Федеральное государственное бюджетное образовательное учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

Расчетное задание

по дисциплине «Защита данных»

**Тема: Программная реализация шифра Полига-Хеллмана.**

Институт ИВТИ

Кафедра Прикладной математики

Группа А-13-16

Бабушкин С.А.

Преподаватель: Хорев П.Б.

Москва, 2020

# Криптосистема Полига-Хеллмана

Схема шифрования Полига – Хеллмана сходна со схемой шифрования RSA. Она представляет собой несимметричный алгоритм, поскольку используются различные ключи для шифрования и расшифрования. В то же время эту схему нельзя отнести к классу криптосистем с открытым ключом, так как ключи шифрования и расшифрования легко выводятся один из другого. Оба  ключа  (шифрования  и  расшифрования)   нужно   держать   в секрете.

Аналогично схеме RSA криптограмма  C и открытый текст  P  определяются из соотношений:

С = Рe mod n,

P = Cd mod n,

где  e\*d ≡ 1  (по модулю некоторого составного числа).

В отличие от алгоритма RSA в этой схеме число n  не определяется через два больших простых числа; число n  должно оставаться частью секретного ключа. Если кто-либо узнает значения e и n, он сможет вычислить значение d.

Не зная значений e или d, противник будет вынужден вычислять значение

e = logp C (mod n).

Известно, что это является трудной задачей.

Схема шифрования Полига-Хеллмана  
***Генерация ключей***  
1. Выбирается большое простое число n – модуль.  
2. Произвольным образом выбираются два числа, удовлетворяющие  
условию:

3. Пара (d, e) – является закрытым ключом.  
4. Число n – является открытым ключом.  
***Шифрование сообщения***Основной шаг алгоритма в точности такой же, как и в алгоритме RSA.  
Шифротекст вычисляется по формуле  
  
***Расшифрование сообщения***  
Для расшифрования сообщения 'c' нужно взять каждый зашифрованный  
блок и вычислить:  


***Обоснование:***

Имеем функцию шифрования:

*, где m – исходное сообщение, e – ключ шифрования, p – простое число*

и функцию дешифрования:

*, где с –шифротекст, d – ключ расшифрования, p – простое число*

Если утверждение e\*d = 1 mod(p-1) является верным, то можно перейти к

e\*d=k(p-1)+1 в силу определения, где k – некоторое натуральное число.

Тогда имеем:

А по малой теореме Ферма:

Следовательно:

# Программная реализация:

Исходный код написан на языке: python

Используемые библиотеки: labmath(вычисление НОД), secrets(randbelow(p) - генерация случайного числа на полуинтервале [0, p))

Большое простое число выбрано из 2048-битной группы ffdhe2048 со значением регистра 256 и посчитанное по формуле:

Значение p – безопасное простое число.

Шестнадцатеричное представление данного числа:

FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1

D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9

7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561

2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935

984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735

30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB

B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19

0B07A7C8 EE0A6D70 9E02FCE1 CDF7E2EC C03404CD 28342F61

9172FE9C E98583FF 8E4F1232 EEF28183 C3FE3B1B 4C6FAD73

3BB5FCBC 2EC22005 C58EF183 7D1683B2 C6F34A26 C1B2EFFA

886B4238 61285C97 FFFFFFFF FFFFFFFF

Запуск программы осуществляется путем выполнения файла PolHelCipher.exe

На вход программе поступает текстовый файл Message.txt с сообщением которое нужно зашифровать. Если файла не существует, то создается новый файл, в который можно занести нужное сообщение.

На выход поступает информация: шифротекст в файле CryptedMessage.txt и файл Parameters.txt, в котором содержится информация о ключах и простом числе p для проверки.

Компиляция программы с помощью модуля pyinstaller.

## Исходный код:

import secrets

from labmath import xgcd

import re

def enc(m, e, p):

return pow(m, e, p)

def dec(c, d, p):

return pow(c, d, p)

def keygen(p):

while True:

e = secrets.randbelow(p)

gcd, x, y = xgcd(e, p-1)

if gcd != 1: continue

d = x % (p-1)

break

return e, d

def main():

try:

with open('Message.txt', 'r') as fr:

s1 = fr.read().encode()

m = int.from\_bytes(s1, 'big')

#safe prime from RFC 7919 (ffdhe2048)

p = 

e, d = keygen(p)

c = enc(m, e, p)

assert m == dec(c, d, p)

s2 = m.to\_bytes((m.bit\_length() + 7) // 8, 'big').decode()

fw = open('CryptedMessage.txt', 'w')

fw.write(str(c) + '\n')

fw.close()

fw = open('DecryptedMessage.txt', 'w')

fw.write(s2 + '\n')

fw.close()

fw = open('Parameters.txt', 'w')

fw.write("Ключи:\n" +

"e = " + str(e) + '\n' +

"d = " + str(d) + '\n' +

"Простое число p = " + str(p))

fw.close()

print("Шифрование прошло успешно!")

except IOError:

with open('Message.txt', 'w') as fr:

print("Нет входного сообщения для шифрования!\nПожалуйста введите его в файл Message.txt!")

if \_\_name\_\_ == '\_\_main\_\_':

main()