Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
—
Институт компьютерных наук и технологий
Кафедра «Информационная безопасность компьютерных систем»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

ТЕСТЫ НА ПРОСТОТУ
по дисциплине «Теоретико-числовые методы в криптографии»
Вариант 36

Выполнил
студент гр. 33508/3 Проценко Е.Г.
____

Руководитель Павленко Е.Ю.
____

Санкт-Петербург
2016

СОДЕРЖАНИЕ

1 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) $n_1 = 3099552058024659 9847$

2) $n_2 = 644404192339707874335895462065541 1924689$

3) $n_3 = 476921747832748874793388092809647376573$

4) n_4 = 2088373155176310475811976190419699572059892968120028416572991374621237770366765 3

2 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, $1 < a < n$. Если число n не проходит тест по основанию a, 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 $\frac{1}{2}$.

2.1 Тест Ферма

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

$$a^{p-1} \equiv 1\ (mod\ p)$$

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

Пример. $n = 561 = 3 * 7 * 11$ – число Кармайкла. Для всех оснований, не делящихся на $3, 11, 17$, 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, $n - 1 \vdots p - 1$. Т.е. число $n - 1$ должно делиться на НОК$(p_1 - 1, \dots, p_s - 1)$.

2.2 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, $1 \leq a \leq n - 1$, взаимно простого с n, выполняется сравнение $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) (mod\ n)$.

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 a.

2.3 Тест Миллера-Рабина

Данный алгоритм основан на малой теореме Ферма и наблюдении, что $n - 1 = 2^k r$, где n и r – нечетные.

$$a^{n-1} - 1 = a^{2^k r} - 1 = \cdots = (a^r - 1)(a^r + 1)(a^{2r} + 1) \dots (a^{2^{k-1} r} + 1)$$

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

3 РЕЗУЛЬТАТЫ РАБОТЫ

Исходный код тестов находится в Приложении А.

3.1 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:
   - $a1 = 3716148314073877145$
   - $a2 = 27178755448348668490$
   - $a3 = 19806197039863831011$
   - $a4 = 1126305995068594587$
   - $a5 = 11291543688517841276$

   $a1^{n_1-1}, a2^{n_1-1}, a3^{n_1-1}, a4^{n_1-1}, a5^{n_1-1}$ не сравнимы с 1 под модулю n_1. Следовательно n_1 – возможно, простое.

2) Второе число – возможно, простое. Пять оснований для которых выполняется условие простоты:
   - a1 = 538043991380674985069147158054974270571З
   - a2 = 1682303010080979402301227348997807943824
   - a3 = 2403743373507505850567292877434704751057
   - a4 = 1819614830958016460427438959835470096769
   - a5 = 3020982571221750269581924548808444220478

$a1^{n_2-1}, a2^{n_2-1}, a3^{n_2-1}, a4^{n_2-1}, a5^{n_2-1}$ не сравнимы с 1 под модулю n_2. Следовательно n_2 – возможно, простое.

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

$$a = 2162409487823462573119518172350891155 78$$

$$a^{n_3-1} \equiv 1\ (mod\ n_3).$$

Следовательно n_3 – составное.

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

a = 13387925443440583780641409053960210099385893357…

…99569392181453006405314184336 20

$$a^{n_4-1} \equiv 1\ (mod\ n_4).$$

Следовательно n_4 – составное.

3.2 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:
    - $a1 = 1454331720921698655 0$
    - $a2 = 15592602064110225529$
    - $a3 = 22466346824751945797$
    - $a4 = 29413684178636097500$
    - $a5 = 14634991986784332298$

    Рассмотрим a1. $a1^{\frac{n_1-1}{2}} \equiv 1 \pmod{n_1}$. $\left(\frac{a}{n_1}\right) = 1$. Эти числа сравнимы по модулю n_1. Аналогично для других оснований a_i. Следовательно n_1 – возможно, простое.

2) Второе число – возможно, простое. Пять оснований для которых выполняется условие простоты:
    - a1 = 177994581980224993802887248106488817273 6
    - a2 = 22416183207786673691566404609646482009 9
    - a3 = 215560667021111039812473449786282920956
    - a4 = 423207928119536904243040146146411283317 5
    - a5 = 22440530531791945569100737572411955142 1

Рассмотрим a1. $a1^{\frac{n_2-1}{2}} \equiv -1 \pmod{n_2}$. $\left(\frac{a}{n_2}\right) \equiv -1 \pmod{n_2}$. Эти числа сравнимы по модулю n_2. Аналогично для других оснований a_i. Следовательно n_2 – возможно, простое.

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

$$a = 4122148680955545488507710625145665055588$$

$$a^{\frac{n_3-1}{2}} \equiv$$

$$1906867911134562191727634720745322260284 \pmod{n_3}.$$

Т.к. результат не равен ни 1, ни -1, видно, что оно не сравнимо с символом Якоби. Поэтому, n_3 составное.

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

a = 740283621312268163004891927821610152993050220092…

…07585382438491823359597582855959

$$a^{\frac{n_3-1}{2}} \equiv$$

11992587811704829656465108403995464086904889630542…

…434025374124725519849255311297 (mod n_4)

Т.к. результат не равен ни 1, ни -1, видно, что оно не сравнимо с символом Якоби. Поэтому, n_4 составное.

3.3 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:
   - $a1 = 1454331720921698655O$
   - $a2 = 706846811740289217 1$
   - $a3 = 427147393850011633 8$
   - $a4 = 2404040457883591895 3$
   - $a5 = 2083912142169957205 8$

   Рассмотрим a1. $n_1 = n^{2^s r}. s = 1.$

   $r = 1549776029012329992 3. a1^r = 1 \equiv 1\ (mod\ n_1).$

   Аналогично для других оснований a_i.

   Поэтому n_1- возможно, простое.

2) Второе число – возможно, простое. Пять оснований для которых выполняется условие простоты:
   - a1 = 9332906395033765027825803253942 31910474
   - a2 = 3553589875925951959886255889401 27423626
   - a3 = 6354987944228604874281716911506 683321423
   - a4 = 4374632624921956609745459265049 412267335
   - a5 = 9263011086570235825268360079366 77272940

   Рассмотрим a1. $n_1 = n^{2^s r}. s = 4.$

   $r = 40275262021231742145993466379096 3245293.$

$a1^{3r} \equiv -1\ (mod\ n_2)$. Соответствующая скобка дает 0.

Соответственно, выражение сравнимо с 0 по модулю n_2

Аналогично для других оснований a_i.

Поэтому n_2- возможно, простое.

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

$$a = 32758239757179562804301922641831260934$$

При разложении ни одна из скобок не будет давать 0. Поэтому, n_3 составное.

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

a = 87921063367066553029040908002618505570555370987 9…

…42992829926309702473534572080 47

При разложении ни одна из скобок не будет давать 0. Поэтому, n_4 составное.

3.4 Работа алгоритмов с числами Кармайкла

Рассмотри следующие 2 числа Кармайкла [2]:

$K_1 = 4754868377601046732119933839981363081972014948522510826417784001$ (*32 десятичных разряда*)

$K_2 = 13347338771470623824869348071051978994960022011138499204965105416 01$ (*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
K_1	60402	15045	0
K_2	60140	15275	0

Таблица 4 – Время работы с числами Кармайкла в секундах

Число Кармайкла\Тест	Ферма	Соловэя-Штрассена	Миллера-Рабина
K_1	34.6146511878	41.4764507752	36.6120775628
K_2	35.6139878956	40.1062298147	34.5262695352

4 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 a_i взаимно простых с основанием n, где $i = \varphi(n)$.

Тест Соловэя-Штрассена умеет распознавать числа Кармайкла, но не для всех оснований, а только для a_i, где $i = \frac{\varphi(n)}{4}$.

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
```

```python
    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,
    4769217478327488747933880928096473765 73,
    20883731551763104758119761904196995720598929681200284165729913746212377703667653,
]
```