 12 |
| 8 | 9 | 8 | 4 |
| 9 | 10 | 9 | 9 |
| 10 | 6 | 10 | 3 |
| 11 | 22 | 11 | 1 |
| 12 | 4 | 12 | 8 |
|  |  | 13 | 18 |
|  |  | 14 | 6 |
|  |  | 15 | 2 |

Знайшли декілька пар чисел. Почнемо перевіряти умови, які нам вже відомі про числа *a* та *b*.

|  |  |  |  |
| --- | --- | --- | --- |
| a | *A=5a mod 23=8* | b | *B=5b mod 23 = 19* |
| 2 | 2 | 16 | 3 |
| 6 | 8 | 4 | 4 |
| 6 | 8 | 15 | 19 |

Отже, ми знайшли цю пару: *а=6, b=15*. Тоді за формулою *s=19a mod 23=196 mod 23=2*

# Виконані завдання

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/Протокол_Діффі_—_Геллмана))

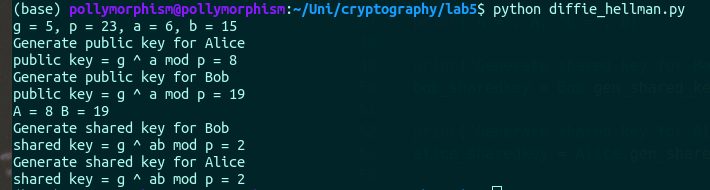
*g* = 5, *p* = 23, *a* = 6, *b* = 15,

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

*Alice’s private key* {*a*}= {6}

*Bob’s private key* {*b*}= {15} ***K* = 2**

Код програми наведено в додатку 1

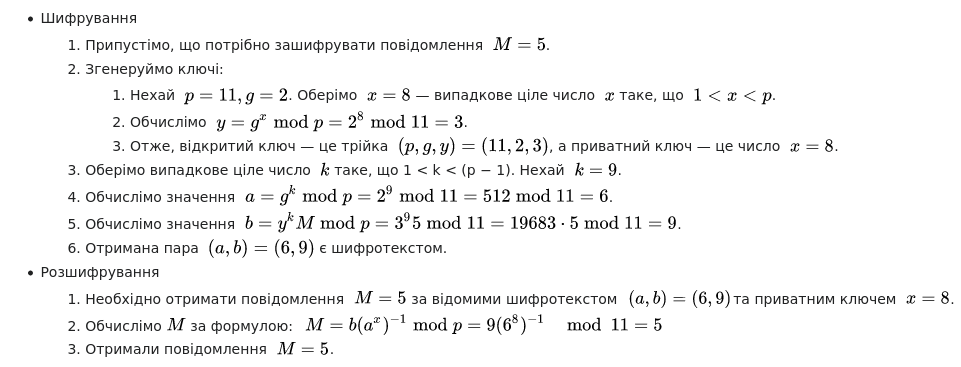


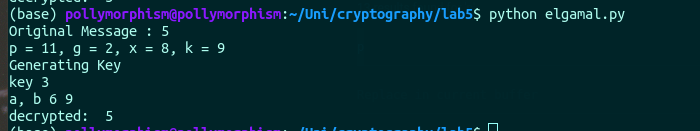
1.*б*. Виконати дії відправника (див. *Таблицю 1*) та згенерувати спільний ключ з одержувачем. Результати записати в таблицю [3 курс завдання Diffi-Hellman](https://docs.google.com/spreadsheets/d/15ADdesRVmGehA9ZxOYUmhRqBaoPYhFHSAbv-YvmoHRE/edit?usp=sharing) (див. 1-ий аркуш ).

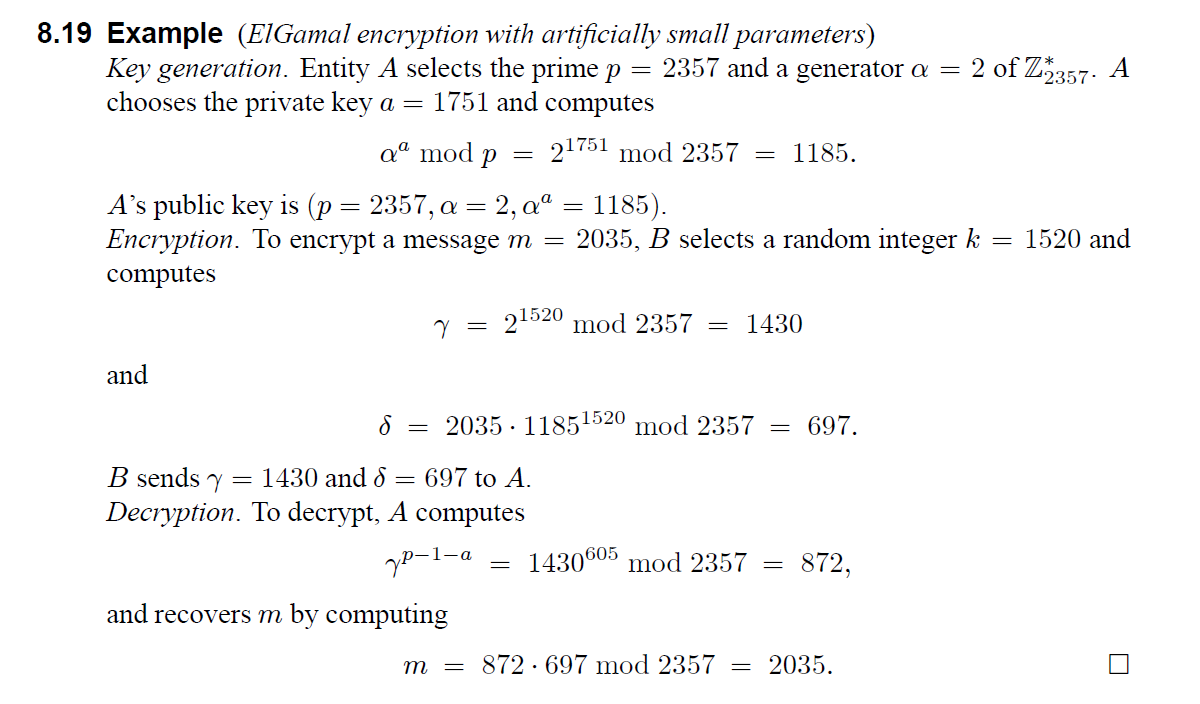
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***f(x) = gx mod p*** | | | | | |
|  |  |  |  |  |  |
|  | ***Верзун Поліна*** |  |  | ***Горбач Костянтин*** |  |
| 1 | Обирає і публікує прості числа *g, p (частини відкритого ключа)* | {11,31} |  |  |  |
| 2 | Обирає секретний ключ ***a*** | **3** |  | Обирає секретний ключ ***b*** | **5** |
| 3 | Обчислюєі публікує *A=g****a*** *mod p* | A=29 |  | Обчислюєі публікує *B = g****b*** *mod p* | B=6 |
| 4 | Обчислює *K=B****a*** *mod p* | **30** |  | Обчислює *K=A****b*** *mod p* | **30** |

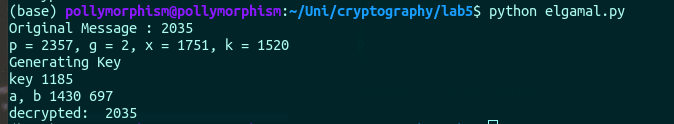
2. Скласти програму, яка дозволяє виконувати шифрування та розшифрування за алгоритмом Ель-Гамаля. Перевірити роботу програми на контрольних прикладах. Навести скріншоти детального виконання алгоритму для контрольних прикладів (*а, б*) та власного завдання (*в*).

2.*а*. Контрольний приклад 1 ([Схема Ель-Гамаля](https://uk.wikipedia.org/wiki/Схема_Ель-Гамаля))

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 [3 курс завдання Diffi-Hellman](https://docs.google.com/spreadsheets/d/15ADdesRVmGehA9ZxOYUmhRqBaoPYhFHSAbv-YvmoHRE/edit?usp=sharing) (**див.** **2-ий аркуш**). Розшифрувати одержані криптотексти. Заповнити *Таблицю 2*.

*Увага! Як визначити М ? 01 січня*  *M* = 101 … 4 *липня*  *M* = 407 … *31 грудня*  *M* = 3112 .

# Завдання 1.б. Обмін ключами за алгоритмом Діффі-Хеллмана

*Таблиця 1.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***f(x) = gx mod p*** | | | | | |
|  |  |  |  |  |  |
|  | ***Верзун Поліна*** |  |  | ***Горбач Костянтин*** |  |
| 1 | Обирає і публікує прості числа *g, p (частини відкритого ключа)* | {11,31} |  |  |  |
| 2 | Обирає секретний ключ ***a*** | **3** |  | Обирає секретний ключ ***b*** | **5** |
| 3 | Обчислюєі публікує *A=g****a*** *mod p* | A=29 |  | Обчислюєі публікує *B = g****b*** *mod p* | B=6 |
| 4 | Обчислює *K=B****a*** *mod p* | **30** |  | Обчислює *K=A****b*** *mod p* | **30** |

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

*Таблиця 2.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Alice*** (ВІДПРАВНИК) |  |  | ***Bob*** (ОДЕРЖУВАЧ) |  |
|  |  |  | 1 | Обирає прості числа *g, p (g - генератор, p - модуль)* | *g* = 3 *p* = 3581 |
|  |  |  | 2 | Обирає секретний ключ ***x*** | **1669** |
|  |  |  | 3 | Обчислює *y=g****x*** *mod p* | *y* = 3433 |
|  |  |  | 4 | Публікує відкритий ключ{*p, g, y*} | 3581, 3, 3433 |
| 5 | Одержує відкритий ключ{*p, g, y*} | 3581, 3, 3433 |  |  |  |
| 6 | Обирає текст для шифрування *M* | 1903 |  |  |  |
| 7 | Обирає випадкове ціле число *k*: *k* < *p – 1, взаємно просте з p-1* | 1721 |  |  |  |
| 8 | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | *a* = 13 *b* = 1721 |  |  |  |
| 9 | Надсилає одержувачу шифротекст *(a; b).* | 13, 1594 |  |  |  |
|  |  |  |  |  |  |
|  | Обирає випадкове ціле число *k*: *k* < *p – 1, взаємно просте з p-1* | 1777 |  |  |  |
|  | Обчислює *a = g^k* mod *p* та *b = (y^k ∙ M)* mod *p* | a = 2932  b = 571 |  |  |  |
|  | Надсилає одержувачу шифротекст *(a; b).* | 2932,  571 |  |  |  |
|  |  |  | 10 | Використовуючи секретний ключ ***x***, розшифровує отриманий шифротекст:  M′ = (*a****^***(*p-1-x*) ∙*b*) mod *p* | M′ = 1903 |

# 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**

**Код до завдання 1**

class DiffieHellman:

""" Class to represent the Diffie-Hellman key exchange protocol """

# Current minimum recommendation is 2048 bit.

def \_\_init\_\_(self, p, g, a):

self.p = p

self.g = g

self.\_\_a = a

def get\_private\_key(self):

""" Return the private key (a) """

return self.\_\_a

def gen\_public\_key(self):

""" Return A, A = g ^ a mod p """

# calculate G^a mod p

A = pow(self.g, self.\_\_a, self.p)

print("public key = g ^ a mod p =", A)

return A

def gen\_shared\_key(self, other\_contribution, sha=0):

""" Return g ^ ab mod p """

self.shared\_key = pow(other\_contribution, self.\_\_a, self.p)

print("shared key = g ^ ab mod p =", self.shared\_key)

if sha:

return hashlib.sha256(str(self.shared\_key).encode()).hexdigest()

else:

return self.shared\_key

g = 5

p = 23

a = 6

b = 15

print("g = 5, p = 23, a = 6, b = 15")

Alice = DiffieHellman(p, g, a)

Bob = DiffieHellman(p, g, b)

print("Generate public key for Alice")

A = Alice.gen\_public\_key()

print("Generate public key for Bob")

B = Bob.gen\_public\_key()

print("A =", A, "B =", B)

print("Generate shared key for Bob")

bob\_sharedkey = Bob.gen\_shared\_key(A)

print("Generate shared key for Alice")

alice\_sharedkey = Alice.gen\_shared\_key(B)

# Додаток 2

# Код до завдання 2

import random

# from math import pow

a = random.randint(2, 10)

def gcd(a, b):

if a < b:

return gcd(b, a)

elif a % b == 0:

return b

else:

return gcd(b, a % b)

# Generating large random numbers

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

return pow(g, x, p)

# Modular exponentiation

def power(a, b, c):

x = 1

y = a

while b > 0:

if b % 2 == 0:

x = (x \* y) % c

y = (y \* y) % c

b = int(b / 2)

return x % c

def decrypt(a, b, q, p):

return (a \*\* (p - 1 - q) \* b) % p

msg = 1903

p = 3581

g = 3 # alpha

x = 1669 # a

k = 1777

print("Original Message :", msg)

print("p = {}, g = {}, x = {}, k = {}".format(p, g, x, k))

print("Generating Key")

key = gen\_key(p, g, x)

print("key", key)

a = gen\_key(p, g, k)

b = ((key \*\* k) \* msg) % p

print("a, b", a, b)

print("decrypted: ", decrypt(a, b, x, p))