МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»
Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему Исследование систем линейных запретов над конечным полем

Обучающийся И.С. Кучерявый
(Инициалы Фамилия) (личная подпись)

Научный руководитель канд. физ.-мат. наук О.В. Лелонд
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Содержание

Перечень условных значений, сокращений и терминов

B_n — множество булевых функций от n переменных

$\langle f \rangle$ — идеал функции f

$Ann\ \langle f \rangle$ — аннулятор идеала функции f

$poly(x)$ — полином от переменной x

$\leq_p$ — полиномиальная сводимость

$\leq_T$ — сводимость по Тьюрингу

$\mathbb{F}_{2^k}$ — поле Галуа, состоящее из 2^k элементов

$\mathbb{F}_{2^k}^*$ — мультипликативная группа поля $\mathbb{F}_{2^k}$

$\mathbb{F}_{2^k}^n$ — множество векторов длины n над полем $\mathbb{F}_{2^k}$

(x, y) — скалярное произведение векторов x и y

$|A|$ — мощность множества A

XOR — добавление по модулю 2

0 — нулевой элемент поля $\mathbb{F}_{2^k}$

1 — единичный элемент поля $\mathbb{F}_{2^k}$

$\overline{0}$ — вектор из нулей над полем $\mathbb{F}_{2^k}$

$\overline{1}$ — вектор из единицы над полем $\mathbb{F}_{2^k}$

$\exists$ — квантор существования

$\forall$ — квантор общности

$\langle a, a_0 \rangle$ — упорядоченная пара, задающая линейный запрет $(a, x) \neq a_0$

$diag(\gamma_1, \gamma_2, \ldots, \gamma_m)$ — диагональная матрица с элементами $\gamma_1, \gamma_2, \ldots, \gamma_m$

$x_{1:n}$ — первые n компонент вектора x

КНФ — конъюнктивная нормальная форма

Введение

Актуальность темы. Стандартные модели алгебраического криптоанализа рассматривают зависимость между открытыми текстами, шифротекстами и ключами в виде системы полиномиальных уравнений над конечным полем. Можно рассмотреть альтернативную задачу, в которой известны лишь ограничения на возможные значения некоторых зависимостей с неизвестными параметрами. Исследование такой задачи является целесообразным, поскольку существует много методов, которые предоставляют возможность получить частичную информацию о промежуточных значениях некоторых параметров в процессе шифрования. Эти методы могут указывать на то, что определенные зависимости с неизвестными параметрами не могут принимать некоторый конечный набор значений. Такая информация может быть получена из побочного канала связи или из особенностей реализации криптосистемы. Учитывая это, возникает задача восстановления неизвестного вектора по частичной информации, представленной в форме определенных линейных зависимостей.

Цель и задачи исследования. Целью исследования является развитие и уточнение алгебраических моделей и методов криптоанализа. Для достижения цели необходимо решить следующие задачи исследование:

- сформулировать определение системы линейных запретов и задачи ее решения; исследовать алгебраические и комбинаторные свойства систем линейных запретов;
- рассмотреть системы линейных запретов с гарантированно однозначным решением и исследовать вопросы восстановления решения таких систем;
- предложить алгоритмы решения частных случаев систем линейных запретов;
- исследовать сводимость задачи решения системы линейных запретов к известным теоретико-сложным задачам; установить класс

сложности данной задачи.

Объектом исследования являются информационные процессы в системах криптографической защиты.

Предметом исследования является система линейных запретов и ее свойства.

Научная новизна. В работе получены следующие результаты:

– определен критерий существования решения системы линейных запретов. Построен полиномиальный вероятностный алгоритм проверки существование решения для некоторых частичных случаев. Доказано свойства решений системы линейных запретов в случае нулевых правых частей;
– сформулирован и доказан ряд утверждений о количестве решений системы линейных запретов в случае, когда система линейных запретов порождена фиксированным неизвестным вектором. Получена нетривиальная оценка точки насыщения в случае ненулевых правых частей системы;
– сформулированы задачи проверки существования и поиска решения системы линейных запретов. Доказана эквивалентность по Тьюрингом этих задачу. Построен вероятностный алгоритм поиска решения системы линейных запретов в некоторых частичных случаях. Построен вероятностный эвристический алгоритм поиска нескольких решений системы линейных запретов в некоторых частичных случаях. Сформулирован ряд смежных задач и доказана принадлежность этих задач соответствующим классам сложности.

Практическое значение. Результаты данной работы могут быть использованы как самостоятельным, так и вспомогательным инструментом для алгебраического криптоанализа потоковых шифров и криптосистем на линейных кодах.

1 Обзор методов алгебраического криптоанализа и теории сложности

В данном разделе приведен обзор методов алгебраического криптоанализа потоковых шифров. Основной задачей алгебраического криптоанализа является построение зависимостей между открытыми текстами, шифротекстами и ключами в виде определенной системы полиномиальных уравнений над конечным двоичным полем и решение этой системы. В общем случае такая задача является вычислительно сложной, но существуют алгоритмы решения этой задачи в определенных частных случаях. После анализа таких алгоритмов описывается ситуация, в которой о неизвестном векторе можно получить некоторую частичную информацию. Выдвигается ряд модельных предположений о виде такой информации и формулируется задача восстановления неизвестного вектора по частичной информации, представленной в форме линейных зависимостей. Также рассматриваются определения и утверждения по теории сложности, необходимые для дальнейших исследований.

1.1 Методы алгебраического криптоанализа потоковых шифров

Одним из видов симметричных криптосистем являются потоковые шифры [1]. Существует много различных конструкций потоковых шифров [2, 3]. Во многих случаях потоковые шифры имеют эффективную реализацию, поэтому они часто используются на практике в устройствах и системах телекоммуникационной связи [4], где обязательным требованием является высокая скорость шифрования и расшифровки. Распространенным типом потоковых шифров является синхронный потоковый шифр [5]. Этот шифр содержит в себе такие составляющие:

– генератор гаммы: алгоритм, который принимает на вход начальное состояние x_0 и порождает последовательность битов $\gamma_1, \gamma_2, \ldots, \gamma_m$,

которая называется гаммой;

- алгоритм формирования начального состояния x_0 по ключу и вектору инициализации;
- закон шифрования: чаще всего бит шифротекста вычисляется как XOR, бит открытого текста с соответствующим битом гаммы.

Распространенным классом атак на потоковые шифры являются атаки на исходное состояние x_0. Допустим в потоковом шифре генератором гаммы является фильтрующий генератор гаммы с регистром сдвига с линейной обратной связью [1], который имеет сопровождающую матрицу C и функцию осложнения f от n переменных с $deg\, f = d$. В таком случае биты гаммы генерируются по формуле $\gamma_i = f\left(C^i \cdot x_0\right)$ для $i = 0,1,\ldots,m$, где т — это длина сгенерированного отрезка гаммы.

По сгенерированным отрезкам гаммы можно составить определенную систему полиномиальных уравнений степени d . Заметим, что в общем случае это будет система уравнений над полем $\mathbb{F}_{2^k}$, где $k \geq 1$, но поскольку любую систему уравнений над $\mathbb{F}_{2^k}$ можно представить в виде системы уравнений над $\mathbb{F}_2$, то большинство существующих методов построено для $\mathbb{F}_2$. Задача состоит в том, чтобы за системой этих уравнений восстановить неизвестный вектор x_0. Такая задача в общем случае является $\mathcal{NP} - \text{полной}$ при $d \geq 2$, но существуют методы, которые позволяют оптимизировать или сократить полный перебор:

- построение уравнений-следствий, которые имеют меньшую степень d;
- решение системы уравнений путем линеаризации;
- представление системы уравнений в виде булевой формулы и применение к ней SAT-решателей.

Рассмотрим необходимые определения и утверждения из теории алгебраических многообразий [6]. Под B_n будем обозначать множество всех булевых функций от п переменных. Идеалом булевой функции будем называть множество $\langle f \rangle = \{g \cdot f, g \in B_n\}$.

Если рассмотреть уравнение $f(x1, x2, \dots, x_n) = 0$, то для всех возможных следствий этого уравнения типа $g(x1, x2, \dots, x_n) = 0$ функции g будут принадлежать $\langle f \rangle$. Таким образом, идеал функции f описывает все возможные уравнения-следствия. Аннулятором идеала функции f, обозначается $Ann\,\langle f \rangle$, будем называть множество $\{g \in B_n | g \cdot f = 0\}$. Для анулятора выполняется такое равенство: $Ann\,\langle f \rangle = \langle f + 1 \rangle$.

Пусть для $1 \le i \le m$ выполняется $f\,(C^i \cdot x_0) = 1$. Допустим, можно выбрать такой $g \in Ann\,\langle f \rangle$, что $deg\ g < d$. Получим уравнение-следствие, но уже в меньшей степени. Убедимся в этом: если $g \in Ann\,\langle f \rangle$, то $g\,(C^i \cdot x_0) \cdot f\,(C^i \cdot x_0) = 0$. Поскольку $f\,(C^i \cdot x_0) = 1$, то $g\,(C^i \cdot x_0) = 0$, следовательно, получили уравнение, степень которого меньше d.

Рассмотрим теперь случай, когда для $1 \le i \le m$ выполняется $f\,(C^i \cdot x_0) = 0$. Предположим, что можно выбрать $h \in Ann\,\langle f + 1 \rangle$ такой, что $deg\ h < d$. Тогда $h\,(C^i \cdot x_0) = 0$ будет уравнением-следствием, но уже в меньшей степени. Убедимся в этом: если $h \in Ann\,\langle f + 1 \rangle$, то $h\,(C^i \cdot x_0) \cdot (f\,(C^i \cdot x_0) + 1) = 0$, откуда $h\,(C^i \cdot x_0) = 0$. Таким образом, если реализуется один из случаев $g \in \langle A \rangle$ или $h \in \langle f + 1 \rangle$, где $deg\ g,\ deg\ h < d$, то можно построить уравнение-следствия меньшей степени, то есть провести алгебраическую атаку на начальное состояние генератора гаммы.

Таким образом, задача поиска уравнений-следствий меньшей степени сводится к задаче поиска функций минимальной степени в идеале $\langle f \rangle$ (или $\langle f + 1 \rangle$). Для поиска функций минимальной степени можно воспользоваться источником [7], где утверждается, что функции с наименьшей степенью в минимальном базисе Гребнера [6] идеала $\langle f \rangle$ имеют наименьшую степень среди всех функций в идеале $\langle f \rangle$. Следовательно, задача поиска функций минимальной степени сводится к задаче построения минимального базиса идеала Гребнера. Для поиска базиса Гребнера существует ряд алгоритмов. Самым известным примером является алгоритм Бухбергера, сложность которого в общем случае является супер-экспоненциальной относительно

количества переменных функции f [8]. Есть более современные алгоритмы поиска базиса Гребнера, такие как F4 [9] и F5 [10], но в общем случае сложность этих алгоритмов является экспоненциальной [11].

Итак, метод поиска уравнений-следствий меньшей степени предоставляет возможность провести атаку на начальное состояние потокового шифра. Эта атака является алгебраической, поскольку базируется на поиске функций меньшей степени с идеала $\langle f \rangle$ или $\langle f + 1 \rangle$, то есть использует алгебраические свойства функции f. Заметим, что даже если такая атака является успешной, то в большинстве случаев не удается построить систему линейных уравнений и возникает ситуация, когда необходимо решать систему полиномиальных уравнений.

Одним из способов решения системы полиномиальных уравнений является использование метода линеаризации. Этот метод состоит в исключении из системы мономов, которые содержат несколько переменных, путем введения новых искусственных переменных и уравнений. После того, как была построена линеаризованная система, можно использовать метод Гаусса для поиска решений этой системы. В результате применения метода Гаусса будет получена новая система уравнений, для которой необходимо выполнить полный перебор значений, который в общем случае является экспоненциальным [12].

Вариацией такого метода является алгоритм XL [13], предложенный Николасом Куртуа. В этом алгоритме фиксируется определенное число $D > d$ (как правило, в качестве D выбирается $d + 1$) и формируется множество мономов, степень которых не превышает $D - d$ (это множество включает так же константу 1). Затем все уравнения умножаются на каждый моном с множественного числа M, поэтому в результате этого преобразования система уравнений имеет размер $m \cdot |M|$. Полученная система линеаризируется и к ней применяется метод Гаусса. Поскольку количество уравнений в системе увеличивается, то в некоторых случаях такой подход к решению системы оказывается более эффективным по сравнению с традиционным методом

линеаризации.

Еще одним алгоритмом, предложенным Николасом Куртуа, является ElimLin [14]. Этот алгоритм применяется к системе квадратичных уравнений. Таким образом, систему полиномиальных уравнений сначала необходимо превратить в систему квадратичных уравнений с помощью введения новых переменных и уравнений. Алгоритм заключается в итеративном повторении таких преобразований:

- система квадратичных уравнений упорядочивается по длине мономов и к ней применяется метод Гаусса, чтобы попытаться избавиться от нелинейных мономов. В результате все уравнения полученной системы можно разделить на линейные, которые формируют множество S_L,, и нелинейные, которые формируют множество St. Если множество нелинейных уравнений является пустым, то алгоритм останавливается;
- в каждом уравнении из S_L выбирается некоторый моном, выражается через остальные переменные и подставляется во все уравнения системы. После такой подстановки линейное уравнение записывается в множество S_L и весь процесс итеративно повторяется.

Если полученное в результате работы алгоритма множество линейных уравнений S_L включает достаточное количество уравнений, то можно найти решение этой системы, а следовательно, и решение исходной квадратичной системы. В противном случае алгоритм ElimLin не срабатывает.

В алгоритмах XL и ElimLin основными методами решения является построение уравнений-следствий и увеличению количества уравнений в системе. Существуют принципиально другие алгоритмы решения систем полиномиальных уравнений, например, алгоритм Раддума-Семаева [15]. В алгоритме Раддума-Семаева уравнение в системе записывается не в виде полинома от многих переменных, а в виде набора битовых строк, которые являются решениями этого уравнения и называются конфигурациями. Все уравнения системы в таком представлении формируют граф конфигурации, в

котором каждая вершина соответствует некоторому уравнению, то есть набора конфигураций для этого уравнения. Далее к этому графу применяется алгоритм передачи сообщений, в результате которого в графе остаются только те конфигурации, которые не противоречат друг другу, то есть решения системы уравнений. Возможны ситуации, когда в начале работы алгоритма конфигурации уже находятся в согласованном состоянии, поэтому на этот случай предлагается набор дополнительных стратегий, которые выводят конфигурации из согласованного состояния.

В качестве еще одного алгоритма, который использует принципиально иной подход к решению системы уравнений, можно привести Zhuang-Zi алгоритм [16]. В этом алгоритме задача решения системы полиномиальных уравнений со многими переменными сводится к задаче решения системы уравнений с одной переменной, но над некоторым расширением поля $\mathbb{F}_2$.

Отдельным подходом к решению систем полиномиальных уравнений является использование SAT-решателей [17]. Как правило, такой подход применяется к системе квадратичных уравнений, поэтому систему полиномиальных уравнений необходимо превратить в систему квадратичных уравнений путем введения новых переменных и уравнений. Для системы квадратичных уравнений над $\mathbb{F}_2$, которая содержит n переменных и m уравнений, опишем принцип построения эквивалентной формулы φ, заданной в КНФ [18]:

- поскольку КНФ не содержит констант, то для константы 1 необходимо ввести дополнительную искусственную переменную. Эта переменная должна содержаться в φ в виде отдельного дизъюнкта, чтобы ее единственным возможным значением было 1;
- для каждого квадратичного монома вида $x_i \cdot x_j$, где $1 \leq \mathrm{i} \leq j \leq n$, выполнить замену $x_{ij} = x_i \cdot x_j$. Поскольку равенство $x_{ij} = x_i \cdot x_j$ эквивалентно равенству $(xi \vee \bar{x}_{ij}) \wedge (xi \vee \bar{x}_{ij}) \wedge (\overline{xi} \vee \bar{x}j \vee xij) = 1$, то добавить в формулу φ соответствующие дизъюнкты. После этого шага в системе уравнений остаются только линейные слагаемые, то

есть система становится линейной;

– для каждого линейного уравнения $x_1 + x_1 + \cdots + x_l = 0$ построить соответствующую ему формулу в КНФ. Для этого можно по таблице истинности булевой формулы $x_1 + x_1 + \cdots + x_{l-1}$ построить КНФ с 2^{l-1} дизъюнктами, а затем в каждый дизъюнкт добавить x_l или $\overline{x}_l$ в зависимости от того, какое значение формула $x_1 + x_1 + \cdots + x_l$ принимает на соответствующем наборе входных значений. Для каждого линейного уравнения необходимо добавить соответствующий этому уравнению набор дизъюнктов размера 2^{l-1} в формулу φ.

Рассмотрим какое количество дизъюнктов добавляет каждый из этапов построения формулы. При замене одного квадратичного монома необходимо добавить три дизъюнкта в формулу, поэтому общее количество таких дизъюнктов будет ограничено полиномом от размера начальной системы. При преобразовании линейного уравнения $x_1 + x_1 + \cdots + x_l = 0$ необходимо добавить экспоненциальное от l количество дизъюнктов. Это можно предотвратить, если разбивать такую сумму на набор частичных сумм, которые связаны друг с другом общими переменными, и рассматривать их как отдельные уравнения [12].

Таким образом, систему квадратичных уравнений над $\mathbb{F}_2$ можно представить в виде булевой формулы в КНФ. Это предоставляет возможность применить к ней известные SAT-решатели[19].

Итак, существует много методов решения системы полиномиальных уравнений и, соответственно, способов восстановления начального состояния фильтрующего генератора гаммы, то есть неизвестного вектора x_0 . В общем случае эти методы требуют экспоненциального вычисления и срабатывают на практике лишь в частных случаях, поэтому возникает необходимость в поиске альтернативных моделей, в рамках которых можно попробовать восстановить неизвестный вектор x_0. Такой моделью может быть восстановление неизвестного вектора x_0 при наличие некоторой частичной информации об

этом векторе. Эта информация может быть получена из побочного канала связи или из особенностей конкретной реализации криптосистемы. Поскольку информация является частичной, то она не касается непосредственно вектора x_0, но может, например, накладывать ограничения на значение некоторых линейных зависимостей с неизвестным вектором x_0. Рассмотрим несколько ситуаций, в которых можно получить такую частичную информацию.

В случае фильтрующего генератора гаммы обновление состояния генератора происходит при помощи умножения начального вектора x_0 на сопровождающую матрицу C. В результате за m тактов работы регистра состояние генератора принимает значения $C^i \cdot x_0$ для $i = \overline{1,m}$. В этом случае частичной информацией могут быть некоторые значения, которые векторы вида $C^i \cdot x_0$, $i = \overline{1,m}$, не могут принимать.

Аналогичные конструкции встречаются в криптосистемах на линейных кодах. В этих криптосистемах в процессе шифрования выполняются матричные операции с открытым текстом и ключом. Аналогично примеру с начальным состоянием генератора гаммы, существует возможность получить определенные ограничения на линейные зависимости, которые появляются во время выполнения этих матричных операций.

Учитывая приведенные модельные предположения и примеры, возникает задача восстановления неизвестного вектора по частичной информации, представленной в форме линейных зависимостей. Для того чтобы иметь возможность применять к сформулированной задачи имеющиеся теоретико-множественные конструкции, необходимо формализовать эту задачу с помощью отдельного математического объекта или даже теории. В данной работе предлагается один из возможных способов такой формализации введением понятий линейного запрета и системы линейных запретов.

1.2 Необходимые определения и утверждения из теории сложности

Рассмотрим основные сведения из теории сложности, которые будут применяться при исследовании систем линейных запретов.

Задача — это некоторый общий вопрос, на который необходимо найти ответ. Задача должна состоять из двух частей: набор параметров или свободных переменных и набор свойств, которым должен удовлетворять ответ на эту задачу, чтобы считать такой ответ решением. Индивидуальная задача или экземпляр задачи — это задача, в которой все параметры зафиксированы конкретными значениями [20].

Задачей распознавания будем называть задачу, в которой два возможных ответа — «Да» или «Нет». Таким образом, набор всех входных данных или индивидуальных задач можно однозначно классифицировать на две группы в зависимости от ответа. Задачей поиска будем называть задачу, в которой необходимо найти объект, удовлетворяющий условиям этой задачи.

Алгоритмом будем называть четко определенную последовательность действий. Под вероятностным алгоритмом будет понимать алгоритм, который имеет доступ к источнику случайности, то есть генератору случайных объектов (в дальнейшем это будут преимущественно элементы конечного поля). Генератор, как правило, будет иметь равновероятное деление на множестве элементов, то все элементы этого множества будут выбираться с одинаковой вероятностью. Сложностью операции генерирования случайных объектов будем пренебрегать.

Для того, чтобы определить сложность задачи, можно доказать принадлежность этой задачи к определенному классу сложности. Дадим определение основным классам сложности по времени $\mathcal{P}$ и $\mathcal{NP}$ [21].

Будем говорить, что алгоритм В решает определенную задачу распознавания П, если $B(x) = 1 \Leftrightarrow \Pi(x) = 1$. Алгоритм В решает задачу за полиномиальное время, если на входе время работы алгоритма (или

количество элементарных операций) не превышает poly(|x|), где poly(|x|) — это полином, степень которого является константой.

Определение 1.1. Задача распознавания принадлежит классу сложности $\mathcal{P}$, если существует алгоритм, который решает эту задачу за полиномиальное от длины входа время.

Таким образом, класс сложности $\mathcal{P}$ состоит из всех задач распознавания, для которых существует эффективный развязок.

Определим алгоритм верификации V как алгоритм, который принимает на вход два аргумента, первый из которых — это входные данные задачи x, а второй — это бинарный ряд y, называется сертификатом. Алгоритм V верифицирует входная строка x, если существует сертификат y такой, что $V(\mathrm{x}, y) = 1$. Аналогично, алгоритм является полиномиальным, если время работы (или количество элементарных операций) является полиномом от длины входа.

Будем говорить, что алгоритм V верифицирует задачу распознавания П за полиномиальное время, если этот алгоритм является полиномиальным и длина сертификата ограничена длиной полинома. Другими словами, алгоритм V верифицирует П за полиномиальное время, если для каждого входа $x \in \{0,1\}^*$ сертификат y, длина которого ограничена полиномом от длины исходных данных, такой что $\Pi(x) = 1 \Leftrightarrow V\ (x,\ y) = 1$, причем час работы V является полиномиальным от длины входа

Определение 1.2. Задача распознавания принадлежит классу сложности $\mathcal{NP}$, если существует алгоритм, который верифицирует эту задачу за полиномиальное от длины входа время. Таким образом, класс сложности $\mathcal{NP}$ состоит из всех задач распознавания, решение которых можно проверить.

Очевидно, что $\mathcal{P} \subseteq \mathcal{NP}$, потому что найденное решение является автоматически проверенным. Вопрос $\mathcal{N}\ \mathcal{P} \subseteq \mathcal{P}$ остается открытым.

Рассмотрим различные типы сводимости между задачами. Сначала рассмотрим полиномиальную сводимость или сводимость по Карпову [22].

Определение 1.3. Задача распознавания Π_1 полиномиально сводится к

задаче распознавания Π_2, обозначается $\Pi_1 \leq_p \Pi_2$, если существует функция $f: \{0,1\}^* \to \{0,1\}^*$, которая вычисляется по полиномиальному времени, и $\forall x \in \{0,1\}^*$ выполняется: $\Pi_1(x) = 1 \Leftrightarrow \Pi_2(f(x)) = 1$.

Если для задач Π_1 и Π_2 выполняется $\Pi_1 \leq_p \Pi_2$ и $\Pi_2 \leq_p \Pi_1$, то будем называть такие задачи эквивалентными относительно полиномиальной сводимости и обозначать $\Pi_1 = \Pi_2$.

Рассмотрим сводимость по Тьюрингу (другое название — сводимость по Куку) [20].

Определение 1.4. Задача распознавания П1 сводится по Тьюрингу к задаче распознавания Π_2, обозначается $\Pi_1 \leq_T \Pi_2$, если существует полиномиальный алгоритм А, который за полиномиальное количество обращений к оракулу, что решает задачу Π_2, решает задачу Π_1.

Замечание. Это определение имеет место не только для задач распознавания, но и для задач поиска, причем сводиться друг к другу могут задачи разных типов.

Сводимость по Карпову можно рассматривать как частичный случай сведения по Тьюрингу, в котором можно сделать лишь один запрос к оракулу Π_2 и необходимо вернуть именно тот ответ, который этот оракул предоставил.

Если для задач Π_1 и Π_{22} выполняется $\Pi_1 \leq_T \Pi_2$ и $\Pi_2 \leq_T \Pi_1$, то будем называть такие задачи эквивалентными относительно сводимости по Тьюрингу и обозначать $\Pi_1 =_T \Pi_2$.

Задача П называется $\mathcal{NP}$-сложной, если для любой задачи $\Pi' \in \mathcal{N}\,\mathcal{P}$ выполняется $\Pi' \leq_p \Pi$. Таким образом, задача является $\mathcal{NP}$-сложной, если какая-либо другая задача из класса $\mathcal{NP}$ полиномиально сводится к этой задачи.

Если задача П принадлежит классу $\mathcal{NP}$ и является $\mathcal{NP}$-сложной, то будем называть ее $\mathcal{NP}$-полной. $\mathcal{NP}$-полные задачи формируют множество самых сложных задач в классе $\mathcal{NP}$.

Для того, чтобы доказать, что некоторая задача является $\mathcal{NP}$-полной, необходимо показать, что:

– эта задача принадлежит классу $\mathcal{NP}$;

– некоторая другая $\mathcal{NP}$-полная задача полиномиально сводится к ней.

Таким образом, имея хотя бы одну $\mathcal{NP}$-полную задачу, можно доказывать $\mathcal{NP}$-полноту других задач пользуясь приведенным фактом и транзитивностью полиномиального сведения.

Существование $\mathcal{NP}$-полной задачи — это чисто технический факт, пример такой $\mathcal{NP}$-полной задачи можно найти в [22]. Более нетривиальным является поиск $\mathcal{NP}$-полной задачи, которая была бы применима на практике. Такая задача была найдена Куком в 1971 году и получила название SAT [23]. Эта задача сыграла фундаментальную роль в развитии теории сложности, поскольку по помощью нее удалось доказать $\mathcal{NP}$-полноту многих задач, которые используются на практике. Первый весомый список $\mathcal{NP}$-полных задач появился в работе Карпа, которая была опубликована в 1972 году [24].

Рассмотрим задачу SAT (формулировка взята из [22, 25]).

Задача 1.1 (SAT). Вход. Булева формула в КНФ ф с п переменными и дизъюнктами.

Необходимо выяснить существует ли такой набор $(x1, x2, \dots, x_n)$, что $\phi(x1, x2, \dots, x_n) = 1$.

Выход. «Да», если такой набор существует, «Нет» иначе.

Часто формулируют SAT в случае, когда количество литералов (литералом будем называть переменную или ее отрицание) в одном дизъюнкте не превышает некоторого фиксированного числа.

Задача 1.2 (kSAT). Вход. Булева формула в КНФ ϕ с п переменными и т дизъюнктами, где каждый из дизъюнктов содержит не более k литералов.

Необходимо выяснить, существует ли такой набор $(x1, x2, \dots, x_n)$, что $\phi(x1, x2, \dots, x_n) = 1$.

Выход. «Да», если такой набор существует, «Нет» иначе.

Известно, что 2SAT принадлежит классу, то есть для этой задачи существует эффективный алгоритм решения, а все kSAT для $k \geq 3$ являются $\mathcal{NP}$ -полными задачами.

Другой вариацией SAT является задача, в которой на вход поступает

булева формула, но все операции дизъюнкции в дизъюнктах заменены на XOR. Для удобства, такие модифицированные дизъюнкта будем называть XOR-выражениями.

Задача 1.3 (XORSAT). Вход. Булева формула ϕ с п переменными и т XOR-выражениями.

Необходимо выяснить существует ли такой набор $(x1, x2, \dots, x_n)$, что $\phi(x1, x2, \dots, x_n) = 1$.

Выход. «Да», если такой набор существует, «Нет» иначе.

Известно, что XORSAT принадлежит классу $\mathcal{P}$ — для эффективного решения этой задачи можно воспользоваться методом Гаусса решения систем линейных уравнений.

В контексте анализа сложности систем линейных запретов важную роль будет играть модификация задачи XORSAT, в которой необходимо установить, что лишь некоторый набор XOR-выражений принимает значение 1, не обязательно все выражения одновременно.

Задача 1.4 (Max-XORSAT). Вход. Булева формула ϕ с п переменными и т XORвыражениями, число l, где $1 \leq l \leq m$.

Необходимо выяснить существует ли такой набор $(x1, x2, \dots, x_n)$, что по крайней мере l XOR-выражений приобретают значение 1.

Выход. «Да», если такой набор существует, «Нет» иначе.

Известно, что задача Max-XORSAT является $\mathcal{NP}$-полной. Если рассмотреть ограниченную версию этой задачи — Max-kXORSAT, то она будет $\mathcal{NP}$-полной начиная с $k \geq 2$ [25]. Для доведения $\mathcal{NP}$-полной Max-XORSAT можно выбрать задачу Max-Cut, которая используется для доводки $\mathcal{NP}$-полноты широкого класса задач.

Еще одним важным примером $\mathcal{NP}$-полной задачи является задача существования решения системы квадратичных уравнений над полем $\mathbb{F}_{2^k}$.

Задача 1.5 (MQ). Вход. $\mathbb{F}_{2^k}$, полиномы $\mathrm{p}_1, \mathrm{p}_2, \dots, p_m$ над полем $\mathbb{F}_{2^k}$, $k > 1$, каждый из полиномов принимает на вход n переменных и $deg\ p_i \leq 2$ для $i = \overline{1, m}$.

Необходимо выяснить существует набор $(x1, x2, \dots, x_n)$ такой, что $p_i(x1, x2, \dots, x_n) = 0$ для $i = \overline{1, m}$.

Выход. «Да», если такой набор существует, «Нет» иначе.

$\mathcal{NP}$ -полнота задачи MQ для случая $\mathbb{F} = \mathbb{F}_{2^k}$ была доказана в работе [26], а в случае произвольного поля $\mathbb{F}$ в работе [27]. Для доказательства $\mathbb{F}$ – полноты в последней работе использовалась техника арифметизации: при сведении задачи 3SAT к MQ все дизъюнкты в булевой формуле записывались в виде полиномов Жегалкина. Степень таких полиномов в случае 3SAT не может превышать трех. Из полученных выражений сформировалась система кубических уравнений со многими переменными относительно XOR. Для избавления от кубических мономов применялась схема, которая заключалась в исключении кубических мономов путем введения новых переменных и уравнений в систему.

Задача MQ имеет широкий спектр применения. В криптографии MQ используется для построения криптосистем: криптосистема SimpleMatrix [28], системы цифровой подписи UOV [29] и Rainbow [30]. MQ также применяется в алгебраическом криптоанализе потоковых шифров, подробнее такое применение MQ исследуется в подразделе 1.1. Проводятся исследования в направлении эффективного решения частных случаев этой задачи в зависимости от вида входной системы [31, 32].

Другим важным классом сложности задач является вероятностный класс $\mathcal{RP}$ [22]. Для того, чтобы определить этот класс, необходимо сформулировать определение вероятностного алгоритма с односторонней ошибкой. Будем говорить, что задача распознавания П решается вероятностным алгоритмом R с односторонней ошибкой, когда выполняются следующие условия:

– если $\Pi(x) = 1$, то $R(x) \geq \frac{1}{2}$;

– если $\Pi(x) = 0$, то $R(x) = 0$.

Вероятностный алгоритм R называется полиномиальным, если время его работы ограничено полиномом от длины входных данных.

Определение 1.5. Задача распознавания П принадлежит классу сложности $\mathscr{RP}$, если для нее существует полиномиальный вероятностный алгоритм с односторонней ошибкой. Известно, что класс $\mathscr{RP} \subseteq \mathcal{NP}$. Аналогично можно определить класс сложности задач со$\mathscr{RP}$, для которых существует вероятностный алгоритм с односторонней ошибкой, но эта ошибка происходит в случае $\Pi(x) = 0$. Примером задачи класса сложности c со$\mathscr{RP}$ является задача, в которой необходимо выяснить равно ли поленом тождественно нулю, где поленом задано над $\mathbb{Z}$. Построение вероятностного алгоритма с односторонней ошибкой для этой задачи использует лемму Шварца-Зипеля [33], которая будет также использоваться в контексте систем линейных запретов.

Выводы к разделу 1

В этом разделе были рассмотрены основные методы алгебраического криптоанализа. Большинство этих методов направлено на поиск решения системы полиномиальных уравнений, поскольку система полиномиальных уравнений является одним из основных предметов исследования алгебраического криптоанализа. Предложено рассмотреть более широкий класс задач, которые направлены на восстановление неизвестного вектора по частичной информации. Выдвинуты модельные предположения о виде этой информации и сформулирована задача восстановления вектора по частичной информации, представленной в форме линейных зависимостей.

Также рассмотрены основные факты из теории сложности, необходимые для дальнейших исследований. Они охватывают определение классов сложности задач и типов сводимости между задачами. Рассмотрены $\mathcal{NP}$-полные задачи, которые играют важную роль в исследовании систем линейных запретов, и оценки сложности задач, связанных с системами линейных запретов.

2 Алгебраические и комбинаторные свойства систем линейных запретов над конечным полем

В данном разделе формализуется задача восстановления неизвестного вектора по частичной информации, представленной в форме линейных зависимостей, путем введения нотации системы линейных запретов над конечным полем. Система линейных запретов рассматривается как отдельный математический объект, поэтому возникает необходимость построения теории, которая бы содержала в себе содержательные утверждения в отношении систем линейных запретов. Построение такой теории требует установления связей с имеющейся математической базой знаний, поэтому большая часть этого раздела посвящена поиску таких связей. В результате этих исследований доказывается ряд алгебраических и комбинаторных свойств системы линейных запретов.

2.1 Определения и свойства систем линейных запретов

Определим линейный запрет и систему линейных запретов по аналогии с линейного уравнения и системы линейных уравнений.

Определение 2.1. Линейным запретом над полем $\mathbb{F}_{2^k}$ будем называть выражение типа

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \neq a_0, \qquad (1)$$

где $a_i \in \mathbb{F}_{2^k}$ для i $= \overline{0,n}$, $x_i \in \mathbb{F}_{2^k}$ для i $= \overline{1,n}$.

Если обозначить $a = (a_1, a_2, \cdots, a_n)$, то можем переписать линейную запрет в виде выражения $(a, x) \neq a_0$, где $(a, x) := \Sigma_{i=1}^{n} a_i x_i$ — это скалярное произведение векторов a и x. Фактически, линейная запрет означает, что $a_1x_1 + a_2x_2 + \cdots + a_nx_n \in \mathbb{F}_{2^k} \setminus \{a_0\}$, а это эквивалентно такой совокупности уравнений:

$$\begin{cases} a_1x_1 + a_2x_2 + \cdots + a_nx_n = a'_1 \\ \cdots \\ a_1x_1 + a_2x_2 + \cdots + a_nx_n = a'_l \end{cases}, \quad (2)$$

где $a'_i \in \mathbb{F}_{2^k} \setminus \{a_0\}$ для i $= \overline{1,n}$ и $l = 2^k - 1$.

Замечание. В зависимости от контекста будем также называть линейным запретом вектор $(a_1, a_2, \cdots, a_n, a_0)$ длины $n+1$ в случае $a_0 \neq 0$ и вектор $(a_1, a_2, \cdots, a_n)$ длины n в случае $a_0 \neq 0$.

Замечание. Если для $x_0 \in \mathbb{F}_{2^k}^n$ выполняется $(a, x_0) = 0$, то будем говорить, что вектор a запрещает x_0. Также в таком случае будем говорить, что векторы a и x_0 является ортогональными.

Определение 2.2. Решением линейного запрета будем называть такой вектор $x_0 \in \mathbb{F}_{2^k}^n$, что $(a, x_0) \neq 0$. Множеством решений линейного запрета будем называть множество векторов $D = \{x \in \mathbb{F}_{2^k}^n | (a, x) \neq a_0\}$. Если множества решений двух линейных запретов совпадают, то будем называть такие линейные запреты эквивалентными.

С учетом определения множества решений линейного запрета, возникает вопрос какие элементарные действия над запретами можно выполнять, получая при этом эквивалентные запреты. Необходимо проверить выполняются ли для линейных запретов элементарные преобразования системы линейных уравнений, такие как перенос слагаемых из одной части в другую и умножение обеих частей на константу.

Утверждение 2.1. Множество решений линейного запрета не меняется при переносе слагаемых из одной части в другую, а также при умножении обеих частей на константу, которая не равна нулю.

Доказательство. Рассмотрим линейный запрет

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \neq a_0. \quad (3)$$

Обозначим $D_0 = \{x \in \mathbb{F}_{2^k} | (a, x) \neq a_0\}, D'_0 = \mathbb{F}_{2^k} \setminus D_0$, то есть D'_0 состоит из $x \in \mathbb{F}_{2^k}^n$ таких, что $(a, x) = 0$.

Сформируем другой линейный запрет

$$a_2x_2 + \cdots + a_nx_n \neq a_1x_1 + a_0. \quad (4)$$

Аналогично определим множества D_1 и D_1'. Множество D_1' состоит из всех $x \in \mathbb{F}_{2^k}^n$ таких, что $a_2x_2 + \cdots + a_nx_n = a_1x_1 + a_0$. Перенесем слагаемое a_1x_1 в левую часть уравнения: $a_1x_1 + a_2x_2 + \cdots + a_nx_n = a_0$.

Отсюда следует, что $D_1' = D_0'$, а следовательно, $D_1 = D_0$, поскольку множества D_0, D_0' и D_1, D_1' не пересекаются.

При умножении на константу доказательство аналогично.

Поскольку поле $\mathbb{F}_{2^k}$ является конечной множеством, то можем вычислить мощность множества решений линейного запрета.

Утверждение 2.2. Пусть $a_1x_1 + a_2x_2 + \cdots + a_nx_n \neq a_0$— линейный запрет над полем $\mathbb{F}_{2^k}$, тогда $|D| = 2^{kn} - 2^{k(n-1)}$.

Доказательство. Рассмотрим соответствующее линейному запрету линейное уравнение $a_1x_1 + a_2x_2 + \cdots + a_nx_n = a_0$. Если имеем дело с нетривиальным случаем $a \neq \bar{O}$, то какое из $a_1, a_{2,\ldots}, a_n$ является ненулевым элементом поля; не ограничивая общности предположим, что это x_n. Тогда можем переписать уравнение в следующем эквивалентном виде:

$$a_n^{-1} \cdot (a_1x_1 + a_2x_2 + \cdots + a_nx_n + a_0) = x_n \quad (5)$$

Пробегая все возможные векторы $(x_1, x_2, \cdots, x_{n-1})$ получаем некоторые значения x_n такие, что $(x_1, x_2, \cdots, x_n)$ являются решением линейного уравнения.

Если D'— это множество решений этого линейного уравнения, то

$$|D'| = \left|\{(x_1, x_2, \cdots, x_{n-1}), x_i \in \mathbb{F}_{2^k}, \mathrm{i} = \overline{1,n}\}\right| = 2^{k(n-1)}. \quad (6)$$

Тогда $D = \mathbb{F}_{2^k} \backslash D'$, а значит

$$|D| = \left|\mathbb{F}_{2^k}^n\right| - |D'| = 2^{kn} - 2^{k(n-1)}. \quad (7)$$

Следует заметить, что другой способ получить этот результат — заметить, что множество решений соответствующего линейного уравнения формирует линейное подпространство размерности $n-1$, мощность которого равна $2^{k(n-1)}$.

Отдельно рассмотрим случай линейного запрета при $n = 1$, то есть $a \cdot x \neq a_0$, где $a \neq 0$. Если $a_0 = 0$, то решениями будут все $x \in \mathbb{F}_{2^k}^*$, количество которых составляет $2^k - 1$. Если $a_0 \in \mathbb{F}_{2^k}^*$, то множеством решений будет $\{z \in \mathbb{F}_{2^k}, z\ a_0 \cdot a^{-1}\}$; его мощность составит $2^k - 1$. Следовательно, в случае $n = 1$ количество решений линейной запрета согласуется с формулой $2^{kn} - 2^{k(n-1)}$.

Определение 2.3. Системой линейных запретов будем называть систему соотношений вида:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \cdots + a_n^{(1)}x_n \neq a_0^{(1)} \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \cdots + a_n^{(2)}x_n \neq a_0^{(2)} \\ \cdots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \cdots + a_n^{(m)}x_n \neq a_0^{(m)} \end{cases}, \quad (8)$$

где $a_i^{(j)} \in \mathbb{F}_{2^k}$ для i $= \overline{0,n}, j = \overline{1,m}, x_i \in \mathbb{F}_{2^k}$ для i $= \overline{1,n}$ и $m > 1$.

Более компактная запись: будем обозначать $a^{(j)} \coloneqq (a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)})$, тогда систему линейных запретов можно переписать в виде:

$$\begin{cases} \left(a^{(1)}, x\right) \neq a_0^{(1)} \\ \left(a^{(2)}, x\right) \neq a_0^{(2)} \\ \cdots \\ \left(a^{(m)}, x\right) \neq a_0^{(m)} \end{cases}, \quad (9)$$

где $a_i^{(j)} \in \mathbb{F}_{2^k}$ для $j = \overline{1,m}$ и $x \in \mathbb{F}_{2^k}^n$.

Еще более компактная запись: пусть A — матрица размера $m \times n$ над полем $\mathbb{F}_{2^k}$ вида

$$A = \begin{pmatrix} a^{(1)} \\ a^{(2)} \\ \dots \\ a^{(m)} \end{pmatrix} = \begin{pmatrix} a_1^{(1)} & a_2^{(1)} & \dots & a_n^{(1)} \\ a_1^{(2)} & a_2^{(2)} & \dots & a_n^{(2)} \\ & & \dots & \\ a_1^{(m)} & a_2^{(m)} & \dots & a_n^{(m)} \end{pmatrix} \quad (10)$$

Тогда систему линейных запретов можно записать в виде:

$$A \cdot x \neq a_0 \text{ , где } a_0 = \begin{pmatrix} a_0^{(1)} \\ a_0^{(2)} \\ \dots \\ a_0^{(m)} \end{pmatrix}. \quad (11)$$

Замечание. Запись $A \cdot x \neq a_0$ является условной и вводится для компактности дальнейших формул. Общепринятая запись $a \neq b$ для некоторых векторов a и b означает, что эти векторы отличаются в некоторой координате, в нашем же случае она означает, что они отличаются во всех координатах.

Аналогично линейному запрету можем определить решение системы линейных запретов.

Определение 2.4. Решением системы линейных запретов будем называть $x_0 \in \mathbb{F}_{2^k}^n$ такой, что $A \cdot x \neq a_0$. Множеством решений системы линейных запретов будем называть множество векторов

$$\widetilde{D} = \{x \in \mathbb{F}_{2^k}^n \mid A \cdot x \neq a_0\} = D_1 \cap D_2 \cap \dots \cap D_m, \quad (12)$$

где D_j — это множество решений соответствующего линейного запрета в системе, $|D_j| = 2^{kn} - 2^{k(n-1)}$, $j = \overline{1,m}$. Две системы линейных запретов будем называть эквивалентными, если у них совпадают множества решений.

Утверждение 2.3. Множество решений системы линейных запретов не изменяется при переносе слагаемых из одной части в другую в любом из запретов и при умножении обеих частей какого-либо запрета на константу,

которая не равна нулю.

Доказательство. Пусть в некотором запрете выполнили перенос слагаемого или умножение на константу. Тогда по утверждению 2.1 множество решений этого запрета не изменилось, а следовательно, не изменилось пересечение всех множеств решений линейных запретов.

Замечание. Для упрощения вычислений в некоторых случаях будем рассматривать систему линейных запретов с нулевыми правыми частями. Также $\mathbb{F} = \mathbb{F}_{2^k}$, хотя в случае $\mathbb{F}$ — произвольного конечного поля и нулевых правых частей большинство результатов будет выполняться, если нет, то об этом будет отдельно сказано.

2.2 Критерий существования решений систем линейных запретов

Рассмотрим критерий существования решения систем линейных запретов над конечным полем. Этот критерий связывает арифметику систем линейных запретов с арифметикой полиномов и позволяет использовать методы, разработанные для полиномов, для изучения систем линейных запретов над конечным полем.

Теорема 2.1. У системы линейных запретов $A \cdot x \neq \overline{0}$ над полем $\mathbb{F}_{2^k}$ существует решение тогда и только тогда, когда

$$\prod_{i=1}^{m}\left(a^{(\mathrm{i})}, x\right) \neq 0. \qquad (13)$$

Доказательство. Обозначим для удобства $F(x) = \prod_{i=1}^{m}\left(a^{(\mathrm{i})}, x\right) \neq 0$.

Будем доказывать этот факт в следующей форме: у системы линейных запретов $A \cdot x \neq \overline{0}$ не существует решений тогда и только тогда, когда $F(x) \equiv 0$. Поскольку доказательства необходимости и достаточности этого критерия являются достаточно сходными, то не будем отдельно рассматривать необходимость на достаточность.

Отсутствие решений системы линейных запретов вида

$$\begin{cases} \left(a^{(1)}, x\right) \neq 0 \\ \left(a^{(2)}, x\right) \neq 0 \\ \cdots \\ \left(a^{(m)}, x\right) \neq 0 \end{cases} \tag{14}$$

равносильно тому, что для каждого $x \in \mathbb{F}_{2^k}^n$ хотя бы один из запретов не выполняется, то есть превращается в равенство, что в виде формулы можно записать таким образом:

$$\forall x \in \mathbb{F}_{2^k}^n : \exists i \in \overline{1, m} : \left(a^{(i)}, x\right) = 0. \tag{15}$$

Рассмотрим полином, состоящий из произведения левых частей всех линейных запретов: $F(x) = \left(a^{(1)}, x\right) \cdot \left(a^{(2)}, x\right) \cdot \cdots \cdot \left(a^{(m)}, x\right)$. Зафиксируем некоторое x, тогда для него будет существовать сомножитель, который равен 0, следовательно все произведение будет превращаться в нуль: $F(x) = 0$. Таким образом, $\forall x : F(x) = 0$. А это фактически и означает, что полином F над полем $\mathbb{F}_{2^k}$ тождественно равен нулю, поскольку он обращается в ноль на каждом возможном входе.

Замечание. В условии теоремы 2.1 в системе линейных запретов все правые части равны нулю. Если они будут ненулевыми, то это не повлияет на ход доказательства. В таком случае полином $F(x)$ будет определяться следующим образом:

$$F(x) = \prod_{i=1}^{m} \left[\left(a^{(\mathrm{i})}, x\right) + a_0^{(i)}\right]. \tag{16}$$

Более того, этот критерий может быть применен к произвольным функциям в левых частях запретов, не обязательно линейных. Но в этом случае степень результирующего полинома $F(x)$ уже не будет равна m, а будет составлять, например, $2 \cdot m$, в случае квадратичных левых частей.

Замечание. Этот критерий показывает, что в системе линейных запретов над конечным полем можно получать запреты-последствия путем умножения левых частей. Пусть есть два полиномы $f\ (x)$ и $g(x)$ над конечным полем, тогда $f\ (x) \neq 0$ и $g(x) \neq 0$ в том и только в том случае, когда $f\ (x) \cdot\ g(x)\ \neq 0$. Несложно в этом убедиться: если $f\ (x) \neq 0$ и $g(x) \neq 0$, то эти полиномы принимают некоторые значения с мультипликативной группы поля, а произведение двух элементов мультипликативной группы поля не может выйти за ее переделы, поэтому $f\ (x) \cdot\ g(x)\ \neq 0$. В иную сторону доказательство аналогичное. Поэтому сформулированный критерий можно рассматривать как вариант использования этого свойства, при котором нашли произведение левых частей всех запретов. Так как в системе линейных запретов необходимо найти такой x, что $f\ (x) \cdot\ g(x)\ \neq 0$, то вопрос существования такого x приводит к вопросу тождественного равенства полинома $f\ (x) \cdot\ g(x)$ нулю.

Замечание. Обратим внимание на то, что приведенный критерий не дает возможности конструктивно искать решения системы линейных запретов. Это связано с тем, что определение полинома, который не равен тождественно нулю, не требует явно задавать значение аргумента, на котором этот полином не обращается в ноль.

Учитывая этот критерий, возникает идея аналитически проверять равен ли тождественно нулю полином $F(x)$. К сожалению, при раскрытии скобок потенциально может возникнуть экспоненциальное количество слагаемых — имеющее дело с полиномом степени m от n переменных, который содержит в общем случае C_{n+m-1}^{m-1} коэффициентов. Этот биномиальный коэффициент нельзя ограничить некоторым полиномом от n и m, поэтому в таком виде этот критерий нельзя использовать для построения полиномиального алгоритма для проверки существования решения системы линейных запретов или полиномиального возведения вопроса существования решения системы к задаче определения или тождественного равенства полинома над некоторым конечным полем нулю. Более того, даже раскрытия скобок не всегда дает ответ

на этот вопрос, поскольку в конечном поле существуют полиномы с ненулевыми коэффициентами, которые равны тождественно нулю.

Но существуют вероятностные тесты, которые по заданным полиномам определяют равен ли тождественно этот полином нулю. В этих тестах не нужно аналитически раскрывать все скобки; достаточно использовать полином как «черный ящик» и вычислять значения этого полинома в некоторых точках. А вычисление значения полинома уже не является экспоненциальным исчислением, даже когда, как в нашем случае, вычисляется значение нескольких функций, то есть левых частей всех линейных запретов в системе.

Самый известный и одновременно самый простой такой тест основан на лемме Шварца-Зипеля, которая в случае поля $\mathbb{F} = \mathbb{F}_{2^k}$ (но эта лемма распространяется на конечное подмножество произвольного поля) утверждает, что вероятность по всем $r_1, r_2, \cdots, r_n$, которые являются элементами поля $\mathbb{F}_{2^k}$, того, что полином $f(r_1, r_2, \cdots, r_n),\ deg\ f\ =\ m$, равна нулю, ограничена сверху величиною $\frac{m}{|\mathbb{F}_{2^k}|}$, иначе говоря:

$$Pr_{r_1, r_2, \cdots, r_n \in \mathbb{F}_{2^k}}[f(r_1, r_2, \cdots, r_n) = 0] \leq \frac{m}{2^k}. \qquad (17)$$

Эта лемма работает только для случая $m \leq 2^k$, поскольку иначе будем получать тривиальную оценку вероятности, которая не является информативной. Более того, если выбрать $m \leq 2^{k-1}$, то получим классический случай, в котором при итеративном повторении проверки условия $f(r_1, r_2, \cdots, r_n) = 0$ ошибка будет экспоненциально уменьшаться, поскольку она ограничена константой $\frac{2^{k-1}}{2^k} = \frac{1}{2}$.

Следовательно, можем сформулировать вероятностный алгоритм проверки того, существует ли решение у системы линейных запретов в случае, когда $m \leq 2^{k-1}$.

Алгоритм 2.1. Вход. $\mathbb{F}_{2^k}$, матрица A размера $m \times n$, вектор a_0 размера $m \times 1$. Выполнить:

а) построить полином $F(x) = \prod_{i=1}^{m}\left[\left(a^{(i)}, x\right) + a_0^{(i)}\right]$, где $deg\ F = m$;

б) повторять d раз:

1) выбрать случайно и равновероятно $(r_1, r_2, \cdots, r_n) \in \mathbb{F}_{2^k}^n$,
2) проверить выполняется ли $F(r_1, r_2, \cdots, r_n) \neq 0$. Если выполняется, вернуть «Да»;

в) вернуть «Нет».

Выход. «Да», если система имеет решение, иначе – «Нет».

Получаем тест с односторонней ошибкой:

– допустим на входе система $A \cdot x = a_0$, в которой существует решение. Тогда $F(x) \equiv 0$. В таком случае вероятность получить ответ «Да» равняется нулю, поскольку «Да» возвращаем только тогда, когда найден вектор, в котором полином не равен нулю, а для тождественно равного нулю полинома такого значения вектора не существует. В таком случае, вероятность получить «Нет» равна 1;
– допустим на входе система $A \cdot x = a_0$, в которой нет решений. Тогда выполняется $F(x) \not\equiv 0$. Все генерации случайных наборов $(r_1, r_2, \cdots, r_n)$ являются независимыми событиями, поэтому вероятность того, что полином принимает значение нуль одновременно на всех этих наборах, равна произведению вероятностей отдельных событий. Поэтому через d итераций имеем оценку:

$$Pr_{r_1, r_2, \cdots, r_n \in \mathbb{F}_{2^k}}[f(r_1, r_2, \cdots, r_n) = 0] \leq \left(\frac{m}{2^k}\right)^d \leq \left(\frac{1}{2}\right)^d. \quad (18)$$

Следовательно, можем оценить вероятность того, что полином не равен тождественно нулю:

$$Pr_{r_1, r_2, \cdots, r_n \in \mathbb{F}_{2^k}}[f(r_1, r_2, \cdots, r_n) = 0] \leq 1 - \left(\frac{1}{2}\right)^d. \quad (19)$$

В этом случае даем правильный ответ, то есть «Да», с вероятностью $1 - \left(\frac{1}{2}\right)^d$, а ответ «Нет» с вероятностью $\left(\frac{1}{2}\right)^d$.

Полученные вычисления можно представить в виде матрицы ошибок 2.1, которая отражает совместимые вероятности всех гипотез и действительных фактов.

Таблица 1 – Матрица ошибок алгоритма для d итераций

	«Да»	«Нет»
$A \cdot x \neq a_0$ имеет решение	$\geq 1 - \left(\frac{1}{2}\right)^d$	$\leq \left(\frac{1}{2}\right)^d$
$A \cdot x \neq a_0$ не имеет решений	0	1

Проанализируем сложность этого алгоритма. Операция вычисления значения полинома в некоторой точке требует: $m \times n$ операций умножения в поле для вычисления промежуточных значений левых частей системы, а также m операций умножения для нахождения общего произведения. Получаем $m \cdot (n + 1)$ операций умножения в поле. Эту процедуру необходимо повторить d раз, следовательно в общем необходимо $d \cdot m \cdot (n + 1) = O(d \cdot m \cdot n)$ операций, поэтому приведенный алгоритм является полиномиально вероятностным. В дальнейшем упрощенная версия этого алгоритма будет использована, чтобы доказать принадлежность задачи существования решения системы линейных запретов (с ограничением на m) к вероятностному классу сложности $\mathcal{RP}$. Ограничения на m могут быть сняты [35], это планируется в дальнейших исследованиях.

Отметим, что в частичных случаях выяснить существуют ли решения системы линейных запретов можно исключительно по виду этой системы. Например, если в системе содержится такой x_i, где $1 < i < n$, что все

коэффициенты $a_i^{(j)} \neq 0$ для $j = \overline{1,m}$, то решения в системе гарантированно существуют. Векторы, у которых на i-й позиции содержится ненулевой элемент, а все остальные компоненты равны нулю, будут такими решениями. В общем случае такой x_i для $1 < i < n$ не всегда существует.

2.3 Свойства решений систем линейных запретов с нулевыми правыми частями

Предположим, что известно одно из решений системы линейных запретов с нулевыми правыми частями. Возникает вопрос можно ли в таком случае сказать о других решений или возможно даже восстановить некоторые из них.

Утверждение 2.4. Пусть $z = (z_1, z_2, \ldots, z_n)$ — решение системы линейных запретов $A \cdot x \neq \overline{0}$ над полем $\mathbb{F}_{2^k}$. Тогда $z' = (b \cdot z_1, b \cdot z_2, \ldots, b \cdot z_n)$, где $b \in \mathbb{F}_{2^k} \setminus \{0\}$ — также решение этой системы.

Доказательство. Пусть z — решение системы линейных запретов, тогда

$$\begin{cases} a_1^{(1)} z_1 + a_2^{(1)} z_2 + \cdots + a_n^{(1)} z_n = y_1 \\ a_1^{(2)} z_1 + a_2^{(2)} z_2 + \cdots + a_n^{(2)} z_n = y_2 \\ \cdots \\ a_1^{(m)} z_1 + a_2^{(m)} z_2 + \cdots + a_n^{(m)} z_n = y_m \end{cases} \quad (20)$$

где $y_1, y_2, \ldots, y_n \in \mathbb{F}_{2^k} \setminus \{0\}$.

Умножим левую и правую части всех уравнений на элемент поля $b \in \mathbb{F}_{2^k} \setminus \{0\}$.

$$\begin{cases} a_1^{(1)} z_1 b + a_2^{(1)} z_2 b + \cdots + a_n^{(1)} z_n b = y_1 b \\ a_1^{(2)} z_1 b + a_2^{(2)} z_2 b + \cdots + a_n^{(2)} z_n b = y_2 b \\ \cdots \\ a_1^{(m)} z_1 b + a_2^{(m)} z_2 b + \cdots + a_n^{(m)} z_n b = y_m b \end{cases} \quad (21)$$

Выполним замену $z_i' = z_i b$ для $i = \overline{1,n}$, $y_j' = y_j b$ для $j = \overline{1,m}$.

Получаем систему

$$\begin{cases} a_1^{(1)} z_1' + a_2^{(1)} z_2' + \cdots + a_n^{(1)} z_n' = y_1' \\ a_1^{(2)} z_1' + a_2^{(2)} z_2' + \cdots + a_n^{(2)} z_n' = y_2' \\ \cdots \\ a_1^{(m)} z_1' + a_2^{(m)} z_2' + \cdots + a_n^{(m)} z_n' = y_3' \end{cases} \quad (22)$$

Поскольку $\mathrm{b} \neq 0$, то новые $\mathrm{y_j^{'}} \neq 0$ для $\mathrm{j} = \overline{1, \mathrm{m}}$. Следовательно, $(\mathrm{z_1^{'}}, \mathrm{z_2^{'}}, \ldots, \mathrm{z_n^{'}})$ — также решение исходной системы линейных запретов.

Итак, по известным решениям можно восстановить часть других решений. Сформулируем утверждение, которое вычисляет количество таких решений.

Утверждение 2.5. Если система линейных запретов $A \cdot x \neq \overline{0}$ над полем $\mathbb{F}_{2^k}$ имеет хотя бы одно решение, то мощность множества решений по крайней мере $2^k - 1$.

Доказательство. Допустим система линейных запретов $A \cdot x \neq \overline{0}$ имеет решение z'.

Определим операцию умножения вектора на скаляр: $b \cdot z' = (b \cdot z_1', b \cdot z_2', \ldots, b \cdot z_n')$ для $z \in \mathbb{F}_{2^k}^n$ и $b \in \mathbb{F}_{2^k}^*$. Рассмотрим множество векторов

$$\left\{ z \in \mathbb{F}_{2^k}^n | z = b \cdot z_2', b \in \mathbb{F}_{2^k}^* \right\}. \quad (23)$$

Покажем, что для элементов $b_1 \neq b_2$, где $b_1, b_2 \in \mathbb{F}_{2^k}^*$, выполняется $b_1 z' \neq b_2 z'$. От противного: предположим, что $b_1 z' = b_2 z'$. Это означает, что эти векторы совпадают во всех компонентах, то есть: $b_1 z' = b_2 z'$ для $i = \overline{1, m}$. Очевидно, что решение исходной системы не может быть нулевым (поскольку в правых частях запретов содержатся нули), поэтому выберем номер j такой, что $z_j' \neq 0$. Тогда для компоненты с этим номером выполняется $b_1 z_j' = b_2 z_j'$, а из этого следует $(b_1 - b_2) z_j' = 0$. Поскольку $z_j' \neq 0$ и поле не имеет делителей нуля, то $b_1 = b_2$, т. Е. приходим к противоречию. Таким образом, все векторы $b \cdot z_j'$ для $b \in \mathbb{F}_{2^k}$ являются разными.

Поскольку элемент b пробегает всю мультипликативную группу поля $\mathbb{F}_{2^k}^*$, мощность которой $2^k - 1$, то число векторов вида $b \cdot z_j'$ ограничена числом $2^k - 1$. Учитывая, что все векторы такого вида являются разными, их количество равно $2^k - 1$. Следовательно, общее количество решений системы линейных запретов составляет минимум $2^k - 1$.

Сформулируем теорему, которая описывает структуру множества решений системы линейных запретов с нулевыми правыми частями.

Теорема 2.2. Пусть $D \subseteq \mathbb{F}_{2^k}^n$ — множество решений системы линейных запретов $A \cdot x \neq \overline{0}$ над полем $\mathbb{F}_{2^k}$, тогда $|D|$ делится на $2^k - 1$.

Доказательство. Рассмотрим бинарное отношение на множестве решений D системы линейных запретов: два вектора $a, b \in D$ находятся в бинарном отношении ~, если существует элемент $c \in \mathbb{F}_{2^k}^*$ такой, что $(a_1, a_2, \dots, a_n) = (c \cdot b_1, c \cdot b_2, \dots, c \cdot b_n)$. Такое отношение будем называть отношением пропорциональности, а векторы, принадлежащие этому отношению, — пропорциональными.

Покажем, что отношение ~ является отношением эквивалентности.

Рефлексивность. Свойство $\forall z \in D : z \sim z$ выполняется всегда, поскольку можем выбрать $c = 1$.

Симметричность. Надо проверить $\forall x, y \in D : x \sim y \Rightarrow y \sim x$. Если x ~ y, то существует $c \in \mathbb{F}_{2^k}^*$ такой, что $x_i = c \cdot y_i$ для $i = \overline{1, n}$. Поскольку $c \in \mathbb{F}_{2^k}^*$, то существует $c^{-1} \in \mathbb{F}_{2^k}^*$, поэтому $y_i = c^{-1} \cdot x_i$ для $i = \overline{1, n}$. Имеем $(y_1, y_2, \dots, y_n) = (c^{-1} \cdot x_1, c^{-1} \cdot x_2, \dots, c^{-1} \cdot x_n)$, а это означает, что $y \sim x$.

Транзитивность. Надо проверить, что $\forall x, y, z \in D : x \sim y, y \sim x$ следует $x \sim z$. Допустим, $x \sim y$ и $y \sim z$, тогда существуют $c_1, c_2 \in \mathbb{F}_{2^k}^*$ такие, что $x_i = c_1 \cdot y_i$ и $y_i = c_2 \cdot z_i$ для $i = \overline{1, n}$. Подставляем $y_i = c_2 \cdot z_i$для $i = \overline{1, n}$ в условие $x \sim y$ и получаем $x_i = c_1 \cdot c_2 \cdot z_i$ для $i = \overline{1, n}$. Поскольку $c_1 \cdot c_2 \in \mathbb{F}_{2^k}^*$, то $x \sim z$.

Отношение эквивалентности на множестве D порождает разбиение множества на классы эквивалентности: $D = D_1 \cup D_2 \cup \dots D_s$, где $D_i \cap D_j = \emptyset$

при $i \neq j$, а s — количество классов эквивалентности [34]. Таким образом, любые два элемента одного класса $D_i, i = \overline{1,s}$, находятся в отношении~, а любые два элемента разных классов $D_i, D_j, i \neq j$, не находятся в отношении эквивалентности.

По утверждению 2.5 количество элементов в одном классе эквивалентности составляет $2^k - 1$. Поскольку все классы не пересекаются, то

$$|D| = s \cdot (2^k - 1), \tag{24}$$

что и требовалось доказать.

Замечание. Количество решений одного линейного запрета составляет $2^{kn} - 2^{k(n-1)} = 2^{k(n-1)}(2^k - 1)$, что согласуется с доказанным фактом.

2.4 Системы линейных запретов с нулевыми правыми частями, сгенерированные неизвестным фиксированным вектором

Предположим, что вектор $z^{(tr)} \in \mathbb{F}_{2^k}^n$, где $n \geq 2$, является фиксированным. Будем перебирать все возможные векторы $a \in \mathbb{F}_{2^k}^n$ и все те, для которых выполняется $(a, z^{(tr)}) \neq 0$, будем записывать в множество A_{true} . Очевидно, что множество A_{true} будет состоять из попарно различных векторов.

Утверждение 2.6. Количество векторов в множестве A_{true} составляет $2^{kn} - 2^{k(n-1)}$.

Доказательство. В утверждении 2.2 было доказано, что количество решений линейной запрета составляет $2^{kn} - 2^{k(n-1)}$. Рассмотрим линейный запрет $(a, x) \neq 0$, в котором x зафиксируем значением $z^{(tr)}$, а вектор a будет пробегать все возможные значения $\mathbb{F}_{2^k}^n$. Тогда по доказанному утверждению количество таких a для фиксированного x составляет точно $2^{kn} - 2^{k(n-1)}$.

Следовательно, $|A_{true}| = 2^{kn} - 2^{k(n-1)}$. Теперь сформируем из всех

векторов этого множества систему линейных запретов и найдем ее решение D_{true}. Таким образом, построили как можно полную систему линейных запретов — добавлять дополнительные запреты нет смысла, поскольку исчерпали все возможные варианты запретов для фиксированного $z^{(tr)}$. Возникает вопрос можем ли по полной системе запрета однозначно восстановить решение, то есть найти тот вектор $z^{(tr)}$, с помощью которого была сгенерирована матрица A_{true}. Оказывается, что да, т. Е. для системы $A_{true} \cdot x \neq \overline{0}$ можем восстановить решения $z^{(tr)}$ с точностью до пропорциональных ему векторов.

Теорема 2.3. Для системы линейных запретов $A_{true} \cdot x \neq 0$ количество решений $|D_{true}|$ равно $2^k - 1$.

Доказательства. Напомним, что $z^{(tr)}$ — это вектор, который использовался при генерировании A_{true}. Очевидно, что у системы линейных запретов $A_{true} \cdot x \neq \overline{0}$ решения существуют, это следует из построения матрицы A_{true}: в начале зафиксировали значение $z^{(tr)}$ и добавляли в A_{true} только те $a \in \mathbb{F}_{2^k}^n$, для которых $(a, z^{(tr)}) \neq 0$, поэтому ни один вектор, ортогональный $z^{(tr)}$, в A_{true} содержаться не может, а потому $z^{(tr)}$ гарантированно есть среди решений.

По утверждению 2.5 имеем, что $|D_{true}| > 2^k - 1$. Теорема 2.2 говорит о структуре решений системы: все решения разбиваются на классы эквивалентности по отношению пропорциональности, каждый из которых имеет мощность $2^k - 1$. Будем доказывать, что среди решений системы линейных запретов $A_{true} \cdot x \neq \overline{0}$ имеет только один класс эквивалентности — это класс элемента $z^{(tr)}$.

В начале, сделаем несколько наблюдений. Пусть i, где $1 \leq i \leq n$, — это позиция в векторе $z^{(tr)}$, на которой стоит ненулевой элемент поля. Тогда запрет вида $a_i = (0, \dots, 0, 1, 0, \dots, 0)$, в которой на i-й позиции стоит единица, а на остальных нули, содержится в A_{true} , поскольку $(a, z_i^{(tr)}) = z_i^{(tr)} \neq 0$. Одновременно $(a_i, z) = z_i$, поэтому z_i не может быть нулем. Таким образом,

если на некоторой позиции $z^{(tr)}$ стоит ненулевой элемент поля, то на соответствующей ему позиции вектора z_i тоже должен стоять ненулевой элемент.

Предположим теперь, что i, где $1 \leq i \leq n$, — это позиция вектора $z^{(tr)}$, на которой стоит ноль. Покажем, что на соответствующий позиции другого произвольного вектора z тоже должен стоять ноль. Будем доказывать от противного: допустим это не выполняется, то есть $z_i \neq 0$. Выберем любой другой элемент вектора $z^{(tr)}$, который не равен нулю (такой существует, поскольку $z^{(tr)} \neq 0$), пусть он находится на позиции j, где $1 \leq j \leq n$. Уже было доказано, что ненулевым элементам $z^{(tr)}$ соответствуют ненулевые элементы z, поэтому $z_j \neq 0$. Построим вектор b следующим образом: $b_i = z_i^{-1}$, $b_j = z_j^{-1}$, а на всех остальных позициях содержатся нули. Тогда $(b, z^{(tr)}) = z_j^{tr} z_j^{-1} \neq 0$, а $(b, z) = z_i \cdot z_i^{-1} + z_j \cdot z_j^{-1} = 0$. Следовательно, z не является решением, а это противоречит предположению. Поэтому нулевым позициям $z^{(tr)}$ соответствуют нулевые позиции z. Итак, в множество кандидатов на решение могут попасть только те векторы, в которых ненулевые позиции точно совпадают с $z^{(tr)}$. В дальнейшем речь будет идти только о векторах такого вида.

Фактически, нужно доказать, что какого бы кандидата не взяли (естественно, он должен быть не пропорциональным по отношению к $z^{(tr)}$), в множестве A_{true} всегда найдется запрет, который будет отбрасывать этого кандидата. Более формально, $\forall z$, который не является пропорциональным к $z^{(tr)}$, существует $a \in A_{true}$ такой, что $(a, z) = 0$.

Рассмотрим более подробно условие пропорциональности. Во-первых, сузим круг кандидатов, то есть тех, в которых по меньшей мере две ненулевые компоненты, поскольку если одна компонента (тогда, как известно, в векторе $z^{(tr)}$ будет единственная ненулевая компонента на той же позиции), то вектор является пропорциональным к $z^{(tr)}$. Также будем рассматривать только позиции с ненулевыми компонентами, поскольку в случае, когда одна

компонента из двух будет нулем, а другая нет, условие пропорциональности не будет нарушаться. Учитывая эти замечания, будем говорить, что вектор z не является пропорциональным к $z^{(tr)}$, если существует пара индексов i, j , где $1 \leq i,j \leq n$ и $i \neq j$, таких, что $z_i z_j^{(tr)} + z_i^{(tr)} z_j = d$, где $d \neq 0$, причем $z_i, z_i^{(tr)}, z_j, z_j^{(tr)} \neq 0$. Если бы на этих двух позициях условие пропорциональности не возбуждалось, то имели бы место равенства $z_i = c \cdot z_i^{(tr)}$, $z_j = c \cdot z_j^{(tr)}$, откуда $c = z_i \cdot \left(z_i^{(tr)}\right)^{-1}$, поэтому $c = z_i \cdot \left(z_i^{(tr)}\right)^{-1} = z_j \cdot \left(z_j^{(tr)}\right)^{-1}$, поэтому $z_i \cdot \left(z_i^{(tr)}\right)^{-1} = z_j \cdot \left(z_j^{(tr)}\right)^{-1}$, а из этого следует $z_i z_i^{(tr)} + z_j^{(tr)} z_j = 0$. Делая итог, необходимо показать, что для кандидатов z таких, что $\exists i \neq j,\ z_i z_j^{(tr)} + z_j^{(tr)} z_j = d \neq 0$, существует запрет $a \in A_{true}$ таков, что $(a, z) = 0$, то есть z не является решением.

Зафиксируем вектор-кандидат z, который удовлетворяет описанным выше условиям. Пусть a — это вектор, который состоит из нулей, кроме позиций i и j, на которых нарушается пропорциональность векторов z и $z^{(tr)}$. Построим вектор a, задав компоненты a_i и a_j :

$$a_i = d \cdot z_i^{-1} + z_j^{(tr)},\ a_j = z_i^{(tr)}. \quad (25)$$

Напомним, что $z_i, z_j, z_i^{(tr)}, z_j^{(tr)} \neq 0$ и

$$d = z_i z_j^{(tr)} + z_i^{(tr)} z_j \neq 0. \quad (26)$$

Проверим принадлежит ли a множеству A_{true}, то есть $(a, z^{(tr)}) \neq 0$:

$$a_i z_i^{(tr)} + a_j z_j^{(tr)} = (d \cdot z_i^{-1} + z_j^{(tr)}) z_j^{(tr)} + z_i^{(tr)} z_j^{(tr)} = d \cdot z_i^{-1} + z_i^{(tr)}. \quad (27)$$

Поскольку $d, z_i^{-1}, z_i^{(tr)} \neq 0$, то этот вектор содержится в A_{true} .

Проверим выполняется $(a, z) = 0$:

$$a_i z_i + a_j z_j = (d \cdot z_i^{-1} + z_j^{(tr)}) z_j^{(tr)} + z_i^{(tr)} z_j = d + z_i z_j^{(tr)} + z_j^{(tr)} z_j = d + d = 0. \quad (28)$$

Эти проверки завершают доказательство.

Доказанная теорема говорит о том, что точно восстановить $z^{(tr)}$ возможно, но можно найти множество пропорциональных векторов, среди которых гарантированно будет содержаться $z^{(tr)}$, если нет можно полную систему запретов. Но возникает проблема, которая заключается в том, что размер $|A_{true}|$ слишком большой, например, для $\mathbb{F}_8$ и $n = 5$, количество всех векторов составляет 32768, а $|A_{true}| = 28672$, то есть 87.5%. В общем

$$\frac{2^{kn} - 2^{k(n-1)}}{2^{kn}} = 1 - \frac{2^{kn-k}}{2^{kn}} = 1 - 2^{-k}, \quad (29)$$

i при $k \to \infty$ стремится к единице. С практической точки зрения, это делает невозможным решение системы линейных запретов, поскольку требует, чтобы она содержала очень большое количество векторов.

Это приводит к вопросу, как по заданным параметрам задачи оценить минимальное количество векторов, которые должны содержаться в системе, чтобы получить в результате $|D_{true}|$ решений, где $|D_{true}|$ примерно равна $|D_{true}|$ (в случае нулевых правых частей $|D_{true}|$ составляет $2^k - 1$).

С точки зрения практики это может понадобиться в такой ситуации: пусть есть неограниченный доступ к векторам $a \in \mathbb{F}_{2^k}^n$, таких что $(a, z^{(tr)}) = 0$ (например, можем обращаться к оракулу, который на каждом шаге случайно предоставляет один из таких векторов). В таком случае есть определенная привязка к количеству запросов к оракулу (или времени), поэтому необходимо понимать сколько минимально надо насобирать таких попарно различных

векторов (и есть ли разница какие именно векторы выбирать, а если есть, то какая стратегия выбора является оптимальной с точки зрения скорейшего уменьшения количества решений), чтобы получить систему линейных запретов с количеством решений, которая является как можно ближе к $|D_{true}| = 2^k - 1$.

Формализуем этот вопрос таким определением.

Определение 2.5. Точкой насыщения будем называть число $S = min_{A'}|A'|$, где минимум исчисляется по всем таким матрицам A', что $A' \subseteq A_{true}$, и $A' \cdot x \neq 0$ имеет $|D_{true}|$ решений. Соответственно, матрицу, на котором этот минимум достигается, будем называть насыщенной матрицей.

Фактически, насыщенная матрица — это такая матрица, которая передает все свойства A_{true}, но при этом может иметь гораздо меньший размер. Если знать точку насыщения, то это даст представление сколько запросов к оракулу нужно сделать, перед тем как начинать решать систему линейных запретов. Если начать решать систему из ненасыщенной матрици, то может возникнуть ситуация, когда мощность множества решений будет слишком большой. Проиллюстрируем на примере, значение точки насыщения на практике намного меньше размера A_{true}.

Пример 2.1. Дано поле $\mathbb{F}_8$ и фиксированный вектор $z^{(tr)}$ размера n = 5. Тогда несложно установить размер матрицы $|A_{true}|$ и множества решений $|D_{true}|$: $|A_{true}| = 28672$ и $|D_{true}| = 7$. Теперь попытаемся найти зависимость количества решений в системе от количества запретов.

На каждой итерации выполняем такую последовательность действий:

- формируем матрицу A_{true},
- инициализируем $D = \mathbb{F}_{2^k}^n$,
- равновероятно и независимо выбираем вектор a из множества A_{true} (выборка осуществляется с возвратом),
- находим решение запрета $(a, z^{(tr)}) \neq 0$, который обозначаем D_j,
- находим пересечение $D := D \cap D_j$ — количество решений системы

линейных запретов на текущем шаге.

Теперь можем построить график зависимости $|D|$ от количества итераций. Этот график изображен на рисунке 2.1.

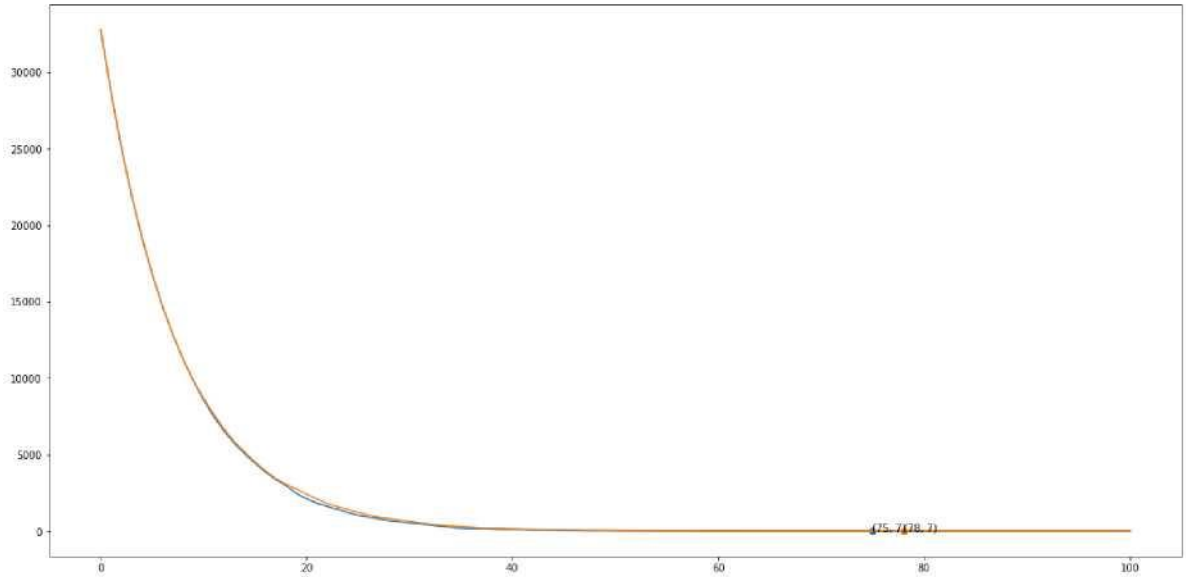

Рисунок 1 – Зависимость числа решений от количества уравнений в системе линейных запретов

Видим, что точка насыщения принимает значение 70-80, что намного меньше $|A_{true}| = 28672$.

Замечание. Если бы имели дело с системой линейных уравнений, то мощность пересечения D спадала бы намного быстрее, поскольку количество решений линейного уравнения составляет $2^{k(n-1)}$, что является незначительным количеством по отношению к 2^{kn}, потому что $2^{k(n-1)}$ разделить на 2^{kn} равен $2^{k(n-1)}$. В этом заключается отличие систем линейных запретов от систем линейных уравнений с комбинаторной точки зрения.

Заметим, что в ситуации, когда неизвестный вектор x является фиксированным, в системе линейных запретов всегда будут существовать решения, то есть вопрос о существовании решений не возникает. Ситуация, в которой система $A \cdot x \neq \overline{0}$ не имеет решений, возможна лишь в том случае, когда некоторые запреты в системе, были порождены другими неизвестными векторами, и такие запреты отрицают допустимые значения друг друга.

2.5 Системы линейных запретов с ненулевыми правыми частями, сгенерированные неизвестным фиксированным вектором

Системы линейных запретов с ненулевыми правыми частями удобно задавать с помощью множества вида $\hat{A} \subset \{\langle a, a_0\rangle | a \in \mathbb{F}_{2^k}^n,\ a_0 \in \mathbb{F}_{2^k}\}$, где каждая пара $\langle a, a_0\rangle$ задает линейный запрет $(a, x) \neq a_0$. Также систему линейных запретов с ненулевыми правыми частями можно рассматривать как матрицу размера $(n+1) \times m$, в которой последний столбец состоит из элементов $a_0 \in \mathbb{F}_{2^k}$. Таким образом, в зависимости от контекста будем рассматривать систему запретов с ненулевыми правыми частями как матрицу из $n+1$ столбиками или как множество пар $\langle a, a_0\rangle$ — все перечисленные объекты будут считаться эквивалентными способами представления системы линейных запретов с ненулевыми правыми частями.

Предположим, что правые части системы линейных запретов могут содержать не только нулевой элемент поля, но и другие. Как и ранее, вектор $z^{(tr)} \in \mathbb{F}_{2^k}^n$ является фиксированным. Будем рассматривать нетривиальный случай $z^{(tr)} \neq 0$. Для произвольного элемента b поля $\mathbb{F}_{2^k}$ определим множество

$$A_{true}^{(b)} = \{\langle a, a_0\rangle, a \in \mathbb{F}_{2^k}^n, |(a, z^{(tr)}) \neq b\}. \quad (30)$$

Заметим, что с точностью до первой компоненты каждой пары $A_{true}^{(0)}$ совпадает с A_{true}. Выберем произвольный ненулевой элемент поля g и сформируем для него множество $A_{true}^{(g)}$. Построим множество

$$\hat{A}_{true} = A_{true}^{(0)} \cup A_{true}^{(g)}, \quad (31)$$

которая будет задавать систему линейных запретов с ненулевыми правыми частями. Через $\widehat{D}_{true}$ будем обозначать количество решений системы

линейных запретов.

Теорема 2.4. Для системы линейных запретов A_{true} количество решений $|\widehat{D}_{true}| = 1$.

Доказательство. Напомним, что $z^{(tr)}$ — это фиксированный вектор, которым было сгенерировано множество $\hat{A}_{true}$. Очевидно, что $z^{(tr)}$является решением такой системы, поскольку в этой системе за построением не содержится ни одного вектора, который бы запрещал $z^{(tr)}$.

Необходимо доказать, что для любого вектора $z \in \mathbb{F}_{2^k}^n$, $z \neq z^{(tr)}$, существует пара $\langle a, a_0 \rangle \in A_{true}$ такая, что $(a, z) = b$. То есть какой бы вектор z, за исключением настоящего решения, не выбрали, в системе всегда найдется запрет, который будет отбрасывать z.

Поскольку $A_{true}^{(0)} \subseteq \hat{A}_{true}$, то можем применить рассуждения из доказательства теоремы 2.3, в котором утверждалось, что произвольный вектор z имеет ненулевые элементы поля на тех же позициях, что и вектор $z^{(tr)}$, иначе z не является решением.

Покажем теперь как отбросить все остальные векторы. Пусть на i-й позиции, $1 \leq i \leq n$, векторов $z^{(tr)}$ и z содержатся ненулевые элементы поля. Покажем, что $z_i \neq z^{(tr)}$ следует, что z не является решением. Построим запрет таким образом: на i-й позиции будет элемент $g \cdot z_i^{-1}$, а на всех остальных нули. Тогда $(a, z^{(tr)}) = g \cdot z_i^{-1} \cdot z_i^{(tr)}$. Этот скалярное произведение будет равно g только в том случае, когда $z_i^{-1} \cdot z_i^{(tr)} = 1$, то есть $z_i = z_i^{(tr)}$, а это не выполняется по предположению, следовательно $(a, z^{(tr)}) \neq g$. В случае некоторого другого вектора z имеем: $(a, z) = g \cdot z_i^{-1} \cdot z_i = g$, следовательно z не может быть решением. С оглядкой на этот факт, можем сделать вывод, что при наличии достаточного количества линейных запретов с нулевыми и ненулевыми частями, можем точно восстановить $z^{(tr)}$, который был сгенерирован $\hat{A}_{true}$. Определение точки насыщения и насыщенной матрицы для множества $\hat{A}_{true}$ является аналогичным определению 2.5. В контексте

системы с ненулевыми правыми частям также встает вопрос поиска матриц с малым количеством запретов, но которые однозначно задают неизвестное фиксированное решение.

Доказательство теоремы 2.4 является конструктивным и предоставляет возможность строить матрицы с ненулевыми правыми частями, которые содержат небольшое количество уравнений, имеющих единственное решение.

Утверждение 2.7. Пусть $\hat{A}_{expl} \subseteq \hat{A}_{true}$ — это система линейных запретов, которая состоит из всех возможных запретов одного из двух типов:

– запреты, в которых одна компонента равна 1, все остальные компоненты равны нулю и правая часть также равна нулю;
– запреты, в которых одна компонента равна некоторому ненулевому элементу поля, все остальные компоненты равны нулю, а правая часть равна фиксированному ненулевому элементу поля g.

Тогда такая система имеет единственное решение $z^{(tr)}$.

Доказательство. Рассмотрим какие векторы запрещает каждая из описанных в условии групп запретов.

Рассмотрим линейные запреты вида $\langle a_i, 0\rangle$ где $a_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0, 0)$ и 1 содержится на позиции с номером $1 \leq i \leq n$. Предположим, что $z_i^{(tr)} \neq 0$, тогда запрет $\langle a_i, 0\rangle$ содержится в системе $\hat{A}_{expl}$. Поэтому для некоторого другого вектора z выполняется $(a_i, z) = z_i$, откуда $z_i \neq 0$. Итак, с $z_i^{(tr)} \neq 0$ вытекает $z_i \neq 0$.

Рассмотрим линейные запреты вида $\langle b_i^{(d)}, g\rangle$ где $b_i^{(d)} = (0, 0, \ldots, 0, d, 0, \ldots, 0, 0)$ и произвольный ненулевой элемент поля d содержится на позиции с номером $1 \leq i \leq n$. Если $z_i^{(tr)} \neq 0$ (а тогда и $z_i \neq 0$ — это было доказано ранее), то для фиксированного номера i в систему попадут все запреты с $d \neq \left(z_i^{(tr)}\right)^{-1} g$, поскольку при $d = \left(z_i^{(tr)}\right)^{-1}$ имеем

$$(b_i^{(d)}, z_i^{(tr)}) = d \cdot z_i^{(tr)} = \left(z_i^{(tr)}\right)^{-1} \cdot g \cdot z_i^{(tr)} = g. \qquad (32)$$

Итак, в системе содержатся запреты вида $\left(b_i^{(d)}, x\right) = g$, где $d \in \mathbb{F}_{2^k}^* \setminus \left\{\left(z_i^{(tr)}\right)^{-1} g\right\}$. Тогда для некоторого другого вектора z рассмотрим, при каких значениях компоненты $z_i \in \mathbb{F}_{2^k}^*$ вектор z не будет решением. Для этого вычислим $\left(b_i^{(d)}, z\right) = d \cdot z_i$, тогда должно выполняться $d \cdot z_i = g$, откуда $z_i = d^{(-1)} \cdot g$. Поскольку d принимает все значения из мультипликативной группы поля, кроме значения $\left(z_i^{(tr)}\right)^{-1} g$, то вектор z не может быть решением для всех таких значений d, за исключением $d = \left(z_i^{(tr)}\right)^{-1} g$. Таким образом, отвергли все возможные значения z_i, кроме

$$z_i = d^{-1} \cdot g = z_i^{(tr)} g^{-1} \cdot g = z_i^{(tr)} \tag{33}$$

Итак, с $z_i^{(tr)} \neq 0$ вытекает $z_i = z_i^{(tr)}$, то есть все ненулевые компоненты вектора $z^{(tr)}$ совпадают со всеми ненулевыми элементами вектора z.

Пусть для фиксированного индекса i, где $1 \leq i \leq n$, компонент $z^{(tr)} = 0$, тогда в систему линейных запретов попадут все запреты $\langle b_i^{(d)}, g \rangle$ с ненулевым d на позиции i. Поэтому для любого другого вектора z выполняется $(b_i^{(d)}, z) = d \cdot z_i \neq g$. Покажем, что такие запреты исключат все возможные векторы, в которых $z_i \in \mathbb{F}_{2^k}^*$. Поскольку d пробегает все элементы $\mathbb{F}_{2^k}^*$, то $z_i \neq d^{-1} \cdot g$, где $\left\{d^{-1} \cdot g | d \in \mathbb{F}_{2^k}^*\right\} = \mathbb{F}_{2^k}^*$, поэтому остается единственный вариант $z_i = 0$. Итак, с $z_i^{(tr)} = 0$ вытекает $z_i = 0$.

Таким образом, для любого $z \in \mathbb{F}_{2^k}^n$ имеем $z = z^{(tr)}$.

Замечание. Доказанное утверждение выполняется для случая $z^{(tr)} = \overline{0}$.

Замечание. Выбор g в определении матриц $\hat{A}_{true}$ и $\hat{A}_{expl}$ является несущественным, главное, чтобы g не был равен нулю. Можно выбрать, например, g = 1, поскольку все результаты с построением и анализом матрицы

$\hat{A}_{expl}$ не зависят от выбора g, а нуждаются лишь в $g \in \mathbb{F}_{2^k}^*$. Но будем приводить все утверждения для произвольного $g \in \mathbb{F}_{2^k}^*$, считая, что он всегда фиксированный некоторой константой.

Оказывается, можно точно вычислить количество векторов в множестве $\hat{A}_{expl}$.

Утверждение 2.8. Количество векторов в матрице $\hat{A}_{expl}$ составляет $(2^k — 1) \cdot n$.

Доказательство. Будем пользоваться введенными в утверждении 2.7 обозначениями a_j и $b_i^{(d)}$, где $1 \leq j \leq n, d \in \mathbb{F}_{2^k}^*$.

Пусть $1 \leq i \leq n$ — фиксированная ненулевая позиция вектора $z^{(tr)}$. Тогда ровно один запрет $\langle a_i, 0 \rangle$ попадет в $\hat{A}_{expl}$. Количество запретов типа $\langle b_i^{(d)}, g \rangle$ будет составлять $2^k - 2$, поскольку только в случае $d = \left(z_i^{(tr)}\right)^{-1} g$ будет выполнено $(b_i^{(d)}, z^{(tr)}) = \mathrm{g}$, то есть нарушится условие принадлежности матрицы $\hat{A}_{expl}$. Другим исключением является d = 0. Итак, одна ненулевая компонента добавляет $2^k - 2 + 1 = 2^k - 1$ запретов $\hat{A}_{expl}$.

Пусть теперь $1 \leq i \leq n$ — фиксированная нулевая позиция вектора $z^{(tr)}$. Тогда ни один запрет $\langle a_i, 0 \rangle$ не попадет в матрицу $\hat{A}_{expl}$, поскольку скалярное произведение $(a_i, z^{(tr)})$ будет равняться нулю. Количество запретов $\langle b_i^{(d)}, g \rangle$, которые попадут в матрицу, будет составлять $2^k - 1$, поскольку какова бы i-и координата вектора $b_i^{(d)}$ не была, скалярное произведение $(b_i^{(d)}, z^{(tr)})$ будет равняться нулю, а фиксированный g гарантировано не является нулем. Остается лишь заметить, что в определении вектора $b_i^{(d)}$ на i-й компоненте стоял ненулевой элемент поля d, поэтому количество таких векторов будет равна $2^k - 1 + 0 = 2^k - 1$. Следовательно, нулевая компонента также добавляет $2^k - 1$ запретов $\hat{A}_{expl}$.

Общее количество запретов в множестве $\hat{A}_{expl}$ будет равна $n \cdot (2^k - 1)$. Заметим также, что количество линейных запретов вида $\langle a_i, 0 \rangle$, где $1 \leq i \leq$

n, совпадает с количеством ненулевых элементов вектора $z^{(tr)}$, поэтому за всеми возможными запретами такого вида можно точно воспроизвести все ненулевые позиции вектора $z^{(tr)}$.

Утверждение 2.9. Для систем линейных запретов с фиксированным решением и запретами, в которых правые части равны нулю или фиксированному $g \neq 0$, точка насыщения $S \leq (2^k - 1) \cdot n$.

Доказательство. По утверждению 2.7 матрица $\hat{A}_{expl}$ имеет ровно одно решение и является подмножеством $\hat{A}_{true}$, поэтому эта матрица является насыщенной по определению. По утверждению 2.8 такая матрица содержит $n \cdot (2^k - 1)$ различных векторов, а поскольку в определении точки насыщения минимум по всем насыщенным матрицам, то величина $n \cdot (2^k - 1)$ будет верхней оценкой для точки насыщения.

Рассмотрим множество линейных запретов с ненулевыми правыми частями:

$$\hat{A}'_{true} = \bigcup_{g \in \mathbb{F}_{2^k}} A^{(g)}_{true} \tag{34}$$

Это множество можно в полном множестве запретов для фиксированного вектора $z^{(tr)}$.

Утверждение 2.10. Пусть $\hat{A}'_{expl} \subset \hat{A}'_{true}$ — это система линейных запретов, которая состоит из всех возможных запретов одного из двух типов:

- запреты, в которых одна компонента равна 1, все остальные компоненты равны нулю и правая часть также равна нулю;
- запреты, в которых одна компонента равна фиксированному ненулевому элементу поля r, все остальные компоненты равны нулю, а правая часть равна любому ненулевому элементу поля s.

Тогда такая система имеет единственное решение $z^{(tr)}$.

Доказательство. Покажем, что построенная таким образом система линейных запретов является эквивалентной системе $\hat{A}_{expl}$ из утверждения 2.7

для фиксированного вектора $z^{(tr)}$, в отношении которых эти системы были сгенерированы. Фиксированную правую части $\hat{A}_{expl}$ будем обозначать g. Построим биективное отображение между множествами запретов $\hat{A}'_{expl}$ и $\hat{A}_{expl}$, а также покажем, что такое отображение для каждой из запретов хранит множество решений.

Будем пользоваться введенными в утверждении 2.7 обозначениями a_j и $b_j^{(r)}$, где $1 \leq j \leq n, x \in \mathbb{F}_{2^k}$.

Построим отображение $f : \hat{A}'_{expl} \to \hat{A}_{expl}$ таким образом:

– если запрет y имеет вид $\langle a_i, 0\rangle$ то $f(y)$ просто возвращает y;

– если запрет y имеет вид $\langle b_i^{(r)}, s\rangle$ где $s, r \neq 0$ и r — фиксированный элемент $\hat{A}'_{expl}$, то f(y) возвращает запрет $\langle b_i^{(rs^{-1}g)}, g\rangle$, где g — фиксированный элемент $\hat{A}_{expl}$.

Поскольку $\hat{A}'_{expl}$ и $\hat{A}_{expl}$ сгенерированы одним и тем же фиксированным вектором $z^{(tr)}$ то для каждого запрета с $\hat{A}'_{expl}$ всегда будет существовать запрет с $\hat{A}_{expl}$ согласно заданного отображения.

Проверим инъективность и сюръективность.

Инъективность. Покажем, что из $y_1 \neq y_2$ следует $f(y_1) = f(y_2)$. Для запретов вида $\langle a_i, 0\rangle$ это следует из инъективнисти тождественной функции. Покажем для запретов второго типа. Пусть для фиксированной позиции $1 \leq i \leq n$ запрета $y_1 = \langle b_i^{(r)}, s_1\rangle$, $y_2 = \langle b_i^{(r)}, s_2\rangle$, $s_1 = s_2$, тогда $f(y_1) = \langle b_i^{(rs_1^{-1}g)}, g\rangle, f(y_1) = \langle b_i^{(rs_2^{-1}g)}, g\rangle$, а совпадать они могут тогда и только тогда, когда $gs_1^{-1}g = rs_2^{-1}g$, то есть когда $s_1 = s_2$, а это противоречит предположению.

Сюръективность. Покажем, что для каждого запрета $y \in \hat{A}_{expl}$ существует решение уравнения f(x) = y. Для запретов вида $\langle a_i, 0\rangle$ это следует из сюръективности тождественной функции. Покажем для запретов второго типа. Пусть $y = \langle b_i^{(d)}, g\rangle$, для некоторого d = 0. Тогда можем положить $x =$

$\langle b_i^{(r)}, g \cdot d^{-1} r \rangle$ и вычислить значение функции f от этого аргумента:

$$f\ (x)\ =\ \langle b_i^{(r \cdot g^{-1} \cdot d r^{-1} g)}, g \rangle = \langle b\ _i^{(d)}, g \rangle \qquad (35)$$

Таким образом, построили биекцию между множествами $\hat{A}'_{expl}$ и $\hat{A}_{expl}$. Из этого следует, что количество запретов в этих системах является одинаковыми.

Теперь покажем, что каждый запрет $x \in \hat{A}'_{expl}$ будет эквивалентен запрету $f(x) \in \hat{A}'_{expl}$, то есть количество их решений совпадает. Для запретов типа $\langle a_i, 0 \rangle$ это очевидно выполняется, поэтому будем доказывать только для запретов второго типа.

Пусть z является решением запрета $\langle b_i^{(r)}, s \rangle \in \hat{A}'_{expl}$ для некоторого $s \neq 0$ и фиксированного $1 \leq i \leq n$. Тогда для этого вектора z выполняется $(b_i^{(r)}, z) \neq s$, т.е. $r \cdot z_i \neq s$, откуда $z_i \neq s \cdot r^{-1}$. Соответствующий ей запрет с $\hat{A}_{expl}$ будет иметь вид $\langle b_i^{(r \cdot s^{-1})}, g \rangle$. Покажем, что любой z, для которого $z_i \neq s \cdot r^{-1}$, будет также решением запрета $\langle b_i^{(r \cdot s^{-1})}, g \rangle$. Перемножим обе части запрета $z_i \neq s \cdot r^{-1}$, на $rs^{-1}g$ имеем $rs^{-1}g \cdot z_i = g$, откуда следует, что такой z является решением запрета $\langle b_i^{(r \cdot s^{-1})}, g \rangle$.

Аналогично можно показать, что любое решение запрета $\langle b\ _i^{(d)}, g \rangle \in \hat{A}_{expl}$ для фиксированных $d \neq 0$ и позиции $1 \leq i \leq n$ является решением запрета $\langle b\ _i^{(r)}, g \cdot d^{-1} r \rangle \in \hat{A}'_{expl}$. Следовательно, множества решений соответствующих линейных запретов совпадают, поэтому совпадают и множества решений систем линейных запретов $\hat{A}_{expl}$ и $\hat{A}'_{expl}$.

Допустим есть источник случайных запретов O_r, который в каждый момент времени генерирует запрет $\langle a, a_0 \rangle$, где $a \in \mathbb{F}_{2^k}^n$ и $a_0 \in \mathbb{F}_{2^k}$. Предположим также, что нельзя построить всю матрицу $\hat{A}'_{true}$, потому что имеющиеся ограниченные по времени ресурсы. Возникает вопрос, можем ли в

таком случае делать определенные локальные предположения о векторе $z^{(tr)}$. В этом случае имеет место такое наблюдение: если O_r еще не сгенерировало запрет $\langle a_i, 0\rangle$ для некоторого $1 \leq i \leq n$, то нельзя точно отличить две ситуации — запрет $\langle a_i, 0\rangle \notin \hat{A}'_{true}$ или O_r еще не успело ее сгенерировать. По этой же причине не можем восстановить ни одну из матриц $\hat{A}_{expl}$ или $\hat{A}'_{expl}$. Возникает вопрос можно ли в таком случае получать некоторую информацию о $z^{(tr)}$.

Утверждение 2.11. Пусть $b_i^{(d)} = (0, \dots, 0, d, 0, \dots, 0)$, где $d \in \mathbb{F}_{2^k}^*$ содержится на позиции с номером $1 \leq i \leq n$. Тогда выполняются следующие утверждения:

- если O_r сгенерировало запрет $\langle b_i^{(d)}, g\rangle$, то i-ая компонента вектора $z^{(tr)} \neq 0$;
- если O_r сгенерировало вектор $\langle b_i^{(d)}, g\rangle$, где $g \neq 0$, то i-ая компонента вектора $z^{(tr)}$ или равна нулю, или не равна нулю и элементу $g \cdot d^{-1}$ одновременно.

Доказательство. Рассмотрим два случая.

Если O_r сгенерировало $\langle b_i^{(d)}, g\rangle$, то $(b_i^{(d)}, z^{(tr)}) = d \cdot z^{(tr)} \neq 0$. Предположим, что $z_i{}^{(tr)} = 0$, тогда $(b_i^{(d)}, z_i{}^{(tr)}) = 0$, а значит, имеем противоречие.

Если O_r сгенерировало вектор $\langle b_i^{(d)}, g\rangle$, то $(b_i^{(d)}, z^{(tr)}) = d \cdot z^{(tr)} \neq g$. Если $z_i{}^{(tr)} = 0$, то $d \cdot z_i{}^{(tr)} = 0 \neq g$. Если $z_i{}^{(tr)} \neq 0$, то $d \cdot z_i{}^{(tr)} \neq g$, откуда следует $z_i{}^{(tr)} \neq g \cdot d^{-1}$.

Таким образом, если запреты имеют определенный вид, то можно делать предположение о неизвестном векторе. Из этого следует, что системы линейных запретов, в которых большое количество запретов с одной ненулевой компонентой в левой части, предоставляют возможность оптимизировать перебор всех возможных вариантов.

2.6 Поиск алгоритма решения систем линейных запретов над конечным полем

Перейдем к попыткам построить алгоритм для решения системы линейных запретов. Самый простой способ — это перебирать все возможные векторы размера n над полем $\mathbb{F}_{2^k}$ и для каждого из векторов проверять есть ли этот вектор решением.

Алгоритм 2.2. Вход. $\mathbb{F}_{2^k}$, матрица A размером $m \times n$ над полем $\mathbb{F}_{2^k}$.

Для каждого вектора $x_0 \in \mathbb{F}_{2^k}^n$:

а) положить $ind = 1, D = 0$,

б) для каждого $j = \overline{1,m}$:

1) проверить выполняется $\left(a^{(j)}, x_0\right) \neq 0$,
2) если не выполняется, то положить $ind = 0$ и прервать внутренний цикл;

в) если $ind = 1$, то добавить x_0 к множеству D.

Выход. Множество решений D.

Сложность метода полного перебора составляет $m \cdot n \cdot 2^{nk}$ операций умножения в поле. Будем учитывать только операции умножения в поле, поскольку операция XOR требует гораздо меньше вычислительных ресурсов и ею можно пренебречь.

При исследовании математического объекта одним из основных инструментов является сокращение имеющихся задач или попытка применить имеющиеся методы. Системы линейных запретов по своей структуре похожи на системы линейных неравенств, системы уравнений с искаженными правыми частями и системе линейных уравнений. Возникает идея попытаться перенести алгоритмы решения для перечисленных объектов на системы линейных запретов. Приведем перечень некоторых методов и аргументируем, почему они не применимы к системам запретов или не придают ускорение по сравнению с методом полного перебора:

– система линейных неравенств состоит из выражений вида $(a, x) < a_0$ и $(a, x) > a_0$. Возникает идея заменить запрет на пару отношений «<» и «>», то есть превратить запрет типа $(a, x) \neq 0$ в $(a, x) < 0$ и $(a, x) > 0$. Тогда бы получили $2 \cdot m$ неровностей и можно было бы применить к ним известные методы, но отношение «<» и «>» не имеют смысла для конечного поля, поскольку на элементах поля не определено отношение порядка. Поэтому с такими линейными запретами нельзя оперировать как с неровностями в случае поля $\mathbb{F}_{2^k}$;
– выполним преобразование над входной системой — добавим к каждому уравнению искусственные переменные, которые будем считать случайным шумом, и заменим все знаки «≠» на знак «=»:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \cdots + a_n^{(1)}x_n + \varepsilon_1 = 0 \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \cdots + a_n^{(2)}x_n + \varepsilon_2 = 0 \\ \cdots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \cdots + a_n^{(m)}x_n + \varepsilon_3 = 0 \end{cases}, \quad (36)$$

где $\varepsilon_j \neq 0$ для $j = \overline{1, m}$. Полученная система напоминает систему с искаженными правыми частями, поэтому возникает гипотеза применить соответствующие методы. Проблема заключается в том, что решение систем с искаженными правыми частями сильно зависит от предположений относительно распределения шума, а в данном случае не можем предъявить никаких (даже чисто практических) гипотез относительно распределения шума, поэтому можем считать, что он является равномерным на всем множестве элементов поля $\mathbb{F}_{2^k}$. В таком случае оценка максимального правдоподобия не сможет найти наиболее вероятных кандидатов в решении среди векторов размера n над полем $\mathbb{F}_{2^k}$;
– для удобства перенумеруем дополнительные переменные, которые были введены в прошлом пункте: $\varepsilon_{n+1}, \varepsilon_{n+2}, \ldots, \varepsilon_{n+m}$. Матрица,

которая соответствует новой системе, имеет размер $m \times (n + m)$:

$$\hat{A} = \begin{pmatrix} a_1^{(1)} & a_2^{(1)} & \cdots & a_n^{(1)} & 1 & 0 & \cdots & 0 \\ a_1^{(2)} & a_2^{(2)} & \cdots & a_n^{(2)} & 0 & 1 & \cdots & 0 \\ & & \cdots & & & & & \\ a_1^{(m)} & a_2^{(m)} & \cdots & a_n^{(m)} & 0 & 0 & \cdots & 1 \end{pmatrix}. \quad (37)$$

Можем применить к этой матрицы метод Гаусса, чтобы свести ее к виду трапеции:

$$\hat{A} = \begin{pmatrix} 1 & b_2^{(1)} & \cdots & b_n^{(1)} & b_{n+1}^{(1)} & \cdots & b_m^{(1)} & b_{m+1}^{(1)} & \cdots & b_{n+m}^{(1)} \\ 0 & 1 & \cdots & b_n^{(2)} & b_{n+1}^{(2)} & \cdots & b_m^{(2)} & b_{m+1}^{(2)} & \cdots & b_{n+m}^{(2)} \\ & & & & & \cdots & & & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & b_{m+1}^{(m)} & \cdots & b_{n+m}^{(m)} \end{pmatrix} \quad (38)$$

Если матрица приведена к виду трапеции, то ранг матрицы равен количеству ненулевых строк, $rank\ \hat{A} = m$. Количество переменных в системе равно $n + m$, поэтому система будет иметь множество решений. Для того чтобы описать все эти решения необходимо выбрать базис системы решений. Если последнюю строку матрицы $\hat{A}$ записать в виде уравнения, то она будет иметь такой вид:

$$\varepsilon_m + b_{m+1}^{(n)}\varepsilon_{m+1} + \cdots + b_{n+m-1}^{(n-1)}\varepsilon_{n+m-1} + b_{n+m-1}^{(n)}\varepsilon_{n+m} = 0 \quad (39)$$

За базис можно выбрать множество $\{\varepsilon_{n+1}, \varepsilon_{n+2}, \dots, \varepsilon_{n+m}\}$, размер которого составляет n. Теперь можем записать ε_m в этом базисе и подставить его в $(m - 1)$ уравнений системы. Таким образом, система уже не будет включать ε_m. При повторении этой процедуры на некотором шаге получим выражение для x_n

$$x_n + b_{n+1}^{(n)}\varepsilon_{n+1} + \cdots + b_m^{(n)}\varepsilon_m + b_{m+1}^{(n)}\varepsilon_{m+1} + \cdots + b_{n+m}^{(n)}\varepsilon_{n+m} = 0 \quad (40)$$

Аналогично выражаем все $x_1, \dots, x_{n-1}$. Следовательно, через этот базис выражаются $m - n$ искусственных переменных $\varepsilon_{n+1}, \varepsilon_{n+2}, \dots, \varepsilon_{n+m}$ и n

исходных переменных $x_1, x_2, \dots, x_n$. Остается подставить некоторый набор значений $(\varepsilon_{m+1}, \dots, \varepsilon_{n+m})$ и вычислить неизвестные переменные, но в этом случае принципиальным является то, что все ε_i для $i = \overline{(n+1), m}$ не могут равняться нулю. Таким образом, приходим к задаче, очень схожей с начальной: необходимо решить систему линейных уравнений относительно ε, учитывая запрет на ε, но эта система имеет простой вид за счет приведения в форму трапеции. Получается, что, применив метод Гаусса, получаем незначительную оптимизацию при переборе значений $(\varepsilon_{m+1}, \dots, \varepsilon_{n+m})$, поскольку полученная система будет содержать больше нулей, но изначальная сложность задачи при этом сохраняется. Из этого можем сделать вывод, что типичными линейными методами систему линейных запретов нельзя решить.

Учитывая рассмотренные попытки, делаем вывод, что нужно искать преобразование другого типа. Более того, это преобразование должно удовлетворять следующего свойства: при введении новых переменных или любых других операциях с переменными не нужно явно накладывать какие-то ограничения на эти переменные (такими ограничениями могут выступать линейные запрета). В случае, когда в рамках преобразования необходимо накладывать запреты, получаем исходную задачу, представленную в другом виде, то есть ее вычислительная сложность сохраняется.

Альтернативным преобразованием может быть так называемый «трюк Рабиновича». Заключается он в следующем: пусть нам нужен запрет $x \neq 0$, для $x \in \mathbb{F}_{2^k}$ заменить на равенство. Можем ввести новую замену $\gamma \in \mathbb{F}_{2^k}$ и заменить данный запрет на уравнение $1 + \gamma \cdot x = 0$. Покажем, что при такой замене множество решений исходной системы запретов остается неизменной с точностью до введенных искусственных переменных.

Утверждение 2.12. Множество решений линейной системы запретов вида

$$\begin{cases} (a^{(1)}, x) \neq 0 \\ (a^{(1)}, x) \neq 0 \\ \cdots \\ (a^{(m)}, x) \neq 0 \end{cases} \tag{41}$$

совпадает с множеством решений линейной системы уравнений вида

$$\begin{cases} 1 + \gamma_1 \cdot (a^{(1)}, x) = 0 \\ 1 + \gamma_2 \cdot (a^{(1)}, x) = 0 \\ \cdots \\ 1 + \gamma_m \cdot (a^{(m)}, x) = 0 \end{cases} \quad (42)$$

с точностью до искусственных переменных $\gamma_1, \gamma_2, \dots, \gamma_m$, где $\gamma_j \in \mathbb{F}_{2^k}$ для $j = \overline{1, m}$.

Доказательство. Пусть $\widetilde{D}$ — это множество решений системы запретов, которая состоит из векторов размера n, а R — множество решений системы уравнений, которая состоит из векторов размера $n + m$. Договоримся, что каждый вектор из множества R изначально содержит $x_1, x_2, \dots, x_n$, а затем $\gamma_1, \gamma_2, \dots, \gamma_m$.

Будем обозначать $x_{1:n}$ — первые n компоненты вектора x.

Покажем, что если $x \in \widetilde{D}$, то существует $y \in R$, такой, что $y_{1:n} = x$. Если x является решением системы линейных запретов, то для него существует такой набор элементов $z_1, z_2, \dots, z_m$, $z_j \neq 0$ для $j = \overline{1, m}$, что выполняются равенства

$$\begin{cases} (a^{(1)}, x) = z_1 \\ (a^{(1)}, x) = z_2 \\ \cdots \\ (a^{(m)}, x) = z_m \end{cases} \quad (43)$$

Подставляем все z_i, $i = \overline{1, m}$, в систему уравнений и получаем:

$$\begin{cases} \gamma_1 \cdot z_1 = 1 \\ \gamma_2 \cdot z_2 = 1 \\ \cdots \\ \gamma_m \cdot z_m = 1 \end{cases} \quad (44)$$

Положим $\gamma_j = \left(z_j\right)^{-1}$ для $j = \overline{1, m}$ и получаем решение, которое принадлежит множеству R, и в котором первые n компоненты совпадают с вектором x.

Покажем, что если $y \in R$, то $y_{1:n} \in \widetilde{D}$. Если y является решением

системы уравнений, то

$$\begin{cases} 1 + \gamma_{n+1} \cdot (a^{(1)}, y_{1:n}) = 0 \\ 1 + \gamma_{n+2} \cdot (a^{(1)}, y_{1:n}) = 0 \\ \cdots \\ 1 + \gamma_m \cdot (a^{(n)}, y_{1:n}) = 0 \end{cases} \quad (45)$$

Отсюда следует, что $(a^{(j)}, y_{1:n})$ не равна нулю для $j = \overline{1,m}$, поскольку в случае, когда для некоторого номера $1 \leq i \leq m$ выражение $(a^{(i)}, y_{1:n})$ превращается в нуль, имеем противоречие $1 = 0$. А это в свою очередь означает, что $y_{1:n} \in \widetilde{D}$.

Следовательно, $\widetilde{D}$ и R совпадают с точностью до n первых компонент вектора.

Замечание. При таком преобразовании количество решений в множестве R совпадает с количеством решений в множестве $\widetilde{D}$, т. Е. $|R| = |\widetilde{D}|$. Это обеспечивает, что в множестве R не содержатся копии некоторого первоначального решения с разными m последними компонентами вектора.

Заметим, что наличие свободных переменных в правых частях никак не влияет на ход доказательства. Следовательно, получили такую систему квадратичных уравнений над полем $\mathbb{F}_{2^k}$:

$$\begin{cases} 1 + a_0^{(1)}\gamma_1 + a_1^{(1)}x_1\gamma_1 + a_2^{(1)}x_2\gamma_1 + \cdots + a_n^{(1)}x_n\gamma_1 = 0 \\ 1 + a_0^{(2)}\gamma_2 + a_1^{(2)}x_1\gamma_2 + a_2^{(2)}x_2\gamma_2 + \cdots + a_n^{(n)}x_n\gamma_2 = 0 \\ \cdots \\ 1 + a_0^{(m)}\gamma_m + a_1^{(m)}x_1\gamma_m + a_2^{(m)}x_2\gamma_m + \cdots + a_n^{(m)}x_n\gamma_m = 0 \end{cases} \quad (46)$$

Полученная система является частным случаем задачи MQ. Известно, что задача MQ является $\mathcal{NP}$-полной задачей. Предложенный подход связывает системы линейных запретов с системами квадратичных уравнений, следовательно к системам линейных запретов могут быть использованы методы исследования систем квадратичных уравнений. Такие методы

рассмотрены в подразделе 1.1.

Введем следующие обозначения:

$$\Gamma = diag(\gamma_1, \gamma_2, \dots, \gamma_m) = \begin{pmatrix} \gamma_1 & 0 = 0 \\ 0 & \gamma_2 = 0 \\ & \dots \\ 0 & 0 = \gamma_m \end{pmatrix}, \hat{a}_0 = \overline{1} + a_0 = \begin{pmatrix} 1 + a_0^{(1)} \\ 1 + a_0^{(2)} \\ \dots \\ 1 + a_0^{(m)} \end{pmatrix},$$

тогда систему можно записать в таком виде:

$$\Gamma \cdot A \cdot x = \hat{a}_0. \quad (47)$$

Заметим, что в таком представлении принципиальным является то, что диагональная матрица Γ содержится в левой части системы. Для того, чтобы отделить Γ от A, необходимо умножить обе части этого равенства на Γ^{-1}. В соответствии с критерием существования обратной матрицы должно выполняться $det\,\Gamma \neq 0$. В случае диагональной матрицы $det\,\Gamma = \prod_{i=1}^{m} \gamma_i$. Если параметризировать произведение γ_i, $i = \overline{1,m}$, некоторым ненулевым элементом g, то есть положить $\prod_{i=1}^{m} \gamma_i = g \in \mathbb{F}_{2^k}^*$, то можно переписать систему в виде $A \cdot x = \Gamma^{-1} \cdot \hat{a}_0$. Таким образом, получаем еще одно преобразование, подобное к введению ненулевого шума в систему, которое имеет несколько недостатков. Во-первых, в системе появляется уравнения $\prod_{i=1}^{m} \gamma_i = g$ степени m, во-вторых, такую систему необходимо рассматривать для всех возможных параметров g, то есть вместо одной исходной системы необходимо рассматривать набор из $2^k - 1$ систем. Из этого можно сделать вывод, что матрицу Γ целесообразно не отделять от левой части системы.

Выводы к разделу 2

Сформулирован и доказан критерий существования решения системы линейных запретов. Этот критерий сводит задачу проверки существования

решений системы линейных запретов к задаче проверки полинома на тождественное равенство нулю над конечным полем. Поскольку полином задан в виде произведения линейных функций, то сложность такого сведения является полиномиальной. На основе этого критерия построен полиномиальный вероятностный алгоритм проверки существования решения для случая $m \leq 2^{k-1}$. Проведен анализ этого алгоритма и вычислена вероятность его односторонней ошибки, которая равна 2^{-d}, где d — это некоторое фиксированное число, которое определяет порядок точности алгоритма. Этот алгоритм был имплементирован и проверен на некотором наборе входных параметров.

Доказаны свойства решения систем линейных запретов в случае нулевых правых частей. Эти свойства характеризуют структуру множества решений системы линейных запретов и предоставляют возможность находить дополнительные решения в ситуации, когда хотя бы одно решение является известным.

3 Исследование сложности задач, связанных с системами линейных запретов

С точки зрения теории сложности, поиск неизвестного решения системы линейных запретов является исчислением, которое требует определенных ресурсов. В практических применениях обязательным условием является эффективность такого вычисления, поскольку в противном случае для работы алгоритма могут потребоваться слишком большие ресурсы. Именно поэтому важным этапом анализа математического объекта является оценка сложности задач, которые связаны с этим объектом. Поскольку различные вычислительные устройства могут иметь различную архитектуру, то необходимо абстрагироваться от конкретного устройства и измерять сложность, задающуюся в количестве определенных элементарных операций. Поэтому универсальным способом оценки сложности задачи является доказательство принадлежности этой задачи к определенному классу сложности. В этом разделе для некоторых задач, связанных с системами линейных запретов, определяются их классы сложности и исследуются свойства этих задач с вычислительной точки зрения.

3.1 Основные задачи, связанные с системами линейных запретов

Сформулируем задачу существования решения произвольной системы линейных запретов над конечным полем.

Задача 3.1 (SLR-decision). Вход. $\mathbb{F}_{2^k}$, матрица A размера $\boldsymbol{m \times n}$ над полем $\mathbb{F}_{2^k}$, вектор $\boldsymbol{a_0}$ размера $\boldsymbol{m \times 1}$.

Необходимо выяснить существует решение, либо нет над полем $\mathbb{F}_{2^k}$ у системы линейных запретов $\boldsymbol{Ax \neq a_0}$.

Выход. «Да», если решение существует, «Нет» – в противном случае.

Заметим, что в формулировке задачи на вход подается все поле $\mathbb{F}_{2^k}$. Это необходимо для того, чтобы при итерации всех элементов поля,

представленных в виде массива из битовых строк, остаться в рамках полиномиального вычисления. Если бы на вход подавалась только степень расширение поля, то цикл по всем элементам поля считался бы не неточным вычислением. Такая кодировка входных данных не является слишком избыточным, поскольку перебор всех возможных векторов фиксированной длины над конечным полем остается экспоненциальным.

Наиболее универсальный способ оценки сложности задачи — это доказательство принадлежности этой задачи некоторому классу сложности. Для задачи SLR-decision можно построить полиномиальный алгоритм проверки решения, поэтому она принадлежит классу сложности $\boldsymbol{\mathcal{NP}}$. Элементарными операциями в последующих утверждениях будем считать операции умножения элементов в поле. Операциями сложения в поле можно пренебречь.

Утверждение 3.1. Задача SLR-decision принадлежит классу задач $\boldsymbol{\mathcal{NP}}$.

Доказательство. Для того, чтобы доказать принадлежность классу $\boldsymbol{\mathcal{NP}}$, покажем для SLR-decision существует сертификат, длина которого ограничена полиномом от длины входа, а также то, что процедура проверки сертификата является полиномиальной за время от длины входа задачи.

Существование полиномиального сертификата. Сертификатом для этой задачи является решение системы линейных запретов $(\boldsymbol{x_1, x_2, \cdots, x_n})$. Длина этого вектора составляет n, что, очевидно, ограничено полиномом от размера входных данных.

Полиномиальная проверка сертификата. Проверка сертификата заключается фактически в подстановке $(\boldsymbol{x_1, x_2, \cdots, x_n})$ в систему линейных запретов и проверке, удовлетворяет ли этот вектор всем запретам. Процедуру проверки можно формализовать таким алгоритмом.

Вход. Вектор $(\boldsymbol{x_1, x_2, \cdots, x_n})$. Выполнять:

а) для каждого $\boldsymbol{j = \overline{1, m}}$:

1) проверить, выполняется ли $a_1^{(j)} x_1 + a_2^{(j)} x_2 + \cdots + a_n^{(j)} x_n \neq a_0^{(j)}$,

2) если не выполняется, то вернуть «Нет»;

б) вернуть «Да».

Выход. «Да», если $(x_1, x_2, \cdots, x_n)$ является решением системы линейных запретов, иначе «Нет».

Процедура проверки требует $n \cdot m$ операций умножения в поле $\mathbb{F}_{2^k}$, следовательно является полиномиальной от длины входных данных задачи.

Напомним, что критерий 2.1 сводил вопрос существования решения к проверки полинома на тождественное равенство нулю, но не давал возможность находить само решение. Поскольку на практике часто необходимо найти само решение, то сформулируем задачу нахождения решения системы линейных запретов.

Задача 3.2 (SLR-search). Вход. $\mathbb{F}_{2^k}$, матрица A размера т х п над полем $\mathbb{F}_{2^k}$, вектор a_0 размера $m \times 1$.

Необходимо найти хотя бы одно решение над полем $\mathbb{F}_{2^k}$ у системы линейных запретов $Ax \neq a_0$.

Выход. Решение системы линейных запретов $Ax \neq a_0$ или «⊥», если его не существует.

Рассмотрим утверждение, которое, в определенном смысле, сводит задачу поиска решения к задаче проверки существования решения.

Утверждение 3.2. Задачи SLR-decision и SLR-search являются эквивалентными относительно сводимости по Тьюрингу:

SLR-decision ≤T SLR-search.

Доказательство. Для того, чтобы доказать эквивалентность по Тьюрингу двух задач, необходимо и достаточно показать, что каждая из этих задач является сводной по Тьюрингу к другой.

SLR-decision ≤T SLR-search. Чтобы доказать сводимость, необходимо показать, что существует полиномиальный алгоритм, который с помощью полиномиальной количества обращений к оракулу, что решает задачу SLR-search, решает задачу SLR-decision.

В этом алгоритме В строится таким образом: получая на вход $(\mathbb{F}_{2^k}, A, a_0)$, В обращается к оракулу SLR-decision, получает х — решение системы

линейных запретов, если он существует; иначе получает «⊥». Соответственно, если он получил x, то возвращает ответ «Да», иначе возвращает «Нет».

SLR-search ≤T SLR-decision. Чтобы доказать сводимость, необходимо показать, что существует полиномиальный алгоритм B, который с помощью полиномиальной количества обращений к оракулу, что решает задачу SLR-decision, решает задачу SLR-search идея заключается в том, чтобы с помощью обращений к оракулу SLR-search, покомпонентно восстанавливать искомый вектор $x = (x_1, x_2, \cdots, x_n)$. Для восстановления первой компоненты x_1 вектора x переносим слагаемое $a_1^{(j)} x_1$ в правую часть для всех $j = \overline{1, m}$ (по утверждению 2.3 это не меняет множество решений системы) и последовательно фиксируем значение x_1 элементами поля $\mathbb{F}_{2^k}$. Если на некотором элементе получили ответ «Да», то фиксируем компонента x_1 этим значением d и модифицируем исходную систему линейных запретов, заменяя правые части $a_0^{(j)}$ на $a_0^{(j)} + a_1^{(j)} d$ для $j = \overline{1, m}$. Переходим к поиску следующей компоненты с модифицированными матрицей A' размерности $m \times (n-1)$ и вектор a'.

С вычислительной точки зрения для восстановления одной компоненты необходимо максимум $|\mathbb{F}_{2^k}|$ — операций обращения к оракулу. Для всех компонент вектора необходимо $n \cdot 2^k$ операций обращения к оракулу, что является полиномиальным количеством от длины входа (вектор входных данных) (поскольку на вход получаем поле $\mathbb{F}_{2^k}$ и значение n).

Корректность алгоритма следует из того, что при поиске каждой последующей координаты фиксируем только то значение, при котором существует решение всей системы. Процесс поиска вектора x можно представить в виде дерева, где каждая вершина имеет $|\mathbb{F}_{2^k}|$ потомков, а количество листов дерева составляет 2^{kn}. Задача поиска заключается в нахождении пути к листу дерева, что соответствует значению вектора x, который является решением системы линейных запретов, а оракул, фактически, помогает предотвратить перебор всех возможных путей, то есть

искать путь, уровень за уровнем, отбрасывая на каждом уровне экспоненциальное количество ошибочных вариантов. Поскольку на каждом шаге выбираем именно то значение компоненты вектора, при котором весь вектор с учетом неизвестных компонент является решением, то гарантировано находим решение. Таким образом, решение не восстановится только в том случае, если на некотором шаге выбрали ветку со значением компоненты вектора, на котором был получен ответ оракула «нет». Если система решений не имеет, то на первой же компоненте получим от оракула все ответы «нет».

3.2 Алгоритмы поиска решения систем линейных запретов

Для получения полиномиального вероятностного алгоритма поиска решения в случае $m \leq 2^{k-1}$ алгоритм 2.1 можно скомбинировать с результатами теоремы 3.2.

Теорема 3.2 описывает способ восстановления вектора-решения системы линейных запретов с помощью $n \cdot 2^k$ обращений к оракулу. Вместо обращения к оракулу будем использовать алгоритм 2.1. Он позволяет с учетом условия $m \leq 2^{k-1}$ решать задачу поиска решения систем линейных запретов с односторонней ошибкой 2^{-d}, где d — это фиксированная точность, которая устанавливается заранее. Сложность этого алгоритма составляет $d \cdot m \cdot n$ операций умножения в поле. Будем обозначать этот алгоритм Sol_exists.

Алгоритм 3.1. Вход. $\mathbb{F}_{2^k}$, матрица A размера $m \times n$ и вектор a_0 размера $m \times 1$, где $m \leq 2^{k-1}$.

Отсортировать элементы поля $\mathbb{F}_{2^k}$ в полиномиальном базисе как битовые строки по порядку роста. Эти битовые строки будут формировать массив, где $|B| = 2^k$.

Зафиксировать $d \in \mathbb{N}$, тогда ошибка одного запуска Sol_exists будет 2^{-d}.

Цикл по i от 1 до n:

а**)** цикл по j от 1 до 2^k:

1) сформировать матрицу A' из матрицы A, удалив в левых частях

всех запретов слагаемое с x_i. Сформировать вектор a'_0 из вектора a_0, присвоив $a_0^{\prime(j)}$ значения $a_0^{\prime(j)} + a_i^{\prime(j)} B_j$ для $j = \overline{1,m}$,

2) вычислить $d = Sol\_exists(\mathbb{F}_{2^k}, \mathrm{A}', a'_0)$,
3) если d = «Да», то положить $x_i = B_j$, $\mathrm{A} = \mathrm{A}'$, $a_0 = a'_0$ и прервать цикл по j;

б**)** если d = «Нет», то вернуть «⊥».

Повернуть вектор $(x_1, x_2, \cdots, x_n)$.

Выход. Вектор $(x_1, x_2, \cdots, x_n)$ или «⊥».

Замечание. Приведен алгоритм поиска решения можно упростить, если алгоритм 2.1, кроме ответа «Да», будет еще возвращать вектор, на котором выполняется $F(\mathrm{x}) \neq 0$.Тогда для поиска решения будет необходимо только проверить проверки для первой компоненты. Но в таком случае алгоритм 3.1 утратит свою универсальность. Поэтому этот алгоритм построен так, чтобы вместо алгоритма 2.1 можно было использовать произвольный полиномиальный алгоритм проверки существования решения с односторонней ошибкой 2^{-d}.

Следовательно, общая сложность алгоритма в худшем случае составляет $O(n^2 \cdot m \cdot 2^k \cdot d)$ и остается полиномиальной от длины входа. Поскольку значение d, которое фиксирует порядок ошибки, устанавливается заранее, то можно выбрать его намного больше, чем количество итераций вероятностного алгоритма, чтобы предотвратить накопление ошибки в рамках всего алгоритма. В таком случае ошибка не будет накапливаться, поскольку количество обращений к Sol_exists является полиномиальной, а функция 2^{-d} уменьшается экспоненциально с увеличением d.

Заметим, что такой алгоритм можно применять и в случае $m > 2^{k-1}$, но тогда он становится эмпирическим, поскольку в таком случае отсутствует теоретическое обоснование корректности его работы. Возможность его использования вызвана тем, что алгоритм 2.1 проверки существования решений имеет одностороннюю ошибку, то есть он может случайно

отбрасывать некоторые действительные решения, но никогда не возвращает векторы, которые не являются решениями. Таким образом может быть ситуация, когда решение системы линейных запретов существовало, а алгоритм поиска решения вернул «⊥», но не может быть ситуации, когда этот алгоритм вернул вектор, который не является решением.

Рассмотрим эмпирический алгоритм, который позволяет искать несколько решений системы линейных запретов. Под $Find\_one\_sol(\mathbb{F}_{2^k}, \mathrm{A}, a_0)$ будем обозначать полиномиальный вероятностный алгоритм 3.1, который находит одно решение системы линейных запретов с высокой вероятностью. Можно применить алгоритм нахождения одного решения таким образом: найдя некоторое решение, наложить на него запрет, и продолжить итеративно запускать вероятностный алгоритм для поиска следующих решений, добавляя для каждого из них соответствующие запреты в систему. В этом случае время работы алгоритма уже не ограничено полиномом, поскольку количество решений системы линейных запретов потенциально может быть экспоненциальным, то есть не ограниченным полиномом от длины входных данных. Для случая нулевых правых частей теорема 2.2 дает представление о структуре решений, а точнее она утверждает, что количество решений составляет $s \cdot (2^k - 1)$, где s — количество классов эквивалентности по отношению к пропорциональности. Таким образом, для случая нулевых правых частей после нахождения одного решения необходимо добавить в множество решений его класс эквивалентности, то есть множество пропорциональных ему векторов.

Чтобы реализовать такой алгоритм, необходимо уметь для найденного решения искать запрет. Таким запретом может быть вектор, ортогональный найденному решению. Задача поиска ортогонального вектора не является сложной, можно воспользоваться детерминированным алгоритмом 3.2.

Алгоритм 3.2 (Orth_search). Вход. Вектор $x \in \mathbb{F}_{2^k}^n$. Выполнять:

– если $x = \overline{0}$, то вернуть вектор $u = (1,0,\dots,0)$;

– если $x \neq \overline{0}$, то существует по крайней мере одна ненулевая

компонента. Будем обозначать позицию этой компоненты i, где

$$1 \leq i \leq n;$$

– положить $I = \{1,2,\dots,n\}$ — множество индексов вектора x. Рассчитать

$$s = \sum_{j \in I \setminus \{i\}} x_j; \quad (48)$$

– вернуть вектор u вида

$$u = (1,1,\dots,1,x_i^{-1} \cdot s,1,\dots,1), \quad (49)$$

где $x_i^{-1} \cdot s$ содержится на позиции с индексом i.

Выход. Вектор $u \neq \overline{0}$ такой, что $(u,\mathrm{x}) = 0$.

Убедимся, что $u \neq \overline{0}$ действительно является ортогональным к x:

$$(u,\mathrm{x}) = 1 \cdot x_1 + \dots + x_i^{-1} s \cdot x_i + \dots + 1 \cdot x_n = s + s = 0. \quad (50)$$

Заметим, что предложенный способ построения ортогонального вектора не единственный. Пусть $1 \leq i \leq n$ — это индекс ненулевой компоненты вектора x, тогда можно случайно сгенерировать $n-1$ элемент поля $u_1, \dots, u_{i+1}, \dots, u_n$ и вычислить

$$s = \sum_{j \in I \setminus \{i\}} r_j u_j \quad (51)$$

Тогда если $u_i = x_i^{-1} \cdot s$, то полученный вектор u тоже будет ортогональным вектору x.

Имея процедуру поиска ортогонального вектора, можем сформулировать алгоритм 3.3 поиска нескольких решений системы линейных запретов.

Алгоритм 3.3. Вход. $\mathbb{F}_{2^k}$, матрица A размера $m \times n$, вектор a_0 размера

$m \times 1$, где $m \leq 2^{k-1}$.

Инициализировать $D = \emptyset, \mathrm{A}' = \mathrm{A}$.

Повторять:

а) $d = Find\_one\_sol\ (\mathbb{F}_{2^k}, A', a_0)$;

б) если d не равно «⊥», то:

1) если $a_0 = \overline{0}$, то $D_0 = \{\mathrm{c} \cdot d, c \in \mathbb{F}_{2^k}^*\}$, иначе $D_0 = \{d\}$,
2) обновить $D = D \cup D_0$,
3) найти вектор $u = Orth\_search(d)$,
4) обновить матрицу A', добавив туда вектор u. Обновить вектор a_0, добавив туда элемент 0.

Выход. Подмножества D множества решений системы линейных запретов $\mathrm{A} \cdot \mathrm{x} \neq a_0$.

Этот алгоритм является эвристическим, поскольку, когда добавляем запрет на каждом шаге, то отбрасываем не гарантировано 1 решение (или $2^k - 1$ решений в случае $a_0 = \overline{0}$,), а по меньшей мере 1 (или $2^k - 1$ в случае $a_0 = \overline{0}$), — может случиться, что во множестве решений нашлось решение, ортогональный запрет, который добавили в систему линейных запретов. Во время практического применения данного алгоритма было эмпирически установлено, что лучшие результаты он показывает, когда мощность множества решений является небольшим. Например, в случае поля $\mathbb{F}_8$ и n = 5 при количестве решений примерно $2 - 10$, алгоритм находил все эти решения. Итак, данный алгоритм целесообразно применять в ситуации, когда система линейных запретов имеет достаточное количество линейных запретов и ее множество решений состоит из небольшого количества векторов.

3.3 Сложность частных случаев задачи проверки существования решения систем линейных запретов

Поскольку пока неизвестна более точная оценка сложности задачи SLR-decision, чем принадлежность класса $\mathcal{NP}$, то сформулируем частные случаи этой задачи и оценим их сложность.

Рассмотрим версию задачи SLR-decision, в которой поле является фиксированным и равно $\mathbb{F}_2$.

Задача 3.3 ($\mathbb{F}_2$-SLR-decision). Вход. Матрица A размера $m \times n$ над полем $\mathbb{F}_2$, вектор a_0 размера $m \times 1$ над полем $\mathbb{F}_2$.

Необходимо найти хотя бы одно решение над полем $\mathbb{F}_2$ системы линейных запретов $\mathrm{A}x \neq a_0$.

Выход. «Да», если решение существует, иначе «Нет».

Рассмотрим утверждение, которое касается сложности задачи $\mathbb{F}_2$-SLR-decision. Заметим, что по своей структуре эта задача подобна задачи XORSAT.

Утверждение 3.3. Задача $\mathbb{F}_2$-SLR-decision принадлежит классу сложности.

Доказательство. Если поле в условии задачи является фиксированным и равно $\mathbb{F}_2$, то для системы линейных запретов

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \cdots + a_n^{(1)}x_n \neq y_1 \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \cdots + a_n^{(2)}x_n \neq y_2 \\ \cdots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \cdots + a_n^{(m)}x_n \neq y_m \end{cases}, \tag{52}$$

где $a_i^{(j)}, y_j \in \mathbb{F}_2$ для $i = \overline{1,n},\ j = \overline{1,m}$, можно получить эквивалентную ей систему уравнений, выполнив замену $\widetilde{y_j} = y_j + 1$ для $j = \overline{1,m}$:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \cdots + a_n^{(1)}x_n \neq \widetilde{y_1} \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \cdots + a_n^{(2)}x_n \neq \widetilde{y_2} \\ \cdots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \cdots + a_n^{(m)}x_n \neq \widetilde{y_m} \end{cases} \tag{53}$$

Для решения таких систем существуют полиномиальные алгоритмы,

например, метод Гаусса. Он может быть использован для вычисления ранга матрицы, которая задает систему линейных уравнений, а имея ранг матрицы, можно дать ответ на вопрос существуют ли решения в исходной системе линейных уравнений. Сложность алгоритма Гаусса является кубической относительно длины входных данных, поэтому она ограничена полиномом от длины входных данных.

Рассмотрим аппроксимационную версию задачи SLR-decision, в которой не обязательно находить решение для всех линейных запретов в системе, а можно найти частичное решение, удовлетворяющее по меньшей мере $1 \leq l \leq m$ линейных запретов.

Задача 3.4 (Max-SLR-decision). Вход. $\mathbb{F}_{2^k}$, матрица A размера $m \times n$ над полем $\mathbb{F}_{2^k}$, вектор a_0 размера $m \times 1$, число l, где $1 \leq l \leq m$.

Необходимо выяснить существует ли решение над полем $\mathbb{F}_{2^k}$ в системе, которая образована по меньшей мере из l линейных запретов входной системы линейных запретов $Ax \neq a_0$.

Выход. «Да», если решение существует, иначе «Нет».

Покажем, что эта задача принадлежит классу сложности $\mathcal{NP}$.

Утверждение 3.4. Задача Max-SLR-decision принадлежит классу задач $\mathcal{NP}$.

Доказательство. Доказательство аналогично утверждению 3.1.

Существование полиномиального сертификата. Сертификатом для этой задачи, как и в случае SLR-decision есть вектор $(x_1, x_2, \cdots, x_n)$.

Полиномиальная проверка сертификата. Проверка сертификата заключается в подставлении вектора $(x_1, x_2, \cdots, x_n)$ в систему линейных запретов и подсчета количества линейных запретов, для которых $(x_1, x_2, \cdots, x_n)$ является решением. Рассмотрим алгоритм проверки сертификата.

Вход. Вектор $(x_1, x_2, \cdots, x_n)$. Выполнить:

а) установить $Counter = 0$;

б) для каждого $j = \overline{1, m}$:

1) проверить, выполняется ли $a_1^{(j)}x_1 + a_2^{(j)}x_2 + \cdots + a_n^{(j)}x_n \neq a_0^{(j)}$,

2) если выполняется, то $Counter = Counter + 1$;

в) если $Counter \geq l$, то вернуть «Да», иначе вернуть «Нет».

Выход. «Да», если x является решением минимум l линейных запретов, иначе «Нет».

Как и в случае прошлой задачи, процедура проверки требует $n \cdot m$ операций умножения в поле $\mathbb{F}_{2^k}$, следовательно является полиномиальной от длины входных данных задачи.

Для доказательства $\mathcal{NP}$-полноты задачи Max-SLR-decision будем использовать задачу Max-XORSAT.

Утверждение 3.5. Задача Max-SLR-decision является NP-сложной.

Доказательство. Для доказательства необходимо показать, что $Max - XORSAT \leq_p Max - SLR - decision.$

Напомним, каким образом формулируется задача Max-XORSAT. На вход задачи подается формула, представленная в КНФ, в которой все операции дизъюнкции в дизъюнктах заменены на операцию XOR. Формула содержит n переменных $(x_1, x_2, \cdots, x_n)$ и m выражений из XOR, которые для сокращение будем называть XOR-выражениями. В XOR-выражение может входить как переменная, так и ее отрицание. Также на вход подается l, где l — натуральное число, не превышающее количества XOR-выражений. Необходимо определить, существует ли набор значений, который удовлетворяет, по меньшей мере, l XOR-выражений в заданной формуле.

Эту формулу можно записать в виде системы уравнений, приравняв каждое XOR-выражение к единице, тогда вопрос будет заключаться в том, можно ли найти набор, который является решением для как минимум l уравнений этой системы. Следует заметить, что все операции отрицания в XOR-выражениях можно заменить на XOR переменные с единицей. Тогда в левых частях уравнений останутся только переменные, а правые части уравнений будут равны нулю в случае, когда в XOR-выражении содержалась нечетное число переменных с отрицанием, и единицы во всех других случаях.

Следовательно, система будет иметь такой вид:

$$\begin{cases} x_{i_1^{(1)}} + x_{i_2^{(1)}} + \cdots + x_{i_{k_1}^{(1)}} = y_1 \\ x_{i_1^{(2)}} + x_{i_2^{(2)}} + \cdots + x_{i_{k_1}^{(2)}} = y_2 \\ \cdots \\ x_{i_1^{(m)}} + x_{i_2^{(m)}} + \cdots + x_{i_{k_1}^{(m)}} = y_m \end{cases}, \quad (54)$$

где $1 \leq k_l \leq n$ для $l = \overline{1,m}$ — это количество переменных в каждом из XOR- выражений, $i_j^{(l)} \in \{1,2,\ldots,n\}$ для $1 \leq j \leq k_l$, $l = \overline{1,m}$ — это индексы которые задают переменные, попавшие в каждый из XOR-выражений (заметим, что в XOR-выражении не могут содержаться переменные с одинаковыми индексами или переменная и ее отрицание одновременно), $y_l \in \{0,1\}$ для $l = \overline{1,m}$ — это набор значений, которые принимают соответствующие XOR-выражения.

Видим, что индексы в такой записи фактически отражают присутствие некоторой переменной из набора $(x_1, x_2, \cdots, x_n)$ в некотором из т уравнений. Эту запись можно упростить, если ввести искусственные переменные, каждая из которых будет индикатором того, присутствует ли некоторая переменная в некотором уравнении. Тогда эта система приобретает такой вид:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \cdots + a_n^{(1)}x_n = y_1 \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \cdots + a_n^{(2)}x_n = y_2 \\ \cdots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \cdots + a_n^{(m)}x_n = y_m \end{cases} \quad (55)$$

где $a_i^{(j)} \in \{0,1\}$ для $i = \overline{1,n}$, $j = \overline{1,m}$ — эти искусственные переменные и $y_j \in \{0,1\}$ для $j = \overline{1,m}$. Заметим, что такое преобразование влияет только на представление системы уравнений и не меняет множество решений.

Поскольку y_i для $i = \overline{1,m}$ принимает только два значения, то можем обозначить $\widetilde{y_i} = y_i + 1, i = \overline{1,m}$. Тогда система имеет вид:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \cdots + a_n^{(1)}x_n \neq \widetilde{y_1} \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \cdots + a_n^{(2)}x_n \neq \widetilde{y_2} \\ \cdots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \cdots + a_n^{(m)}x_n \neq \widetilde{y_m} \end{cases} \quad (56)$$

Получили систему линейных запретов над полем $\mathbb{F}_2$. Итак, задача Max-XORSAT фактически является частным случаем задачи Max-SLR-decision. Функция сведения f, которая ставит в соответствие каждому экземпляру задачи Max-XORSAT экземпляр задачи Max-SLR-decision, — это описанное выше преобразование булевой формулы с XOR-выражениями в систему линейных запретов над $\mathbb{F}_2$.

Функция сохраняет при отображении множество экземпляров задачи Max-XORSAT с ответом «Да», то есть Max-XORSAT(x) = 1 тогда и только тогда, когда Max-SLR(x) = 1. Это следует из построения функции f — на каждом шаге выполнялись только эквивалентные преобразования початковой булевой формулы, поэтому в результате была построена система линейных запретов, в которой множество решений совпадает с множеством входов булевой формулы, на которых она принимает значение 1. Сложность вычисления функции f ограничена полиномом от входных данных, поскольку она требует лишь нескольких итераций матрицы A и вектора a_0.

Следствие 3.1. Задача Max-SLR-decision является NP-полной.

Сформулируем версию задачи SLR-decision, наложив ограничения $m \leq 2^{k-1}$ количество линейных запретов в системе. Заметим, что сформулированная таким образом задача является задачей типа promise. Это означает, что в рамках этой задачи не на все возможные входные данные необходимо предоставлять ответ «Да» или «Нет» — на часть входов, для которых $m > 2^{k-1}$, можно ничего не возвращать.

Задача 3.5 (Restricted-SLR-decision). Вход. $\mathbb{F}_{2^k}$, матрица A размера $m \times n$ над полем $\mathbb{F}_{2^k}$, вектор a_0размера $m \times 1$, где $m \leq 2^{k-1}$.

Необходимо выяснить существует ли решение над полем $\mathbb{F}_{2^k}$ у системы линейных запретов $\mathrm{A}x = a_0$.

Выход. «Да», если решение существует, иначе «Нет».

Покажем, что задача $Restricted - SLR - decision$ принадлежит классу сложности RP.

Утверждение 3.6. Задача $Restricted - SLR - decision$ принадлежит классу задач RP.

Доказательство. Чтобы доказать этот факт, необходимо показать, что существует полиномиальный вероятностный алгоритм, такой что:

– если $Restricted - SLR(\mathbb{F}_{2^k}, \mathrm{A}, a_0) = 1$, то $Pr[B(\mathbb{F}_{2^k}, \mathrm{A}, a_0) = 1] \geq \frac{1}{2}$;

– если $Restricted - SLR(\mathbb{F}_{2^k}, \mathrm{A}, a_0) = 0$, то $Pr[B(\mathbb{F}_{2^k}, \mathrm{A}, a_0) = 1] = 0$.

Построим такой алгоритм B.

Вход. $\mathbb{F}_{2^k}$, матрица A размера $m \times n$, вектор a_0 размера $m \times 1$, где $m \leq 2^{k-1}$. Выполняем следующие шаги:

– случайно равновероятно генерируем $(r_1, r_2, \cdots, r_n)$, где $r_i \in \mathbb{F}_{2^k}$ для $i = \overline{1,n}$. Вычисляем $F(r_1, r_2, \cdots, r_n)$, где $F(\mathrm{x}) = \prod_{i=1}^{m}[(a^{(i)}, x) + a_0^{(i)}]$;

– если $F(r1, r2, \ldots, rn) \neq 0$, то возвращаем «Да», иначе возвращаем «Нет».

Выход. «Да», если существует решение системы $\mathrm{A}x \neq a_0$, иначе «Нет».

В этом случае воспользовались критерием существования решений системы линейных запретов 2.1: решение существует тогда и только тогда, когда полином, состоящий из произведения всех левых частей, тождественно не равен нулю.

Этот алгоритм является полиномиальным от длины входа, поскольку требует $m \times n$ операций умножения в поле.

Убедимся в выполнении условий.

Предположим, что $Restricted - SLR(\mathbb{F}_{2^k}, \mathrm{A}, a_0) = 0$, тогда $F(x) \equiv 0$. В таком случае во всех возможных случаях будет ответ «Нет», поскольку для полинома, тождественно равного нулю, не существует набора значений, при

котором его значение будет отличаться от нуля.

Предположим, что $Restricted - SLR(\mathbb{F}_{2^k}, A, a_0) = 1$, тогда $F(x) \not\equiv 0$. В таком случае вероятность того, что $F(x)$ будет равняться нулю можно оценить с помощью леммы Шварца-Зипеля:

$$Pr_{(r_1,r_2,\cdots,r_n)\in\mathbb{F}_{2^k}}[F(r_1, r_2, \cdots, r_n) = 0] \leq \frac{m}{2^k} \leq \frac{1}{2} \quad (57)$$

Можем вычислить вероятность противоположного события:

$$Pr_{(r_1,r_2,\cdots,r_n)\in\mathbb{F}_{2^k}}[F(r_1, r_2, \cdots, r_n) = 0] \geq \frac{1}{2} \quad (58)$$

Поскольку ответ «Да» предоставляется только в случае, когда полином не равен нулю, то вероятность того, что алгоритм В даст ответ «Да», больше 0.5.

Матрица ошибок алгоритма приведена в таблице 3.1.

Таблица 2 – Матрица ошибок алгоритма решения задачи Restricted-SLR-decision

	«Да»	«Нет»
$Restricted - SLR(y) = 1$	$\geq \frac{1}{2}$	$\leq \frac{1}{2}$
$Restricted - SLR(y) = 0$	0	1

Вычисление ошибок алгоритма завершает доказательство.

Учитывая рассмотренные частные случаи SLR-decision и аналитические свойства систем линейных запретов, возникло две гипотезы относительно уточнения сложности задачи SLR-decision.

Первая гипотеза заключается в том, что задача SLR-decision является NP-полной. В пользу этой гипотезы можно привести следующие наблюдения:

– множество решений системы линейных запретов не имеет определенной структуры, как в случае систем линейных уравнений. Это указывает на то, что проверка существования решения произвольной системы линейных запретов в общем случае требует экспоненциального вычисления;
– систему линейных запретов можно представить в виде системы квадратичных уравнений над конечным полем, а задача MQ в общем случае является NP-полной. Конечно, при преобразовании системы линейных запретов на систему квадратичных уравнений полученная система имеет особую структуру, но пока не известно, как воспользоваться этой структурой;
– в случае NP-полной задачи SAT ее упрощенная версия XORSAT принадлежит классу P , а ее аппроксимационная упрощенная версия Max-XORSAT является NP-полной задачей. Такая же ситуация с частичными случаями $\mathbb{F}_2$-SLR-decision и Max-SLR-decision. Конечно, в задачи Max-SLR-decision выбор поля не ограничивается полем $\mathbb{F}_2$, но при доказывании NP-полноты достаточно лишь случая $\mathbb{F}_2$.

Вторая гипотеза заключается в том, что для задачи SLR-decision существует полиномиальный, возможно вероятностный алгоритм решения. В пользу этой гипотезы можно привести следующие наблюдения:

– все левые части запретов в системе являются линейными относительно XOR. Поэтому нельзя применить рассуждения, аналогичные доказательству $\mathcal{NP}$-полноты задачи MQ, поскольку в задаче 3SAT каждый дизъюнкт в КНФ имеет нелинейную относительно XOR структуру;
– задача SLR-decision эквивалентна задаче проверки тождественного равенства нулю полинома в случае конечного поля, а та же самая задача в случае бесконечного поля принадлежит классу сложности RP;
– задача Restricted-SLR-decision принадлежит классу сложности RP.

Уточнение оценки сложности для задачи SLR-decision планируется в дальнейших исследованиях.

Выводы к разделу 3

В данном разделе определены задачи проверки существования и поиска решения системы линейных запретов над конечным полем. Доказана эквивалентность по Тьюрингом этих задач, т. е. решение одной из этих задач можно применить для решения другой. Следовательно, если будет построен эффективный алгоритм решения SLR-decision, то для задачи SLR-search такой эффективный алгоритм также можно построить.

Поскольку доказательства эквивалентности задач SLR-decision и SLR-search является конструктивным, то вероятностный алгоритм проверки существования решения в случае $m \geq 2^{k-1}$ был использован для построения алгоритма поиска решения в случае $m \geq 2^{k-1}$. Проведен анализ этого алгоритма и установлено, что он является полиномиальным. Также этот алгоритм был имплементирован и проверен на некотором наборе входных данных. На основе алгоритма поиска решения было построен эвристический алгоритм поиска нескольких решений. Этот алгоритм использует процедуру поиска ортогонального вектора, чтобы исключать из множества решений системы найденное решение. В общем случае, такой алгоритм не является полиномиальным и не гарантирует нахождение всех решений. Этот алгоритм был имплементирован и проверен на некотором наборе входных данных. Также экспериментально установлено, что в случае достаточного количества запретов в системе и небольшой мощности множества решений, этот алгоритм восстанавливает большинство этих решений.

Сформулированы частные случаи задачи SLR-decision и определен класс сложности для каждого из этих случаев. Предложено несколько гипотез относительно уточнения сложности задачи SLR-decision и рассмотрены факторы, которые потенциально указывают на каждую из этих гипотез.

Заключение

В ходе исследования выполнены следующие задачи:

– формализована задача восстановления неизвестного вектора по частичной информации, представленной в форме линейных зависимостей, с помощью введения нотации системы линейных запретов над конечным двоичным полем. Сформулирован и доказан критерий существования решения системы линейных запретов. На основе этого критерия построен полиномиальный вероятностный алгоритм проверки существования решения для случая $m \leq 2^{k-1}$. Проведен анализ этого алгоритма и вычислена вероятность ошибки, которая является односторонней и составляет 2^{-d}, где d — фиксированное число, которое определяет порядок точности алгоритма. Этот алгоритм был имплементирован и проверен на определенных наборах входных параметров;

– рассмотрен случай, когда систему линейных запретов сгенерирована неизвестным фиксированным вектором, и исследована принципиальная возможность восстановления неизвестного вектора в таком случае. Это исследование позволило построить систему линейных запретов, которая содержит небольшое количество линейных запретов (в сравнении с количеством всех возможных линейных запретов в системе), но имеет единственное решение. Этот результат дал возможность получить нетривиальную оценку на точку насыщения в случае ненулевых правых частей;

– сформулированы задачи проверки существования решения и поиска решений для системы линейных запретов. Доказана эквивалентность этих задач по Тьюрингу, что дало возможность построить полиномиальный вероятностный алгоритм поиска хотя бы одного

решения системы линейных запретов в случае $m \leq 2^{k-1}$. Алгоритм был имплементирован и апробирован на определенных наборах входных данных. На основе этого алгоритма был построен другой эвристический алгоритм для поиска нескольких решений системы линейных запретов. В ходе апробации было обнаружено, что лучшие результаты этот алгоритм показывает в случае, когда мощность множества решения системы линейных запретов составляет примерно 1 — 10 векторов;

- установлен класс сложности задачи проверки существования решения системы линейных запретов. Также для этой задачи сформулированы определенный набор задач, которые являются частичными случаями исходной задачи. Для всех частичных случаев установлена их принадлежность соответствующим классам сложности. Эти результаты позволили выдвинуть гипотезы по уточнению сложности задачи проверки существования решения системы линейных запретов.

В дальнейших исследованиях планируется выполнить поиск полиномиального вероятностного алгоритма проверки существования решения системы линейных запретов над конечным полем в общем случае. Также планируется исследовать возможность уточнения оценки сложности задачи проверки существования решения системы линейных запретов.

Список используемой литературы

1. Баранова Е., Бабаш А. – Информационная безопасность и защита. Инфра – М. 2017 – 324 с
2. Блинов А.М. Информационная безопасность. – СПб: СПбГУЭФ, 2015 - 96с.
3. Белобородова Н. А. Информационная безопасность и защита информации: учебное пособие / Н. А. Белобородова; Минобрнауки России, Федеральное гос. бюджетное образовательное учреждение высш. проф. образования "Ухтинский гос. технический ун-т" (УГТУ). - Ухта : УГТУ, 2016. - 69 с.
4. Винберг Е. Бы. Курс алгебры / Эрнест Борисович Винберг. - Москва: МЦНМО, 2019. -592 с.
5. Громов, Ю.Ю. Информационная безопасность и защита информации: Учебное пособие / Ю.Ю. Громов, В.О. Драчев, О.Г. Иванова. - Ст. Оскол: ТНТ, 2010. - 384 с.
6. Никифоров С. Н. Защита информации : учебное пособие / С.Н. Никифоров. - Санкт-Петербург : СПбГАСУ, 2017. – 76с.
7. Шевченко В. Н. Линейное и целочисленное линейное программирование / В. Н. Шевченко, Н. Ю. Золотых, 2009, 154 с.
8. Ars G. Algebraic Immunities of functions over finite fields / G. Ars, J. Faugere. - 2005.
9. A short overview on modern parallel SAT-solvers [Электронный ресурс] / [S. Holldobler, N. Manthey, V. Nguyen и др.]. - 2011. – Режим доступа к ресурсу: https://www.researchgate.net/publication/254048189_A_short_overview_on_modern_parallel_sat-solvers.
10. Arora S. Computational Complexity: A Modern Approach / S. Arora, B.

Barak. - New York: Cambridge University Press, 2009. - 608 с.

11. Bard G. Algebraic Cryptanalysis / Gregory Bard., 2009. - (392).

12. Bard G. Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers [Электронный ресурс] / G. Bard, N. Courtois, C. Jefferson. - 2001. - Режим доступа к ресурсу: https://eprint.iacr.org/2007/024.pdf.

13. Charles Faugere. - New York: Association for Computing Machinery, 2002. - (ISSAC; 2). - С. 75-83

14. Courtois N. About the XL Algorithm over GF(2) / N. Courtois, J. Patarin // Proceedings of the 2003 RSA Conference on The Cryptographers' Track / N. Courtois, J. Patarin. - San Francisco: Springer-Verlag. - (CT-RSA; 3). - С. 141-157.

15. Cormen T. H. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest., 2009. - 1320 с.

16. Cook S. The Complexity of Theorem-Proving Procedures / Stephen Cook. // Proceedings of the Third Annual ACM Symposium on Theory of Computing. - 1971. - №71. - С. 151-158.

17. Dube T. The Structure of Polynomial Ideals and Grobner Bases / Thomas Dube. // Society for Industrial and Applied Mathematics. - 1990. - №19. - С. 750-773.

18. Ding J. Zhuang-Zi: A New Algorithm for Solving Multivariate Polynomial Equations over a Finite Field [Электронный ресурс] / J. Ding, J. Gower, D. Schmidt. - 2006. - Режим доступа к ресурсу: https://eprint.iacr.org/2006/038.pdf.

19. Ding J. Rainbow, a New Multivariable Polynomial Signature Scheme / J. Ding, D. Schmidt // Proceedings of the Third International Conference on Applied Cryptography and Network Security / J. Ding, D. Schmidt. – New York: Springer-Verlag, 2005. - (ACNS; 5). - С. 164-175.

20. ElimLin Algorithm Revisited / N. Courtois, P. Sepehrdad, P. Susil, S.

Vaudenay // Proceedings of the 19th International Conference on Fast Software Encryption / N. Courtois, P. Sepehrdad, P. Susil, S. Vaudenay. - Washington: Springer-Verlag, 2012. - (FSE; 12). - С. 306-325.

21. Faugere J. A new efficient algorithm for computing Grobner bases (F4) / Jean-Charles Faugere. - 1999.

22. Faugere J. A New Efficient Algorithm for Computing Grobner Bases without Reduction to Zero (F5) / Jean Charles Faugere // Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation / Jean

23. Faugere J. On the Complexity of the F5 Grobner basis Algorithm [Электронный ресурс] / J. Faugere, M. Bardet, B. Salvy. - 2013. - Режим доступа к ресурсу: https://arxiv.org/abs/1312.1655.

24. Fraenkel A. Complexity of problems in games, graphs and algebraic equations / A. Fraenkel, Y. Yesha. // Discrete Applied Mathematics. - 1979. - №1. - С. 15-30.

25. Fraenkel A. Complexity of Solving Algebraic Equations / A. Fraenkel, Y. Yesha. // Inf. Process. Lett.. - 1980. - №10. - С. 178-179.

26. Garey M. Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Произведенной / M. Garey, D. Johnson. - New York: W. H. Freeman & Co., 1979. - 338 с.

27. Goldreich O. Computational Complexity: A Conceptual Perspective / Oded Goldreich. - New York: Cambridge University Press, 2008. - 632 с.

28. Huang H. Algorithm for Solving Massively Underdefined Systems of Multivariate Quadratic Equations over Finite Fields [Электронный ресурс] / H. Huang, W. Bao. - 2015. - Режим доступа к ресурсу: http://arxiv.org/abs/1507.03674.

29. Karp R. Reducibility among combinatorial problems / Richard Karp // Complexity of Computer Computations / Richard Karp. - New York: Plenum Press, 1972. - С. 85-103.

30. Kipnis A. Unbalanced Oil and Vinegar Signature Schemes / A. Kipnis, J. Patarin, L. Goubin. // Advances in Cryptology — EUROCRYPT ’99. - 1999. - C. 206--222.

31. Lewin D. Checking Polynomial Identities over any Field: Towards a Derandomization? / D. Lewin, S. Vadhan // Proceedings of 30th Annual ACM Symposium on Theory of Computing / D. Lewin, S. Vadhan. - Dallas: ACM, 1998. - C. 438-447.

32. McDonald C. An Algebraic Analysis of Trivium Ciphers based on the Boolean Satisfiability Problem [Электронный ресурс] / C. McDonald, C. Charnes, J. Pieprzyk. - 2007. - Режим доступа к ресурсу: https://eprint. iacr.org/2007/129.pdf.

33. Moore C. The Nature of Computation / C. Moore, S. Mertens. - New York: Oxford University Press, Inc., 2011. - 1032 с.

34. Raddum H. New Technique for Solving Sparse Equation Systems [Электронный ресурс] / H. Raddum, I. Semaev. - 2006. - Режим доступа к ресурсу: https://eprint.iacr.org/2006/475.pdf.

35. Simple Matrix - A Multivariate Public Key Cryptosystem (MPKC) for Encryption / C. Tao, H. Xiang, A. Petzoldt, A. Ding. // Finite Fields Appl.. - 2015. - №35. - С. 352-368.

36. Solving Underdefined Systems of Multivariate Quadratic Equations [Электронный ресурс] / N. Courtosis, L. Goubin, W. Meier, J. Tacier. - 2002. - Режим доступа к ресурсу: http://www.goubin.fr/papers/CG02.pdf.