НАО «Восточно-Казахстанский технический университет имени Д. Серикбаева»

Дисциплина «Теоретико-числовые методы в криптографии»

САМОСТОЯТЕЛЬНЫЙ ПРАКТИКУМ К ЛЕКЦИИ 12

Квадратные корни по простому модулю. Алгоритм Тонелли — Шенкса

Методические указания и задание к выполнению

Статус материала. Дополнительный самостоятельный практикум к лекции 12. В рабочей программе у практической темы 12 указаны «Убрать?» и 0 часов, поэтому этот материал не добавляет аудиторную практическую работу. Он продолжает практическую тему 11 и может использоваться для самостоятельного изучения или по решению преподавателя.

Цель: связать законы квадратичных символов с вычислением корней; освоить алгоритм Тонелли — Шенкса для любого нечётного простого модуля и проверить корректность его шагов.

Результат работы: письменные решения своего варианта, программа на Python, таблица промежуточных состояний алгоритма и проверка найденных корней. Предварительно изучаются методика и примеры разделов 1–5; индивидуальные задания приведены в разделе 6.

Средства: Python 3.8 или новее, любой текстовый редактор. Внешние пакеты не требуются. Все числа в вариантах малы, поэтому перебор можно использовать для независимого контроля ответа.

1 От символа к квадратному корню

При нечётном простом p символ Лежандра позволяет определить число корней сравнения $x^2 \equiv a \pmod p$: нулевой символ означает один корень 0; знак -1 — отсутствие корней; знак 1 — два противоположных ненулевых корня. Однако само значение символа не сообщает эти корни.

Если $p \equiv 3 \pmod 4$ и $(a/p) = 1$, один корень получается как $a^{(p+1)/4} \bmod p$. Для $p \equiv 1 \pmod 4$ показатель $(p+1)/4$ не является целым, и эту формулу применять нельзя. Алгоритм Тонелли — Шенкса устраняет данное ограничение.

Порядок решения. Сначала приводят a по модулю p и обрабатывают случаи 0 и невычета. Затем для ненулевого вычета запускают алгоритм извлечения корня, восстанавливают второй корень и проверяют оба возведением в квадрат. Подстановка необходима и для проверки ручных вычислений, и для проверки программы.

2 Методика вычисления символов

2.1 Лемма Гаусса и взаимность: образец оформления

Для вычисления $(11/7)$ берём остатки чисел $11i$, где $1 \le i \le 3$. Получается список 4,1,5. Порог $7/2$ превышают два остатка, поэтому $\nu = 2$ и символ равен $(-1)^2 = 1$.

Числа 7 и 11 оба имеют вид $4k + 3$. Следовательно, при их перестановке знак меняется: $(7/11) = -(11/7) = -1$.

Дополнительно проверим формулу через целые части:

$$S(11,7) = \lfloor 11/7 \rfloor + \lfloor 22/7 \rfloor + \lfloor 33/7 \rfloor = 8,$$

$$S(7,11) = \lfloor 7/11 \rfloor + \lfloor 14/11 \rfloor + \lfloor 21/11 \rfloor + \lfloor 28/11 \rfloor + \lfloor 35/11 \rfloor = 7.$$

Сумма 15 равна $((7-1)/2)((11-1)/2)$. Чётность первой суммы даёт знак 1, чётность второй — знак -1. Полное доказательство этой связи и иллюстрация подсчёта точек приведены в лекции 12.

2.2 Как интерпретировать символ Якоби

Для произведения различных нечётных простых $n = \ell v$ и взаимно простого с ним b имеем $(b/n) = (b/\ell)(b/v)$. Квадратный корень существует ровно при паре местных знаков (1,1). Пара $(-1,-1)$ также даёт положительный символ Якоби, но корня не даёт.

Пример. Для $n = 77$, $b = 6$ знаки $(6/7)$ и $(6/11)$ равны -1. Значит, $(6/77) = 1$, однако корней нет. Всего обратимых классов 60, из них 30 имеют символ Якоби 1 и 15 являются квадратами. Последние числа следуют из китайской теоремы об остатках: четыре пары знаков встречаются по 15 раз.

3 Алгоритм Тонелли — Шенкса

3.1 Подготовка и начальные значения

Пусть $a \not\equiv 0 \pmod p$ и $(a/p) = 1$. Разложим

$$p - 1 = q2^s, \qquad q \text{ нечётно}, \quad s \geq 1.$$

Буква q здесь обозначает нечётную часть числа $p - 1$ и **не обязана обозначать простое число**. Выберем невычет z, то есть $(z/p) = -1$. Для совпадения промежуточных состояний с ответами преподавателя во всех вариантах выбирается наименьшее такое $z \geq 2$.

Зададим начальное состояние:

$$m = s, \quad c = z^q \bmod p, \quad t = a^q \bmod p, \quad R = a^{(q+1)/2} \bmod p.$$

Значение R пока не обязательно является корнем. Оно удовлетворяет равенству $R^2 \equiv at \pmod p$. Переменная t показывает множитель, который ещё требуется устранить.

3.2 Один шаг алгоритма

Если $t = 1$, вычисление закончено: R — искомый корень. Иначе последовательно возводим t в квадрат и находим наименьшее целое $i \geq 1$, при котором $t^{2^i} \equiv 1 \pmod p$. Затем вычисляем

$$b = c^{2^{m-i-1}} \bmod p$$

и обновляем состояние:

$$R \leftarrow Rb, \qquad t \leftarrow tb^2, \qquad c \leftarrow b^2, \qquad m \leftarrow i.$$

Все произведения берутся по модулю p. В формуле для b используются **старые** значения c, m. После получения b величины R, t, c можно вычислять в указанном порядке. Затем повторяется проверка $t = 1$.

3.3 Почему алгоритм работает

В начале и после каждого шага выполняются три утверждения — **инварианты**:

$$R^2 \equiv at \pmod p, \qquad \operatorname{ord}_p(c) = 2^m, \qquad t^{2^{m-1}} \equiv 1 \pmod p.$$

Начальное равенство для R следует из $R^2 = a^{q+1} = at$. По критерию Эйлера $z^{(p-1)/2} = -1$, поэтому $c^{2^{s-1}} = -1$, а $c^{2^s} = 1$. Значит, порядок c точно равен 2^s. Для вычета a критерий Эйлера даёт $t^{2^{s-1}} = 1$.

Если $t \neq 1$, его порядок равен 2^i для некоторого $1 \leq i < m$. Поэтому показатель $m - i - 1$ неотрицателен. Выбранное b^2 имеет порядок 2^i, а его степень 2^{i-1} равна -1. По минимальности i такая же степень старого t равна -1. Следовательно,

$$(tb^2)^{2^{i-1}} = (-1)(-1) = 1.$$

Это доказывает третий инвариант для нового $m = i$. Второй следует из порядка b^2. Первый сохраняется, поскольку $(Rb)^2 = R^2b^2 = atb^2$. При этом m строго уменьшается. Бесконечный цикл невозможен; при $m = 1$ третий инвариант уже даёт $t = 1$.

На выходе $R^2 \equiv a$, и все корни — R и $p - R$. Для ненулевого a они различны. Если $s = 1$, цикл сразу пропускается и начальное значение R совпадает с формулой для $p \equiv 3 \pmod 4$.

После выбора невычета число модульных умножений в основном алгоритме оценивается как $O(\log p + s^2)$. Поиск z перебором и проверка простоты модуля в эту оценку не включены. В учебном коде они выделены в отдельные действия; число шагов зависит от входных данных.

4 Полностью разобранные примеры

4.1 Извлечение корня из 5 по модулю 41

Разложение: $40 = 5 \cdot 2^3$, поэтому $q = 5, s = 3$. Число 5 — вычет: $(5/41) = (41/5) = 1$. Число 2 также является вычетом, а наименьший невычет — $z = 3$. Начальные значения:

$$c = 3^5 \bmod 41 = 38, \quad t = 5^5 \bmod 41 = 9, \quad R = 5^3 \bmod 41 = 2.$$

Первый шаг. $9^2 \equiv 40 \neq 1$, но $9^4 \equiv 1$. Значит, $i = 2$. Получаем $b = 38^{2^0} = 38$, $b^2 \equiv 9$. Новое состояние: $R = 2 \cdot 38 \bmod 41 = 35$, $t = 9 \cdot 9 \bmod 41 = 40$, $c = 9, m = 2$.

Второй шаг. $40^2 \equiv 1$, поэтому $i = 1, b = 9$. Получаем $R = 35 \cdot 9 \bmod 41 = 28$, $t = 40 \cdot 40 \bmod 41 = 1$, $c = 40, m = 1$. Алгоритм завершён.

Состояние	m	c	t	R	i шага	b шага
Начало	3	38	9	2	—	—
После шага 1	2	9	40	35	2	38
После шага 2	1	40	1	28	1	9

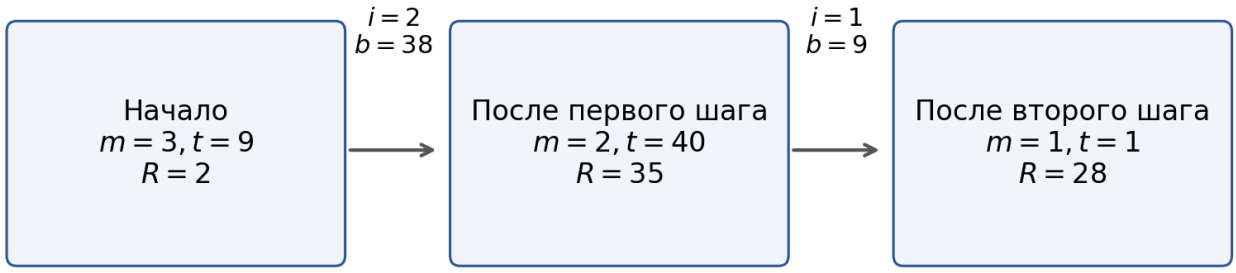


Рисунок 1 — Уменьшение параметра m в разобранном примере

Итак, корни равны 28 и $41 - 28 = 13$. Проверка: $13^2 = 169 = 4 \cdot 41 + 5$; $28^2 = 784 = 19 \cdot 41 + 5$.

Обратите внимание: после первого шага $R = 35$ ещё не является корнем. Выполняется только инвариант $35^2 \equiv 5 \cdot 40 \pmod{41}$.

4.2 Квадратное сравнение общего вида

Решим $2x^2 + 3x + 21 \equiv 0 \pmod{41}$. Дискриминант равен $\Delta \equiv 5$, а $(2A)^{-1} = 4^{-1} \equiv 31$. Используем корни 13 и 28 из предыдущего примера:

$$x_1 \equiv (-3 + 13)31 \equiv 23, \qquad x_2 \equiv (-3 + 28)31 \equiv 37 \pmod{41}.$$

Проверка: $2 \cdot 23^2 + 3 \cdot 23 + 21 = 1148 = 28 \cdot 41$; $2 \cdot 37^2 + 3 \cdot 37 + 21 = 2870 = 70 \cdot 41$.

4.3 Особые входные данные

Для $a = 10, p = 97$ имеем $(10/97) = (2/97)(5/97) = 1 \cdot (-1) = -1$. Поиск корней заканчивается до запуска основного цикла. Для $a = 0, p = 97$ ответ — единственный корень 0, также без цикла. Пустая таблица состояний в этих двух случаях имеет разные причины; их необходимо указать словами.

5 Учебная реализация на Python

Сохраните семь блоков ниже в одном файле в указанном порядке. Функции `ts_initial` и `ts_step` являются внутренними: входные данные проверяет `tonelli`. Функция `pow(a, k, p)` вычисляет остаток степени, а `pow(a, -1, p)` — обратный элемент.

5.1 Проверка модуля и символ Лежандра

```
from math import isqrt

def is_prime(n):
    if n < 2:
        return False
    return all(n % d for d in range(2, isqrt(n) + 1))

def check_prime(p):
    if p == 2 or not is_prime(p):
        raise ValueError("Нужен нечётный простой модуль")

def legendre(a, p):
    check_prime(p)
    v = pow(a % p, (p - 1) // 2, p)
    return -1 if v == p - 1 else v
```

Пробное деление достаточно для малых модулей вариантов. Символ переводит остаток $p - 1$ в целое число -1.

5.2 Независимое вычисление по лемме Гаусса

```
def gauss(a, p):
    check_prime(p)
    if a % p == 0:
        return 0, [], 0
    r = [(a * j) % p for j in range(1, (p + 1) // 2)]
    nu = sum(2 * x > p for x in r)
    return (-1) ** nu, r, nu

def floor_sum(a, p):
    return sum(a * j // p for j in range(1, (p + 1) // 2))
```

`gauss` возвращает символ, список остатков и число ν. Список пуст при нулевом числителе, поскольку лемма для него не применяется. Для `floor_sum` в заданиях используются положительные нечётные взаимно простые аргументы.

5.3 Начальные значения

```
def ts_initial(a, p):
    # Внутренняя функция: a != 0 и (a/p) = 1.
    q, s = p - 1, 0
    while q % 2 == 0:
        q //= 2
        s += 1
    z = 2
    while legendre(z, p) != -1:
        z += 1
    c = pow(z, q, p)
    t = pow(a, q, p)
    R = pow(a, (q + 1) // 2, p)
    return q, s, z, c, t, R
```

Невычет выбирается последовательно: 2, 3, 4 и далее до первого знака -1.

5.4 Один шаг с проверкой инварианта

```
def ts_step(m, c, t, R, p):
    # Внутренняя функция: t != 1, инварианты выполнены.
    i, u = 0, t
    while u != 1:
        u = u * u % p
        i += 1
        if i >= m:
            raise ArithmeticError("Нарушен инвариант")
    b = pow(c, 1 << (m - i - 1), p)
    R = R * b % p
    t = t * b * b % p
    c = b * b % p
    return i, c, t, R, i, b
```

Операция `1 << k` означает 2^k. В возвращаемом кортеже первые четыре позиции — новое состояние (m, c, t, R), последние две — использованные на шаге i, b. Совпадение первого и пятого значений отражает присваивание $m \leftarrow i$.

5.5 Полный алгоритм и журнал шагов

```
def tonelli(a, p):
    check_prime(p)
    a %= p
    if a == 0:
        return [0], []
    if legendre(a, p) == -1:
        return [], []
    q, s, z, c, t, R = ts_initial(a, p)
    trace = [(s, c, t, R, None, None)]
    while trace[-1][2] != 1:
        m, c, t, R = trace[-1][:4]
        trace.append(ts_step(m, c, t, R, p))
    R = trace[-1][3]
    return sorted({R, (-R) % p}), trace
```

Функция возвращает два объекта: отсортированный список всех корней и журнал состояний. В начальной строке значения `None` обозначают отсутствие предыдущего шага. Если уже начальное $t = 1$, журнал состоит из одной строки.

5.6 Решение квадратного и вырожденного сравнения

```
def quadratic_roots(A, B, C, p):
    check_prime(p)
    A, B, C = A % p, B % p, C % p
    if A == 0:
        if B != 0:
            return [(-C * pow(B, -1, p)) % p]
        return list(range(p)) if C == 0 else []
    delta = (B * B - 4 * A * C) % p
    r, _ = tonelli(delta, p)
    inv = pow(2 * A, -1, p)
    return sorted({(-B + x) * inv % p for x in r})
```

При $\Delta = 0$ используется один корень дискриминанта. Множество в последней строке устраняет повторения. Если все коэффициенты равны нулю по модулю p, возвращаются все классы от 0 до $p - 1$.

5.7 Запуск и ожидаемый результат

```
roots, trace = tonelli(5, 41)
print("Корни:", roots)
print("m, c, t, R, i, b:")
for row in trace:
    print(row)
print("Квадратное сравнение:", quadratic_roots(2, 3, 21, 41))
print("Невычет:", tonelli(10, 97)[0])
print("Нулевой случай:", tonelli(0, 97)[0])
assert all(x * x % 41 == 5 for x in roots)
assert gauss(11, 7)[0] == legendre(11, 7) == 1
```

Ожидаемый вывод

```
Корни: [13, 28]
m, c, t, R, i, b:
(3, 38, 9, 2, None, None)
(2, 9, 40, 35, 2, 38)
(1, 40, 1, 28, 1, 9)
Квадратное сравнение: [23, 37]
Невычет: []
Нулевой случай: [0]
```

Это учебная реализация с журналом и зависимым от данных числом шагов. В приложении I.4 к RFC 9380 описана отдельная версия Тонелли — Шенкса с постоянным временем выполнения; здесь изучаются математические инварианты классической схемы.

6 Индивидуальные задания

Выбор данных. Номер варианта одинаков в таблицах 1 и 2. В таблице 1 числа ℓ, v — различные нечётные простые, $n = \ell v$. В таблице 2 p — модуль извлечения корня и квадратного сравнения. Не смешивайте v из таблицы 1 и нечётную часть q разложения $p - 1$.

Задание 1. Лемма Гаусса. Для (v/ℓ) выпишите все наименьшие положительные остатки $vi \bmod \ell$, где $1 \le i \le (\ell - 1)/2$. Найдите ν, вычислите символ и сверьте его с критерием Эйлера.

Задание 2. Взаимность. Найдите (ℓ/v) по закону взаимности. Вычислите суммы $S(v, \ell)$ и $S(\ell, v)$; проверьте их чётности и равенство суммы произведению полуразностей. Объясните знак при перестановке числителя и знаменателя.

Задание 3. Тонелли — Шенкс. Решите $x^2 \equiv a \pmod p$. Укажите q, s, z, начальное состояние и все последующие состояния (m, c, t, R) с использованными i, b. Выбирайте наименьший невычет $z \ge 2$. На каждом шаге проверьте $R^2 \equiv at$ и строгое уменьшение m. Найдите оба корня и проверьте их подстановкой.

Задание 4. Общее квадратное сравнение. Решите $Ax^2 + Bx + C \equiv 0 \pmod p$. Найдите дискриминант, его символ Лежандра и, когда они существуют, квадратные корни дискриминанта.

Вычислите $(2A)^{-1}$ и все корни исходного сравнения. Для этого задания полная таблица шагов Тонелли — Шенкса не требуется, но корни дискриминанта должны быть получены алгоритмом и проверены.

Задание 5. Символ Якоби и квадраты. Для $n = \ell v$ вычислите (b/ℓ), (b/v) и (b/n). Определите существование корней из b по модулю n по местным символам. Найдите число обратимых классов с символом Якоби 1 и число обратимых квадратов. Объясните, достаточен ли знак Якоби для вывода в вашем варианте. Перечислять корни по составному модулю не требуется.

Задание 6. Программная проверка. Запустите код на своём варианте. Сравните списки корней заданий 3 и 4 с полным перебором классов от 0 до $p - 1$. Отдельно проверьте `tonelli(0, 97)`, `tonelli(10, 97)`, случай начального $t = 1$ на `tonelli(3, 13)` и отказ для `tonelli(4, 15)`. Объясните различия результатов, включая длину журнала. В отчёт включите код проверки и полученный вывод.

Таблица 1 — Данные для леммы Гаусса, взаимности и символа Якоби

№	ℓ	v	b
1	7	11	4
2	7	13	2
3	7	17	13
4	7	19	3
5	11	3	4
6	11	5	3
7	11	7	2
8	11	13	2
9	11	17	4
10	11	19	3
11	13	3	7
12	13	5	2
13	13	7	4
14	13	11	10
15	13	17	2
16	17	3	5
17	17	5	4
18	17	7	13
19	19	3	10
20	19	5	2

Таблица 2 — Данные для извлечения корня и сравнения $Ax^2 + Bx + C \equiv 0 \pmod{p}$

№	p	a	A	B	C
1	13	3	2	3	11
2	17	8	3	5	1

№	p	a	A	B	C
3	29	7	4	7	24
4	37	12	5	9	28
5	41	23	1	11	20
6	53	28	2	13	0
7	61	39	3	15	38
8	73	48	4	17	36
9	89	55	5	19	72
10	97	72	1	21	86
11	101	95	2	23	13
12	109	7	3	25	61
13	113	30	4	27	70
14	137	15	5	29	20
15	149	26	1	31	54
16	157	47	2	33	77
17	173	54	3	35	155
18	181	79	4	37	137
19	193	98	5	39	187
20	197	135	1	41	174

7 Содержание отчёта и оценивание

В отчёте укажите дисциплину, тему, Ф. И. О., группу и номер варианта. Затем приведите исходные данные, ручные решения заданий 1–5, таблицу шагов алгоритма, исходный код и результаты проверки задания 6. Завершите объяснением роли символов и условия окончания алгоритма.

Элемент работы	**Баллы**
Задание 1: остатки, число ν, символ	15
Задание 2: взаимность и суммы целых частей	15
Задание 3: начальные данные, шаги, инвариант, все корни	25
Задание 4: дискриминант и решение сравнения	20
Задание 5: местные символы, Якоби, число классов	15
Задание 6: независимый перебор и особые случаи	10
Всего	100

Типичные ошибки. Использование формулы для $p \equiv 3 \pmod 4$ при другом модуле; смешение старого и нового m в показателе для b; поиск i без проверки минимальности; объявление R корнем до достижения $t = 1$; потеря второго корня; вывод о разрешимости только по положительному символу Якоби.

8 Контрольные вопросы

1. Что означает нечётная часть q и почему она не обязана быть простым числом?
2. Почему для подготовки алгоритма нужен именно квадратичный невычет z?
3. Каким равенством связаны R, a, t до завершения алгоритма?
4. Почему минимальное i при $t \neq 1$ удовлетворяет $1 \leq i < m$?
5. Почему показатель $m - i - 1$ неотрицателен?
6. Как изменение c уменьшает возможный порядок t?
7. Какое целое число строго убывает и гарантирует завершение?
8. Как обработать нулевой числитель и квадратичный невычет?
9. Что означает журнал, состоящий из одной начальной строки?
10. Может ли другой выбор невычета изменить журнал и множество конечных корней?
11. Почему проверка корней перебором подходит для вариантов, но не заменяет изучаемый алгоритм?
12. Как значения местных символов отличают квадрат от невычета с положительным символом Якоби?

9 Литература и связь с лекцией

1. Лекция 12 данного комплекта: определения символов, лемма Гаусса, доказательство взаимности и подсчёт классов по составному модулю.
2. Мартынов Л. М. **Алгебра и теория чисел для криптографии.** 2020. § 7.7.1–7.7.2.
3. Trappe W., Washington L. C. **Introduction to Cryptography with Coding Theory.** 2nd ed., 2006. § 3.9–3.10.
4. Faz-Hernandez A. et al. **Hashing to Elliptic Curves.** RFC 9380, 2023. Приложение I.4: вариант Тонелли — Шенкса с постоянным временем выполнения, для дополнительного сравнения.