**ZK-STARK**

**Аннотация**

Человеческое достоинство требует, чтобы личная информация, например, медицинские и судебные сведения, была скрыта от общественности. Но завесой секретности, предназначенной для сохранения конфиденциальности, также могут злоупотреблять для сокрытия лжи и обмана учреждения, которым доверены конфиденциальные данные, несправедливо нанося ущерб гражданам и подрывая доверие к подобным организациям.

Системы доказательства нулевого знания (ZK) являются оригинальным криптографическим решением разногласий между принципами персональной конфиденциальности и институциональной целостности, обеспечивая соблюдение последних и не нанося ущерб первому. Общественное доверие требует прозрачности от систем ZK, что подразумевает функционирование без опоры на какую-либо доверенную сторону и отсутствие уязвимостей, которые могли бы быть использованы влиятельными сторонами для фальсификации. Для систем ZK, работающих с большими данными, крайне важно, чтобы процесс публичной проверки масштабировался сублинейно относительно размера данных. Прозрачные доказательства ZK, которые могут быть проверены за время, экспоненциально меньшее размера данных, были впервые описаны в 1990-х годах, но ранние версии были непрактичны, и до сих пор ни одна ZK система, реализованная в коде (в том числе используемая криптовалютами, такими как Zcash), не обеспечивает одновременно прозрачность и экспоненциальное ускорение проверки для общих вычислений.

В данной работе приводится первая реализация прозрачной системы ZK (ZK-STARK), в которой проверка масштабируется экспоненциально быстрее, чем размер базы данных, и, кроме того, это экспоненциальное ускорение проверки наблюдается конкретно для значимых и последовательных вычислений, описанных далее. Описываемая система использует несколько последних достижений в области **интерактивных доказательств с оракулом** (IOP), таких как “быстрая” (за линейное время) IOP система для кодов корректировки ошибок.

Данная система проверки концепции позволяет полиции доказать общественности, что профиль ДНК кандидата в президенты не фигурирует в базе данных криминалистических профилей ДНК, которую ведет полиция. Доказательство, которое генерируется полицией, не зависит от какой-либо внешней доверенной стороны и не раскрывает никакой дополнительной информации о содержимом БД или о профиле кандидата. В частности, никакая информация ДНК профиля не раскрывается никакой стороне за пределами полиции. Доказательство короче, чем размер БД ДНК, и верифицируется быстрее, происходит наивный перебор данных в БД.

1. **Введение**

**Масштабируемая проверка целостности вычислений на конфиденциальных данных**

Проблема, рассматриваемая здесь, лучше всего иллюстрируется гипотетическим примером: предположим, что полиция (P), отвечающая за национальную БД судебной экспертизы профилей ДНК (D), утверждает, что профиль ДНК (p) кандидата в президенты, который вскоре будет назначен и предположительно будет коррумпирован, не фигурирует в D. Могут ли криптографические протоколы убедить сомневающуюся общественность поверить в это утверждение без ущерба для D или p, не полагаясь на какую-либо внешнюю доверенную сторону (например, главный судья), и с “разумными” вычислительными ресурсами?

Пример соответствия профиля ДНК (DPM) является частным случаем более общей проблемы. Произвольная сторона (P), выполняющая вычисления (C) на наборе данных (D), может иметь стимул подделывать правильный выход (С(D)), вызывая проблему вычислительных целостности (CI) — гарантировать, что P действительно предоставила C(D) а не результат, по тем или иным причинам более благоприятный для P. Когда набор данных D является общедоступным, любая сторона (V), заинтересованная в проверке CI, может наивно выполнить С на D и сравнить результат с полученным от P. Однако такое решение не масштабируется, потому что время, потраченное верификатором (ТV) также велико, как и время, необходимое для выполнения программы (ТC) и V должен прочитать весь датасет D. Схемы обязательств, основанные на криптографических хеш-функциях, широко используются для вычисления нескольких неизменяемых “отпечатков” cmt для состояния в момент времени t на большом наборе данных Dt. Обычно cmt имеет незначительную длину по сравнению с Dt и может быть легко размещен в блокчейне как публичное уведомление. Таким образом, искомое решение CI должно иметь масштабируемую проверку, в которой время проверки и сложность связи масштабируются примерно так же, как log TC и |cmt| (битовая длина cmt), а не как TC и |Dt|; по крайней мере, время проверки/связи должно быть строго меньше, чем TC и |Dt|.

Когда набор данных D содержит конфиденциальную информацию, наивное решение больше не может быть реализовано, и сторона P, ответственная за D, может скрывать нарушения вычислительной целостности под предлогом секретности. Доминирующие методы обеспечения CI над конфиденциальными данными полагаются на “доверенную сторону”, например, аудитора или бухгалтера, которая наивно проверяет расчеты от имени общественности. Это решение по-прежнему не обеспечивает масштабирования, как и в случае публичных данных, и, что еще хуже, требует, чтобы общественность доверяла третьей стороне, что создает потенциальную единственную точку сбоя в протоколе, поскольку эта третья сторона — в той мере, в какой это может быть согласовано — может быть скомпрометирована злоумышленниками.

**Системы доказательств и аргументов с нулевым разглашением** представляют собой автоматизированные протоколы, заменяющие реальных аудиторов в качестве средств обеспечения вычислительной целостности на конфиденциальных данных для любых эффективных вычислений, устраняя коррумпированность и снижая вычислительные затраты. ZK-система S для вычислений C представляет собой пару вероятностных алгоритмов S = (P, V):

* P – алгоритм доказательства вычислительной целостности;
* V – алгоритм верификации доказательства P.

Полнота и обоснованность S подразумевают, что P может эффективно доказать все истинные утверждения, но не сможет убедить V в каких-либо ложных (кроме, быть может, ничтожной вероятности). Самые первые теоретические разработки систем ZK с масштабируемыми верификаторами для общих вычислений, обсуждавшиеся в начале 1990-х годов, были основаны на вероятностно проверяемых доказательствах (PCP). Знаменитая теорема PCP предложила удивительный компромисс между временем выполнения, затраченным доказывающим на построение доказательства (TP), и временем выполнения, затраченным проверяющим, верифицирующим это доказательство (TV). Этот компромисс означает, что время доказательства увеличивается полиномиально по сравнению с наивным временем вычисления (TP = TСO(1)), в то время как время проверки уменьшается экспоненциально по отношению к нему (TV = logO(1) TC).

Система ZK, основанная на теореме PCP (ZK-PCP), имеет три дополнительных преимущества, которые необходимы для обеспечения доверия к целостности вычислений. Во — первых, предположения, на которых основана безопасность этого подхода - существование устойчивых к коллизиям хэш-функций для интерактивных решений и общий доступ к случайной функции (“модель случайного оракула”) для неинтерактивных — делают данное решение устойчивым к атакам крупномасштабных квантовых компьютеров (постквантово безопасные). Ожидаемое увеличение масштабов квантовых компьютеров и призыв к постквантовым криптографическим протоколам, например, Национальным институтом стандартов и Технологии (NIST), подчеркивают важность постквантового безопасного решения ZK.

Во-вторых, ZK-PCP являются системами доказательства знаний (proof of knowledge - POK) или, в случае описанной выше реализации, системами аргументации знаний (argument of knowledge - ARK). Неформально, в контексте примера DPM, ZK-ARK является доказательством того, что полиция использовала “истинный” набор данных Dt и профиль ДНК p кандидата в президенты.

В-третьих, что наиболее важно, ZK-PCP прозрачны (“публичная случайность”), т.е. случайность, используемая верификатором, является общедоступной; в частности, настройка ZK-PCP не требует внешнего доверенного этапа настройки, в отличие от более новых решений ZK, в том числе используемых криптовалютой Zcash. Прозрачность необходима для постоянного общественного доверия, поскольку она серьезно ограничивает способность даже самых могущественных сторон P злоупотреблять системой, и, таким образом, прозрачные системы - это те, которым общественность может надежно доверять, пока в наблюдаемой вселенной существует что-либо непредсказуемое (источники случайности).

Подводя итог, можно сказать, что ZK-PCP системы являются отличным методом обеспечения общественного доверия к IC на конфиденциальных данных и имеют шесть основных достоинств:

* Прозрачность (i);
* Универсальность (ii) — распространяются на любые эффективные вычисления С, даже если они требуют дополнительного (а возможно, и конфиденциального) входа типа Dt;
* конфиденциальность (ZK) (iii) — не компрометируют вспомогательные входы типа Dt
* пост-квантовая безопасность (iv);
* доказательство/аргумент знания (v);
* масштабируемость проверки (vi).

Хотя ZK-PCP известны с середины 1990-х годов, до сих пор ни один из них не был реализован в коде, потому что, согласно недавнему исследованию [110], “доказательства, вытекающие из теоремы PCP (несмотря на асимптотические улучшения) были настолько длинными и сложными, что для их создания и проверки потребовались бы тысячи лет, а для их хранения понадобилось бы больше битов памяти, чем атомов во Вселенной”. Следовательно, недавние усилия по реализации систем ZK для общих вычислений посвящены альтернативным методам, которые не реализуют всех перечисленных выше свойств, хотя некоторые из них чрезвычайно эффективны на практике для конкретных размеров схем и для амортизированных вычислений.

**Интерактивные доказательства с оракулом и масштабируемые доказательства**

Для улучшения масштабируемости доказывающего алгоритма без ущерба для свойств (i)–(vi) недавно была предложена новая модель [22, 94], называемая **интерактивным доказательством с оракулом** (IOP), являющаяся обобщением моделей IP, PCP и интерактивного PCP (IPCP). Как и в PCP, верификатору IOP не нужно читать сообщения доказывающего целиком, а вместо этого запрашивать их в разных местах; как и в IP, проверяющий и верификатор взаимодействуют в течение нескольких раундов. Как и в случае с ZK-PCP, система ZK-IOP может быть преобразована в интерактивную ARK систему, предполагающую семейство устойчивых к коллизиям хэш-функций, и может быть превращена в неинтерактивный аргумент в случайной модели с оракулом, которая обычно реализуется с использованием стандартной хэш-функции. В качестве строгого обобщения IP/PCP/IPCP модель IOP предлагает несколько преимуществ.

Наиболее важным для этой работы является улучшенная масштабируемость IOP, описанная ниже; это преимущество имеет место как асимптотически — при размере входных данных n → ∞, так и для конкретных длин входных данных, возникающих на практике. Основываясь на этой эффективности, недавно была представлена доказательная реализация концепции ВГД под кодовым названием SCI; однако SCI не имеет ZK, и его конкретная длина аргумента и время доказательства все еще довольно велики. Недавно были описаны IOP системы с (идеальным) ZK и масштабируемыми верификаторами, сначала для NP, затем для NEXP. В обеих работах время выполнения проверки (TP) ограничено TC · logO(1) TC; мы называем это масштабируемым временем доказательства (также известным как квазилинейное время доказательства).

Отныне мы будем называть (универсальную) систему ZK (vi’) полностью масштабируемой или просто масштабируемой, если время выполнения как проверяющего, так и верификатора масштабируемо; это оправдано, поскольку обе величины являются почти оптимальными, с точностью до полилогарифмических коэффициентов. Система ZK-IOP, удовлетворяющая свойствам (i)–(v) и полной масштабируемости (vi’) будет называться **масштабируемым прозрачным IOP знаний** (ZK-STIK). Подводя итог, можно отметить, что недавно были представлены теоретические варианты построения систем ZK-STIK, но их конкретная эффективность и применимость к “практическим” вычислениям до сих пор не были продемонстрированы.

Мы представляем новый вариант (вдвойне) масштабируемой и прозрачной системы ZK в модели IOP (ZK-STIK). Мы реализуем эту систему как ZK-STARK и применяем ее для доказательства концепции “осмысленных” вычислений, которые являются очень последовательными по своей природе - проблема DPM, представленная ранее. Наша реализация обеспечивает (i) время проверки, которое строго меньше, чем наивное время выполнения (TV < TC) и (ii) сложность связи, которая строго меньше размера свидетельства (?). Основным нововведением и источником улучшенной производительности в этой системе является расширенная зависимость от модели IOP, включающая **протокол быстрого интерактивного доказательства близости с оракулом Рида-Соломона** (Fast Reed-Solomon (RS) IOP of Proximity (IOPP) (FRI) protocol) и новую процедуру арифметизации. Мы подчеркиваем, что экспоненциальное ускорение времени проверки и размера свидетельства, описанное далее (и показанное на рисунке 1), применимо к любому вычислению, определенному для произвольно большого размера свидетельства, хотя конкретный момент, в котором это ускорение проявляется, зависит от сложности вычисления.

**2. Методы**

В этом разделе приводятся основные нововведения, лежащие в основе (двойной) масштабируемости и конкретной эффективности описываемой реализации ZK-STARK; экспозиция короткая и неформальная. В разделе 3 приводится формальное описание используемой теоретической модели; в приложении В формально описываются шаги алгоритма сокращения (доказанные в последующих разделах).

**Обзор**

Многие ZK-системы (включая нашу) используют арифметизацию - метод, впервые примененный для доказательства нижних границ схем (каких?), а затем адоптированный для систем интерактивных доказательств. **Арифметизация** — это сведение вычислительных задач к алгебраическим задачам, которые включают многочлены “низкой степени” над конечным полем F; “низкая степень” в данном контексте означает, что степень значительно меньше размера поля.

Отправной точкой для арифметизации во всех системах доказательств является утверждение о вычислительной целостности, которое проверяющий хочет доказать, например:

|  |  |
| --- | --- |
| *“α является результатом выполнения C для T шагов на (публичном) входе x”* | (\*\*) |

Обратите внимание, что утверждение DPM (\*) является частным случаем (\*\*). Для ZK-STARK и для связанных с ним более ранних реализаций подобных систем результатом арифметизации является пара **проблем проверки близости Рида-Соломона** (RPT), и масштабируемость ZK-STARK основана на новом решении проблемы RPT, которое обсуждалось ранее; позже мы более подробно объясним процесс арифметизации.

**2.1 Быстрое интерактивное доказательство близости с оракулом Рида-Соломона (FRI3rd)**

Для S ⊂ F и параметра ρ ∈ (0, 1) код Рида-Соломона RS[F, S, ρ] представляет собой семейство функций f : S → F, которые являются оценками многочленов степени < ρ|S|. Проблема RPT предполагает, что верификатору предоставляется oracle-доступ к f и вспомогательной информации, такой как вероятностно проверяемое доказательство близости (PCPP) или интерактивное доказательство близости с оракулом (IOPP); задачей проверки - отличить с высокой степенью вероятности и с небольшим количеством запросов к f и вспомогательным PCPP/IOPP оракулам, случай F ∈ RS[F, S, ρ] и случай, когда F составляет 0,1-далеко от (всех членов) RS[F, S, ρ] в относительном расстоянии Хэмминга. Поиск решений проблемы RPT (частный случай проблемы “тестирования с низкой степенью”) является основным узким местом для прозрачных систем.

ZK-STARK использует новый протокол для решения проблемы RPT, называемый **быстрым RS IOPP (FRI)**. FRI - это первое решение проблемы RPT для достижения **строго линейной арифметической сложности формирования доказательства** — 6 · |S| арифметических операций в F — и **строго логарифмической арифметической сложности проверки доказательства** - 21 · log |S| арифметических операций; кроме того, доказательство может быть построено за log |S| циклов на параллельной машине и структурировано таким образом, чтобы привести к коротким аргументам. FRI значительно улучшает, как асимптотически, так и конкретно, предыдущие решения проблемы RPT, которые требовали арифметической сложности квазилинейных доказательств (θ(|S| · logO(1) |S|)).

**2.2 Арифметизация I — Алгебраическое промежуточное представление (AIR)**

Арифметизация состоит из нескольких этапов, которые аналогичны другим процессам компиляции программ и схем, поэтому мы заимствуем используемую там терминологию и адаптируем ее к нашему процессу.

Первая фаза арифметизации заключается в построении алгебраического промежуточного представления (AIR) программы C. Неофициально, AIR - это множество:

полиномов низкой степени с коэффициентами в F над парой множеств переменных:

которые представляют соответственно текущее и следующее состояние вычисления. AIR определяет отношение перехода вычисления C в том смысле, что пара соответствует единственному допустимому переходу (или “циклу”) C тогда и только тогда, когда:

т. е. тогда и только тогда, когда является общим решением AIR-системы P. Следующие параметры P определяют сложность проверки и верификатора, поэтому их минимизация является основной целью данного этапа. **Степень AIR** равна deg(P) = ; **ширина (состояния)** — это количество переменных (w), необходимых для описания состояния; **(AIR) размер** – это число ограничений (s), и **счетчик цикла** - это количество машинных циклов, необходимое для исполнения C; когда программа обрабатывает большое число (n) элементов данных, как в случае с тестом DPM, нас интересует количество циклов на элемент, обозначаемое как *c*; общее количество циклов для *n* элементов *c\*n*. Если вычисление “расширено” до схемы (как обычно происходит в других решениях, описанных в разделе 1.3), количество циклов является нижней границей глубины схемы; для сравнения с другими системами мы вычисляем в крайнем правом столбце таблицы 4 общее количество элементов умножения для этой расширенной схемы, поскольку эта мера вместе с глубиной схемы являются мерами сложности, которые определяют сложность проверки и проверки.

Таблица 4. Основные параметры сложности базовых криптографических примитивов и бенчмарка ДНК соответствия профиля (DPM).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Размер состояния *w*** | **Число циклов *c*** | **Степень *d*** | **Размер системы *s*** | **#× gates** | **#× gates** | **Total #× gates** |
| **SHA2** | 56 | 3762 | 11 | 1065 | 720 | 3194 | 2708640 |
| **AES128+DM** | 62 | 48 | 8 | 327 | 486 | 1033 | 23328 |
| **Rij160+DM** | 68 | 58 | 8 | 318 | 891 | 1390 | 51678 |
| **DPM** | 81 | 62 | 8 | 626 | 1520 | 1520 | 90954 |

Основной вклад в сложность проверки в наших бенчмарках вносит стоимость доказательства вычислительной целостности повторных вызовов криптографической хэш-функции; другие вычисления незначительны по сравнению с этой стоимостью. Таким образом, выбор конкретной хэш-функции (H) имеет большое значение, как и ее определение в терминах P. ZK-STARK использует двоичное (характеристика 2) поле потому что (i) оно позволяет выполнять эффективные арифметические операции (например, сложение эквивалентно xor) и (ii) его алгебраическая структура необходима для протокола FRI3rd. Таким образом, криптографическая хэш-функция, которую мы ищем, является “дружественной к двоичным полям”, т.е. неформально говоря, ее AIR имеет небольшие параметры сложности при определении над двоичными полями. В таблице 4 приведены основные параметры сложности AIR для бенчмарка DPM, описанного в разделе 1, и для трех хэш–функций: Secure Hash Algorithm 2 (SHA2) и хэш Дэвиса-Мейера [111] на основе блочного шифра Рейндаэля [44] со 128 битами (AES128+DM) и 160 битами (Rij160+DM).

**2.3. Арифметизация II — Алгебраически связывающее интерактивное доказательство с оракулом (ALI)**

Основной причиной сложности по пространству и времени алгоритма доказательства является стоимость выполнения полиномиальной интерполяции и обратной к ней операции — многоточечной полиномиальной оценки. Основная мера сложности выполнения этих операций — это максимальная степень многочлена, который алгоритм доказательства должен интерполировать и/или оценить; для вычисления на наборе данных размером *n* обозначим эту степень как *dmax(n)*. Предыдущая эталонная модель давала для этого параметра следующую оценку:

|  |  |
| --- | --- |
|  | (3) |

что вело к конкретно большим значениям (первая колонка таблицы 5). ZK-STARK уменьшает это значение до:

|  |  |
| --- | --- |
|  | (4) |

Улучшенная эффективность ZK-STARK обусловлена двумя причинами, описываемыми ниже. Первая полностью убирает второе слагаемое из (1), а вторая позволяет избавиться от компоненты *w* в первом слагаемом.

**Алгебраически связываемые интерактивные доказательства с оракулом (ALI)**

Второе слагаемое (3) возникает из-за того, что нашему доказывающему алгоритму необходимо применить “**локальную карту**”, индуцированную AIR-системой P. Предыдущие эталонные системы использовали локальную карту, которая проверяет каждое ограничение AIR отдельно, что приводит к возникновению в (3) второго слагаемого. Вместо этого ZK-STARK использует один раунд взаимодействия, чтобы свести все ограничения *s* к единственному ограничению, представляющему собой случайную линейную комбинацию P1, ..., Ps. Этот раунд взаимодействия полностью удаляет второе слагаемое в (3).

**Кодирование на основе регистров**

Наивное вычисление, выполняемое проверяющим, может быть записано с помощью трассировки выполнения, двумерного массива с *c · n* строками и *w* столбцами, в котором каждая строка представляет состояние вычисления в определенный момент времени (для простоты рассматриваются пространственно-ограниченные вычисления), а каждый столбец соответствует алгебраическому регистру, отслеживаемому во всех циклах *c · n*. Предыдущие системы кодировали полную трассировку выполнения одним кодовым словом Рида-Соломона, что приводило к степени *n · c · w*; затем эта степень умножается на *d* для учета применения вышеупомянутой “локальной карты” к кодовому слову, в результате чего получается первое слагаемое (3).

ZK-STARK использует отдельное кодовое слово Рида-Соломона для каждого регистра, что приводит к возникновению *w* кодовых слов, каждое из которых имеет более низкую степень *n · c.* На первый взгляд этот компромисс может показаться расточительным, потому что теперь нам нужно решить проблему RPT для каждого из *w* кодовых слов. Однако взаимодействие и использование случайности, допускаемые моделью IOP, снова приходят нам на помощь: достаточно решить одну задачу RPT, применяемую к случайной линейной комбинации из всех *w* кодовых слов. Использование одного кодового слова на регистр также помогает снизить сложность связи, как описано в разделе 2.5 ниже.

В таблице 5 сравнивается максимальное значение dmax для ZK-STARK с предыдущим эталоном, а также показан мультипликативный коэффициент уменьшения 6.5 × 104-1.8 × 105 для вычислений, описанных в разделе 2.2 и таблице 4.

Таблица 5. Максимальная степень вычислений, описанных в разделе 2.2.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **SHA2** | 6323922 | 41382 | 18491 |
| **AES128+DM** | 39504 | 384 | 6584 |
| **Rij160+DM** | 49996 | 464 | 6896 |
| **DPM** | 78988 | 496 | 10192 |

**2.4 Расширения низкой степени и степень композиции**

Уменьшение dmax позволяет ZK-STARK достичь большей масштабируемости. Однако, с ростом размера входных данных *n* растет dmax. Основное «бутылочное горлышко» в реализации доказывающего и проверяющего – это вычисление расширения низкой степени (LDE) трасировки исполнения, определяемое далее.

**Определение 2.1 (Low degree extension (LDE)).** Даны конечные множества S, S’ поля F, такие, что |S’|>|S|, и функция f: S -> F. Расширением низкой степени f на S’ называется функция f’:S’ ->F, имеющая тот же интерполяционный полином, что и f.

LDE обычно вычисляется с помощью полиномиальной интерполяции, за которой следует шаг полиномиальной мультиточечной оценки. Эталонные алгоритмы интерполяции и оценки полиномов над бинарными полями известны как аддитивные FFT-алгоритмы, так как они используют быстрое преобразование Фурье. Для увеличения масштабируемости доказывающего ZK-STARK использует недавнюю новую реализацию аддитивного FFT. Исследование данного алгоритма показывает, что его применение ведет к арифметической сложности в *3|S’|log|S’|* для множеств S’. Аддитивные FFT требуют операций. Более того, паттерн доступа к памяти данного кодирующего алгоритма приводит к желаемому времени исполнения, сравнимому с предшествующими реализациями.

Использование аддитивного FFT, имеющего строго квазилинейную арифметическую сложность, позволяет также получить первую ZK-STIK систему, в которой сложность доказательства строго квазилинейна, а сложность верификации строго логарифмическая, при размере трассировки исполнения *c \* n \* w*.

2.5 Минимизация сложности пути аутентификации (APC) и коммуникационной сложности (CC)

Наибольший вклад в коммуникационную сложность, а также во временную и пространственную сложность верификации в ZK-STARK вносит стоимость реализации модели IOP через деревья Меркла. Далее описывается, как ZK-STARK минимизирует эту стоимость.

Схема фиксации-раскрытия (commit-reveal) Килиана требует, чтобы доказывающий выполнял фиксацию (commit) для каждого оракула путем отправки корня дерева Меркла, листья которого выбраны оракулом. IOP включает несколько оракулов, а следовательно, и несколько деревьев Меркла и несколько корней\фиксаций. После того, как доказывающий выполнил фиксацию для всех оракулов, верификатор запрашивает у этих оракулов данные на случайных позициях. Когда доказывающий раскрывает (reveal) ответы оракулов на эти запросы, каждый ответ должен быть дополнен **путем аутентификации,** доказывающим, что ответы на запросы консистентны по отношению к ранне зафиксированным корням деревьев Меркла.

Пусть – число битов результата криптографической хеш-функции, используемой для конструирования дерева Меркла; пусть APtotal обозначает общее число узлов в пути аутентификации во всех поддеревьях деревьев Меркла, чьи листья являются ответами на запросы, и пустть qtotal обозначает число запросов ко всем доказывающим оракулам. Тогда общая коммуникационная сложность системы доказательства имеет вид:

Помимо того, что ZK-STARK уменьшает первое слагаемое с помощью улучшенного анализа обоснованности, она также уменьшает второе слагаемое двумя различными путями:

1. верификатор ZK-STARK запрашивает строки трассировки исполнения LDE, каждая строка состоит из w элементов поля и представляет собой состояние вычисления в определенный момент времени. Чтобы снизить коммуникативную сложность, доказывающий в ZK-STARK помещает каждую такую строку в отдельное поддерево дерева Меркла, в связи с чем затем для каждой строки требуется только один единственный путь аутентификации.
2. Во время фазы QUERY протокола FRI запрашиваются аффинные классы смежности фиксированного подпространства. Доказывающий в ZK-STARK помещает каждый такой класс смежности в отдельное поддерево дерева Меркла, снижая таким образом число путей аутентификации на каждый класс смежности.

Под конец с целью оптимизации времени исполнения и дельнейшего снижения коммуникационной сложности в качестве хеш-функции в ZK-STARK используются хеши Дэвиса-Мейера вместе с AES.

**3 STIK и STARK – формальные определения и первичные конструкции**

Недетерминированная машина Тьюринга M, распознающая язык за время (n – размер экземпляра), индуцирует бинарное отношение R­M, состоящее из всех пар , где и – последовательность недетерминированных выборов M(), приводящая к **принимающему состоянию**. В таком случае говорим, что R = RM индуцировано L b неявно полагаем, что M фиксирована и известна. Наиболее интересующий нас язык – это **NEXP-полный язык вычислительной целостности.**

**Определение 3.1 (Вычислительная целостность).** Бинарное отношение RCI – это множество пар , где:

* , M – недетерминированная машина Тьюринга, x и y – вход и выход соответственно, T >= S – целые числа, обозначающие ограничения по времени и пространству соответственно.
* – описание недетерминированных выборов в M на входе x, приводящее к достижению выхода y за не более чем T шагов при использовании не более чем S ячеек ленты данных (не считая ячеек, в которых записано x).

**Вычислительная целостность** языка LCI есть проекция бинарного отношения RCI на его первую координату, т. е. .

**3.1 Масштабируемые прозрачные IOP знания (STIK)**

STARK, который будет определен позднее, есть реализация STIK, описываемого в этом разделе.

**Определение 3.2 (IOP).** Пусть R – бинарное отношение, индуцированное недетерминированным языком L, а обозначает **ошибку обоснованности**. **Система интерактивного доказательства с оракулом** S для R с ошибкой – это пара интерактивных рандомизированных алгоритмов S = (P, V), удовлетворяющих следующим свойствам:

* **Операционная схема**: на вход верификатору поступает , на вход доказывающему поступает для некоторой строки . Число интерактивных раундов, обозначаемое через , называется **раундовой сложностью системы**. В течение одного раунда доказывающий шлет сообщение (которое может зависеть от и предыдущих сообщений), а верификатор предоставляет доступ оракулу к этому сообщению, затем верификатор отвечает сообщением доказывающему. Обозначим через выход V после взаимодействия с P; выход может принимать два значения – *accept* или *reject*.
* Полнота: если , тогда
* Обоснованность: если , тогда

**4 От AIR до ZK-STARK**

**Экземпляр вычислительной целостности**, обозначаемый через , определяется **отношением перехода** над пространством **машинных состояний** и множеством **граничных условий** (таких, как входы и выходы). **Свидетелем** целостности является валидное выполнение вычисления, предоставляемое **трассировкой исполнения** – последовательностью машинных состояний, соответствующих граничным и переходным условиям вычисления. Приведение утверждений вычислительной целостности, как например (\*\*), решается простым применением теоремы Кука-Левина.

Процесс включает в себя 4 части:

1. Отправная точка – естественное алгебраическое промежуточное представление и , обозначаемое через и соответственно. Под алгебраическим понимается представление, в котором состояния трассировки исполнения представляются в виде последовательностей элементов в конечном поле F, а отношение перехода определяется множеством полиномов над переменными, описывающими текущий и следующий шаг трассировки исполнения.
2. Затем AIR преобразовывается в представление, в котором состояния трассировки исполнения расположены в узлах аффинного графа так, что последовательные состояния соединены ребром графа. Неформально можно сказать, что аффинный граф представляет собой схему с алгебраической топологией. Процесс расположения машинных состояний в узлах графа напоминает процесс размещения и маршрутизации, часто используемый в проектировании компьютерных схем, однако наше пространство проектирования ограничено алгебраическими, а не физическими законами. Мы называем данное преобразование **алгебраическим размещением и маршрутизацией** (APR), а результатом его является пара . Данное преобразование является детерминированным со стороны верификатора, то есть не предполагает случайности и взаимодействий со стороны верификатора. Оно также обладает совершенной полнотой и обоснованностью (доказывающий использует случайность для получения нулевого знания; данное решение не влияет на полноту и обоснованность).
3. APR затем используется для получения с помощью одного раунда IOP пары экземпляров **проблемы проверки близости Рида-Соломона** (RPT). Экземпляры определяются параметрами **кода Рида-Соломона**. Оба кода определяются над одним и тем же полем F, но имеют различные области оценки и рейты . В данном случае свидетель — это пара кодовых слов . Случайность верификатора в однораундовом IOP используется для связывания многочисленных ограничений отношения перехода в единое (случайное). Мы называем данный шаг **алгебраическим связыванием IOP (ALI)**.
4. Под конец для каждой из двух функций (оракулов) вызывается **протокол быстрого RS IOP близости**, и на этом преобразование завершается.