|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_Программное обеспечение ЭВМ и информационные технологии\_\_

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | | Карпухин Александр Сергеевич | |
| Группа |  | *фамилия, имя, отчество*  ИУ7-21М | |
| Тип практики | | технологическая | |
| Название предприятия | | ОАО «ЭРРАЙВАЛ РУС» | |
| Студент | |  |  |
| Руководитель практики от предприятия | |  | *Подпись, дата фамилия, и.о.* |
| Оценка | |  | *Подпись, дата фамилия, и.о.* |
| Руководитель практики от МГТУ | |  | Исаев А.Л. |

*Подпись, дата фамилия, и.о.*

*2021 г.*

**Оглавление**

[**Введение** 3](#_Toc81835306)

[**1** **Аналитический раздел** 5](#_Toc81835307)

[**1.1** **Обзор предметной области** 5](#_Toc81835308)

[**1.1.1** **Доказательства знания** 5](#_Toc81835309)

[**1.1.2** **Интерактивные доказательства** 9](#_Toc81835310)

[**1.1.3** **Интерактивные доказательства с несколькими доказывающими** 10](#_Toc81835311)

[**1.1.4** **Вероятностно проверяемые доказательства** 10](#_Toc81835312)

[**1.1.5** **Доказательство с нулевым разглашением** 11](#_Toc81835313)

[**1.1.5.1** **Идеальное нулевое разглашение** 11](#_Toc81835314)

[**1.1.5.2** **Вычислительное нулевое разглашение** 12](#_Toc81835315)

[**1.1.5.3** **Статистическое нулевое разглашение** 13](#_Toc81835316)

[**1.2** **Описание существующих реализаций** 13](#_Toc81835317)

[**1.2.1** **ZK-SNARK** 13](#_Toc81835318)

[**1.2.2** **ZK-STARK** 13](#_Toc81835319)

[**1.2.2.1 Масштабируемая верификация** 13](#_Toc81835320)

[**1.2.2.2 Интерактивные масштабируемые доказательства с оракулом** 17](#_Toc81835321)

[**1.2.3** **Сравнение существующих решений** 18](#_Toc81835322)

[**2** **Технологический раздел** 20](#_Toc81835323)

[**2.1** **Платформа Etherium** 20](#_Toc81835324)

[**2.1.1** **Общие сведения** 20](#_Toc81835325)

[**2.1.2** **Состояние системы** 20](#_Toc81835326)

[**2.1.3** **Виртуальная машина Etherium** 21](#_Toc81835327)

[**2.1.4** **Транзакции** 22](#_Toc81835328)

[**2.1.5** **Смарт-контракты** 23](#_Toc81835329)

[**Список использованных источников** 24](#_Toc81835330)

# **Введение**

При организации распределенных вычислений и многоагентных систем, состоящих из множества самостоятельных интеллектуальных сущностей, фундаментальной является проблема достижения общей надежности при условии наличия неисправных компонентов. Решение состоит в обеспечении консенсуса или согласования некоторого состояния системы и данных, его описывающих, которое необходимо во время вычислений.

Проблема консенсуса - основная проблема в управлении публичными распределенными системами, такими, как, например, блокчейн-сети. Решение проблемы должно отвечать на вопрос о том, каким образом узлам сети достичь единой точки зрения на журнал транзакций, руководствуясь лишь общими правилами обработки сообщений в сети, при условии, что произвольные узлы могут отказывать или быть скомпрометированы. Большинство алгоритмов консенсуса основаны на публичном хранении информации обо всех операциях в сети с целью предоставления любому участнику сети доступа к ним для подтверждения достоверности.

Большой объем верифицируемых данных и их публичность привели к возникновению необходимости создания алгоритмов, которые позволили бы сократить объем вычислений для обеспечения консенсуса и не требовали бы для этого раскрытия информации обо всех пользовательских операциях. Одним из направлений повышения эффективности алгоритмов обеспечения консенсуса является идея сокращения объем данных, содержащихся непосредственно в распределенной системе, а также обеспечение консенсуса не для каждой операции в сети, а только в строго определенных «реперных» точках. На текущий момент наиболее известными формализациями, позволяющими заменить данные и цепочку операций над ними в распределенной системе короткой подписью (доказательством), являются протоколы доказательства с нулевым раскрытием ZK-SNARK и ZK-STARK. Обе эти формализации могут использоваться в рамках более широкой группы подходов для повышения эффективности блокчейн-сетей, многие из которых предполагают хранение оригинальных данных вне блоков сети.

В рамках практической работы ожидается выполнение следующих задач:

* изучение теоретических основ доказательств с нулевым разглашением;
* изучение и анализ существующих протоколов доказательства с нулевым разглашением и их реализаций;
* реализация протокола ZK-STARK в виде смарт-контракта проверки доказательства для некоторого простого частного случая (генерация самого доказательства может быть выполнена за пределами сети с использованием существующих библиотек реализации ZK-STARK).

# **Аналитический раздел**

# **Обзор предметной области**

Понятие доказательства является одним из центральных в современной криптографии и теории сложности. К примеру, класс NP описывает множество языков, чья принадлежность классу может быть определена детерминированным верификатором, выполняющим проверку за полиномиальное время путем чтения строк доказательства полиномиальной длины; данный класс отражает традиционное понятие математического доказательства. За последние три десятилетия исследователи изучили и представили системы доказательства, обобщающие приведенное выше традиционное понятие. В этом разделе приводятся основные теоретические сведения, необходимое для определения исследуемых алгоритмов.

# **Доказательства знания**

Интуитивно можно сказать, что двухсторонний протокол является **системой доказательства знания,** если одна сторона, называемая **проверяющим V**, убеждается в том, что другая сторона, именуемая **доказывающим P**, в действительности обладает некоторой информацией.

Положим для простоты, что проверяющий всегда будет убежден, т. е. после взаимодействия с доказывающим он всегда находится в принимающем состоянии. Говоря, что доказывающему известно убеждающее утверждение, имеют ввиду, что он «может быть изменен» так, чтобы произвести это утверждение. Идея «изменения машины M» рассматривается эффективными алгоритмами, использующими M в качестве оракула. Вернее, существует эффективный алгоритм, называемый **экстрактором знания**, который для некоторой формулы *φ* на входе и при наличии доступа оракула к «хорошему» доказывающему (т. е. такому, который всегда убеждает проверяющего на входе *φ*), способен сформировать на выходе убеждающее утверждение для *φ*.

Проблема возникает в общем случае, когда доказывающий может убедить проверяющего с некоторой вероятностью . В случае постоянного трудностей не возникает, и можно потребовать, чтобы даже в этом случае экстрактор знания успешно производил убеждающее утверждение за ожидаемое полиномиальное время (или же формировал такое утверждение за полиномиальное время с вероятностью, экспоненциально близкой к 1). Данная интерпретация верна также, когда – произвольная не пренебрежимо малая функция от длины входа *φ*. Однако это не так в случае, когда доказывающий не убеждает проверяющего с не пренебрежимо малой вероятностью.

Данное утверждение обусловлено тем, что последовательность вероятностей может одновременно не быть ни пренебрежимо малой (т. е. меньше чем для всех и всех достаточно больших *n*), ни не пренебрежимо малой (т. е. больше чем для некоторого и всех достаточно больших *n*), то есть последнее не обязательно подразумевает первое, а, следовательно, неверно их отождествлять.

Основным аспектом приводимого далее определения является отсутствие различия между доказывающими, убеждающими проверяющего с не пренебрежимо малой вероятностью или нет. В данном случае основным требованием является утверждение о том, что экстрактор знания всегда успешно убеждает проверяющего, и среднее число выполняемых им шагов обратно пропорционально вероятности того, что доказывающий убедит проверяющего.

**Случайным оракулом** называется идеализированная хэш-функция, на каждый новый запрос выдающая случайный ответ, равномерно распределенный по области значений. При этом для одного и того же запроса ответ будет одинаковым.

Пусть - бинарное отношение, . Если , говорят, что *y* является свидетельством *x*.

Система доказательства, определяемая далее, представляет собой двухсторонний протокол, каждая сторона которого моделируется интерактивной функцией, определение которой приводится далее.

**Интерактивная функция A** ставит в соответствие каждому (общий вход) и (префикс диалога) вероятностное распределение на , обозначаемое . Случайно выбранный из этого распределения элемент обозначается через .

Взаимодействие доказывающего и проверяющего на некотором общем входе x состоит из последовательности ходов, в каждый из которых одна сторона отправляет сообщение другой. Участники чередуют ходы, и для простоты полагаем, что первый ход выполняет доказывающий, а последний – проверяющий. Обозначим через и случайные переменные, представляющие сообщения, отправляемые доказывающим и проверяющим соответственно на *i*-ом шаге взаимодействия. Каждое сообщение определяется интерактивной функцией от общего входа и всех предыдущих сообщений:

Эти случайные переменные определены над вероятностными выборами обоих интерактивных функций.

Положим, что существует функция , обозначающая число раундов взаимодействия и зависящая только проверяющего *V* и общего входа *x*), такая, что -ый ход проверяющего содержит сообщение о принятии или отклонении доказательства. **Транскриптом взаимодействия** называется случайная переменная со строковым значением, хранящая сообщения V и P от начала взаимодействия и до принятия проверяющим решения:

Так как транскрипт содержит вердикт проверяющего, то для каждого общего входа *x* можно говорить о множестве принимающих транскриптов и множестве отклоняющих транскриптов . Таким образом, вероятность того, что проверяющий достигает принимающего состояния, есть величина .

Пусть K – машина со случайным оракулом, и A – интерактивная функция. Тогда есть случайная переменная, описывающая выход K с оракулом и входом *x*. Утверждение о том, что является оракулом, означает, что K задает некоторую строку , и на очередном шагу получает случайный элемент из .

Пусть *p(x)* – вероятность того, что доказывающий *P* убедит проверяющего *V* на входе *x*, а *k(x)* интуитивно определяется как вероятность того, что проверяющий достигнет принимающего состояния даже при ложном доказательстве.

Пусть *R* - бинарное отношение, , *V* – интерактивная функция, вычислимая за вероятностное полиномиальное время. Говорят, что *V* является **проверяющим знания для отношения *R*** с **ошибкой знания *k*** если выполняются следующие условия:

* **нетривиальность (полнота)**: существует интерактивная функция , такая, что для всех все возможные взаимодействия V с на общем входе x принимаются, т. е. ;
* **валидность (с ошибкой k)**: существует константа *c > 0* и машина со случайным оракулом *K*, такая, что для каждой интерактивной функции P и каждого машина *K* удовлетворяет следующему условию:

если , тогда на входе *x* и при наличии доступа к оракулу машина *K* выдает на выходе строку из множества *R(x)* за ожидаемое число шагов, ограниченное числом .

Машина с оракулом *K* называется **универсальным экстрактором знания**, а *k* называется **функцией ошибки знания** [1].

# **Интерактивные доказательства**

Интерактивные доказательства были представлены в работе [2] в 1989 году. В ***k*-раундовом интерактивном доказательстве** вероятностный полиномиальный алгоритм проверяющего V обменивается *k* сообщениями с доказывающим P, причем предполагается, что последний обладает неограниченными вычислительными ресурсами. В результате проверяющий либо принимает, либо отвергает доказательство. IP[*k*] – это класс языков с *k*-раундовым интерактивным доказательством.

Формальное определение: пусть . Пара (P, V) называется **системой интерактивного доказательства** для языка L, если:

где числа и – константы, удовлетворяющие условиям:

Независимо от определения интерактивного доказательства в работе [3] были описаны так называемые игры Артура-Мерлина: ***k*-раундовой игрой Артура-Мерлина** является *k*-раундовое интерактивное доказательство, в котором все сообщения проверяющего являются единообразно и независимо случайными. AM[*k*] – это класс языков с *k*-раундовой игрой Артура-Мерлина.

Голдвассер и Сипсер в [4] показали, что обе модели являются в равной степени мощными – для полиномиального *k* выполняется . Было также доказано, что интерактивные доказательства соответствуют множеству языков, разрешимых в полиномиальном пространстве, то есть .

# **Интерактивные доказательства с несколькими доказывающими**

**Интерактивные доказательства с несколькими доказывающими** были описаны в работе [5] и представляют собой *k*-раундовые интерактивные доказательства, в которых проверяющий *k* раз взаимодействует с *p* независимыми не коммуницирующими доказывающими и принимает или отвергает доказательство. Класс языков с *k-*раундовыми интерактивными доказательствами с *p* доказывающими обозначается через .

В работе [5] авторы показывают, что двух доказывающих всегда достаточно, то есть , и что все языки из NP в данной модели имеют идеальное доказательство с нулевым разглашением. Далее в [6] было продемонстрировано, что взаимодействие с двумя доказывающими эквивалентно взаимодействию с единственным доказывающим при наличии доступа некоторого оракула к строке доказательства, и отсюда было получено, что . Помимо этого, было показано, что любой язык из NEXP имеет однораундовое интерактивное доказательство с двумя доказывающими, т. е. .

# **Вероятностно проверяемые доказательства**

Вероятностно проверяемые доказательства были описаны в работах [6, 7, 8]. В таком доказательстве вероятностный полиномиальный алгоритм проверяющего V имеет возможность предоставить доступ машины Тьюринга с оракулом к строке доказательства; в данном случае обозначает класс языков, для которых проверяющий использует максимум *r* бит случайных данных и совершает максимум *q* запросов к различным частям доказательства (при этом максимальная длина доказательства составляет 2r). Из утверждения для MIP было получено, что . Последующие работы показали, что . Данное утверждение известно как PCP-теорема.

# **Доказательство с нулевым разглашением**

Пусть (P, V) – протокол с нулевым разглашением, т.е. проверяющий V убеждается в корректности утверждения доказывающего P, не узнав ничего нового о его закрытых входных данных. Для этого необходимо ограничить вычислительные ресурсы V полиномом, зависящим от размера его входных данных. Далее приводится описание следующих аспектов нулевого разглашения:

* идеальное нулевое разглашение;
* вычислительное нулевое разглашение;
* статистическое нулевое разглашение.

# **Идеальное нулевое разглашение**

Если случайные величины в транскрипте доказательства равномерно распределены в соответствующем вероятностном пространстве и независимы от общих входных данных P и V, то бессмысленно утверждать, что они допускают утечку информации [9]. Положим, что в данном случае P взаимодействует с V на языке, не обладающем избыточностью, т.е. имеющем наибольшую энтропию. Следовательно, не важно, насколько большими вычислительными ресурсами обладает V, так как он не сможет извлечь из транскрипта никакой дополнительной информации.

Протокол интерактивного доказательства *(P, V)* для языка L называется **идеальным протоколом с нулевым разглашением**, если для любого предложения транскрипт доказательства можно создать с помощью некоторого алгоритма , время работы которого полиномиально зависит от длины *x*, с одним и тем же вероятностным распределением. Эффективный алгоритм называется **имитатором ZK-протокол**. Однако в случае, когда *(P, V)* – идеальный протокол с нулевым разглашением, алгоритм называется **уравнителем**.

# **Вычислительное нулевое разглашение**

Пусть и – два множества ансамблей, в которых и – случайные величины из конечного выборочного пространства S. Обозначим через . Пусть – вектор случайных переменных, которые порождены в результате эксперимента *E* или , где *l* – величина, ограниченная полиномом, зависящим от *k*.

**Классификатор** пары проедставляет собой вероятностный алгоритм, работающий за время, ограниченное полиномом, зависящим от *k*. Результат работы принадлежит множеству *{0, 1}* и удовлетворяет следующим условиям:

* тогда и только тогда, когда ;
* тогда и только тогда, когда .

Говорят, что классификатор различает пару с преимуществом , где:

Ансамбли E и называются полиномиально неразличимыми, если для пары не существует ни одного классификатора с преимуществом , которое при всех достаточно больших значениях k не было бы пренебрежимо малым.

Протокол интерактивного доказательства *(P, V)* для языка L называется **вычислительным протоколом с нулевым разглашением**, если для любого предложения транскрипт доказательства можно подделать с помощью полиномиального алгоритма S(x), причем вероятностное распределение поддельной стенограммы полиномиально неотличимо от истинного.

# **Статистическое нулевое разглашение**

Понятие статистического протокола с нулевым разглашением было введено в работе [10]. Протокол интерактивного доказательства называется статистическим протоколом с нулевым разглашением, если существует эффективный имитатор, позволяющий подделать транскрипт доказательства с точностью, которая не поддается статистическому распознаванию. Статистический алгоритм распознавания аналогичен классификатору, введенному выше, за исключением того, что время его работы не обязательно полиномиально ограничено.

# **Описание существующих реализаций**

# **ZK-SNARK**

ZK-SNARK – протокол краткой неинтерактивной аргументации знаний с нулевым разглашением. Протокол основан на PCP-теореме, согласно которой языки из класса NP имеют «короткое» вероятностно проверяемое доказательство.

# **ZK-STARK**

# **1.2.2.1 Масштабируемая верификация**

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 S(P, V) 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.

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 ZK-STARK. 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 [12], 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.

# **1.2.2.2 Интерактивные масштабируемые доказательства с оракулом**

Для улучшения масштабируемости доказывающего алгоритма без ущерба для свойств (i)–(vi) недавно была предложена новая модель [13], 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.

Далее будем называть (универсальную) систему ZK 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 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.

ZK-STARK представляет новый вариант масштабируемой и прозрачной системы ZK в модели IOP (ZK-STIK). Данная реализация обеспечивает время проверки, которое строго меньше, чем наивное время выполнения (TV < TC) и 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. Важно отметить, что экспоненциальное ускорение времени проверки и размера свидетельства применимо к любому вычислению, определенному для произвольно большого размера свидетельства, хотя конкретный момент, в котором это ускорение проявляется, зависит от сложности вычисления.

# **Сравнение существующих решений**

Сравнение описанных выше протоколов ZK-SNARK и ZK-STARK приведено в таблице 1.2.3.1.

Таблица 1.2.3.1 Сравнение протоколов ZK-SNARK и ZK-STARK

|  |  |  |
| --- | --- | --- |
| **Критерий** | **ZK-SNARK** | **ZK-STARK** |
| Алгоритмическая сложность P | O(N log N) | O(N polylog N) |
| Алгоритмическая сложность V | O(1) | O(polylog N) |
| Коммуникационная сложность (число переданных сообщений) | O(1) | O(polylog N) |
| Размер доказательства | 200 байт; фиксированный | От нескольких десятков до сотен килобайт; нефиксированный |
| Фаза доверительной установки | Требуется | Не требуется |
| Устойчивость к атакам квантовых компьютеров | Нет | Да |
| Криптографические допущения | Строгие | Существование устойчивых к коллизиям хеш-функций |

Как видно из сравнения, ZK-STARK имеет более высокую алгоритмическую и коммуникационную сложность, а также генерирует доказательства нефиксированного большого размера. При этом основным аргументом в пользу ZK-STARK считается отсутствие необходимости в проведении доверительной установки, что полностью исключает участие третьих лиц в процессе построения доказательств и обеспечивает максимальную вычислительную целостность на приватных данных. В дополнение к этому ZK-STARK является также постквантовым протоколом и основывается на менее строгих криптографических допущениях, предполагающих наличие устойчивых к коллизиям хеш-функций.

# **Технологический раздел**

# **Платформа Etherium**

# **Общие сведения**

**Etherium** – платформа для децентрализованных вычислений с использованием смарт-контрактов на базе блокчейн-сети. Последняя представляет собой криптографически защищенную децентрализованную транзакционную систему с единым состоянием, где:

* **криптографическая защищенность** означает, что безопасность обеспечивается криптографически стойкими протоколами, атаки на которые требуют обладания недостижимыми на практике вычислительными ресурсами либо настолько значительные временные затраты на раскрытие, что раскрываемая информация к тому времени утратит свою актуальность;
* **децентрализованность** подразумевает, что все клиенты являются равноправными участниками сети;
* **транзакционность** означает, что все операции в сети выполняются в режиме транзакций, и их результаты не будут отражены в ней до фиксации соответствующей транзакции в очередном блоке;
* **единое состояние** подразумевает, что существует только одно заданное состояние системы, на основе которого выполняются все транзакции, и это состояние является единым для всех участников сети.

# **Состояние системы**

Общее состояние системы представляет собой отображение адресов (160-битных идентификаторов) на состояния аккаунтов. Данное состояние хранится в виде модифицированного префиксного дерева Меркла, в котором лист представляет собой состояние аккаунта, а путь до него от корня дерева – его идентификатор (хэш). Будучи неизменяемой структурой данных, данный способ хранения позволяет откатится к предыдущему состоянию простым изменением хэш-значения корня дерева, так как все они хранятся в сети.

Глобальное состояние системы формируется на основе состояний ее аккаунтов:

* **внешние аккаунты** – учетные записи, создаваемые пользователями сети с помощью закрытых ключей и не имеющие никакого связанного с ними кода;
* **аккаунты смарт-контрактов** – учетные записи программ для EVM; обращение к ним приводит к вызову связанного с ними кода.

Состояние любого аккаунта в системе характеризуется следующими полями:

* **nounce** – число транзакций с данного адреса для пользовательских аккаунтов или количество созданных контрактов для аккаунта смарт-контракта;
* **balance** – число Wei (минимальная единица внутренней валюты системы), которым владеет данный аккаунт;
* **storageRoot** – 256-битное хэш-значение корневого узла дерева состояния, которое содержит хранилище (storage) данного аккаунта;
* **codeHash** – хэш кода EVM, который должен быть исполнен при получении сообщения данным адресом (для аккаунтов смарт-контрактов).

# **Виртуальная машина Etherium**

**Виртуальная машина Etherium (EVM)** поддерживается в виде децентрализованной вычислительной системы, исполняемой на всех узлах сети, подключенных к ней через клиент Etherium. EVM можно представить в виде распределенного конечного автомата, состояние которого хранится в сети.

EVM – стековая виртуальная машина с глубиной стека в 1024 записи. Каждая запись представляет собой 256-битное слово, размер которого был выбран в связи с повсеместным использованием 256-битной криптографии. При исполнении контрактов машина поддерживает изменяемую область памяти - **memory**, которая существует только во время исполнения транзакции. При этом контракты имеют постоянное хранилище - **storage**, содержащееся в глобальном состоянии системы в дереве Меркла.

# **Транзакции**

**Транзакции** – это действия, инициируемые внешними пользовательскими аккаунтами. Транзакции, меняющие состояние EVM, должны быть распространены на всю сеть; любой узел сети может распространить запрос на исполнение транзакции, после чего майнер исполнит ее и распространит результирующее состояние на всю остальную сеть. Таким образом, транзакции должны «майниться», чтобы считаться валидными.

Любая транзакция включает в себя следующую информацию:

* **recipient** – адрес получателя (если это внешний аккаунт - транзакция осуществляет передачу значения, если это контракт – будет исполнен код контракта);
* **signature** – идентификатор отправителя, генерируемый в момент подписания транзакции приватным ключом отправителя и подтверждающий, что отправитель авторизовал транзакцию;
* **value** – объем внутренней валюты, пересылаемый в рамках транзакции;
* **data** – опциональное поле, включающее дополнительные данные (аргументы конструктора, если транзакция адресована контракту);
* **gasLimit** – максимальный объем «газа» - внутренней валюты, которая может быть потрачена на исполнение транзакции; фактически «газ» специфицирует количество вычислительных инструкций;
* **gasPrice** – стоимость единицы «газа» в ETH.

После того, как отправитель сформировал и подписал транзакцию, выполняются следующие шаги:

* генерируется хэш транзакции;
* транзакция передается в сеть и включается в пул транзакций;
* майнер выбирает транзакцию и включает в блок для верификации;
* после подтверждения блок включается в цепочку.

Транзакция имеет так называемый подтверждающий номер, отражающий число блоков, включенных в цепочку после блока с данной транзакцией. Чем больше это число, тем выше вероятность, что транзакция была исполнена сетью. Это связано с тем, что существует вероятность отката всех транзакций в блоке, если он не будет верифицирован достаточным количеством участников сети или окажется в более коротком ответвлении цепочки блоков.

# **Смарт-контракты**

Смарт-контракты представляют собой программы, исполняемые в EVM. Они включают в себя данные и методы, исполняемые при получении транзакции по адресу контракта.

Данные контракта могут храниться в двух различных местах:

* **storage** – постоянное хранилище, в котором хранится состояние контракта; существует перманентно с момента загрузки контракта в сеть;
* **memory** – временная память, существующая только во время исполнения контракта; так как не является перманентной, более эффективна в обращении.

# **Заключение**

В ходе практики были изучены теоретические основы доказательств и аргументации знаний, а также протоколов доказательств и аргументации знаний с нулевым разглашением.

Был проведен анализ и сравнение основных существующих протоколов аргументации знаний с нулевым разглашением – ZK-SNARK и ZK-STARK, в результате которого были выявлены сильные и слабые стороны обеих моделей.

Для реализации протокола ZK-STARK была выбрана блокчейн-платформа Etherium, использующая в качестве языка смарт-контрактов Solidity.

# **Список использованных источников**

1. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO ’92, pages 390–420, 1993.
2. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. SIAM J. Comp., 1989.
3. Laszlo Babai. Trading group theory for randomness. In STOC ’85, 1985.
4. Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In STOC ’86, 1986.
5. Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In STOC ’88, 1988.
6. Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. 1988.
7. Laszlo Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In STOC ’91, 1991.
8. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. JACM, 1998
9. Мао В. Современная криптография: Теория и практика / пер. Д. А. Клюшина — М.: Вильямс, 2005. — 768 с. — ISBN 978-5-8459-0847-6
10. Goldwasser, Shafi & Micali, Silvio & Rackoff, Chales. (2019). The knowledge complexity of interactive proof-systems. 10.1145/3335741.3335750.
11. Wood, Daniel Davis. “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER.” (2014).
12. Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. Commun. ACM, 58(2):74–84, 2015**.**
13. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs, pages 31–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.