1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский политехнический университет Петра Великого
3. —
4. Институт компьютерных наук и технологий
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ОТЧЕТ**

**по лабораторной работе № 2**

# Тесты на простоту

1. по дисциплине «Теоретико-числовые методы в криптографии»
2. Вариант 36
3. Выполнил
4. студент гр. 33508/3 Проценко Е.Г.

\_\_\_\_\_\_\_\_\_\_

Руководитель Павленко Е.Ю.

\_\_\_\_\_\_\_\_\_\_

1. Санкт-Петербург
2. 2016

1. Содержание

[1 Цель работы 3](#_Toc466252134)

[2 Теоретические сведения 4](#_Toc466252135)

[2.1 Тест Ферма 5](#_Toc466252136)

[2.2 Тест Соловэя-Штрассена 5](#_Toc466252137)

[2.3 Тест Миллера-Рабина 5](#_Toc466252138)

[3 Результаты работы 6](#_Toc466252139)

[3.1 Исследование теста Ферма 6](#_Toc466252140)

[3.2 Исследование теста Соловэя-Штрассена 8](#_Toc466252141)

[3.3 Исследование теста Миллера-Рабина 10](#_Toc466252142)

[3.4 Работа алгоритмов с числами Кармайкла 12](#_Toc466252143)

[4 Вывод 13](#_Toc466252144)

[Список используемых источников 14](#_Toc466252145)

[Приложение А 15](#_Toc466252146)

# Цель работы

Каждое из указанных чисел, согласно варианту, проверить на простоту с помощью тестов: Ферма, Соловэя-Штрассена, Рабина-Миллера.

Числа, которые нужно проверить на простоту (вариант 36):

1)

2)

3)

4) = 208837315517631047581197619041969957205989296812 00284165729913746212377703667653

# Теоретические сведения

Тесты на простоту – это алгоритмы, которые определяют является ли число простым.

Простые числа широко применяются в области защиты информации. Например, в ассиметричном алгоритме шифрования RSA. В соответствии с ГОСТ длина ключа должны быть не менее 254 бит. Для таких больших чисел вопрос определения простоты является крайне сложным. Определение простоты числа необходимо при взломе информации, зашифрованной или подписанной с использованием RSA. Для вскрытия такого сообщения необходимо уметь разлагать число на два простых сомножителя, что при больших размерах исходного числа является нетривиальной задачей [1].

Для определения простоты числа будут рассмотрены следующие алгоритмы:

1. тест Ферма;
2. тест Соловэя-Штрассена;
3. тест Миллера-Рабина.

Данные алгоритмы являются вероятностными. Для каждой итерации цикла генерируется случайное a, . Если число n не проходит тест по основанию а, то n – составное, в противном случае считается, что n, вероятно, простое.

После t независимых тестов, вероятность того, что составное n будет t раз объявлено простым, не превосходит .

## Тест Ферма

Этот тест, определяющий простоту натурального числа n, основан на малой теореме Ферма:

Минусом теста Ферма является то, что он не справляется с **числами Кармайкла.**

Пример. – число Кармайкла. Для всех оснований, не делящихся на , тест Ферма скажет, что n, вероятно, простое.

**Критерий Корселта.** Составное целое нечетное n является числом Кармайкла тогда и только тогда, когда:

1. n – свободно от квадратов;
2. для каждого простого делителя p числа n, . Т.е. число должно делиться на .

## Тест Соловэя-Штрассена

В основе этого алгоритма лежит Критерий Эйлера.

**Критерий Эйлера.** Целое нечетное n является простым тогда и только тогда, когда для любого целого a, , взаимно простого с n, выполняется сравнение .

В отличие от теста Ферма данный алгоритм распознает числа Кармайкла как составные, т.е. не существует составных чисел, которые были бы эйлерово псевдо простых по всем основаниям а.

## Тест Миллера-Рабина

Данный алгоритм основан на малой теореме Ферма и наблюдении, что , где n и r – нечетные.

Тогда в крайнем произведении хотя бы одна из скобок делиться на n.

# Результаты работы

Исходный код тестов находится в Приложении А.

## Исследование теста Ферма

Таблица 1 – Исследование теста Ферма с заданными числами

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число\Итерация | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | + | + | + | + | + | + | + | + | + | + |
| 2 | + | + | + | + | + | + | + | + | + | + |
| 3 | - | - | - | - | - | - | - | - | - | - |
| 4 | - | - | - | - | - | - | - | - | - | - |

1. Первое число – возможно, простое. Пять оснований для которых выполняется условие простоты:

не сравнимы с 1 под модулю . Следовательно – возможно, простое.

1. Второе число – возможно, простое. Пять оснований для которых выполняется условие простоты:

* a1 = 5380439913806749850691471580549742705713
* a2 = 1682303010080979402301227348997807943824
* a3 = 2403743373507505850567292877434704751057
* a4 = 1819614830958016460427438959835470096769
* a5 = 3020982571221750269581924548808444220478

не сравнимы с 1 под модулю . Следовательно – возможно, простое.

1. Третье число составное. Основание, для которого нарушается условие простоты:

.

Следовательно – составное.

1. Четвертое число составное. Основание, для которого нарушается условие простоты:

a = 13387925443440583780641409053960210099385893357… …9956939218145300640531418433620

.

Следовательно – составное.

## Исследование теста Соловэя-Штрассена

Таблица 2 – Исследование теста Соловэя-Штрассена с заданными числами

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число\Итерация | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | + | + | + | + | + | + | + | + | + | + |
| 2 | + | + | + | + | + | + | + | + | + | + |
| 3 | - | - | - | - | - | - | - | - | - | - |
| 4 | - | - | - | - | - | - | - | - | - | - |

1. Первое число – возможно, простое. Пять оснований для которых выполняется условие простоты:

Рассмотрим a1. . . Эти числа сравнимы по модулю . Аналогично для других оснований . Следовательно – возможно, простое.

1. Второе число – возможно, простое. Пять оснований для которых выполняется условие простоты:

* a1 = 1779945819802249938028872481064888172736
* a2 = 224161832077866736915664046096464820099
* a3 = 2155606670211111039812473449786282920956
* a4 = 4232079281195369042430401461464112833175
* a5 = 224405305317919455691007375724119551421

Рассмотрим a1. . . Эти числа сравнимы по модулю . Аналогично для других оснований . Следовательно – возможно, простое.

1. Третье число составное. Основание, для которого нарушается условие простоты:

.

Т.к. результат не равен ни 1, ни -1, видно, что оно не сравнимо с символом Якоби. Поэтому, составное.

1. Четвертое число составное. Основание, для которого нарушается условие простоты:

a = 74028362131226816300489192782161015299305022092… …07585382438491823359597582855959

11992587811704829656465108403995464086904889630542… …434025374124725519849255311297 (mod )

Т.к. результат не равен ни 1, ни -1, видно, что оно не сравнимо с символом Якоби. Поэтому, составное.

## Исследование теста Миллера-Рабина

Таблица 3 – Исследование теста Миллера-Рабина с заданными числами

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число\Итерация | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | + | + | + | + | + | + | + | + | + | + |
| 2 | + | + | + | + | + | + | + | + | + | + |
| 3 | - | - | - | - | - | - | - | - | - | - |
| 4 | - | - | - | - | - | - | - | - | - | - |

1. Первое число – возможно, простое. Пять оснований для которых выполняется условие простоты:

Рассмотрим a1.. .

. .

Аналогично для других оснований .

Поэтому - возможно, простое.

1. Второе число – возможно, простое. Пять оснований для которых выполняется условие простоты:

* a1 = 933290639503376502782580325394231910474
* a2 = 355358987592595195988625588940127423626
* a3 = 6354987944228604874281716911506683321423
* a4 = 4374632624921956609745459265049412267335
* a5 = 926301108657023582526836007936677272940

Рассмотрим a1.. .

.

. Соответствующая скобка дает 0.

Соответственно, выражение сравнимо с 0 по модулю

Аналогично для других оснований .

Поэтому - возможно, простое.

1. Третье число составное. Основание, для которого нарушается условие простоты:

При разложении ни одна из скобок не будет давать 0. Поэтому, составное.

1. Четвертое число составное. Основание, для которого нарушается условие простоты:

a = 879210633670665530290409080026185055705553709879… …4299282992630970247353457208047

При разложении ни одна из скобок не будет давать 0. Поэтому, составное.

## Работа алгоритмов с числами Кармайкла

Рассмотри следующие 2 числа Кармайкла [2]:

4754868377601046732119933839981363081972014948522510826417784001 (*32 десятичных разряда*)

1334733877147062382486934807105197899496002201113849920496510541601 (*33 десятичных разряда*)

Рассмотрим 10000 тысяч итераций каждого алгоритма с каждым из чисел.

Таблица 4 – Кол-во ошибок с числами Кармайкла

|  |  |  |  |
| --- | --- | --- | --- |
| Число Кармайкла\Тест | Ферма | Соловэя-Штрассена | Миллера-Рабина |
|  | 60402 | 15045 | 0 |
|  | 60140 | 15275 | 0 |

Таблица 4 – Время работы с числами Кармайкла в секундах

|  |  |  |  |
| --- | --- | --- | --- |
| Число Кармайкла\Тест | Ферма | Соловэя-Штрассена | Миллера-Рабина |
|  | 34.6146511878 | 41.4764507752 | 36.6120775628 |
|  | 35.6139878956 | 40.1062298147 | 34.5262695352 |

# Вывод

Я познакомился с тестами на простоту, и зачем они вообще нужны.

Было проверено экспериментально, что числа Кармайкла, считаются, возможно, простыми для теста Ферма для всех оснований взаимно простых с основанием n, где .

Тест Соловэя-Штрассена умеет распознавать числа Кармайкла, но не для всех оснований, а только для , где .

Тест Миллера-Рабина является самым точным, т.к. ни одно число Кармайкла не прошло тест. Так же данный тест показал не худший показатель по времени работы, что в итоге делает тест Миллера-Рабина лучшим из рассмотренных.

# Список используемых источников

1. <https://ru.wikipedia.org/wiki/RSA>
2. <https://arxiv.org/pdf/math/0604376v1.pdf>
3. «Теоретико-числовые методы в криптографии» Е.Б.Маховенко. 2006

# Приложение А

**import** **random**

**def** **fermat\_primality\_test**(n):

"""

Probabilistic test to determine whether a number is a probable prime.

input:

n >= 5, n is not even

output:

False - n is composite

True - n is probable prime

"""

# step 1

a = random.randint(**2**, n - **2**)

# step 2

r = pow(a, n - **1**, n)

# step 3

**return** [**True** **if** r == **1** **else** **False**, a]

**def** **is\_even**(x):

**return** x & **0x1** == **0**

**def** **jacobi\_symbol**(a, n):

"""

input:

n >= 3, n is not even

a - integer, 0 <= a < n

output:

jacobi symbol result

"""

# step 1

g = **1**

**while** **True**:

# step 2

**if** a == **0**:

**return** **0**

# step 3

**if** a == **1**:

**return** g

# step 4

k = **0**

a1 = a

**while** is\_even(a1):

a1 >>= **1**

k += **1**

# step 5

**if** is\_even(k):

s = **1**

**else**:

**if** n % **8** **in** [**1**, **7**]:

s = **1**

**if** n % **8** **in** [**3**, **5**]:

s = -**1**

# step 6

**if** a1 == **1**:

**return** g \* s

# step 7

**if** n % **4** == **3** **and** a1 % **4** == **3**:

s = -s

# step 8

a = n % a1

n = a1

g \*= s

**def** **solovay\_strassen\_primality\_test**(n):

"""

Probabilistic test to determine whether a number is a probable prime.

input:

n >= 5, n is not even

output:

False - n is composite

True - n is probable prime

"""

# step 1

a = random.randint(**2**, n-**2**)

# step 2

r = pow(a, (n - **1**) >> **1**, n)

# step 3

**if** r != **1** **and** r != n - **1**:

**return** [**False**, a]

# step 4

s = jacobi\_symbol(a, n)

# step 5

**if** pow(r - s, **1**, n) != **0**:

**return** [**False**, a]

**else**:

**return** [**True**, a]

**def** **miller\_rabin\_primality\_test**(n):

"""

Probabilistic test to determine whether a number is a probable prime.

input:

n >= 5, n is not even

output:

False - n is composite

True - n is probable prime

"""

# step 1

s = **0**

r = n - **1**

**while** is\_even(r):

r >>= **1**

s += **1**

# step 2

a = random.randint(**2**, n - **2**)

# a = 1365252518031410025084500935292129408721

# step 3

y = pow(a, r, n)

# step 4

**if** y **not** **in** [**1**, n - **1**]:

# step 4.1

j = **1**

# step 4.2

# if j <= s - 1 and y != n - 1:

**while** j <= s - **1** **and** y != n - **1**:

# step 4.2.1

y = pow(y, **2**, n)

# step 4.2.2

**if** y == **1**:

# if y not in [1, n - 1]:

**return** [**False**, a]

#step 4.2.3

j += **1**

# step 4.3

**if** y != n - **1**:

**return** [**False**, a]

# step 5

**return** [**True**, a]

numbers = [

**30995520580246599847**,

**6444041923397078743358954620655411924689**,

**476921747832748874793388092809647376573**,

**20883731551763104758119761904196995720598929681200284165729913746212377703667653**,

]