Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №7

По дисциплине «Методы защиты информации»

По теме «Криптография с использованием эллиптических кривых»

Выполнила:

Студентка гр. 653501

Д. А. Гуринович

Проверил:

В. С. Артемьев

Минск 2019

Содержание

[1. Постановка задачи](#_2et92p0)

2. Краткие теоретические сведения

3. Схема алгоритма

4[. Результаты выполнения](#_tyjcwt)

5[. Вывод](#_3dy6vkm)

[ПРИЛОЖЕНИЕ 1. Исходный код программы.](#_1t3h5sf)

# 1. Постановка задачи

В современном мире остро стоит вопрос о безопасности. Обеспечение безопасности является важным аспектом деятельности любой компании. Для обеспечения безопасности используется множество различных средств, как аппаратных, так и программных. Одним из таких средств является шифрование.

Шифрование — обратимое преобразование информации в целях сокрытия от [неавторизованных](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) лиц, с предоставлением, в это же время, [авторизованным](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) пользователям доступа к ней. Главным образом, шифрование служит задачей соблюдения [конфиденциальности](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) передаваемой информации. Важной особенностью любого алгоритма шифрования является использование [ключа](https://ru.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%D0%B8%D1%8F)), который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма.

В ходе данной лабораторной работы необходимо изучить теоретическую часть о шифровании с использованием эллиптических кривых и алгоритме Диффи-Хелмана, на основании которых создать приложение, использующие данные технологии.

**2. Краткие теоретические сведения**

Протокол Ди́ффи — Хе́ллмана — [криптографический протокол](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB), позволяющий двум и более сторонам получить общий секретный [ключ](https://ru.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%D0%B8%D1%8F)), используя незащищенный от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов [симметричного шифрования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Схема открытого распределения ключей, предложенная Диффи и Хеллманом, произвела настоящую революцию в мире шифрования, так как снимала основную проблему классической криптографии — проблему распределения ключей.

В чистом виде алгоритм Диффи — Хеллмана уязвим для модификации данных в канале связи, в том числе для атаки «[Man-in-the-middle (человек посередине)](https://ru.wikipedia.org/wiki/%D0%A7%D0%B5%D0%BB%D0%BE%D0%B2%D0%B5%D0%BA_%D0%BF%D0%BE%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5)», поэтому схемы с его использованием применяют дополнительные методы односторонней или двусторонней аутентификации.

Ниже показана наглядная схема обмена между двумя пользователями. Алиса и Боб хотят использовать общий ключ для шифрования переписки. Рассмотрим детально этот процесс, используя краски вместо чисел:

1. Алиса и Боб выбрали общую краску.
2. Алиса и Боб выбрали по одной секретной краске.
3. Алиса и Боб смешали общую и секретную краску.
4. Алиса и Боб обменялись получившимися смешанными красками.
5. Алиса смешала полученную смешанную краску от Боба со своей секретной краской.
6. Боб смешал полученную смешанную краску от Алисы со своей секретной краской.
7. Теперь у Алисы и Боба есть общая секретная краска.

Предположим, существует два абонента: Алиса и Боб. Обоим абонентам известны некоторые два числа, которые не являются секретными и могут быть известны также другим заинтересованным лицам. Для того, чтобы создать неизвестный более никому секретный ключ, оба абонента генерируют большие случайные числа: Алиса — число, Боб — число. Затем Алиса вычисляет [остаток от деления](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BA%D0%BE%D0%BC) и пересылает его Бобу, а Боб вычисляет [остаток от деления](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BA%D0%BE%D0%BC) (2) и передаёт Алисе. Предполагается, что злоумышленник может получить оба этих значения, но не модифицировать их (то есть, у него нет возможности вмешаться в процесс передачи).

На втором этапе Алиса на основе имеющегося у неё и полученного по сети вычисляет значение.

Боб на основе имеющегося у него и полученного по сети вычисляет значение.

Как нетрудно видеть, у Алисы и Боба получилось одно и то же число.

Его они и могут использовать в качестве секретного ключа, поскольку здесь злоумышленник встретится с практически неразрешимой (за разумное время) проблемой вычисления (3) или (4) по перехваченным, если числа выбраны достаточно большими.

При работе алгоритма каждая сторона:

1. генерирует случайное [натуральное число](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) a — закрытый ключ
2. совместно с удалённой стороной устанавливает открытые параметры p и g (обычно значения p и g генерируются на одной стороне и передаются другой), где  
   p является [случайным простым числом](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE)(p-1)/2 также должно быть [случайным простым числом](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) (для повышения безопасности)[[5]](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D0%B8%D1%84%D1%84%D0%B8_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0#cite_note-5)g является [первообразным корнем](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D1%80%D0%B5%D0%BD%D1%8C_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%87%D0%B8%D1%81%D0%B5%D0%BB)) [по модулю](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E) p (также является простым числом)
3. вычисляет открытый ключ A, используя преобразование над закрытым ключом  
   A = ga mod p
4. обменивается открытыми ключами с удалённой стороной
5. вычисляет общий секретный ключ K, используя открытый ключ удаленной стороны B и свой закрытый ключ a  
   K = Ba mod p  
   К получается равным с обеих сторон, потому что:  
   Ba mod p = (gb mod p)a mod p = gab mod p = (ga mod p)b mod p = Ab mod p

В практических реализациях для a и b используются числа порядка 10100 и p порядка 10300. Число g не обязано быть большим и обычно имеет значение в пределах первого десятка.

**3. Схема алгоритма**

# 

# 4. Результаты выполнения

# 

Рисунок 4.1. Исходные данные

# 

Рисунок 4.2. Результат работы программы

**5. Вывод**

В рамках данной лабораторной работы были рассмотрены шифрование с использованием эллиптических кривых и алгоритм Диффи-Хелмана.

Для демонстрации работы описанных выше алгоритмов было разработано и реализовано приложение с использованием указанного выше алгоритма.

# ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ

import collections

def inv(n, q):

for i in range(q):

if (n \* i) % q == 1:

return i

assert False, "unreached"

def sqrt(n, q):

assert n < q

for i in range(1, q):

if i \* i % q == n:

return (i, q - i)

pass

raise Exception("not found")

Coord = collections.namedtuple("Coord", ["x", "y"])

class EC(object):

"""System of Elliptic Curve"""

def \_\_init\_\_(self, a, b, q):

assert 0 < a and a < q and 0 < b and b < q and q > 2

assert (4 \* (a \*\* 3) + 27 \* (b \*\* 2)) % q != 0

self.a = a

self.b = b

self.q = q

self.zero = Coord(0, 0)

def is\_valid(self, p):

if p == self.zero: return True

l = (p.y \*\* 2) % self.q

r = ((p.x \*\* 3) + self.a \* p.x + self.b) % self.q

return l == r

def at(self, x):

assert x < self.q

ysq = (x \*\* 3 + self.a \* x + self.b) % self.q

y, my = sqrt(ysq, self.q)

return Coord(x, y), Coord(x, my)

def neg(self, p):

return Coord(p.x, -p.y % self.q)

def add(self, p1, p2):

if p1 == self.zero: return p2

if p2 == self.zero: return p1

if p1.x == p2.x and (p1.y != p2.y or p1.y == 0):

return self.zero

if p1.x == p2.x:

l = (3 \* p1.x \* p1.x + self.a) \* inv(2 \* p1.y, self.q) % self.q

else:

l = (p2.y - p1.y) \* inv(p2.x - p1.x, self.q) % self.q

x = (l \* l - p1.x - p2.x) % self.q

y = (l \* (p1.x - x) - p1.y) % self.q

return Coord(x, y)

def mul(self, p, n):

r = self.zero

m2 = p

while 0 < n:

if n & 1 == 1:

r = self.add(r, m2)

pass

n, m2 = n >> 1, self.add(m2, m2)

pass

return r

def order(self, g):

assert self.is\_valid(g) and g != self.zero

for i in range(1, self.q + 1):

if self.mul(g, i) == self.zero:

return i

pass

raise Exception("Invalid order")

pass

class DiffieHellman(object):

def \_\_init\_\_(self, ec, g):

self.ec = ec

self.g = g

self.n = ec.order(g)

pass

def gen(self, priv):

assert 0 < priv and priv < self.n

return self.ec.mul(self.g, priv)

def secret(self, priv, pub):

assert self.ec.is\_valid(pub)

assert self.ec.mul(pub, self.n) == self.ec.zero

return self.ec.mul(pub, priv)