УДК 519.1

С.Л. Крывый, Г.И. Гогерчак

АЛГОРИТМ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ В ПОЛЕ F_{p^k}

Ключевые слова: системы диофантовых уравнений, расширение поля, задача о математическом сейфе.

Введение

Рассматривается задача построения базиса множества решений систем линейных уравнений (СЛУ) над конечным полем F_{p^k}. Поле F_{p^k} строится как расширение поля вычетов F_p с помощью неприводимого в поле F_p полинома степени k, где p — простое число. Приведем необходимые понятия и определения.

Кольцом вычетов по модулю числа m называется алгебра $Z_m = (A = \{0, 1, ..., m-1\}, \Sigma = \{+, \cdot, -, ^{-1}, 0, 1\})$, где $+$ и $\cdot$ — бинарные ассоциативные коммутативные операции сложения и умножения по модулю m, связанные законом дистрибутивности, операции $-$ и $^{-1}$ — унарные операции взятия противоположного и обратного элементов относительно операций $+$ и $\cdot$ соответственно, а 0 и 1 — нульарные операции: аддитивный ноль и мультипликативная единица.

Полем вычетов по модулю m называется ассоциативно-коммутативное кольцо вычетов, в котором мультипликативная полугруппа является группой. Известно, что кольцо вычетов по модулю m является полем, если и только если m — простое число. Поле вычетов будем обозначать F_m.

Противоположным элементу a в поле F_m называется элемент b такой, что $a + b \equiv 0 \pmod m$. Очевидно, что сравнения $a_1x_1 + \ldots + a_jx_j + \ldots + a_nx_n \equiv 0 \pmod m$ и $a_1x_1 + \ldots - b_jx_j + \ldots + a_nx_n \equiv 0 \pmod m$ эквивалентны ($-b_j$ — противоположный a_j), т.е. каждое решение первого сравнения является решением второго, и наоборот.

В поле F_p при $a \neq 0$ сравнение $ax \equiv b \pmod p$ всегда имеет решение, и это решение единственно.

Полином над полем F_p называется тривиальным, если он является константой. Полином $q(x)$ неприводим в поле F_p, если его нельзя разложить в произведение двух нетривиальных полиномов в этом же поле.

Например, полиномы x^2+x+1 и x^3+x+1 неприводимы в поле F_2, в то же время в поле F_3 полином x^3+x+1 разлагается в следующее произведение: $x^3+x+1=(x+2)\cdot(x^2+x+2)$ и, следовательно, неприводимым не является.

Если некоторый полином $P(x)$ неприводим в поле F_p, то и произвольный полином вида $Q(x)=i\cdot P(x)$, где $0<i<p$, тоже неприводим. Поэтому в случае их поиска для заданного p достаточно найти все неприводимые полиномы, первый коэффициент которых равен единице — так называемые нормализованные полиномы. Недостающие элементы этой совокупности можно легко получить, домножая полученные поочередно на ненулевые элементы соответствующего поля.

Расширением поля F_p по модулю неприводимого полинома $q(x)$ степени k называется поле F_{p^k} полиномов степени $n<k$ над полем F_p. Поле F_{p^k} имеет характеристику p и состоит из p^k элементов. Его построение сводится к вычислению остатков от деления полиномов, полученных при сложении (умножении) элементов поля, на неприводимый над полем F_p полином $q(x)$. Примерами такого типа полей являются поля F_{2^2} и F_{3^2}, таблицы сложения и умножения для которых получены с помощью неприводимых полиномов x^2+x+1 и x^2+x+2 над полями F_2 и F_3 соответственно (табл. 1, 2).

Таблица 1

+	0	1	2	3		×	0	1	2	3
0	0	1	2	3		0	0	0	0	0
1	1	0	3	2		1	0	1	2	3
2	2	3	0	1		2	0	2	3	1
3	3	2	1	0		3	0	3	1	2

Здесь полином x обозначен числом 2, а полином $x+1$ — числом 3.

Таблица 2

+	0	1	2	3	4	5	6	7	8		×	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7	8		0	0	0	0	0	0	0	0	0
1	1	2	0	4	5	3	7	8	6		1	0	1	2	3	4	5	6	7
2	2	0	1	5	3	4	8	6	7		2	0	2	1	6	8	7	3	5
3	3	4	5	6	7	8	0	1	2		3	0	3	6	7	1	4	5	8
4	4	5	3	7	8	6	1	2	0		4	0	4	8	1	5	6	2	3
5	5	3	4	8	6	7	2	0	1		5	0	5	7	4	6	2	8	1
6	6	7	8	0	1	2	3	4	5		6	0	6	3	5	2	8	7	4
7	7	8	6	1	2	0	4	5	3		7	0	7	5	8	3	1	4	2
8	8	6	7	2	0	1	5	3	4		8	0	8	4	2	7	3	1	6

Здесь полином x обозначен числом 3, $x+1$ — числом 4, $x+2$ — числом 5, $2x$ — числом 6, $2x+1$ — числом 7, $2x+2$ — числом 8.

Рассмотрим способ построения такого типа полей.

1. Построение поля F_{p^k}

Поскольку операции сложения и умножения в поле F_{p^k} нетрадиционны, полноценная работа с этим полем возможна только после построения таблиц этих

ISSN 0572-2691

операций. Построение расширения поля требует нахождения неприводимого полинома соответствующей степени над F_p, с помощью которого строятся таблицы операций.

1.1. Поиск неприводимого полинома над полем вычетов. Поиск неприводимых полиномов в заданном поле F_p может осуществляться их полным перебором и последующей проверкой неприводимости каждого из них. Поскольку количество полиномов степени k с коэффициентами из поля F_p составляет p^k (при условии рассмотрения лишь нормализованных полиномов), сложность такого алгоритма — $O(p^k)\cdot T(p,k)$, где $T(p,k)$ — временная сложность алгоритма проверки неприводимости.

Для проверки неприводимости воспользуемся тестом Рабина [6]. Пусть l_1, $l_2,\ldots, l_n$ — простые делители числа k и $k/l_i = m_i$. Полином $P(x)$ степени k над полем F_p неприводим, если и только если:

1) $P(x)\,|\,(x^{p^k} - x)$, т.е. $P(x)$ — делитель $x^{p^k} - x$;

2) $\gcd(P(x),\, x^{p^{m_i}} - x) = 1$ для всех $1 \le i \le k$,

где сокращение gcd — наибольший общий делитель.

Пример 1. Пусть имеется полином $P(x) = x^6 + x^5 + 1$ степени 6 в поле F_2. Деление полинома $Q(x) = x^{2^6} = x^{64}$ на этот полином дает в остатке полином $R(x) = x$, таким образом, $P(x)$ удовлетворяет первому условию теста Рабина. Полиномы $x^{2^{6/2}} - x = x^8 - x$ и $x^{2^{6/3}} - x = x^4 - x$ попарно взаимно просты с полиномом $P(x)$, что в совокупности на основании теста Рабина дает возможность считать полином $x^6 + x^5 + 1$ неприводимым в поле F_2.

Если же рассмотреть полином $P(x) = x^2 + x$ степени 2 в том же поле F_2, то остаток от деления полинома $Q(x) = x^{2^2} = x^4$ на этот полином тоже дает в остатке полином $R(x) = x$. Но $P(x)$ очевидно неприводимым не является, поскольку $x^2 + x = x\cdot(x+1)$. Действительно, полиномы $x^{2^{2/2}} - x = x^2 - x$ и $P(x) = x^2 + x$ не являются взаимно простыми, поэтому второе условие теста Рабина в этом случае не выполняется. ♠

Из вышесказанного следует такой алгоритм проверки полинома на неприводимость:

```
function is-irreducible(P(x))
begin
  if x^(p^k) ≡ x (mod P(x)) then
  begin
    irreducible = true;
    for l_i in factors(deg(P(x))) do
      begin
      if gcd(P(x), x^(p^(k/l_i)) - x (mod P(x))) ≠ 1 then
          begin
            irreducible = false;
            break;
          end;
      end;
      return irreducible;
  end
  else return false;
end;
```

Здесь $\gcd(P(x), q(x))$ — функция вычисления наибольшего общего делителя полиномов алгоритмом Евклида, $\deg((P(x)))$ — степень полинома $P(x)$, а $\text{factors}(n)$ — функция, которая находит простые делители числа n.

Использование в приведенном выше алгоритме бинарного возведения в степень для вычисления полинома x^{p^k} дает возможность выполнить сравнения $x^{p^k} \equiv x \pmod{P(x)}$ и $\gcd(P(x), x^{p^{k/l_i}} - x \pmod{P(x)}) \neq 1$ с помощью не более $2 \cdot \log p^k$ операций. А использование эффективных операций умножения и деления полиномов позволяет достичь для алгоритма в целом сложности $O(k^2 \log^2 k \log p \log\log k)$ [6].

Вооружившись эффективным алгоритмом проверки полинома на неприводимость, для формирования каталога неприводимых нормализованных полиномов воспользуемся алгоритмом с полным перебором:

```
function get-irreducible(p, k)
begin
  irreducible-polynomials = [];
  for P(x) in all-polynomials(p, k) do
  begin
    if is-irreducible(P(x)) then
    begin
      irreducible-polynomials.add(P(x));
    end;
  end;
  return irreducible-polynomials;
end;
```

Здесь all-polynomials (*p*, *k*) — генератор нормализованных полиномов степени k над полем F_p.

В результате выполнения указанной выше функции для разных значений p и k можно сформировать таблицу неприводимых нормализованных полиномов, фрагмент которой приведен ниже (табл. 3).

Таблица 3

p	k=2	k=3	k=4	k=5
p = 2	x^2+x+1	x^3+x^2+1 x^3+x+1	x^4+x^3+1 x^4+x+1 $x^4+x^3+x^2+x+1$	x^5+x^3+1 x^5+x^2+1 $x^5+x^4+x^3+x^2+1$ $x^5+x^4+x^3+x+1$ $x^5+x^4+x^2+1$
p = 3	x^2+1 x^2+x+2 x^2+2x+2	x^3+2x^2+1 x^3+2x^2+x+1 x^3+2x+1 x^3+x^2+2x+1 x^3+2x^2+2x+1	$x^4+x^3+x^2+1$ $x^4+2x^3+x^2+1$ x^4+2x^3+x+1 x^4+x^2+x+1 x^4+x^3+2	x^5+2x^4+1 $x^5+x^4+2x^3+1$ $x^5+x^4+x^2+1$ $x^5+2x^3+x^2+1$ x^5+2x^4+x+1
p = 5	x^2+x+1 x^2+4x+1 x^2+2 x^2+x+2 x^2+4x+2 x^2+3	x^3+x^2+1 x^3+2x^2+1 x^3+x+1 x^3+3x^2+x+1 x^3+4x^2+x+1 x^3+2x+1	$x^4+x^3+x^2+1$ $x^4+4x^3+x^2+1$ $x^4+2x^3+2x^2+1$ $x^4+3x^3+2x^2+1$ $x^4+x^3+3x^2+1$ $x^4+4x^3+3x^2+1$	x^5+4x^4+1 x^5+2x^3+1 $x^5+3x^4+2x^3+1$ $x^5+4x^4+3x^3+1$ $x^5+x^4+4x^3+1$ $x^5+2x^4+4x^3+1$

Узким местом алгоритма поиска неприводимых полиномов, очевидно, является их полный перебор. Если в случае поиска всех таких полиномов достичь большей эффективности не удается, то для поиска хотя бы одного неприводимого полинома вместо полного перебора можно ограничиться случайным выбором некоторого полинома над соответствующим полем с последующей его проверкой. Действительно, вероятность выбора неприводимого полинома среди полиномов степени n близка к $1/n$, поэтому после n повторений этой операции можно получить неприводимый полином с вероятностью, большей $1-1/e \approx 0,63$ [5].

1.2. Построение таблиц операций. После выбора неприводимого полинома над полем F_p можно приступать к построению его расширения F_{p^k}. Для этого необходимо построить таблицы операций сложения и умножения для всех элементов поля. Элементами поля F_{p^k} являются полиномы степеней $0 \le s < k$ над полем F_p, а операции сложения и умножения в этом поле рассматриваются по модулю выбранного неприводимого полинома.

Пример 2. Пусть дано поле F_2 и неприводимый в этом поле полином $P(x) = x^2 + x + 1$. Таким образом, элементами этого поля будут полиномы степеней 0 и 1 над полем F_2, а следовательно, их совокупность можно обозначить $\{0, 1, x, x+1\}$. Для построения таблицы сложения выполним следующие вычисления:

$$0+0=0 \quad 1+0=1 \quad x+0=x \quad (x+1)+0=x+1$$
$$0+1=1 \quad 1+1=0 \quad x+1=x+1 \quad (x+1)+1=x$$
$$0+x=x \quad 1+x=x+1 \quad x+x=0 \quad (x+1)+x=1$$
$$0+(x+1)=x+1 \quad 1+(x+1)=x \quad x+(x+1)=1 \quad (x+1)+(x+1)=0.$$

Заметим, что операция сложения не предусматривает получения полиномов высших степеней, поэтому осуществляется через прибавление соответствующих коэффициентов в поле F_2 без необходимости вычисления остатков.

В результате выполненных вычислений получим таблицу сложения для поля F_{p^k} (табл. 4).

Таблица 4

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Если рассматривать элементы поля как функции от одной переменной $f(x)$, их нумерацию можно задать, вычислив для каждого элемента значение в точке $x = 2$. Так, элементы 0 и 1 в этом случае сохранят свое представление, а элементы x и $x+1$ будут обозначены числами 2 и 3 соответственно. В этом случае таблица сложения преобразуется к следующему виду (табл. 5).

Таблица 5

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

В случае умножения полиномов результатом будет остаток от деления, полученного в результате обычного произведения полинома на неприводимый полином x^2+x+1 (рассмотренные здесь сравнения вычисляются по модулю полинома x^2+x+1):

$$0\cdot 0=0 \qquad 1\cdot 0=0 \qquad x\cdot 0=0 \qquad (x+1)\cdot 0=0$$

$$0\cdot 1=0 \qquad 1\cdot 1=1 \qquad x\cdot 1=x \qquad (x+1)\cdot 1=x+1$$

$$0\cdot x=0 \qquad 1\cdot x=x \qquad x\cdot x=x^2\equiv x+1 \qquad (x+1)\cdot x=x^2+x\equiv 1$$

$$0\cdot (x+1)=0 \qquad 1\cdot (x+1)=x+1 \qquad x\cdot (x+1)=x^2+x\equiv 1 \qquad (x+1)\cdot (x+1)=x^2+1\equiv x.$$

В результате выполненных вычислений получим таблицу умножения для поля F_{p^k} (табл. 6) (справа — ее эквивалент с использованием указанной выше нумерации).

Таблица 6

×	0	1	x	$x+1$		×	0	1	2	3
0	0	0	0	0		0	0	0	0	0
1	0	1	x	$x+1$		1	0	1	2	3
x	0	x	$x+1$	1		2	0	2	3	1
$x+1$	0	$x+1$	1	x		3	0	3	1	2

Следует заметить, что предложенная выше нумерация не является единственно верной. Так, задать преобразование полиномов в числовое представление можно любым способом, который предполагает различные метки для разных элементов. ♠

Обобщая вышеуказанный пример, можно предложить следующий алгоритм вычисления таблиц операций:

```
function get-tables(p, P(x))
begin
  k = deg(P(x));
  elements = get-elements(p, k);
  len = length(elements);
  sum-table = array[len][len];
  product-table = array[len][len];
  for i = 0 to len – 1 do
  begin
    for j = 0 to len – 1 do
    begin
      sum-table[i][j] = poly+(elements[i], elements[j], p);
      product-table[i][j] = poly*(elements[i], elements[j], p) mod P(x);
    end;
  end;
  return sum-table;
end;
```

Здесь get-elements — генератор элементов поля F_{p^k}, а операции сложения и произведения между полиномами имеют вышеизложенный смысл.

Оценим сложность построения таблиц операций. Общее количество элементов в поле F_{p^k} определяется количеством полиномов степени $0 \le s < p$ и равно p^k. Таким образом, в предложенном выше алгоритме следует выполнить по p^{2k} операций сложения, умножения и деления (для вычисления остатка) двух полиномов.

Операция сложения выполняется за линейное время от степени полинома, а операции умножения и деления полиномов, в случае эффективной их реализации, имеют сложность порядка $O(k\log k\log\log k)$ каждая. Следовательно, сложность построения таблиц сложения и умножения для поля F_{p^k} имеет порядок $O(p^{2k}k\log k\log\log k)$ и требует $O(kp^{2k})$ памяти для хранения результатов в виде полиномов.

1.3. Противоположный и обратный элементы в поле F_{p^k}. Напомним, что противоположным элементу $a \in A$ в поле $F_{p^k} = (A, \Sigma)$ относительно операции сложения называется элемент $b \in A,$ для которого выполняется равенство $a + b = 0,$ где 0 — аддитивный ноль. В кольце вычетов Z_m противоположный элемент всегда существует и равен $b = (m - a)$ (здесь операция разницы имеет обычный арифметический смысл).

Обратным элементу $a \in A$ относительно операции умножения называется элемент $a^{-1} \in A,$ для которого выполняется равенство $a \cdot a^{-1} = 1,$ где 1 — мультипликативная единица. В кольце вычетов обратный элемент может не существовать, если оно с делителями нуля, т.е. модуль этого кольца — составное число, а если он существует, то его поиск производится с помощью расширенного алгоритма Евклида.

Поскольку поле F_{p^k} не содержит делителей нуля (ведь базовый полином, по которому оно строится, неприводим), для каждого его элемента существует обратный и его можно найти с помощью расширенного алгоритма Евклида для полиномов, т.е. требует в худшем случае $O(k)$ делений полиномов для нахождения остатка.

С другой стороны, поскольку операция сложения в этом поле теряет свой привычный арифметический смысл, нахождение противоположного элемента теперь требует большего количества операций. Так, если оперировать непосредственно самими полиномами, то для произвольного полинома $P(x) = a_0 + a_1 x + ...$ $... + a_{k-1} x^{k-1}$ над полем F_p обратным является полином $P^{-1}(x) = (p - a_0) +$ $+ (p - a_1)x + \ldots + (p - a_{k-1})x^{k-1}$, формирование которого требует $O(k)$ операций. В случае работы с численными представлениями полиномов, в общем случае противоположный элемент ищется полным перебором элементов, сложность которого $O(p^k)$.

Если же нумерация элементов поля заранее известна, алгоритм вычисления противоположного элемента можно упростить. Так, для нумерации, заданной с помощью подстановки в полином модуля p вместо $x,$ имея числовое представление полинома n, можно восстановить коэффициенты полинома и рассчитать порядковый номер противоположного элемента за $O(k)$ элементарных операций. Например, в поле F_{2^2} со стандартной нумерацией, имея номер n элемента, можно разложить его в сумму $n = [n/2] * 2 + n \bmod 2.$ Тогда противоположный элемент имеет номер $n^{-1} = ((2 - [n/2]) \bmod 2) * 2 + ((2 - n) \bmod 2) \bmod 2 = ([n/2] \bmod 2) * 2 +$ $+ n \bmod 2 = [n/2] * 2 + n \bmod 2 = n.$ Следовательно, для F_{2^2} каждый из элементов противоположен сам себе. В этом также можно убедиться, обратившись к таблице сложения, построенной выше.

2. Алгоритм решения СЛУ в поле F_{p^k}

Идея *TSS* -метода и его реализаций для различных областей рассматривалась и обосновывалась в работах [2–4], но для поля F_{p^k} такого обоснования не было. Поэтому приведем *TSS* -алгоритм и его обоснование для этой области.

Рассмотрим сначала одно линейное однородное уравнение (ЛОУ):

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0. \qquad (1)$$

Пусть $a_1 \neq 0$, найдем указанным выше способом противоположный ему элемент $c = -a_1$ и, комбинируя c с каждым ненулевым коэффициентом ЛОУ (1), построим множество таких векторов:

$$B = \{s_1 = (a_2, c, \ldots, 0, 0), \ldots, s_{n-1} = (a_n, 0, \ldots, 0, c)\} \cup M_0,$$

где M_0 — множество единичных векторов, соответствующих нулевым коэффициентам ЛОУ.

Имеет место лемма 1.

Лемма 1. Множество B является базисом множества всех решений ЛОУ (1), сложность построения которого $O(\max(n^2, T_{op}(p, k)))$, где $T_{op}(p, k)$ — сложность нахождения противоположного элемента.

Доказательство. Пусть $d = (d_1, d_2, \ldots, d_n)$ — произвольное решение ЛОУ (1). Рассмотрим вектор

$$d - c^{-1}d_2s_1 - c^{-1}d_3s_2 - \ldots - c^{-1}d_ns_{n-1} = ((d_1 - c^{-1}a_2d_2 - c^{-1}a_3d_3 - \ldots$$

$$\ldots c^{-1}a_nd_n), 0, \ldots, 0) = ((cd_1 - a_2d_2 - a_3d_3 - \ldots - a_nd_n)c^{-1}, 0, \ldots, 0) =$$

$$= ((-a_1d_1 - a_2d_2 - a_3d_3 - \ldots - a_nd_n)c^{-1}, 0, \ldots, 0) = \bar{0}$$

на том основании, что $c = -a_1$ и что d — решение ЛОУ (1). Отсюда получаем требуемое представление $d = c^{-1}d_2s_1 + c^{-1}d_3s_2 + \ldots + c^{-1}d_ns_{n-1}$, где c^{-1} — обратный элемент к c.

Сложность построения базиса составляют величины: поиск противоположного элемента — построения базисных решений, $O(n^2)$ и a — $T_{op}(p, k)$. Тогда общая сложность выражается величиной $O(\max(n^2, T_{op}(p, k)))$.

В зависимости от представления элементов поля F_{p^k} полиномами, полиномами с фиксированной нумерацией или произвольной нумерацией сложность этого построения выражается величинами $O(\max(n^2, k))$, $O(\max(n^2, k))$ или $O(\max(n^2, p^k))$ соответственно. ■

Пример 3. Найти базис множества всех решений ЛОУ $2x + 3y + 4z + 5u = 0$ в поле $F_9 = F_{3^2}$.

Возьмем $a_3 = 4$, его противоположный равен -8 в поле F_9 (см. табл. 2). Строим базис: $s_1 = (8, 0, 2, 0)$, $s_2 = (0, 8, 3, 0)$, $s_3 = (0, 0, 5, 8)$.

Нетрудно убедиться (используя табл. 2), что решением ЛОУ среди прочих будет вектор $d = (0, 1, 1, 1)$, представление которого в полученном базисе имеет вид

$$d = 6s_2 + 6s_3 = (0, 1, 5, 0) + (0, 0, 8, 1) = (0, 1, 5 + 8, 1) = (0, 1, 1, 1)$$

согласно таблицам сложения и умножения в поле F_9. ♠

Из леммы 1 очевидно вытекает следующий алгоритм:

```
function LHE(coefficients, field)
begin
  solution = [];
  while (i < length(coefficients) and coefficients[i] == 0)
  begin
    basis-vector = array[length(coefficients)];
    basis-vector[i] = 1;
    solution.append(basis-vector);
  end;
  if (i < length(coefficients))
  begin
    opposite = field.product-opposite(coefficients[i]);
    for j = i + 1 to length(coefficients) – 1 do
    begin
      basis-vector = array[length(coefficients)];
      if (coefficients[j] = 0)
      begin
        basis-vector[j] = 1;
      end;
      else
      begin
        basis-vector[j] = opposite;
        basis-vector[i] = coefficients[j];
      end;
      solution.append(basis-vector);
    end;
  end;
  return solution;
end;
```

Если дано линейное неоднородное уравнение (ЛНУ), то его решение требует построения базиса множества всех решений ЛОУ, которое ему соответствует, и нахождения частного решения. Частное решение находим из сравнения: $a_i x \equiv b$ в поле F_{p^k}, где $a_i \not\equiv 0$. Например, для ЛНУ $2x + 3y + 4z + 5u = 1$ получаем частное решение для $a_3 = 4$ путем решения сравнения $4x \equiv 1$ в поле F_9. Из таблиц операций этого поля находим решение $x = 3$ и частное решение ЛНУ имеет вид $s^1 = (0, 0, 3, 0)$.

Сложность решения вышеуказанного сравнения, эквивалентного операции нахождения обратного элемента, имеет сложность $O(k^2 \log k \log\log k)$ при условии оперирования полиномами и использования эффективных реализаций их умножения и деления, или же оперирования известной фиксированной нумерацией полиномов.

Общее решение ЛНУ $a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$ записывается в виде

$$s = s^1 + c_1 s_1 + \ldots c_{n_1} s_{n-1},$$

где $c_i \in F_{p^k}$, $s_i \in B$, s^1 — частное решение ЛНУ, B — базис множества всех решений ЛОУ, соответствующего ЛНУ.

Рассмотрим метод решения СЛОУ и СЛНУ.

2.1. Алгоритм решения СЛОУ в поле F_{p^k}. Рассмотрим СЛОУ

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1n}x_n = 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2n}x_n = 0, \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ L_q(x) = a_{q1}x_1 + \ldots + a_{qn}x_n = 0, \end{cases} \tag{2}$$

где $a_{ij}, b_i, x_i \in F_{p^k}$, $i = 1, \ldots, n$, $j = 1, \ldots, q$.

Построим базис $B_1 = \{e_1, \ldots, e_m\}$ множества всех решений первого ЛОУ системы S, описанным выше способом. Возьмем функцию $L_2(x) = a_{21}x_1 + \ldots + a_{2n}x_n$ и рассмотрим ЛОУ:

$$L_2(e_1)\,y_1 + L_2(e_2)\,y_2 + \ldots + L_2(e_m)\,y_m = 0. \tag{3}$$

Если все $L_2(e_i) = 0$, то $L_2(x)$ линейно выражается $L_1(x)$, и его можно удалить из СЛОУ S. Поэтому можно считать, что все уравнения в S линейно независимы.

Найдем базисные решения $B' = \{r_1, r_2, \ldots, r_{m-1}\}$ ЛОУ (3) и построим по этим векторам из B' соответствующие линейные комбинации векторов из B_1. Обозначим полученное множество $M = \{s_1, s_2, \ldots, s_{m-1}\}$.

Лемма 2. Множество *M* является базисом множества всех решений СЛОУ:

$$S_1 = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1n}x_n = 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2n}x_n = 0. \end{cases} \tag{4}$$

Доказательство. Очевидно, что все элементы M — решения СЛОУ S_1. Пусть $x = (x_1, \ldots, x_n)$ — произвольное решение СЛОУ S_1, тогда x, будучи решением первого уравнения, имеет представление $x = d_1e_1 + \ldots + d_me_m$, где $e_i \in B_1$, $i = 1, \ldots, m$. Подставляя x в $L_2(x)$, получаем ЛОУ:

$$d_1L_2(e_1) + \ldots + d_mL_2(e_m) = c_1d_1 + \ldots + c_md_m = 0,$$

т.е. вектор $(d_1, d_2, \ldots, d_m)$ является решением ЛОУ (3) и, следовательно, представляется в виде неотрицательной линейной комбинации векторов из B':

$$(d_1, \ldots, d_m) = f_1r_1 + \ldots + f_{m-1}r_{m-1}.$$

Но тогда

$$x = d_1e_1 + \ldots + d_me_m = f_1s_1 + \ldots + f_{m-1}s_{m-1},$$

а это значит, что x представляется в виде неотрицательной линейной комбинации векторов из M. В силу произвольности вектора x лемма справедлива. ■

Теорема 1. Множество решений *M*, построенное вышеописанным способом для СЛОУ (2), является базисом множества всех решений этой СЛОУ.

Сложность построения базиса пропорциональна величине $q\max(n^2, T_{op}(p, k))$, где q — число уравнений СЛОУ, n — число неизвестных СЛОУ, а $T_{op}(p, k)$ — сложность нахождения противоположного элемента.

Доказательство выполняется индукцией по числу q уравнений в СЛОУ S.

Базис индукции при $q = 2$ имеет место в силу леммы 2.

Шаг индукции. Предположим, что теорема справедлива для всех $k < q$. Тогда множество M решений СЛОУ S', которая состоит из первых $q - 1$ уравнений, в силу предположения индукции является базисом множества решений S'.

Повторяя выкладки применительно к M и $L_q(x) = 0$, аналогичные тем, которые использовались при доказательстве леммы 2, получаем справедливость теоремы.

Оценка временной сложности, которая приведена в формулировке теоремы, очевидным образом следует из леммы 1. ■

Пример 4. Найти базис множества всех решений СЛОУ:

$$S = \begin{cases} 2x_1 + 1x_2 + 3x_3 + 5x_4 + 2x_5 = 0, \\ 0x_1 + 3x_2 + 6x_3 + 1x_4 + 8x_5 = 0, \\ 7x_1 + 1x_2 + 4x_3 + 0x_4 + 5x_5 = 0 \end{cases}$$

в поле $F_9 = F_{3^2}$ с операциями из табл. 2.

Базисными решениями первого уравнения системы с противоположным -1 для коэффициента $a_{11} = 2$ есть

$$B_1 = \{(1, 1, 0, 0, 0), (3, 0, 1, 0, 0)\ (5, 0, 0, 1, 0), (2, 0, 0, 0, 1)\}.$$

Значения второго уравнения на этих решениях: 3, 6, 1, 8. Составляем сравнение $3x + 6y + z + 8u = 0,$ решениями которого с противоположным -6 для коэффициента $b_1 = 3$ есть (6, 6, 0, 0), (1, 0, 6, 0), (8, 0, 0, 6). Линейные комбинации векторов из базиса B_1 принимают вид

$$B_2 = \{(2, 6, 6, 0, 0), (6, 1, 0, 6, 0)\ (2, 8, 0, 0, 6)\}.$$

Значения на этих решениях: 1, 5, 6. Составляем сравнение $x + 5y + 6z = 0,$ решением которого с противоположным -2 для коэффициента $b_1 = 1$ есть (5, 2, 0) и (6, 0, 2). Линейные комбинации за этими решениями векторов из B_2 дают базисные решения СЛОУ S:

$$s_1 = (1, 7, 8, 3, 0),\ s_2 = (4, 2, 7, 0, 3).\ \spadesuit$$

Из теоремы 1 вытекает такой алгоритм:

```
function SLHE(equations, field)
begin
  solution = LHE(equations[0], field);
  for I = 1 to length(equations) do
  begin
    coefficient-equation = [];
    for j = 0 to length(solution) – 1 do
    begin
      result-on-equation = equations[I](solution[j]);
      coefficient-equation.append(result-on-equation);
    end;
    coefficient-equation-solution = LHE(coefficient-equation, field);
    new-solution = [];
    for j = 0 to length(coefficient-equation-solution) – 1 do
    begin
      new-solution.append(combine(solution, coefficient-equation-solution));
    end;
    solution = new-solution;
  end;
  return solution;
end;
```

2.2. Алгоритм решения СЛНУ. Пусть дана СЛНУ

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1n}x_n = b_1, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2n}x_n = b_2, \\ \ldots \quad \ldots \quad \ldots \quad \ldots \ldots \quad \ldots \quad \ldots \quad \ldots \\ L_q(x) = a_{q1}x_1 + \ldots + a_{qn}x_n = b_q, \end{cases} \tag{5}$$

где $a_{ij}, b_i, x_i \in F_p, i = 1, \ldots, n,\ j = 1, \ldots, q,\ q < n.$

Поскольку общее решение СЛНУ состоит из частного решения и линейной комбинации базисных решений соответствующей ей СЛОУ, задача сводится к нахождению частного решения СЛНУ. Для этого преобразуем систему к виду

$$S' = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1n}x_n - b_1x_0 = 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2n}x_n - b_2x_0 = 0, \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ L_q(x) = a_{q1}x_1 + \ldots + a_{qn}x_n - b_qx_0 = 0. \end{cases} \tag{6}$$

Найдем базис $B' = \{s_{1'}, s_{2'}, \ldots, s_{n'}\}$ множества всех решений СЛОУ (6). Выделим из этих решений то, у которого последняя координата (соответствующая значению дополнительной неизвестной x_0) не равна нулю. Если таких решений нет, то исходная систем S несовместна. Если же эта координата равна единице, то данное решение без последней координаты является частным решением СЛНУ (5). Если эта координата не равна единице, то для поиска частного решения S нужно решить сравнение $ax \equiv 1$, где a — значение последней координаты выбранного решения. Далее, умножив все координаты этого решения на найденное значение x и удалив последнюю координату, получим частное решение СЛНУ S.

Отсюда легко найти оценку временной сложности алгоритма построения общего решения СЛНУ в поле F_{p^k}. Действительно, к оценке, полученной в предыдущей теореме, добавляется оценка вычисления частного решения СЛНУ. Эта оценка включает в себя сложность решения сравнения $ax \equiv b$ и построения вектора, являющегося частным решением. Сложность решения этого сравнения оценивается выражением $O(k^2 \log k \log\log k)$ при условии оперирования полиномами и использования эффективных реализаций их умножения и деления, или же оперирования известной фиксированной нумерацией полиномов. Если известна лишь таблица произведения, а способ нумерации элементов произвольный, то сложность сравнения пропорциональна p^k.

Таким образом, имеет место следующая теорема.

Теорема 2. Пусть $s' = (a_1, a_2, \ldots, a_n, a)$ — базисное решение СЛОУ (6). Если существует решение сравнения $ax \equiv 1 \pmod{p^k}$, то СЛНУ (5) совместна над полем F_{p^k}, и ее общее решение имеет вид $u = x^1 + \sum_{i=1}^{k} b_i e_i$, где x^1 — частное решение (5), e_i — базисные векторы множества решений СЛОУ, которая соответствует данной СЛНУ.

Временная сложность построения общего решения СЛНУ (5) в общем случае имеет порядок $O(q(\max(n^2, T_{op}(p,k), T_{inv}(p,k))))$, где q — количество уравнений в системе, n — число неизвестных в СЛНУ, а $T_{op}(p,k)$ и $T_{inv}(p,k)$ — сложность поиска противоположного и обратного элементов соответственно.

Пример 5. Найти общее решение СЛНУ:

$$S = \begin{cases} 2x_1 + 1x_2 + 3x_3 + 5x_4 + 2x_5 = 1, \\ 0x_1 + 3x_2 + 6x_3 + 1x_4 + 8x_5 = 2, \\ 7x_1 + 1x_2 + 4x_3 + 0x_4 + 5x_5 = 3 \end{cases}$$

в поле $F_9 = F_{3^2}$ с операциями из табл. 2.

Преобразовывая СЛНУ к СЛОУ так, как в доказательстве теоремы 2, получаем СЛОУ:

$$S_1 = \begin{cases} 2x_1 + 1x_2 + 3x_3 + 5x_4 + 2x_5 - 1x_0 = 0, \\ 0x_1 + 3x_2 + 6x_3 + 1x_4 + 8x_5 - 2x_0 = 0, \\ 7x_1 + 1x_2 + 4x_3 + 0x_4 + 5x_5 - 3x_0 = 0, \end{cases}$$

которая после замены отрицательных значений их противоположными, преобразуется к виду

$$S_1 = \begin{cases} 2x_1 + 1x_2 + 3x_3 + 5x_4 + 2x_5 + 2x_0 = 0, \\ 0x_1 + 3x_2 + 6x_3 + 1x_4 + 8x_5 + 1x_0 = 0, \\ 7x_1 + 1x_2 + 4x_3 + 0x_4 + 5x_5 + 6x_0 = 0. \end{cases}$$

Находим базис множества всех решений этой СЛОУ так, как описано выше. В данном случае базисными решениями являются векторы

$$B' = \{s_1 = (1, 7, 8, 3, 0, 0),\ s_2 = (4, 2, 7, 0, 3, 0)\ s_3 = (6, 5, 3, 0, 0, 3)\}.$$

Поскольку последняя компонента одного из векторов ненулевая, система совместна. Первые два решения без последней координаты составляют базис множества решений СЛОУ, которая соответствует СЛНУ S.

Вектор с ненулевой последней координатой позволяет найти частное решение СЛНУ. Из табл. 2 следует, что обратным элементу 3 является элемент 4, поэтому частное решение вычисляется следующим образом: $x^1 = 4 \cdot s_3 = 4 \cdot (6, 5, 3, 0, 0) =$ $= (2, 6, 1, 0, 0)$.

Итак, общее решение СЛНУ S принимает вид

$$x = x^1 + as_1 + bs_2, \text{ где } a, b \in F_{p^k}. \spadesuit$$

2.3. Реализация *TSS* -алгоритма. Описанные выше алгоритмы реаизованы в программном обеспечении «Решение СЛНУ в поле F_{p^k}», созданном с помощью языка программирования C++ и инструментария разработки оконных приложений Qt для этого языка. Программа расчитана на операционную систему Windows и позволяет выполнять поиск неприводимых полиномов в поле F_{p^k}, расчет таблиц сложения и умножения для этого поля, а также решать произвольные системы линейных уравнений в этом поле, представляя решение через перечень базисных векторов и вектор — частное решение системы.

Среднее время вычисления таблиц сложения и умножения поля F_{p^k} для некоторых значений p и k представлено в табл. 7. При замерах времени использовался персональный компьютер с операционной системой Windows 10, процессором Intel Core i5 частотой 2.71 ГГц и оперативной памятью 6 Гб.

Таблица 7

p	$k = 2$, с	$k = 3$, с	$k = 4$, с
$p = 2$	≈ 0	≈ 0	≈ 0
$p = 3$	≈ 0	0,001	0,011
$p = 5$	0,010	0,016	0,381
$p = 7$	0,007	0,096	5,348
$p = 11$	0,021	1,781	458

Среднее время решения СЛНУ для разных количеств уравнений m и переменных n над полем F_{13^3} указано в табл. 8.

Таблица 8

m	$n = 100$, с	$n = 150$, с	$n = 200$, с	$n = 300$, с
$m = 50$	2,72	5,02	9,22	27,72
$m = 100$	3,84	8,46	16,78	69,82
$m = 150$	–	9,81	29,63	93,18
$m = 200$	–	–	35,61	119,38

3. Применение: решение задачи о математическом сейфе

Рассмотрим использования предложенного TSS -алгоритма к решению одной игровой задачи.

Математическим сейфом называется система $Z = (z_1, z_2, \ldots, z_n)$ взаимосвязанных замков такая, что, когда поворачивается ключ в одном замке, это же происходит и в замках, связанных с данным [1]. В рамках статьи заменим традиционный для этой задачи поворотный замок, осуществляющий прибавление к текущим позициям связанных замков единицы, циферблатом с p^k кнопками на каждом замке, где кнопка i прибавляет к текущему состоянию число i .

Математический сейф может задаваться двумя способам: с помощью прямоугольной матрицы, элементы которой соответствуют замкам, а значения ее элементов — позициям замков, т.е. в виде матрицы $Z = \| z_{ij} \|$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, и с помощью графа, вершины которого соответствуют замкам. При матричном задании математического сейфа каждый замок z_{ij} взаимосвязан с замками, расположенными в той же строке и том же столбце. А при задании математического сейфа с помощью графа связанными с данным замком являются те замки, которые соответствуют замкам, расположенным в смежных вершинах. Исходное состояние сейфа Z задается матрицей $B = \| b_{ij} \|$. Каждый из замков может находиться в одной из нескольких позиций. Всех возможный позиций конечное число: $0, 1, \ldots, k-1$. Замок открыт, когда он находится в позиции 0. В любой другой позиции он считается закрытым. При этом, если в каком-то замке осуществляется нажатие на кнопку i, то все замки, связанные с данным замком, увеличивают свои позиции на элемент i в соответствии со смыслом операции сложения.

Перейдем к решению следующей задаче. Исходя из начального состояния сейфа, представленного матрицей $B = \| b_{ij} \|$, где $b_{ij} \in \{0, 1, \ldots, k-1\}$, найти такую последовательность нажатия кнопок на циферблате, чтобы сейф перешел в положение открытого, т.е. когда все замки находятся в позиции 0. Рассмотрим сначала задание математического сейфа с помощью матрицы.

3.1. Решение задачи о математическом сейфе на матрицах. Пусть матрица $X = \| x_{ij} \|$ — решение задачи, где x_{ij} соответствует кнопке в замке z_{ij}, которая увеличивает положение замка на элемент x_{ij}. Тогда условием того, что элемент b_{ij} преобразуется матрицей X в нуль, представляется соотношением

$$\sum_{s=1}^{n} x_{is} + \sum_{s=1, s \neq i}^{m} x_{sj} + b_{ij} = 0, \qquad (7)$$

где $i = 1, \ldots, m$, $j = 1, \ldots, n$, а операции сложения выполняются в соответствии с таблицей сложения в поле F_{p^k}.

ISSN 0572-2691

Обозначим $\bar{x} = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, x_{m1}, \ldots, x_{mn})$ вектор-столбец, полученный из матрицы X последовательной записью ее строк. Аналогично из матрицы B получим вектор-столбец $\bar{b}$. Кроме того, пусть J_n — матрица размера $n \times n$, состоящая из единиц, E_n — единичная матрица того же размера. Тогда решение задачи о математическом сейфе сводится к преобразованию (7) для всей матрицы B и записывается в виде системы уравнений

$$A\bar{x} + \bar{b} = 0, \tag{8}$$

где матрица A размера $mn \times mn$ состоит из m^2 клеток:

$$A = \begin{bmatrix} J_n & E_n & E_n & \cdots & \cdots & E_n \\ E_n & J_n & E_n & \cdots & \cdots & E_n \\ E_n & E_n & J_n & \cdots & \cdots & E_n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ E_n & E_n & E_n & \cdots & \cdots & J_n \end{bmatrix}. \tag{9}$$

Поскольку открытие сейфа $m \times n$ сводится к решению СЛНУ с mn уравнениями и mn неизвестными, вычислительная сложность этой задачи в поле F_{p^k} имеет порядок $O(mn(\max((mn)^2, T_{op}(p,k), T_{inv}(p,k))))$, где $T_{op}(p,k)$ и $T_{inv}(p,k)$ — сложность нахождения противоположного и обратного элементов соответственно.

Когда известен способ нумерации элементов поля и по номеру есть возможность восстановить соответствующий ему полином, то сложность решения задачи $O(\max((mn)^3,\ mnk^2 \log k \log\log k))$. Если же способ нумерации полиномов поля неизвестен, сложность увеличивается до значения $O(\max((mn)^3, mnp^k))$.

Пример 6. Найти решение задачи о математическом сейфе в поле F_{3^2} с операциями из табл. 2 с исходным состоянием

$$A = \begin{pmatrix} 8 & 4 & 4 \\ 0 & 7 & 8 \end{pmatrix}. \tag{10}$$

В этом случае $\bar{b} = (8, 4, 4, 0, 7, 8)$ и система $Ax + \bar{b} = 0$ в поле F_{3^2} принимает вид

$$Ax + \bar{b} = \begin{cases} 1\ 1\ 1\ 1\ 0\ 0\ 8 \\ 1\ 1\ 1\ 0\ 1\ 0\ 4 \\ 1\ 1\ 1\ 0\ 0\ 1\ 4 \\ 1\ 0\ 0\ 1\ 1\ 1\ 0 \\ 0\ 1\ 0\ 1\ 1\ 1\ 7 \\ 0\ 0\ 1\ 1\ 1\ 1\ 8 \end{cases} \equiv 0.$$

Применив TSS-алгоритм для решения этой системы, получим базис, состоящий из единственного вектора $s = (8, 3, 4, 5, 1, 1, 2)$.

Чтобы найти комбинацию, которая открывает сейф, нужно привести s к виду, в котором его последняя координата равна единице. Из таблицы умножения находим, что элемент 2 является обратным самому себе, следовательно, искомая комбинация принимает вид $(8 \cdot 2, 3 \cdot 2, 4 \cdot 2, 5 \cdot 2, 1 \cdot 2, 1 \cdot 2) = (4, 6, 8, 7, 2, 2)$. Убедимся, что эта комбинация действительно открывает данный сейф:

$$\begin{pmatrix} 8 & 4 & 4 \\ 0 & 7 & 8 \end{pmatrix} \to (x_{11} = 4) \begin{pmatrix} 0 & 8 & 8 \\ 4 & 7 & 8 \end{pmatrix} \to (x_{12} = 6) \begin{pmatrix} 6 & 5 & 5 \\ 4 & 4 & 8 \end{pmatrix} \to (x_{13} = 8) \begin{pmatrix} 5 & 1 & 1 \\ 4 & 4 & 4 \end{pmatrix} \to$$

$$\to (x_{21} = 7) \begin{pmatrix} 0 & 1 & 1 \\ 2 & 2 & 2 \end{pmatrix} \to (x_{22} = 2) \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \to (x_{23} = 2) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \spadesuit$$

Для решения задачи о сейфе в этом поле необходимо выполнение условия: если $m+n-1 \equiv 0 \pmod p$, то $\sum\limits_{i=1}^{mn-1} b_i = 0,$ где сумма вычисляется по таблице сложения поля F_{p^k}. Действительно, если $m+n-1 \equiv 0 \pmod p$, то сумма уравнений системы (8) равна нулю по этому модулю, а сумма свободных членов отлична от нуля, из чего следует справедливость условия.

Пример 7. Найти решение задачи о математическом сейфе в поле F_{2^2} с операциями из табл. 1 с исходным состоянием

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \end{pmatrix}.$$

В этом случае $\bar b = (1, 2, 3, 0, 1, 2)$, $m+n-1 = 4 \equiv 0 \pmod 2$, $\sum\limits_{i=1}^{6} b_i = 3$ (в поле F_{2^2}). Следовательно, система несовместна и решения не существует. ♠

3.2. Вариация задачи о сейфе. Одна из вариаций задачи о сейфе в поле F_{p^k} — случай, когда сейф считается открытым при определенной комбинации замков. Это значит, что система (8) принимает вид

$$A\bar x + \bar b = \bar c, \tag{11}$$

где $\bar c$ — состояния замков, при которых сейф будет открыт.

Очевидно, что данная задача сводится к рассмотренной выше, поскольку система (11) сводится к СЛОУ

$$A\bar x + \bar d = 0, \tag{12}$$

где $\bar d = \bar b + (-\bar c)$. Здесь $-\bar c$ — вектор противоположных элементов к соответствующим компонентам вектора $\bar c$, а сложение выполняется по таблице поля F_{p^k}.

Криптографическая ценность этой вариации состоит в следующем. Если комбинация замков $\bar c$ заранее неизвестна, то, имея в распоряжении начальное состояние сейфа для его открытия, следует решить СЛНУ (12) для каждого варианта конечной комбинации. Таким образом, поскольку для каждого замка возможны p^k значений, решать систему в худшем случае приходится p^{kmn} раз, что делает этот вариант решения задачи более сложным. Если же комбинация $\bar c$ известна, сложность решения в худшем случае оценивается, как было сказано выше, выражением $O(\max((mn)^3, mnp^k))$, но с использованием соответствия номеров полиномов сложность можно уменьшить до $O(\max((mn)^3, mnk^2 \log k \log\log k))$.

3.3. Решение задачи о математическом сейфе на графах. Предположим, что в вершинах графа $G = (V, E)$ находятся замки, которые могут быть в одной из p^k позиций, каждая из которых соответствует некоторому элементу поля F_{p^k}.

Если в вершине u замок находится в позиции i, нажатие кнопки j переводит его, а также все замки, находящиеся в смежных вершинах, в позицию $i + j$ в соответствии с таблицей сложения поля F_{p^k}. Исходные позиции замков в вершинах задаются вектором $\bar{b} = (b_1, \ldots, b_{|V|})$.

Решение задачи в этом случае выполняется с помощью тех же алгоритмов, что и при задании с помощью матриц. Но следует заметить, что в данном случае матрица СЛНУ $Ax + \bar{b} = \bar{c}$ над полем F_{p^k} может быть произвольной и является матрицей смежности графа G: элемент a_{ij} равен единице, если существует ребро из j в i или $i = j$, и нулю в противном случае.

Рассмотрим некоторые примеры.

Пример 8. Найти решение задачи о математическом сейфе в поле F_{3^2} с операциями из табл. 2 с исходным состоянием $\bar{b} = (7, 2, 2, 8, 4)$ и графом замков $G = (V, E)$, представленным ниже (рис. 1).

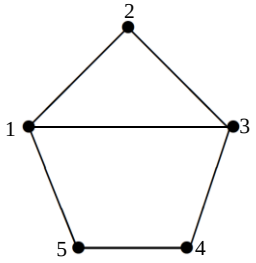

Рис. 1

В этом случае система $Ax + \bar{b} = 0$ в поле F_{3^2} принимает вид

$$Ax + \bar{b} = \begin{cases} 1\ 1\ 1\ 0\ 1\ 7 \\ 1\ 1\ 1\ 0\ 0\ 2 \\ 1\ 1\ 1\ 1\ 0\ 2 = 0. \\ 0\ 0\ 1\ 1\ 1\ 8 \\ 1\ 0\ 0\ 1\ 1\ 4 \end{cases}$$

Применив *TSS*-алгоритм для решения этой системы, получим базис, состоящий из единственного вектора $s = (8, 3, 0, 0, 8, 2)$.

Из табл. 2 видно, что элемент 2 является обратным самому себе, следовательно, искомая комбинация принимает вид $(8 \cdot 2, 3 \cdot 2, 0, 0, 8 \cdot 2) = (4, 6, 0, 0, 4)$. Убедимся что эта комбинация действительно открывает наш сейф:

$$\begin{pmatrix} 7 \\ 2 \\ 2 \\ 8 \\ 4 \end{pmatrix} \to (x_1 = 4) \begin{pmatrix} 2 \\ 3 \\ 3 \\ 8 \\ 8 \end{pmatrix} \to (x_2 = 6) \begin{pmatrix} 8 \\ 0 \\ 0 \\ 8 \\ 8 \end{pmatrix} \to (x_5 = 4) \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \spadesuit \qquad (13)$$

Пример 9. Найти решение задачи о математическом сейфе в поле F_{3^2} с операциями из табл. 2 с исходным состоянием $\bar{b} = (2, 2, 4, 3, 8)$ и орграфом замков, представленным ниже, при условии, что сейф открывается при состояниях $\bar{c} = (3, 6, 3, 8, 5)$.

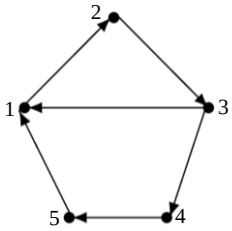

Рис. 2

В этом случае система $Ax + \bar{b} = 0$ в поле F_{3^2} принимает вид

$$Ax + \bar{b} = \begin{Bmatrix} 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 1 & 0 & 0 & 2 \\ 1 & 0 & 1 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 1 & 3 \\ 1 & 0 & 0 & 0 & 1 & 8 \end{Bmatrix} = \begin{pmatrix} 3 \\ 6 \\ 3 \\ 8 \\ 5 \end{pmatrix}.$$

Применив *TSS* -алгоритм для решения этой системы, получим базис, состоящий из двух векторов: $s_1 = (1, 2, 1, 1, 2, 0)$ и $s_1 = (6, 7, 0, 5, 0, 1)$.

Убедимся, что комбинация $(6, 7, 0, 5, 0)$ действительно открывает сейф:

$$\begin{pmatrix} 2 \\ 2 \\ 4 \\ 3 \\ 8 \end{pmatrix} \to (x_1 = 6) \begin{pmatrix} 8 \\ 2 \\ 1 \\ 3 \\ 5 \end{pmatrix} \to (x_2 = 7) \begin{pmatrix} 3 \\ 6 \\ 1 \\ 3 \\ 5 \end{pmatrix} \to (x_4 = 5) \begin{pmatrix} 3 \\ 6 \\ 3 \\ 8 \\ 5 \end{pmatrix}. \spadesuit$$

Заключение

В статье рассмотрены эффективные алгоритмы поиска неприводимого полинома над полем вычетов и построения таблиц сложения и умножения поля F_{p^k} по найденному неприводимому полиному.

Предложен *TSS* -алгоритм решения системы из q линейных уравнений с n неизвестными в поле F_{p^k}, который позволяет находить решение в худшем случае за время $O(q(\max(n^2, p^k)))$, которое можно уменьшить до $O(q(\max(n^2, k^2 \log k \log\log k)))$ при условии работы непосредственно с самими полиномами или с заблаговременно известной их нумерацией.

Системы линейных уравнений в поле F_{p^k} могут использоваться для решения игровых задач, включая задачу о математическом сейфе, в задачах кодирования и криптографии. В кодировании и криптографии чаще всего используются кольца и поля с модулем 2^n [7, 8]. Это связано со спецификой задач в этих областях. Предложенные алгоритмы могут применяться и в этих областях.

С.Л. Кривий, Г.І. Гогерчак

АЛГОРИТМ РОЗВ'ЯЗКУ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ У ПОЛІ F_{p^k}

Розглянуто базові теоретичні поняття в області скінченних полів, зокрема поняття поля залишків та розширення поля залишків. Наведено комплекс алгоритмів, необхідних для побудови розширень полів залишків: тест Рабіна для перевірки поліномів на незвідність, його використання для пошуку незвідних поліномів, алгоритм побудови таблиць додавання та множення за модулем незвідного полінома, шляхи обчислення протилежного та оберненого елементів на основі цих таблиць. Запропоновано шляхи покращення ефективності пошуку незвідних поліномів з використанням імовірнісного підходу. Описано особливості нумерації елементів розширень скінченних полів F_{p^k} і вплив вибору нумерації на ефективність виконання базових операцій над елементами поля, зокрема для пошуку протилежного і оберненого елементів. Запропоновано алгоритм побудови базису множини розв'язків систем лінійних однорідних рівнянь та алгоритм побудови загального розв'язку системи неоднорідних рівнянь над розширенням скінченного поля F_{p^k} у вигляді суми комбінації векторів базису множини розв'язків відповідної однорідної системи та часткового розв'язку неоднорідної системи. Алгоритми розв'язання систем рівнянь мають поліноміальні оцінки часової складності, які наведено в таблицях для конкретних прикладів систем лінійних рівнянь та різних параметрів алгоритмів. Розглянуто застосування запропонованих алгоритмів розв'язання систем лінійних рівнянь до задачі про математичний сейф, її класичне формулювання та адаптацію до поля F_{p^k}. Описано варіанти представлення математичного сейфа: за допомогою матриць та графів. Наведено умови існування розв'язку, алгоритми розв'язання задачі в цих випадках та їх ефективність в полі F_{p^k}. Вказано можливі шляхи застосування систем рівнянь в полі F_{p^k} в кодуванні та криптографії.

Ключові слова: системи діофантових рівнянь, розширення поля, задача про математичний сейф.

S.L. Kryvyi, H.I. Hoherchak

ALGORITHM FOR SOLVING SYSTEMS OF LINEAR EQUATIONS IN FIELD F_{p^k}

Basic theoretical concepts of finite fields area are considered, including concepts of residue field and extension of residue field. The algorithms necessary for constructing extensions ef residue fields are given: a Rabin test for checking irreducibility of polynomials, its application to irreducible polynomials search, algorithm for construction of addition and multiplication tables by modulo of irreducible polynomial, ways of opposite and inverse elements calculation based on these tables. Ways of efficiency improvement for irreducible polynomials search with probabilistic approach are introduced. The features of the numbering of elements extensions of finite fields F_{p^k} and numbering choice influence on the efficiency of performing basic operations on field elements, including search for opposite and inverse elements, are described. An algorithm for constructing a basis for a set of solutions of homogeneous systems of linear equations and an algorithm for building of common solution ef inhomogeneous systems of linear eqations over a finite field F_{p^k} as a sum of linear combination of corresponding

homogeneous system set of solutions basis and partial solution of inhomogeneous system are presented. Proposed algorithms have polinomial estimations of time complexity as demonstrated by tables for different systems of equations and different parameters of these algorithms. Applications for systems of linear equations in the mathematical safe problem is considered. Classic formulation of this problem and its adaptation for the field F_{p^k} is proposed. Various cases of the representation of a mathematical safe: using matrixes and graphs are described. Conditions for solution existence, algorithms for solving a problem in these cases, and their effectiveness in the field F_{p^k} are considered. Possible applications of systems of equations over the field F_{p^k} in coding and cryptography are indicated.

Keywords: systems of diophantine equations, field extension, math safe problem.

1. Донец Г.А. Решение задачи о сейфе на (0, 1)-матрицах. *Кибернетика и системный анализ.* 2002. № 1. С. 98–105.
2. Крывый С.Л. Алгоритмы решения систем линейных диофантовых уравнений в полях вычетов. *Кибернетика и системный анализ.* 2007. № 2. С. 15–23.
3. Кривий С.Л. Лінійні діофантові обмеження та їх застосування. Чернівці-Київ : Букрек, 2015. 224 с.
4. Сергієнко І.В., Кривий С.Л., Провотар О.І. Алгебраїчні аспекти інформаційних технологій (ч. 1). Київ : Інтерсервіс, 2018. 410 с.
5. Lidl R., Niederreiter H. Finite fields. 2nd edition. Cambridge University Press. 1996. P. 755. DOI https://doi.org/10.1017/CBO9780511525926.
6. Rabin M.O. Probabilistic algorithms in finite fields. *SIAM Journal on Computing.* 1980. N 9(2). P. 273–280. DOI https://doi.org/10.1137/0209024.
7. Гаврилкевич М.В., Солодовников В.И. Эффективные алгоритмы решения задач линейной алгебры над полем из двух элементов. *Обозрение прикл. промышл. матем.* 1995. **2**. Вып. 3. С. 400–437.
8. Алексейчук А.Н., Игнатенко С.М. Метод оптимизации алгоритмов решения систем линейных уравнений с искаженной правой частью над кольцом вычетов по модулю 2^n. *Реєстрація, зберігання і обробка даних.* 2005. **7**, № 1. С. 21–29.

Получено 05.03.2019
После доработки 24.05.2019