**Введение**

Как известно, в настоящее время одним из наиболее надежных и эффективных методов защиты информации является шифрование, представляющее собой метод преобразования информации в зашифрованный текст с той целью, чтобы доступ к ней смог получить лишь пользователь, у которого есть необходимый ключ для дешифровки.

Ранее информация шифровалась и расшифровывалась при помощи одного и того же криптографического ключа. Однако в этом случае существует проблема: как передать такой ключ получателю, чтобы он с его помощью смог расшифровать ваше сообщение? Он должен передаваться либо при личной встрече, либо по надежным защищенным каналам связи, что позволит предотвратить перехват ключа посторонними. Но это не всегда удобно и достаточно проблематично.

Данная проблема решается при помощи ассиметричной криптографии. В данном случае у пользователя имеется так называемая ключевая пара, состоящая из закрытого ключа (Private Key) и открытого ключа (Public Key). Открытый ключ предназначен для массового распространения — вы отправляете его другим пользователям, публикуете на открытых серверах ключей и т. д. для того чтобы все желающие могли зашифровать с его помощью сообщение для вас.

Интересно то, что после того, как сообщение зашифровано, расшифровать его сможет лишь владелец закрытого ключа, который находится в паре с открытым ключом, то есть только вы и никто другой, даже отправитель сообщения.

Это возможно по тому, что открытый и закрытый ключи связаны между собой по особой математической зависимости, но получить из открытого ключа закрытый не представляется возможным.

В основе ассиметричной криптографии лежит алгоритм Диффи-Хеллермана, двух ученых, сформулировавших модель криптографической системы с открытым ключом. Впоследствии три других ученых Р. Ривест, А. Шамир и Л. Адлеман создали ассиметричный алгоритм шифрования RSA (от первых букв фамилий его создателей), который сегодня получил повсеместное распространение и используется как для шифрования/дешифрования сообщений, так и для создания электронно-цифровых подписей.

Ассиметричная криптография сегодня еще известна как криптография на основе инфраструктуры открытых ключей (Public Key Infrastructure), она широко используется по всему миру как серьезными структурами и организациями (например, министерством обороны США), так и рядовыми пользователями.

## **Алгоритм RSA**

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**

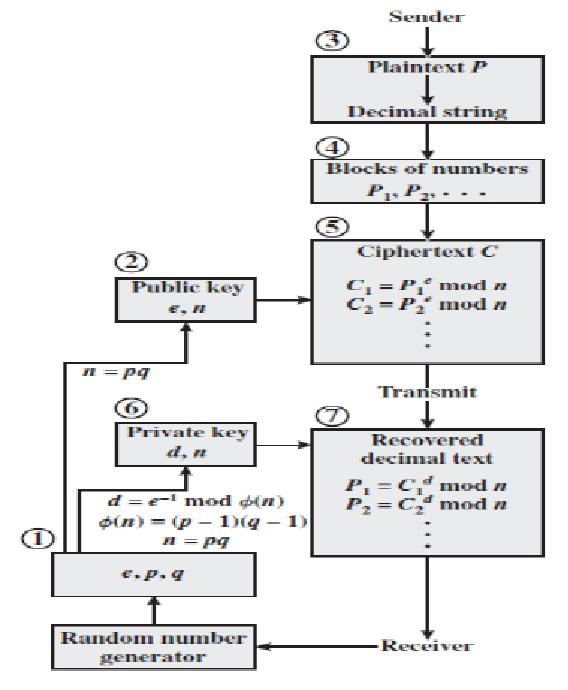
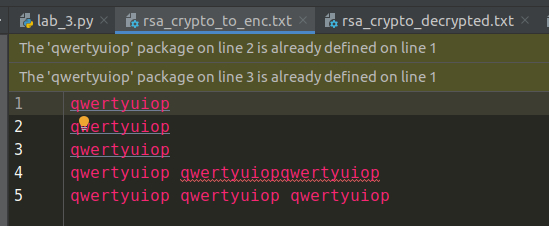
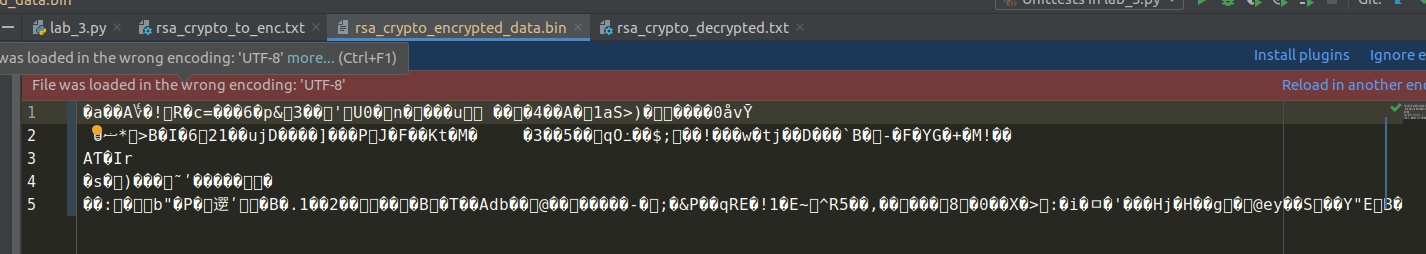


Рис.1. Схема алгоритма

**Пример работы программы**





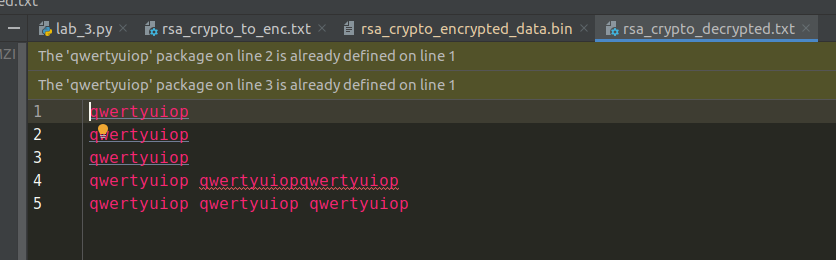


Рис.2. Пример работы

**Код программы**

*import* random

'''

Euclid's algorithm for determining the greatest common divisor

Use iteration to make it faster for larger integers

'''

*def* \_\_gcd(*a*, *b*):

*while b* != 0:

a, b = *b*, *a* % *b*

*return a*

'''

Euclid's extended algorithm for finding the multiplicative inverse of two numbers

'''

*def* \_\_multiplicative\_inverse(*e*, *phi*):

d = 0

x1 = 0

x2 = 1

y1 = 1

temp\_phi = *phi*

*while e* > 0:

temp1 = temp\_phi // *e*

temp2 = temp\_phi - temp1 \* *e*

temp\_phi = *e*

e = temp2

x = x2 - temp1 \* x1

y = d - temp1 \* y1

x2 = x1

x1 = x

d = y1

y1 = y

*if* temp\_phi == 1:

*return* d + *phi*

*return* d

'''

Tests to see if a number is prime.

'''

*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 = pq

n = *p* \* *q*

# Phi is the totient of n

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

# Choose an integer e such that e and phi(n) are coprime

e = random.randrange(1, phi)

# Use Euclid's Algorithm to verify that e and phi(n) are comprime

g = \_\_gcd(e, phi)

*while* g != 1:

e = random.randrange(1, phi)

g = \_\_gcd(e, phi)

# Use Extended Euclid's Algorithm to generate the private key

d = \_\_multiplicative\_inverse(e, phi)

# Return public and private keypair

# Public key is (e, n) and private key is (d, n)

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

*def* rsa\_encrypt(*pk*, *plaintext*):

# Unpack the key into it's components

key, n = *pk*

# Convert each letter in the plaintext to numbers based on the character using a^b mod m

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

# Return the array of bytes

*return* cipher

*def* rsa\_decrypt(*pk*, *ciphertext*):

# Unpack the key into its components

key, n = *pk*

# Generate the plaintext based on the ciphertext and key using a^b mod m

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

# Return the array of bytes as a string

*return* ''.join(plain)

**Вывод**

* ходе написания лабораторной работы были изучены алгоритмы шифрования и дешифрования RSA, а также написаны их программные реализации. Были получены навыки усложнения и увеличения криптостойкости алгоритма RSA, а также изучены модификации и режимы работы алгоритма RSA.