Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 3

“Компьютерная реализация алгоритмов криптографии с открытым ключом на примере RSA”

Выполнил студент гр. 753504

Горбачёнок К. Н.

Проверил

Протько М.И.

Минск, 2020

# Постановка задачи

Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма RSA.

# Описание алгоритма

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других.

Криптографические системы с открытым ключом используют так называемые односторонние функции, которые обладают следующим свойством:

– если известно x, то f(x) вычислить относительно просто;

– если известно y=f(x), то для вычисления x нет простого (эффективного) пути.

Под односторонностью понимается не теоретическая однонаправленность, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования (обратной операции) за разумное время необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложение числа на простые множители.

В криптографической системе с открытым ключом каждый участник располагает как открытым ключом (англ. public key), так и закрытым ключом (англ. private key). В криптографической системе RSA каждый ключ состоит из пары целых чисел. Каждый участник создаёт свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно или даже публиковать их. Открытый и закрытый ключи каждого участника обмена сообщениями в криптосистеме RSA образуют «согласованную пару» в том смысле, что они являются взаимно обратными, то есть:

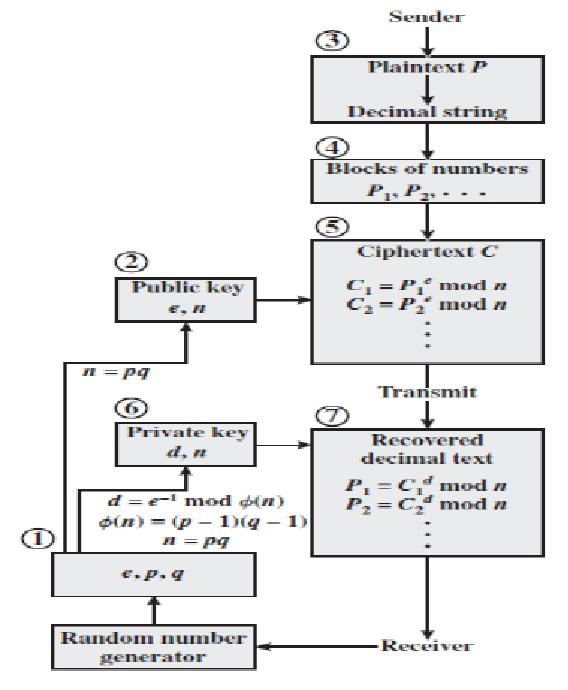
Для любых допустимых пар открытого и закрытого ключей (p,s)

существуют соответствующие функции шифрования Ep(x) и расшифрования Ds(x) такие, что для любого сообщения m из M, где M — множество допустимых сообщений, m = Ds(Ep(m)) = Ep(Ds(m)).

RSA-ключи генерируются следующим образом:

* Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).
* Вычисляется их произведение n=p\*q, которое называется модулем.
* Вычисляется значение функции Эйлера от числа n:
* phi(n)=(p-1)\*(q-1)
* Выбирается целое число e (1<e<phi(n)), взаимно простое со значением функции phi (n). Число e называется открытой экспонентой (англ. public exponent). Обычно в качестве e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые из чисел Ферма: 17, 257 или 65537, так как в этом случае время, необходимое для шифрования с использованием быстрого возведения в степень будет меньше. Слишком малые значения e, например 3, потенциально могут ослабить безопасность схемы RSA.
* Вычисляется число d, мультипликативно обратное к числу e по phi(n), то есть число, удовлетворяющее сравнению:
* d\*e =1 (mod phi(n)). Число d называется секретной экспонентой. Обычно оно вычисляется при помощи расширенного алгоритма Евклида.
* Пара (e,n) публикуется в качестве открытого ключа RSA (англ. RSA public key).
* Пара (d,n) играет роль закрытого ключа RSA (англ. RSA private key) и держится в секрете.

# Блок-схема алгоритма



***Блок-схема алгоритма RSA***

# Результаты работы программы

***Исходный файл***

******

***Результат работы программы***

# Прграммный код

import random

import math

def mod\_invers(a, m):

a = a % m;

for x in range(1, m) :

if ((a \* x) % m == 1) :

return x

return 1

def is\_prime(num):

if num == 2:

return True

if num < 2 or num % 2 == 0:

return False

for n in range(3, int(num \*\* 0.5) + 2, 2):

if num % n == 0:

return False

return True

def generate\_rsa\_key\_pair(p, q):

if not (is\_prime(p) and is\_prime(q)):

raise ValueError('Both numbers must be prime.')

elif p == q:

raise ValueError('p and q cannot be equal')

n = p \* q

phi = (p - 1) \* (q - 1)

e = random.randrange(1, phi)

g = math.gcd(e, phi)

while g != 1:

e = random.randrange(1, phi)

g = math.gcd(e, phi)

d = multiplicative\_inverse(e, phi)

return (e, n), (d, n)

def rsa\_encrypt(pk, plaintext):

key, n = pk

cipher = [(ord(char) \*\* key) % n for char in plaintext]

return cipher

def rsa\_decrypt(pk, ciphertext):

key, n = pk

plain = [chr((char \*\* key) % n) for char in ciphertext]

return ''.join(plain)

import rsa

def file\_info(filename="file.txt"):

with open(filename) as file\_handler:

text = file\_handler.readline().strip()

return text

def main():

pub\_key, pr\_key = rsa.generate\_rsa\_key\_pair(19, 23)

text = file\_info()

enc\_text = rsa.rsa\_encrypt(pub\_key, text)

print("\033[95m\nENCODED:\033[0m ", ''.join(str(i) for i in enc\_text))

dec\_text = rsa.rsa\_decrypt(pr\_key, enc\_text)

print("\033[95m\nDECODED:\033[0m ", dec\_text)

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

main()

# Вывод

В ходе написания лабораторной работы был изучен алгоритмы шифрования и дешифрования RSA, а также написана программная реализация.