Министерство образования и науки Российской Федерации

ФГБОУ «Петрозаводский государственный университет»

Институт математики и информационных технологий

Отчет по лабораторной работе

Разработка программы шифрования и расшифрования алгоритмом RSA

**Выполнили:**

**студент 3 курса 22305 Д. И. Кнауб**

**студентка 3 курса 22305 Д. С. Ефимова**

**Преподаватель по лабораторной**

**работе: д.т.н., доцент,**

**профессор кафедры ПМиК**

**Р. В. Воронов**

подпись принявшего работу

**Петрозаводск – 2022**

**Оглавление**

[**Формулировка задания** 3](#_Toc96465172)

[**Описание метода решения** 4](#_Toc96465173)

[Шаг 1) Генерация ключей 4](#_Toc96465174)

[Тест Рабина-Миллера 4](#_Toc96465175)

[Возведение в степень по модулю в кольце 5](#_Toc96465176)

[Алгоритм Евклида 5](#_Toc96465177)

[Расширенный алгоритм Евклида 6](#_Toc96465178)

[Шаг 2) Шифрование 6](#_Toc96465179)

[Шаг 3) Расшифровка 6](#_Toc96465180)

[Китайскаая теорема об остатках 7](#_Toc96465181)

[**Примеры кода программы** 9](#_Toc96465182)

[**Тестовые данные** 12](#_Toc96465183)

# **Формулировка задания**

**Вариант 1.6.**

Напишите программу шифрования и расшифрования алгоритмом RSA. Рекомендуется использовать библиотеку для работы с длинными числами. В случае применения этой библиотеки разрешается использовать функции сложения, вычитания, умножения, целочисленного деления, вычисления остатка от деления. Функции возведения числа в степень, нахождения наибольшего общего делителя, обратного элемента в мультипликативной группе вычетов, генерации простого числа реализовать самостоятельно. Для ускорения вычислений использовать китайскую теорему об остатках.

Выполняемые функции программы:

1) генерация пары открытый/закрытый ключ, при этом число e задается пользователем;

2) шифрование данных (целого числа);

3) расшифрование шифртекста (целого числа).

# **Описание метода решения**

Алгоритм RSA является асимметричным алгоритмом, который используется для приложений безопасности в информационном обществе, включая аутентификацию по идентификатору и электронную транзакцию в сценариях электронных транзакций.

## ***Шаг 1)*** Генерация ключей

Принцип алгоритма **генерации ключей** (публичного и приватного) гласит, что он должен сначала сгенерировать два больших простых числа p и q, вычислив модуль . Затем он вычислит функцию Эйлера и получает (в случае метода решения поставленной задачи) число е – открытую экспоненту. Данное число должно быть: простым, меньше f(n) и взаимно простым с f(n), то есть НОД(f(n), e) = 1. Теперь пара чисел {e, n} – открытый ключ, который необходим для зашифровки сообщения. Позже необходимо вычислить число d, обратное е по модулю f(n), то есть . Пара чисел {d, n} – секретный ключ, который нужен для расшифровки. Безопасность RSA зависит от сложности разложения большого целого числа n.

Быстрое создание больших простых чисел необходимо для RSA. В методе решения используется следующий подход к генерации больших простых чисел: случайным образом генерируется большое целое число, а затем выполняется тест этого числа на простоту, который представляет из себя вероятностный метод теста Рабина-Миллера.

Тест Рабина-Миллера (Листинг 1) позволяет эффективно определять, является ли данное число составным, однако, с его помощью нельзя строго доказать простоту числа. На вход тесту подается большое целое число n, которое каждый раз генерируется функцией random.randit(диапозон от, до) до тех пор, пока не найдется простое число, а также подается число итераций k. Каждая итерация Рабина-Миллера уменьшает вероятность того, что число является составным в ¼ раза. Оптимальным количеством итераций является число 40, которое и используется в алгоритме. Сразу же проверяем: если число четное, то оно составное.

Число n – 1 однозначно представляется в виде Это можно сделать последовательным делением n – 1 на 2.

Далее повторяем цикл k раз: выбирается случайное целое число а в отрезке [2, n – 1]. Данное число называется свидетелем простоты числа n, если:

* , то продолжаем генерировать новое число а, пока не пройдем все k проверок. x вычисляется с помощью возведения в степень по модулю.

Иначе, повторить цикл s – 1 раз (внутренний цикл): если , то перейти на следующую итерацию внешнего цикла. Иначе вернуть False.

Если все k итераций пройдены, то вернуть True (простое).

Обычно при реализации необходимо возводить число в очень большую степень по модулю, что при использовании функции python pow(a,b,c) занимает большое количество времени. Поэтому была реализована функция возведения в степень по модулю в кольце (Листинг 2).

Возведение в степень по модулю в кольце выглядит следующим образом: пока степень p больше квадрата, то

* Если степень нечетная, то получение четной степени происходит за счет выноса множителя, перемножая с уже имеющимся и находя остаток при делении этого произведения. Сама степень p понижается на 1.
* Если степень четная, то понижаем ее, возводя число в квадрат и находя остаток от деления, а степень уменьшаем в 2 раза. Это возможно сделать, так как по модулю есть большое количество эквивалентных чисел данному.

В завершении (когда степень сократится до квадрата или меньше) оставшийся элемент, возведенный в получившуюся степень по модулю, перемножается с полученным результатом, поделенное на модуль.

Так как число е (открытую экспоненты) вводит сам пользователь, было необходимо сделать проверки: простое, меньше f(n) и взаимно простое с f(n). Для проверки на простоту используется тест Рабина-Миллера (Листинг 1). Для проверки взаимно простых чисел используется алгоритм Евклида (Листинг 3). Числа являются взаимно простыми, если их наибольший общий делитель равен 1.

Алгоритм Евклида — это алгоритм, который используется для нахождения наибольшего делителя двух целых чисел.

Идея такой реализации достаточна проста, алгоритм уменьшает числа до тех пор, пока одно из них не станет равно нулю, а другое искомому делителю. Для этого используется цикл while, в котором большее число делится на меньшее и становится равным остатку от деления. Таким образом, функция вернёт наибольший общий делитель (НОД) её двух аргументов.

Так как одно из чисел всегда становится равным нулю, то функция всегда будет возвращать делитель, благодаря логическому оператору или, который используется вместе с return.

Таким образом, на первом шаге получена пара чисел {e, n}, которая является открытым ключом и отправляется для зашифровки. Но также необходимо получить закрытый ключ. Нужно вычислить число d, обратное е по модулю f(n). То есть остаток от деления по модулю f(n) должен быть равен 1: .

Расширенный алгоритм Евклида используется для нахождения числа d (Листинг 4), который также применяется для нахождения обратного элемента в кольце по модулю. Работа расширенного алгоритма Евклида заключается в следующем:

Пусть функция egcd получает на вход числа a и b и возвращает кортеж из трех чисел g, x, y, где g - наибольший общий делитель a и b, а x и y - такие целые числа, что ax+by = g.

Условием окончания рекурсии является a = 0. В этом случае g = b, x = 1, y = 0. Если же a != 0, то вызовем функцию рекурсивно для чисел a и b mod a  и получим ответ для исходных чисел.

*Для нахождения обратного элемента в кольце по модулю:*

Элемент a называют обратным элементом к элементу b, если a\*b = 1 (mod n), то есть число a\*b дает остаток 1 при делении на n. Для нахождения обратного элемента можно использовать расширенный алгоритм Евклида (описанный выше).

Заметим, что для того, чтобы a\*b давало остаток 1 при делении на n необходимо, чтобы b и n были взаимно простыми. То есть НОД для чисел b и n равен 1.

Применив к числам b и n расширенный алгоритм Евклида, найдем такие числа x и y, что b\*x + n\*y = 1. Тогда (поскольку n\*y делится на n) b\*x будет давать остаток 1 при делении на n, то есть x и будет обратным элементом к b в кольце вычетов по модулю n.

## Шаг 2) Шифрование

Шифрование сообщения М выполняется по следующему алгоритму:

Сообщение возводитеся в степень е по модулю n. То есть с помощью возведения в степень по модулю в кольце. Сообщение M не должно быть больше n = pq.

## Шаг 3) Расшифровка

Имея на руках закрытый ключ, можно расшифровать полученное сообщение. Это не получится сделать открытым ключом.

В криптосистеме RSA при расшифровки требуется вычислить , n=pq, где p, q – большие простые числа, d – закрытый ключ, который также может быть большим числом. Тогда данная операция может оказаться очень затратной по времени и памяти. Использование китайской теоремы об остатках позволяет ускорить вычисления для расшифровки сообщения.

Китайская теорема об остатках утверждает, что можно восстановить целое число по множеству его остатков от деления на числа из некоторого набора попарно взаимно простых чисел.

То есть для любого (m1, m2), . Должен быть уникальный m, , при чем

m1 = m mod p

m2 = m mod q

Другими словами, при заданном a (m1, m2) должно существовать только m, удовлетворяющее вышеуказанному уравнению. Таким образом, процесс расшифровки , можно разложить на p также как и , а затем вычислить m. Но показатель степени d все еще относительно велик, и операция по прежнему является дорогостоящей, поэтому необходимо понизить индекс. может быть уменьшено до . Аналогично и со вторым уравнением. Заранее можно рассчитать . Но их вычисление также может быть проще через вычисление обратного значения е к p-1, q-1 соответственно.

Итак, алгоритм заключается в следующем (Листинг 5):

* d является обратным элементом e к f(n), теперь нужны два других обратных элемента (для (p-1) и (q-1)):
  + 1: вычислить dp так, чтобы dp\*e = 1 mod(p-1)
  + 2: вычислить dq так, чтобы dq\*e = 1 mod(q-1)
* Кроме того, необходим третий элемент, который является обратным q к p.
  + 3: вычислить qInv так, чтобы qInv \* q = 1 mod p

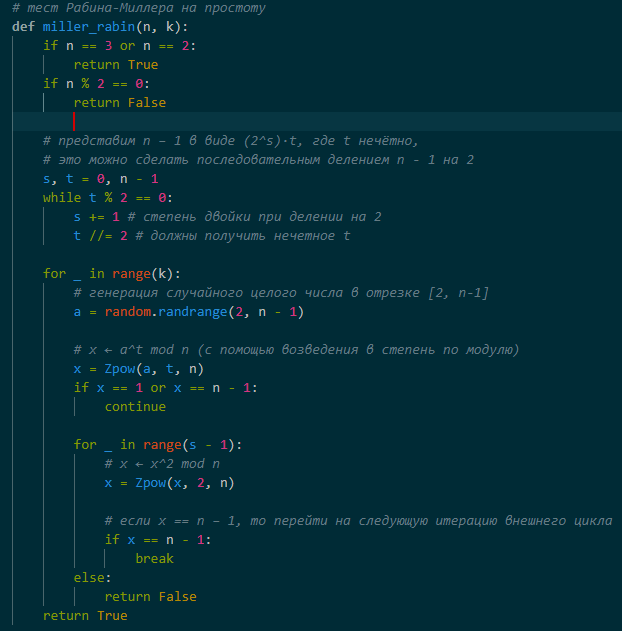
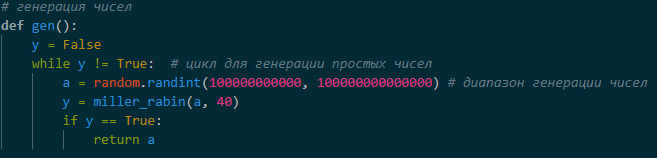
Все числа dp, dq, qInv являются частью закрытого ключа, поэтому скрытые.

В итоге m – расшифрованное сообщение.

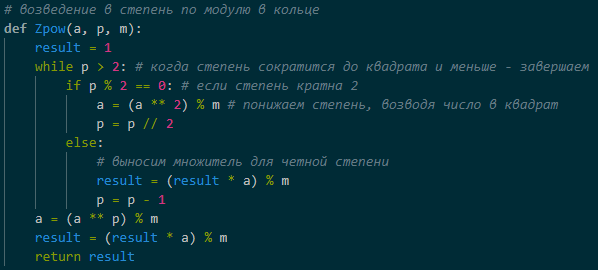
Работа программы проверяется тестами, результат которых можно увидеть в конце документа в главе Тестовые данные.

# **Примеры кода программы**

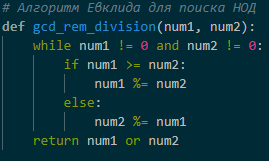
Листинг 1. Генерация чисел и проверка их на простоту через тест Рабина-Миллера

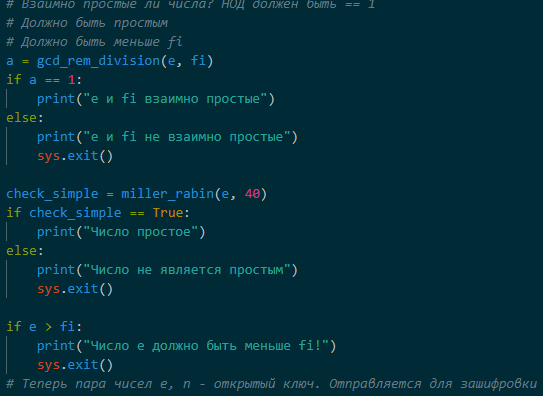


Листинг 2. Функция возведения в степень по модулю в кольце

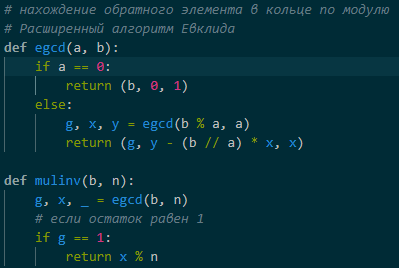


Листинг 3. Алгоритм Евклида для поиска наибольшего общего делителя (взаимно простых чисел в случае данной программы)

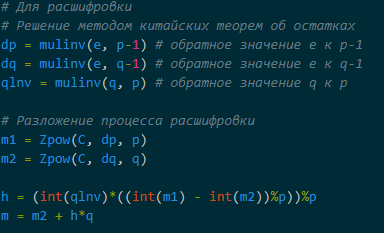




Листинг 4. Расширенный алгоритм Евклида для нахождения обратного элемента по модулю



Листинг 5. Расшифровка сообщения с помощью китайской теоремы об остатках



# **Тестовые данные**

Проверка работы алгоритма проводилась вручную на тестовых данных (результаты 10 тестов приведены ниже). Для проверки также было засечено время выполнения программы. Проанализировав полученные результаты, можно сказать, что алгоритм работает безошибочно как с малыми, так и с большими числами, а скорость его выполнения невысока.

Тест 1.

|  |  |
| --- | --- |
| P | 61 |
| q | 439 |
| n | 26779 |
| fi | 26280 |
| e | 11 |
| d | 23891 |
| Сообщение для шифрования | 1000 |
| Зашифрованное сообщение | 19313 |
| Расшифрованное сообщение | 1000 |
| Время выполнения | 0.0002751000001808279 |

Тест 2.

|  |  |
| --- | --- |
| P | 65981 |
| q | 75037 |
| n | 4951016297 |
| fi | 4950875280 |
| e | 11 |
| d | 2994643099 |
| Сообщение для шифрования | 46564 |
| Зашифрованное сообщение | 546192419 |
| Расшифрованное сообщение | 46564 |
| Время выполнения | 0.0007101999999576947 |

Тест 3.

|  |  |
| --- | --- |
| P | 4348471 |
| q | 775627 |
| n | 3372791516317 |
| fi | 3372786392220 |
| e | 1307 |
| d | 3019250251643 |
| Сообщение для шифрования | 8989678768 |
| Зашифрованное сообщение | 2209270861983 |
| Расшифрованное сообщение | 8989678768 |
| Время выполнения | 0.001109099999666796 |

Тест 4.

|  |  |
| --- | --- |
| P | 178958993 |
| q | 465684559 |
| n | 83338439734289087 |
| fi | 83338439089645536 |
| e | 7 |
| d | 11905491298520791 |
| Сообщение для шифрования | 8989678768 |
| Зашифрованное сообщение | 81167063641171147 |
| Расшифрованное сообщение | 8989678768 |
| Время выполнения | 0.0010530000004109752 |

Тест 5.

|  |  |
| --- | --- |
| P | 73239322063 |
| q | 89527475621 |
| n | 6556931620493799926123 |
| fi | 6556931620331033128440 |
| e | 2069 |
| d | 2944122511013789162069 |
| Сообщение для шифрования | 323665786767352343245 |
| Зашифрованное сообщение | 5864946314692614600247 |
| Расшифрованное сообщение | 323665786767352343245 |
| Время выполнения | 0.0034290999997210747 |

Тест 6.

|  |  |
| --- | --- |
| P | 7582106454941 |
| q | 6817147546159 |
| n | 51688338414017352651121619 |
| fi | 51688338414002953397120520 |
| e | 3529 |
| d | 16492226991829789040849449 |
| Сообщение для шифрования | 324537686563464572 |
| Зашифрованное сообщение | 917283248079369947130561 |
| Расшифрованное сообщение | 324537686563464572 |
| Время выполнения | 0.005864400000064052 |

Тест 7.

|  |  |
| --- | --- |
| P | 302091110247653 |
| q | 565842159119983 |
| n | 170935886073484795883671149899 |
| fi | 170935886073483927950401782264 |
| e | 2633 |
| d | 19865697356052442822948327145 |
| Сообщение для шифрования | 433245376865634645722 |
| Зашифрованное сообщение | 63288233721606106782587507383 |
| Расшифрованное сообщение | 433245376865634645722 |
| Время выполнения | 0.005296600000292528 |

Тест 8.

|  |  |
| --- | --- |
| P | 60096268282530938449 |
| q | 41149114399963176311 |
| n | 2472908218568744141483963082189675881639 |
| fi | 2472908218568744141382717699507181766880 |
| e | 463 |
| d | 662290754001132340240727850407970926767 |
| Сообщение для шифрования | 537686563464572 |
| Зашифрованное сообщение | 2125591125907350695011275136555714854779 |
| Расшифрованное сообщение | 537686563464572 |
| Время выполнения | 0.0034356000001025677 |

Тест 9.

|  |  |
| --- | --- |
| P | 354063316038356244496417 |
| q | 586194218847977226553181 |
| n | 207549868967828725506421704154169400286014452477 |
| fi | 207549868967828725506420763896634513952543402880 |
| e | 53 |
| d | 74404670007334826124943292717661429530157068957 |
| Сообщение для шифрования | 8654853768656346457248478 |
| Зашифрованное сообщение | 147918397972845928149566805445822477810901526801 |
| Расшифрованное сообщение | 8654853768656346457248478 |
| Время выполнения | 0.003410299999814015 |

Тест 10.

|  |  |
| --- | --- |
| P | 245905276517613962936737 |
| q | 942026725792433069123507 |
| n | 231649342492970759304083226490396701476280576659 |
| fi | 231649342492970759304082038558394391429248516416 |
| e | 35317 |
| d | 24104859802974984864017371002692736883157921053 |
| Сообщение для шифрования | 68646548537686563464572484785646 |
| Зашифрованное сообщение | 69282699953521617605838018099203754069125906629 |
| Расшифрованное сообщение | 68646548537686563464572484785646 |
| Время выполнения | 0.2890957999998136 |