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

Дисциплина «Теоретико-числовые методы в криптографии»

ПРАКТИЧЕСКАЯ РАБОТА 8

ПОСТРОЕНИЕ КОНЕЧНЫХ ПОЛЕЙ. ЛИНЕЙНЫЕ УРАВНЕНИЯ В ПОЛЕ ГАЛУА

Методические указания и задание к выполнению

Аудиторная нагрузка: 4 академических часа. Сначала изучите методические сведения и разобранные примеры, затем выполните задания своего варианта. Теория, необходимая для работы, включена в эту методичку.

1 Цель работы

Построить конечное поле с помощью неприводимого многочлена, освоить операции над его элементами и решить линейные уравнения и системы. Связать ручные вычисления с программным представлением элементов поля и обратимым аффинным преобразованием.

2 Планируемые результаты

После выполнения работы студент умеет:

- отличать поле $\mathbb{F}_{p^n}$ от кольца целых вычетов $\mathbb{Z}_{p^n}$ при $n > 1$;
- обосновывать неприводимость многочлена степени 2 или 3 и задавать элементы поля;
- складывать, вычитать, умножать и обращать элементы в полиномиальном представлении;
- решать линейное уравнение, различать совместные и несовместные системы;
- проверять решения подстановкой и воспроизводить вычисления на Python.

3 Программное обеспечение и организация

Достаточно Python 3 и обычного текстового редактора или среды разработки; дополнительные пакеты не требуются. Код раздела 5 запускается последовательно в одном файле. Полные таблицы операций формируются программно, а выбранные клетки и решения уравнений объясняются вручную.

Рекомендуемое распределение четырёх этапов по 45 минут: построение поля и представление элементов; ручная арифметика и обращение; уравнения и системы; программная проверка и оформление отчёта. Если занятия объединены в пары, сохраните эту последовательность работы.

4 Методические сведения и разобранные примеры

4.1 Поле и его характеристика

Поле — коммутативное кольцо с единицей $1 \neq 0$, в котором каждый ненулевой элемент имеет мультипликативный обратный. Следовательно, деление на ненулевой элемент всегда определено и даёт единственный результат.

Для простого числа p вычеты по модулю p образуют поле $\mathbb{F}_p$. Конечное поле имеет $q = p^n$ элементов, где p простое и $n \geq 1$. Его **характеристика** равна p: сумма p единиц равна нулю. Характеристика поля обязательно проста, иначе произведение двух меньших ненулевых сумм единиц дало бы ноль. Как векторное пространство над $\mathbb{F}_p$ конечное поле имеет некоторую размерность n и поэтому содержит ровно p^n элементов.

Обозначения $\mathrm{GF}(q)$ и $\mathbb{F}_q$ означают поле Галуа из q элементов. При $n > 1$ оно не совпадает с $\mathbb{Z}_{p^n}$. Например, в $\mathbb{Z}_8$ ненулевые вычеты 2 и 4 имеют нулевое произведение. В поле из восьми элементов произведение двух ненулевых элементов нулём быть не может.

4.2 Конструкция через неприводимый многочлен

Многочлен $P(T) \in \mathbb{F}_p[T]$ степени $n \geq 1$ называется **неприводимым**, если его нельзя представить произведением двух многочленов положительной степени. В работе используются многочлены со старшим коэффициентом 1.

Для степени 2 или 3 неприводимость равносильна отсутствию корней в $\mathbb{F}_p$. Если многочлен такой степени приводим, один из множителей обязательно линейный; его корень будет корнем исходного многочлена. Для степеней 4 и выше проверка только корней недостаточна.

Построение поля выполняется по следующему алгоритму:

1. Выбрать простое p и неприводимый многочлен $P(T)$ степени n.
2. Рассмотреть все остатки $A(T)$ степени меньше n, включая нулевой многочлен.
3. Складывать и вычитать их коэффициенты по модулю p.
4. При умножении сначала перемножить многочлены, затем взять остаток по модулю $P(T)$, также считая коэффициенты по модулю p.

Полученная структура обозначается $\mathbb{F}_p[T]/(P(T))$. В ней ровно p^n элементов, поскольку каждый из n коэффициентов имеет p возможных значений.

Почему получается поле? Для ненулевого остатка $A(T)$ степени меньше n неприводимость $P(T)$ даёт $\gcd(A, P) = 1$. Алгоритм Евклида для многочленов, как и для целых чисел, позволяет записать $U(T)A(T) + V(T)P(T) = 1$. После перехода к остаткам произведение классов U и A равно 1 — обратный элемент существует. Остальные правила кольца наследуются от многочленов.

Если взять приводимый многочлен, это рассуждение нарушается. Например, над $\mathbb{F}_2$ имеем $T^2 + 1 = (T + 1)^2$. В кольце остатков по этому многочлену ненулевой класс $T + 1$ имеет нулевой квадрат. Такое кольцо полем не является.

4.3 Построение поля из восьми элементов

Выберем $P_8(T) = T^3 + T + 1$ над $\mathbb{F}_2$. Проверка даёт $P_8(0) = 1$ и $P_8(1) = 1$ в $\mathbb{F}_2$. Корней нет; степень равна 3, следовательно, многочлен неприводим.

Обозначим через α класс многочлена T. Тогда

$$\mathbb{F}_8 = \mathbb{F}_2[T]/(T^3 + T + 1), \qquad \alpha^3 = \alpha + 1.$$

Элементы имеют вид $a_0 + a_1\alpha + a_2\alpha^2$, где каждый коэффициент равен 0 или 1. Коэффициенты складываются по модулю 2, поэтому $A + A = 0$ и $A - B = A + B$ для любых элементов этого поля.

Пример сложения. Для $A = \alpha^2 + 1$ и $B = \alpha + 1$ получаем $A + B = \alpha^2 + \alpha$. Две единицы сокращаются, поскольку $1 + 1 = 0$.

Пример умножения. Сначала раскрываем скобки, затем заменяем α^3 на $\alpha + 1$:

$$(\alpha^2 + 1)(\alpha + 1) = \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^2.$$



Рисунок 1. Коэффициенты уменьшаются по модулю 2, а произведение заменяется остатком по модулю выбранного неприводимого многочлена.

Пример обращения. Над $\mathbb{F}_2$ выполняется тождество

$$T^3+T+1=(T^2+T)(T+1)+1.$$

Поэтому $(\alpha^2+\alpha)(\alpha+1)=1$ и $(\alpha+1)^{-1}=\alpha^2+\alpha$. Проверка прямым умножением даёт $\alpha^3+\alpha=(\alpha+1)+\alpha=1$.

Другой способ обращения следует из теоремы Лагранжа. Ненулевые элементы $\mathbb{F}_q$ образуют группу порядка $q-1$, поэтому для $A\neq 0$

$$A^{q-1}=1, \qquad A^{-1}=A^{q-2}.$$

В $\mathbb{F}_8$ обратный элемент можно найти как шестую степень; в $\mathbb{F}_9$ — как седьмую. Эти формулы не применяются к нулю.

4.4 Поле из девяти элементов и отличие вычитания

Над $\mathbb{F}_3$ многочлен $P_9(T)=T^2+1$ имеет значения 1, 2, 2 при $T=0,1,2$ соответственно. Корней нет, поэтому он неприводим. Обозначая класс T через β, получаем

$$\mathbb{F}_9=\mathbb{F}_3[T]/(T^2+1), \qquad \beta^2=-1=2.$$

Элемент имеет вид $a_0+a_1\beta$, где $a_0,a_1\in\{0,1,2\}$. Здесь $1+1=2$, а $1+1+1=0$. Сложение и вычитание уже не совпадают.

Пусть $A=\beta+1$, $B=2\beta+2$. По модулю 3 получаем

$$A+B=0, \qquad A-B=2\beta+2.$$

Произведение равно $2\beta^2+4\beta+2=4+\beta+2=\beta$. Обратный к A элемент — $\beta+2$, поскольку

$$(\beta+1)(\beta+2)=\beta^2+3\beta+2=2+2=1.$$

Сокращение коэффициентов выполняется по модулю 3, а не 9. Именно это правило отличает арифметику данного поля от арифметики целых вычетов по модулю 9.

4.5 Коды элементов для таблиц и программы

Для хранения элемента используем целое число, которое кодирует его коэффициенты:

$$\operatorname{code}(a_0+a_1\theta+\cdots+a_{n-1}\theta^{n-1})=\sum_{i=0}^{n-1}a_i\,p^i.$$

Здесь $\theta=\alpha$ для $\mathbb{F}_8$ и $\theta=\beta$ для $\mathbb{F}_9$. Кортеж коэффициентов в таблице и функции `digits` записывается от младшей степени к старшей: сначала a_0.

Код	Элемент $\mathbb{F}_8$	Коэффициенты	Элемент $\mathbb{F}_9$	Коэффициенты
0	0	(0,0,0)	0	(0,0)

Код	Элемент $\mathbb{F}_8$	Коэффициенты	Элемент $\mathbb{F}_9$	Коэффициенты
1	1	(1,0,0)	1	(1,0)
2	α	(0,1,0)	2	(2,0)
3	$\alpha+1$	(1,1,0)	β	(0,1)
4	α^2	(0,0,1)	$\beta+1$	(1,1)
5	α^2+1	(1,0,1)	$\beta+2$	(2,1)
6	$\alpha^2+\alpha$	(0,1,1)	2β	(0,2)
7	$\alpha^2+\alpha+1$	(1,1,1)	$2\beta+1$	(1,2)
8	—	—	$2\beta+2$	(2,2)

Числа в таблице вариантов и в выводе программы — коды элементов. Они не разрешают выполнять обычные целочисленные операции с последующим взятием остатка по модулю q. Например, код 3 в $\mathbb{F}_8$ обозначает $\alpha+1$, а код 3 в $\mathbb{F}_9$ обозначает β.

Для рассмотренного произведения в $\mathbb{F}_8$ коды входов равны 5 и 3, код результата равен 4. Обычное целое произведение $5\cdot 3$ по модулю 8 дало бы 7, то есть другой результат. Для сложения в $\mathbb{F}_8$ можно использовать побитовое XOR кодов; для $\mathbb{F}_9$ это правило не подходит.

4.6 Линейное уравнение и аффинное уравнение

Решим уравнение $AX=B$ в поле $\mathbb{F}_q$. Если $A\neq 0$, умножение на A^{-1} даёт единственное решение $X=A^{-1}B$. Если $A=0$ и $B=0$, подходит любой из q элементов. Если $A=0$, но $B\neq 0$, решений нет.

Пример в $\mathbb{F}_8$. Уравнение $(\alpha+1)X=\alpha^2+1$ имеет решение

$$X=(\alpha^2+\alpha)(\alpha^2+1)=\alpha+1.$$

Подстановка подтверждает ответ: $(\alpha+1)^2=\alpha^2+1$. В кодах элементов исходные данные равны 3 и 5, а ответ равен 3.

Пример в $\mathbb{F}_9$. В уравнении $(\beta+1)X=2\beta+2$ правая часть равна $2(\beta+1)$, поэтому $X=2$. Через обратный элемент получаем тот же результат: $(\beta+2)(2\beta+2)=2$.

Для аффинного уравнения $AZ+B=C$ сначала вычитают B, затем умножают на A^{-1}:

$$Z=A^{-1}(C-B),\qquad A\neq 0.$$

Все операции выполняются в выбранном поле. В частности, в характеристике 3 нельзя автоматически заменять вычитание сложением.

4.7 Системы двух линейных уравнений

Рассмотрим систему над $\mathbb{F}_q$:

$$\begin{cases} ax+by=u,\\ cx+dy=v. \end{cases}$$

В индивидуальных вариантах $a\neq 0$. Вычтем из второй строки первую, умноженную на ca^{-1}. Получим эквивалентную систему

$$ax+by=u,\qquad Dy=E,$$

где $D=d-ca^{-1}b$ и $E=v-ca^{-1}u$. Определитель $\Delta=ad-bc$ удовлетворяет $D=a^{-1}\Delta$.

- При $D\neq 0$ решение единственно: $y=D^{-1}E$, затем $x=a^{-1}(u-by)$.
- При $D=0$ и $E\neq 0$ вторая строка требует $0=E$; система несовместна.

- При $D = E = 0$ вторая строка не даёт нового ограничения. Полагаем $y = t$ и получаем $x = a^{-1}u - a^{-1}bt$, где $t \in \mathbb{F}_q$. Решений ровно q.

Операции над строками допустимы по тем же причинам, что и над вещественными числами: строки можно менять местами, умножать на ненулевой элемент поля и прибавлять кратную другой строки. Если в другой задаче $a = 0$, сначала выбирают ненулевой ведущий коэффициент, меняя строки или порядок неизвестных; делить на ноль нельзя.

Пример с единственным решением в $\mathbb{F}_8$.

$$\begin{cases} \alpha x + y = \alpha, \\ x + \alpha y = 0. \end{cases}$$

Прибавим к первой строке вторую, умноженную на α. Члены с x сократятся, и останется $(\alpha^2 + 1)y = \alpha$. Поскольку $\alpha(\alpha^2 + 1) = 1$, получаем $y = \alpha^2$. Из второй строки $x = \alpha y = \alpha^3 = \alpha + 1$. Проверка первой строки: $\alpha(\alpha + 1) + \alpha^2 = \alpha$; второй: $(\alpha + 1) + \alpha^3 = 0$.

Пример с несколькими решениями.

$$\begin{cases} \alpha x + y = \alpha, \\ \alpha^2 x + \alpha y = \alpha^2. \end{cases}$$

Вторая строка равна первой, умноженной на α. Поэтому $x = t, y = \alpha + \alpha t$, где $t \in \mathbb{F}_8$. Получается восемь пар. Если заменить правую часть второй строки на $\alpha^2 + 1$, после исключения возникнет $0 = 1$ — решений не будет. Над конечным полем совместная вырожденная система имеет конечное число решений.

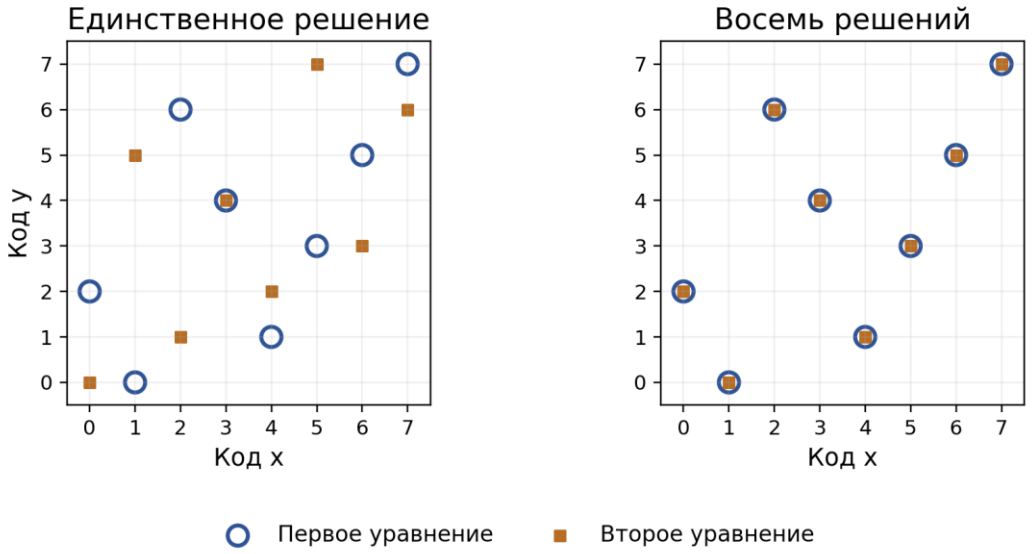


Рисунок 2. Решения системы — общие точки двух множеств. Оси перечисляют коды элементов; это не графики прямых над вещественными числами.

4.8 Учебное обратимое преобразование

Пусть $a \neq 0$. Отображение $\mathcal{E}(x) = ax + b$ переставляет элементы поля, а обратное отображение имеет вид $\mathcal{D}(y) = a^{-1}(y - b)$. Действительно, $\mathcal{D}(\mathcal{E}(x)) = x$ для каждого x.

Для $a = \alpha + 1$ и $b = \alpha^2 + 1$ в $\mathbb{F}_8$ таблица образов в порядке кодов 0,1, … ,7 равна [5,6,3,0,2,1,4,7]. Все коды различны, значит, преобразование биективно. Это учебный пример применения полевой арифметики; обратимость и криптографическая стойкость — разные свойства.

4.9 Типичные ошибки

1. Использовать вычисление `(a*b) % q` вместо умножения многочленов с редукцией.
2. Сокращать коэффициенты по модулю q, хотя требуется модуль p.

3. Проверять только отсутствие корней у многочлена степени 4 или выше и считать это доказательством неприводимости.
4. Применять формулу обратного элемента к нулю или к классу в кольце по приводимому многочлену.
5. Считать, что нулевой определитель всегда означает отсутствие решений. Необходимо проверить правые части после исключения.
6. Считать числовой код элемента его значением в простом подполе. Например, код 3 в $\mathbb{F}_9$ — элемент β, а сумма трёх единиц в этом поле равна нулю.

5 Программные примеры перед заданиями

5.1 Поля и представление коэффициентов

Код рассчитан на два заранее обоснованных поля этой работы. Кортежи многочленов содержат коэффициенты от свободного члена к старшему. Функции не проверяют произвольные новые многочлены на неприводимость: для другого поля такое обоснование нужно выполнить отдельно.

```
FIELDS = {
    "F8": (2, (1, 1, 0, 1)),  # 1 + T + T**3
    "F9": (3, (1, 0, 1)),     # 1 + T**2
}

def digits(a, F):
    p, f = F
    n = len(f) - 1
    if type(a) is not int or not 0 <= a < p**n:
        raise ValueError("Неверный код элемента")
    return [(a // p**i) % p for i in range(n)]

def encode(cs, F):
    p, f = F
    if len(cs) != len(f) - 1:
        raise ValueError("Неверное число коэффициентов")
    return sum((c % p) * p**i for i, c in enumerate(cs))
```

Функция `digits` переводит код в коэффициенты, а `encode` выполняет обратное действие. Например, для `F8` код `5` соответствует списку `[1, 0, 1]` и элементу $1 + \alpha^2$.

5.2 Сложение, противоположный элемент и вычитание

Сначала выполняются покоэффициентные действия, затем `encode` уменьшает каждый коэффициент по модулю p.

```
def add(a, b, F):
    return encode([x + y for x, y in
                   zip(digits(a, F), digits(b, F))], F)

def neg(a, F):
    return encode([-x for x in digits(a, F)], F)

def sub(a, b, F):
    return add(a, neg(b, F), F)
```

5.3 Умножение с редукцией

Первый двойной цикл формирует коэффициенты произведения. Затем старшие члены последовательно уничтожаются вычитанием подходящего кратного $P(T)$. Старший коэффициент $P(T)$ равен единице, поэтому дополнительного деления не требуется.

```
def mul(a, b, F):
    p, f = F
    A, B = digits(a, F), digits(b, F)
    n = len(A)
    c = [0] * (2*n - 1)
    for i in range(n):
        for j in range(n):
            c[i+j] = (c[i+j] + A[i]*B[j]) % p
    for k in range(2*n - 2, n - 1, -1):
        lead = c[k]
        for j in range(n + 1):
            c[k-n+j] = (c[k-n+j] - lead*f[j]) % p
    return encode(c[:n], F)
```

При уменьшении степени коэффициенты уже лежат в $\mathbb{F}_p$. Возвращаются первые n коэффициентов остатка. Алгоритм использует порядка n^2 операций над коэффициентами при данном прямом способе умножения и редукции.

5.4 Степень и обратный элемент

Бинарное возведение в степень использует порядка $\log k$ умножений поля для $k > 0$. При нулевом показателе функция возвращает единицу как пустое произведение. Обратный элемент вычисляется только для ненулевого аргумента.

```
def fpow(a, k, F):
    digits(a, F)
    if type(k) is not int or k < 0:
        raise ValueError("Показатель должен быть целым >= 0")
    r = 1
    while k:
        if k % 2:
            r = mul(r, a, F)
        a = mul(a, a, F)
        k //= 2
    return r

def inv(a, F):
    digits(a, F)
    if a == 0:
        raise ZeroDivisionError("У нуля нет обратного")
    p, f = F
    return fpow(a, p**(len(f)-1) - 2, F)
```

5.5 Уравнения и независимая проверка системы

Функция `solve_linear` возвращает список всех решений, включая пустой список. Системы здесь проверяются полным перебором пар: максимум $9^2 = 81$ пара. Ручное исключение из раздела 4.7 и такой перебор дают два разных способа проверки. Перебор подходит для небольших полей этой работы; его стоимость растёт как q^2 проверок пар.

```
def solve_linear(a, b, F):
    digits(a, F)
    digits(b, F)
    p, f = F
    q = p**(len(f)-1)
    if a != 0:
        return [mul(inv(a, F), b, F)]
    return list(range(q)) if b == 0 else []

def solve_system(a, b, c, d, u, v, F):
    for value in (a, b, c, d, u, v):
        digits(value, F)
    p, f = F
    q = p**(len(f)-1)
    return [(x, y) for x in range(q) for y in range(q)
            if add(mul(a, x, F), mul(b, y, F), F) == u
            and add(mul(c, x, F), mul(d, y, F), F) == v]
```

5.6 Таблицы операций и аффинное преобразование

Индекс строки и индекс столбца совпадают с кодами входных элементов. В клетке записан код результата. Для аффинного преобразования используйте $a \neq 0$.

```
def operation_tables(F):
    p, f = F
    q = p**(len(f)-1)
    plus = [[add(a, b, F) for b in range(q)]
            for a in range(q)]
    times = [[mul(a, b, F) for b in range(q)]
             for a in range(q)]
    return plus, times

def affine(x, a, b, F):
    return add(mul(a, x, F), b, F)

def restore(y, a, b, F):
    return mul(inv(a, F), sub(y, b, F), F)
```

Чтобы вывести таблицу, вызовите `plus, times = operation_tables(FIELDS["F8"])` и напечатайте строки нужного списка. В отчёте подпишите поле и порядок кодов в заголовках.

5.7 Воспроизводимый пример

Во всех строках следующего вывода числа, относящиеся к элементам поля, являются их кодами. В частности, надпись `3*x=5` означает уравнение с коэффициентом $\alpha + 1$ и правой частью $\alpha^2 + 1$ в $\mathbb{F}_8$.

```
F = FIELDS["F8"]
print("F8:", add(5, 3, F), mul(5, 3, F), inv(3, F))
print("3*x=5:", solve_linear(3, 5, F))
print("Система:", solve_system(2, 1, 1, 2, 2, 0, F))
print("E(x):", [affine(x, 3, 5, F) for x in range(8)])
F = FIELDS["F9"]
print("F9:", add(4, 8, F), sub(4, 8, F), mul(4, 8, F))
print("4*x=8:", solve_linear(4, 8, F))
```

Ожидаемый вывод

```
F8: 6 4 6
3*x=5: [3]
Система: [(3, 4)]
E(x): [5, 6, 3, 0, 2, 1, 4, 7]
F9: 0 8 3
4*x=8: [2]
```

5.8 Проверки арифметики и решений

Следующий блок проверяет кодирование, обратные элементы, коммутативность, дистрибутивность и решение каждого линейного уравнения в обоих полях. Это проверка реализации; математическое обоснование построения поля дано в разделе 4.2.

```
for F in FIELDS.values():
    p, f = F
    q = p**(len(f)-1)
    for a in range(q):
        assert encode(digits(a, F), F) == a
        assert add(a, neg(a, F), F) == 0
        assert mul(a, 1, F) == a
        assert fpow(a, q, F) == a
        if a:
            assert mul(a, inv(a, F), F) == 1
        for b in range(q):
            assert add(a, b, F) == add(b, a, F)
            assert mul(a, b, F) == mul(b, a, F)
            expected = [x for x in range(q)
                        if mul(a, x, F) == b]
            assert solve_linear(a, b, F) == expected
            for c in range(q):
                left = mul(a, add(b, c, F), F)
                right = add(mul(a, b, F), mul(a, c, F), F)
                assert left == right
```

Дополнительно проверим вырожденные системы, восстановление после преобразования и обработку деления на ноль.

```
F = FIELDS["F8"]
assert len(solve_system(2, 1, 4, 2, 2, 4, F)) == 8
assert solve_system(2, 1, 4, 2, 2, 5, F) == []
assert len({affine(x, 3, 5, F) for x in range(8)}) == 8
for x in range(8):
    assert restore(affine(x, 3, 5, F), 3, 5, F) == x
try:
    inv(0, F)
except ZeroDivisionError:
    pass
else:
    raise AssertionError("Деление на ноль не обнаружено")
print("Все проверки пройдены")
```

При успешном выполнении появляется строка `Все проверки пройдены`. Для своего варианта дополнительно сравните ручные ответы с `solve_linear`, `solve_system` и результатами `fpow`.

6 Порядок выполнения работы

1. Выберите номер варианта из раздела 8. Выпишите поле, его характеристику, многочлен и исходные элементы в полиномиальном виде.
2. Выполните ручные вычисления заданий 1–4, показывая редукцию степеней и действия над строками.
3. Постройте таблицу аффинного преобразования из задания 5.
4. Запустите код раздела 5, замените входные данные своими и выполните проверки задания 6.
5. Оформите отчёт с ручным решением, таблицами и результатами программной проверки. Подготовьтесь объяснить любое использованное правило.

7 Индивидуальные задания

Во всех вариантах коэффициент a ненулевой. Все буквы в формулах ниже обозначают элементы поля, указанные своими кодами в таблице. Показатель k — обычное целое число.

Задание 1. Построение поля. Докажите неприводимость выбранного $P(T)$, укажите характеристику и порядок поля. Выпишите все элементы, коэффициенты и коды. Сформируйте полные таблицы сложения и умножения; вручную объясните клетки для $a + b$, ab и $a \cdot a^{-1}$. Объясните, почему $\mathbb{Z}_q$ не заменяет это поле.

Задание 2. Арифметика. Найдите $a + b$, $a - b$, ab, a^{-1} и a^k. Результаты запишите в полиномиальном виде и кодами. Обращение проверьте умножением, а степень — последовательностью квадратов или допустимым уменьшением показателя.

Задание 3. Линейные уравнения. Решите $aX = b$ и $aZ + b = c$. Подставьте оба ответа в соответствующие уравнения. Отдельно укажите множества решений уравнений $0X = 0$ и $0X = 1$ в своём поле.

Задание 4. Система. Решите $ax + by = u$, $cx + dy = v$. Найдите Δ, D и E по разделу 4.7. Для единственного решения приведите пару и проверку обеих строк; для нескольких решений — параметрическую запись и полный список пар кодов; при несовместности — полученное противоречие.

Задание 5. Аффинное преобразование. Постройте таблицу $\mathcal{E}(x) = ax + b$ для всех элементов поля в порядке кодов. Запишите обратное преобразование и докажите его формулу. Проверьте отсутствие повторов в таблице и восстановление каждого входа.

Задание 6. Программная проверка. Воспроизведите примеры и проверки раздела 5, затем вычислите ответы своего варианта. Полным перебором проверьте множество решений системы. При расхождении с ручным расчётом найдите первый неверный шаг и покажите исправление.

8 Варианты индивидуальных заданий

Для вариантов 1–10 используется $\mathbb{F}_8$ с $P_8(T) = T^3 + T + 1$; для вариантов 11–20 — $\mathbb{F}_9$ с $P_9(T) = T^2 + 1$. Соответствие кодов элементам приведено в разделе 4.5.

№	Поле	a	b	c	d	u	v	k
1	$\mathbb{F}_8$	5	5	4	7	6	1	14
2	$\mathbb{F}_8$	1	7	5	2	3	0	17
3	$\mathbb{F}_8$	4	1	6	5	0	7	20
4	$\mathbb{F}_8$	7	3	4	1	5	3	23
5	$\mathbb{F}_8$	3	5	2	6	2	4	26
6	$\mathbb{F}_8$	6	7	2	6	7	4	29
7	$\mathbb{F}_8$	2	1	3	1	4	3	32
8	$\mathbb{F}_8$	5	3	4	4	1	2	35
9	$\mathbb{F}_8$	1	5	4	2	6	5	38
10	$\mathbb{F}_8$	4	7	2	6	3	5	41
11	$\mathbb{F}_9$	3	7	6	1	2	7	44
12	$\mathbb{F}_9$	6	0	7	4	7	5	47
13	$\mathbb{F}_9$	1	2	8	7	3	3	50
14	$\mathbb{F}_9$	4	4	3	3	8	6	53
15	$\mathbb{F}_9$	7	6	7	6	4	5	56
16	$\mathbb{F}_9$	2	8	3	7	0	6	59
17	$\mathbb{F}_9$	5	1	4	1	5	4	62
18	$\mathbb{F}_9$	8	3	5	4	1	2	65
19	$\mathbb{F}_9$	3	5	8	3	6	4	68
20	$\mathbb{F}_9$	6	7	2	8	2	4	71

9 Требования к отчёту

Укажите дисциплину, номер работы, Ф. И. О., группу и вариант. Отчёт должен содержать обоснование поля, таблицу кодов, таблицы операций, ручные решения, таблицу аффинного преобразования, код и существенные результаты запуска.

Для системы обязательно запишите преобразованную вторую строку $Dy = E$ и объясните число решений. Для множества из q решений приведите параметрическую формулу и список пар; один найденный пример полным ответом не является. Завершите отчёт кратким выводом о правилах полевой арифметики, которые оказались существенными в вашем варианте.

10 Критерии оценивания

Составляющая работы	Баллы
Построение поля, доказательство неприводимости, коды и таблицы	20
Арифметика, обратный элемент и степень с обоснованием	20
Два линейных уравнения и случаи нулевого коэффициента	15
Система: исключение, полное множество решений и проверка	25
Аффинное преобразование и восстановление	10
Программная проверка, воспроизводимость и оформление	10
Итого	100

11 Контрольные вопросы

1. Чем поле отличается от кольца? Почему ноль не имеет обратного элемента?
2. Что означают порядок и характеристика конечного поля?
3. Почему поле из восьми элементов не совпадает с $\mathbb{Z}_8$?
4. Почему отсутствие корней достаточно для неприводимости в степенях 2 и 3, но недостаточно в общем случае?
5. Как образуются элементы $\mathbb{F}_p[T]/(P(T))$ и почему их p^n?
6. Какие два разных уменьшения используются при умножении элементов в полиномиальном представлении?
7. Почему сложение и вычитание совпадают в $\mathbb{F}_8$ и не совпадают в $\mathbb{F}_9$?
8. Как найти обратный элемент с помощью тождества Безу и с помощью степени?
9. Какие множества решений может иметь уравнение $AX = B$ в поле?
10. Почему нулевого определителя недостаточно для вывода о совместности системы?
11. Почему вырожденная совместная система из примера имеет восемь, а не бесконечно много решений?
12. Как расшифровать код элемента и почему обычное умножение кодов по модулю q неверно?
13. Почему при $a \neq 0$ отображение $ax + b$ является подстановкой поля?
14. Чем программный перебор решений дополняет ручное исключение неизвестных?

Литература

1. Мартынов Л. М. **Алгебра и теория чисел для криптографии.** 2020. §14.2.1 «Описание конечных полей»; §14.2.2 «Алгоритм построения конечного поля порядка p^n»; §12.1 — теорема Лагранжа.
2. Paar C., Pelzl J. **Understanding Cryptography: A Textbook for Students and Practitioners.** 2010. §4.3 — поля Галуа, полиномиальное представление и операции в расширениях двоичного поля.
3. Trappe W., Washington L. C. **Introduction to Cryptography with Coding Theory.** 2nd ed., 2006. §3.11 «Finite Fields».

Числовые примеры, программная реализация и варианты составлены для этой работы. Формулы и код используют одно и то же представление полей.