МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики

Кафедра вычислительных систем и программирования

**КУРСОВАЯ РАБОТА**

по дисциплине:

**«Языки и методы программирования»**

Тема: «Реализация и исследование шифрования Эль-Гамаля»

Направление: 01.03.02 Прикладная математика и информатика

Студент: Карпова Софья

Группа: ПМ1701 Подпись:

Проверил: Фёдоров Дмитрий Юрьевич

Должность: Старший преподаватель

Оценка: Дата:

Подпись:

Санкт-Петербург

2018

**Оглавление**

[Введение 3](#_Toc532821955)

[1. Описание схемы шифрования Эль-гамаля 5](#_Toc532821956)

[1.1. Генерация ключей 5](#_Toc532821957)

[1.2. Шифрование 5](#_Toc532821958)

[1.3. Дешифрование 5](#_Toc532821959)

[2. сОЗДАНИЕ ПРОТОТИПА ПРОГРАММЫ ШИФРОВАНИЯ, ОСНОВАННОЙ НА СХЕМЕ эЛЬ-гАМАЛЯ 7](#_Toc532821960)

[2.1. Генерация ключей 7](#_Toc532821961)

[2.2. Реализация шифрования 11](#_Toc532821962)

[2.3. Реализация дешифровки 13](#_Toc532821963)

[3. Исследование алгоритма шифрования Эль-Гамаля 15](#_Toc532821964)

[3.1. Криптографическая стойкость 15](#_Toc532821965)

[3.2. Недостатки схемы шифрования Эль-Гамаля 15](#_Toc532821966)

[Заключение 17](#_Toc532821967)

[Список использованной литературы 19](#_Toc532821968)

# **Введение**

Защищённое хранение и передача информации. Эта проблема появилась ещё в древние времена, и тогда появилось решение - шифрование. До наших дней дошли упоминания о шифрах в Древнем Египте, Древней Греции и Древнем Риме. С ходом технологического прогресса известные методы шифрования совершенствовались, а также создавались новые. Сегодня шифрование является неотъемлемой частью нашей повседневной жизни. С бурным развитием цифровых технологий значительно возрастает объём информации, которую хочется или необходимо защищать от посторонних лиц, а также методы кражи этой информации. Поэтому шифрование является одной из важнейших задач современной науки, в частности криптографии.

Шифрование с открытым ключом - одно из решений данной проблемы. Строгое определение звучит следующим образом: Криптографическая система с открытым ключом (разновидность асимметричного шифрования) - система шифрования и/или электронной подписи (ЭП), при которой открытый ключ передаётся по открытому (то есть незащищённому) каналу и используется для проверки ЭП и шифрования сообщения. Закрытый ключ используется для генерации ЭП и для расшифровки сообщений и строго охраняется. Начало асимметричным шифрам было положено в работе «Новые направления в современной криптографии» Уилфреда Диффи и Мартина Хеллмана, опубликованной в 1976 году. Под влиянием работы Ральфа Меркла о распространении открытого ключа, они предложили метод получения секретных ключей, используя открытый канал. Этот метод, известный как метод Диффи-Хеллмана, был первым опубликованным практичным методом для установления разделения секретного ключа между заверенными пользователями канала.

В 1985 году египетский учёный Тахер Эль-Гамаль предложил разработанную им новую схему шифрования, которая является усовершенствованным вариантом схемы Диффи-Хеллмана. Схема Эль-Гамаля основана на трудности вычисления дискретных логарифмов в конечном поле и включает в себя алгоритм шифрования и алгоритм цифровой подписи. В отличие от схемы RSA, схема Эль-Гамаля не была запатентована, поэтому стала более дешёвой альтернативой, так как не требовалась оплата лицензии. Схема Эль-Гамаля лежала в основе бывших стандартов электронной цифровой подписи США (DSA) и России (ГОСТ P.34 10-94).

Цель данной курсовой работы изучить схему шифрования Эль-Гамаля, выявить её преимущества и недостатки и реализовать программу шифрования – дешифрования на практике.

Для достижения поставленной цели будут решены следующие задачи:

1. изучение теоретической стороны задачи;
2. создание прототипа программы для генерации ключей, шифрования и дешифрования информации;
3. создание функций, необходимых для реализации прототипа программы.
4. выявление преимуществ и недостатков схемы шифрования Эль-Гамаля;

# **Описание схемы шифрования Эль-гамаля**

Работу алгоритма шифрования можно разделить на несколько этапов:

1. генерация ключей;
2. шифрование;
3. дешифрование;

### **Генерация ключей**

* генерируется большое случайное простое число;
* выбирается число ***g*** – первообразный корень ;
* выбирается случайное целое число , такое что ;
* вычисляется: (1)
* тройка чисел – открытый ключ, – закрыты ключ;

### **Шифрование**

* ***M*** – сообщение для шифрования. Число ***M*** должно быть меньше числа ;
* вычисляется пара чисел ***(a, b),*** такие что:

(2)

(3)

Пара  ***–*** является шифротекстом;

* число – надёжно удаляется из памяти;

### **Дешифрование**

Дешифровка происходит по следующей формуле:

(4)

где ***M*** – исходное сообщение. Нетрудно проверить корректность формулы:

(5)

Схематическое изображение схемы шифрования Эль-Гамаля можно увидеть на рисунке 1:

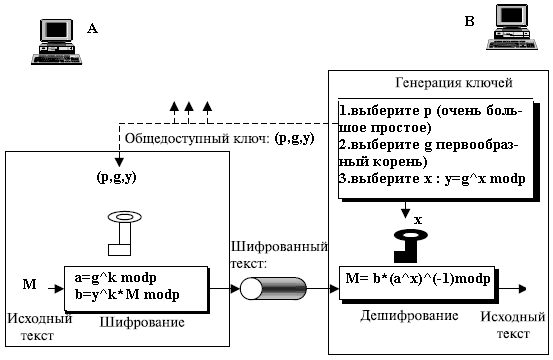


Рисунок 1 – Схема шифрования

# **сОЗДАНИЕ ПРОТОТИПА ПРОГРАММЫ ШИФРОВАНИЯ, ОСНОВАННОЙ НА СХЕМЕ эЛЬ-гАМАЛЯ**

## **Генерация ключей**

При решении задачи генерации открытого и закрытого ключей возникло несколько важных подзадач, среди которых:

1. Генерация случайных простых чисел
2. Нахождение первообразных корней
3. Возведение больших чисел в степень в поле

Эти подзадачи можно решать стандартными методами, однако в этом случае производительность программы будет низкой.

* + 1. **Генерация случайных простых чисел**

Для решения этой задачи были реализованы такие методы, как решето Эратосфена и решето Аткина. Функции ***eratos()*** и ***atkins()*** принимают в качестве входных данных число ***n***, а в качестве результата выдают ***n***-ое простое число (из-за громоздкости текста кода, в работе он приведён не будет). Для того чтобы понять, кокой из методов эффективнее всего использовать при шифровании, сравним время работы этих функций для ***n*** от 100000 до 750000 (См рисунок 2) и для ***n*** от 1000000 до 7500000 (См рисунок 3).

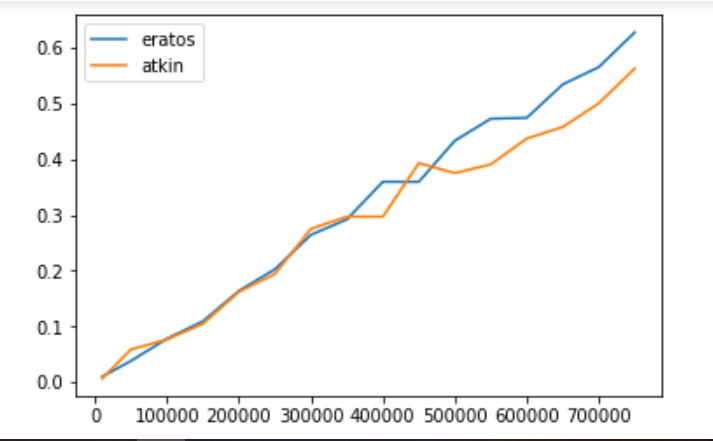


Рисунок 2 – сравнение быстродействия решета Аткина и решета Эратосфена

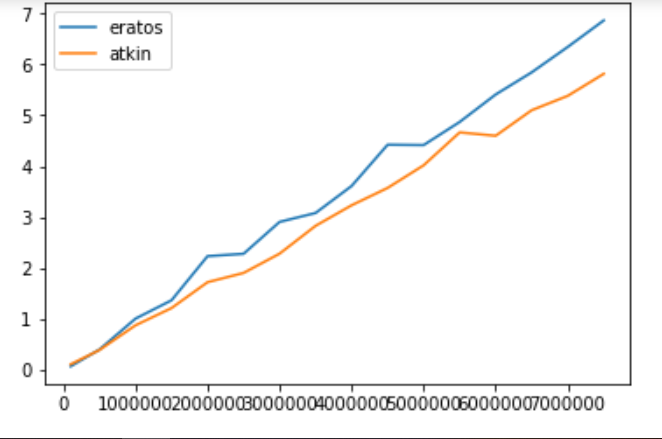


Рисунок 3 – сравнение быстродействия решета Аткина и решета Эратосфена

Функции ***generate\_prime\_atkin()*** и ***generate\_prime\_eratos(),*** которые используют в себе функции ***eratos()*** и ***atkins()*** соответственно***,*** генерируют случайные простые числа в диапазоне от 9973 до 999983 (т.е. от 10000-го простого числа до 1000000-го). На рисунках 4 и 5 показано время работы этих функций: по оси абсцисс – номер запуска, по оси ординат – время.

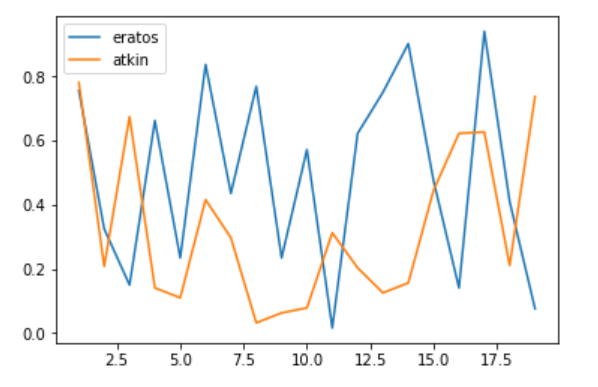
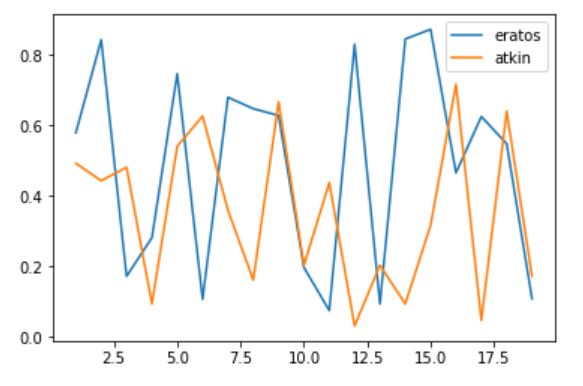


Рисунок 4 – Сравнение быстродействияРисунок 5– Сравнение быстродействия

Из графиков на рисунках 2, 3, 4 и 5 видно, что в большинстве случаев, решето Аткина использовать выгоднее, чем решето Эратосфена.

* + 1. **Нахождение первообразных корней**

Для нахождения первообразного корня используется функция ***get\_primitive\_root()***, которая принимает на вход число ***p***, для которого методом полного перебора находит все первообразные корни (он не один). В начале работы функция генерирует случайное число ***i*** – порядковый номер первообразного корня. Когда при полном переборе вычисляется ***i***–тый первообразный корень, функция прерывается, и в качестве результата возвращается вычисленный случайный первообразный корень числа ***p.*** Текст функции ***get\_primitive\_root()*** представлен ниже:

def get\_primitive\_root2(p):

phi = p – 1

n = random.randint(1,(p-1)/2)

k = 1

r = 0

for i in range(int((p-1)/2),p):

if pow(i,phi,p) == 1:

lst = [1]

for j in range(1,phi):

lst.append(pow(i,j,p))

lst = list(set(lst))

if lst == list(range(1,p)):

r = i

if k == n:

break

else:

k = k + 1

return r

* + 1. **Возведение больших чисел в степень в поле**

В рамках этой работы используется базовая функция языка Python - **a%b,** которая находит остаток от деления числа **a** на **b**. Такое решения не является оптимальным, так как затратно по памяти и времени, поэтому требует усовершенствования.

* + 1. **Генерация ключей**

Функция ***get\_keys()*** по запросу пользователя в соответствии с алгоритмом Эль-Гамаля высчитывает открытый и закрытый ключи, которые записываются в виде списка ***[close\_key, open\_key]***, где ***open\_key*** – список ***[g, p, y].*** Текст функции ***get\_keys()*** представлен ниже:

def get\_keys():

p = generate\_prime\_atkin()

g = get\_primitive\_root2(p)

x = random.randint(1,p-1)

y = pow(g,x,p)

open\_key = [g,p,y]

close\_key = x

return [close\_key, open\_key]

## **Реализация шифрования**

* + 1. **Шифрование числа**

При шифровании числа используется функция ***encrypt(message, open\_k),*** принимающая на вход число ***message***, открытый ключ - ***open\_k,*** и возвращающая в качестве результата список ***cipher***, вида ***[a,b]*** – шифротекст. Числа ***a, b*** высчитываются в соответствии с алгоритмом шифрования Эль-Гамаля, описанном в пункте 1.1.2. Текст функции ***encrypt()*** представлен ниже:

def encrypt(message,open\_k):

p = open\_k[1]

g = open\_k[0]

y = open\_k[2]

k = random.randint(1, p)

cipher = []

#message = 764

a = pow(g,k,p)

b = ((y^k)\* message)% p

cipher.append(a)

cipher.append(b)

y = 0

k = 0

return (cipher, open\_k, close\_k)

* + 1. **Шифрование строки и текстового файла**

Функция ***encrypt\_text(message, open\_k)*** :

* + 1. принимает на вход строку ***message*** и открытый ключ - ***open\_k***
    2. преобразует строку символов в список, где каждый элемент – код символа ASCII
    3. каждый элемент списка преобразуется в шифрограмму вида ***[a,b]*** и записывается в общий список ***cipher.*** Таким образом на выходе получается список вида ***[[a1,b1], [a2,b2], … , [an,bn]]***, где ***n*** – длина строки. Текст функции ***encrypt\_text()*** представлен ниже:

def encrypt\_text(message,open\_k):

m = []

for i in message:

m.append(ord(i))

p = open\_k[1]

g = open\_k[0]

y = open\_k[2]

k = random.randint(1, p)

cipher = []

a = g^k % p

b = y^k % p

for i in m:

r = []

c = b\* i

r.append(a)

r.append(c)

cipher.append(r)

y = 0

k = 0

return cipher

Для шифрования текстового файла используется функция ***encrypt\_text\_from\_file2newfile(file, newfile, open\_k)*** :

1. входные параметры ***file, newfile, open\_k,*** где ***file, newfile*** – имена файлов откуда берётся текст для шифрования и куда будет записываться шифрограмма соответственно, а ***open\_k*** – открытый ключ;
2. к каждой строке текстового файла ***file*** применяется функция ***encrypt\_text***() и в новый файл ***newfile*** построчно записывается шифротекст;

Текст функции ***encrypt\_text\_from\_file2newfile()*** представлен ниже:

def encrypt\_text\_from\_file2newfile(file,newfile,open\_k):

f = open(file, 'r')

c = open(newfile, 'a')

for line in f:

l = encrypt\_text(line,open\_k)

c.write(str(l[:-1]) + "\n")

c.close()

f.close()

return(close\_k)

## **Реализация дешифровки**

Дешифровка происходит аналогично шифрованию:

1. Функция ***decipher(cipher, open\_key, close\_key)*** дешифрует шифротекст - список вида ***[a,b],*** в число. Текст функции ***decipher()*** представлен ниже:

def decipher(cipher,open\_key,close\_key):

a = cipher[0]

b = cipher[1]

m = int(b / (a^int(close\_key) % int(open\_key[1])))

return(m)

1. Функция ***decipher\_text(cipher, open\_key, close\_key)*** дешифрует список ***cipher*** вида ***[[a1,b1], [a2,b2], … , [an,bn]]*** – в строку длины ***n***. Текст функции ***decipher\_text()*** представлен ниже:

def decipher\_text(cipher,open\_key,close\_key):

dem = []

decipherment = ""

for i in cipher:

a = i[0]

b = i[1]

m = int(b / (a^int(close\_key) % int(open\_key[1])))

dem.append(m)

for i in dem:

decipherment = decipherment + chr(i)

return decipherment

1. Функция ***decipher\_from\_file2newfile(file, newfile, open\_key, close\_key)*** дешифрует текстовый файл ***file*** с шифротекстом. Построчно считывая из ***file*** строки вида “***[[a1,b1], [a2,b2], … , [an,bn]]***”, функция дешифрует их помощью функции ***decipher\_text()*** и записывается в новый текстовый документ ***newfile***. Текст функции ***decipher\_from\_file2newfile()*** представлен ниже:

def decipher\_from\_file2newfile(file,newfile,open\_key,close\_key):

f = open(file, 'r')

c = open(newfile, 'a')

for l in f:

a = from\_str2lst(l)

c.write(decipher\_text(a,open\_key,close\_key))

f.close()

c.close()

Функция ***decipher\_from\_file2newfile()*** построчно считывает шифротекст из файла ***file***. Но функция ***decipher\_text()*** на вход принимает объект типа ***list***, а не ***string***. Поэтому для корректной работы используется функция ***from\_str2lst(st)***, которая переделывает строку ***st*** вида “***[[a1,b1], [a2,b2], … , [an,bn]]*** ” в список. Текст функции ***from\_str2lst()*** представлен ниже:

def from\_str2lst(st):

m = []

for i in range(len(st)-1):

if st[i] == "[":

a = 0

c = []

if st[i] >= "0" and st[i] <= "9":

a = a\*10 + int(st[i])

if st[i + 1] == ",":

c.append(a)

a = 0

if st[i + 1] == "]":

c.append(a)

m.append(c)

return m[:-1]

# **Исследование алгоритма шифрования Эль-Гамаля**

## **Криптографическая стойкость**

В настоящее время криптосистемы с открытым ключом считаются наиболее перспективными. К ним относится и схема Эль-Гамаля, крипто стойкость которой основана на вычислительной сложности проблемы дискретного логарифмирования, где по известным требуется вычислить ,удовлетворяющий сравнению:**.** Проблема дискретного логарифма состоит в том, что, зная основание степени и получившийся после возведения результат по модулю простого числа, невозможно за обозримое время определить, в какую именно степень было возведено основание.

Так как при шифровании вводится случайная величина , то шифр Эль-Гамаля можно назвать шифром многозначной замены. Из-за случайности выбора числа такую схему ещё называют схемой вероятностного шифрования. Вероятностный характер шифрования является преимуществом для схемы Эль-Гамаля, так как у схем вероятностного шифрования наблюдается большая стойкость по сравнению со схемами с определённым процессом шифрования.

Криптостойкость схемы Эль-Гамаля сравнима с криптостойкостью схемы RSA и составляет MIPS для ключа 1300 бит, но из-за большой трудоёмкости операции возведения в степень больших чисел быстродействие RSA ниже.

## **Недостатки схемы шифрования Эль-Гамаля**

Одним из недостатков схемы шифрования Эль-Гамаля является удвоение длины зашифрованного текста по сравнению с начальным текстом. Так шифротекстом одного числа (символа) является пара чисел **,** объём передаваемого сообщения увеличивается вдвое.

Другим недостатком схемы является трудность задачи вычисления дискретных логарифмов в поле. Операции возведения в степень больших чисел в конечном поле достаточно трудоёмки даже для современных процессоров, даже если производятся по оптимизированным алгоритмам. Поэтому процесс вычисления новых ключей шифрования занимает достаточно долгое время. Например, в рамках этой курсовой работы среднее вычисление ключей занимает от 10 до 30 секунд, где случайное простое число ***p*** находится в промежутке от 1000-го до 100000-го простого числа (991 и 99991 соответственно). Однако на практике используются ключи большего размера (как правило 768б 1024 и 1536 бит), и время вычисления одного ключа занимает до нескольких минут.

# **Заключение**

В процессе написания курсовой работы была изучен метод шифрования Эль-Гамаля, его теоретическое обоснование, принцип работы и, как результат, создан прототип программы, которая позволяет генерировать открытый и закрытый ключи для шифрования, шифровать и дешифровывать сообщения – числа, строки символов, текстовые сообщения. На данный момент времени криптосистемы с открытым ключом, к которым относится схема шифрования Эль-Гамаля, считаются наиболее перспективными.

Главным преимуществом схемы шифрования является её высокая криптостойкость. Как было показано в работе, это достигается благодаря сложности процесса вычисления степеней больших чисел в конечном поле, а также использованию случайных величин при шифровании. Благодаря этому схема шифрования Эль-Гамаля широко применялась для шифрования данных как в частных, так и в государственных структурах.

Однако сложность вычислительных процессов и, как следствие, большое потребление ресурсов для них (время) является одним из главных недостатков данной схемы шифрования. Так при приблизительно одинаковой криптостойкости, алгоритм шифрования RSA быстродейственнее примерно в тысячу раз. Также в процессе шифрования длина сообщения увеличивается в два раза, что также является большим недостатком схемы шифрования. Как результат, алгоритм шифрования не является оптимальным по затрачиваемым ресурсам, времени шифрования и конечным объёмом зашифрованного сообщения.

Вследствие этих недостатков, схема шифрования Эль-Гамаля не применяется в чистом виде в настоящее время. Сегодня для шифрования больших объёмов данных применяются более быстрые схемы шифрования (обычно симметричные) с использованием сессионного ключа, а этот ключ уже шифруется с помощью ассиметричной схемы шифрования (в частности схема Эль-Гамаля) и уже передаётся получателю. Таким образом схема шифрования Эль-Гамаля применяется в более совершенных современных схемах шифрования. На её основе было разработано и используются большое количество современных алгоритмов. К ним относятся алгоритмы DSA, ECDSA, KCDSA, схема Шнорра.

**Список использованной литературы**

1. TAHER ELGAMAL, MEMBER, IEEE, «A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms», IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-31, NO. 4, JULY 1985 - Режим доступа: <https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf>, свободный. – Загл. с экрана. – Язык английский.
2. Википедия: статья: «Криптосистема с открытым ключём» 2018 - Режим доступа: <https://ru.wikipedia.org/wiki/Криптосистема_с_открытым_ключом>, свободный. – Загл. с экрана. – Язык русский.
3. Википедия: статья: «Протокол Диффи-Хеллмана» 2018 - Режим доступа: <https://ru.wikipedia.org/wiki/Протокол_Диффи_—_Хеллмана>, свободный. – Загл. с экрана. – Язык русский.
4. Википедия: статья: «Схема Эль-Гамаля» 2018 - Режим доступа: <https://ru.wikipedia.org/wiki/Схема_Эль-Гамаля>, свободный. – Загл. с экрана. – Язык русский.
5. «Ещё раз о поиске простых чисел» 21.11.2012 - Режим доступа: <https://habr.com/post/133037/>, свободный. – Загл. с экрана. – Язык русский.
6. «3.2. Алгоритм шифрования Эль Гамаль» 2006 - Режим доступа: <https://studfiles.net/preview/909637/page:10/> свободный. – Загл. с экрана. – Язык русский.
7. «6.4. Нахождение первообразных корней по простому модулю» 2007 - Режим доступа: <https://studfiles.net/preview/6268704/page:25/> свободный. – Загл. с экрана. – Язык русский.
8. «Криптосистема Эль-Гамаля» 15.01.2014 - Режим доступа: <https://studopedia.su/10_105628_kriptosistema-el-gamalya.html> свободный. – Загл. с экрана. – Язык русский.