# *Lelantus Spark: безопасные и гибкие приватные транзакции*

Арам Дживанян (Aram Jivanyan)[1,2*] и Аарон Фейкерт (Aaron Feickert)3
1 Firo
2 Ереванский государственный университет
3 Cypher Stack

**Аннотация**. Нами предлагается модификация протокола приватных транзакций Lelantus, обеспечивающая приватность получателя, более высокий уровень безопасности, а также дополнительные практичные свойства. Наша схема децентрализованных анонимных платежей (DAP), Spark, предусматривает наличие неинтерактивной одноразовой адресации, скрывающей адрес получателя при совершении транзакций. Модифицированный формат адреса обеспечивает гибкость видимости транзакций. Владельцы адресов могут при желании без ущерба для безопасности позволить третьим сторонам просматриваться входящие или все транзакции, связанные с адресом; эта функциональная возможность позволяет снизить нагрузку, связанную со сканированием блокчейна и вычислением баланса, не делегируя полномочий стороны, осуществляющей трату средств. Также предусматривается возможность делегирования затратных операций, связанных с доказательством, без ущерба для полномочий стороны, осуществляющей трату средств, при построении транзакций. Кроме того, техническое решение совместимо со схемой проведения операций с простой линейной мультиподписью, что позволяет взаимно не доверяющим друг другу сторонам совместно принимать и строить транзакции, связанные с адресом с мультиподписью. Нами приводятся доказательства того, что Spark соответствует формальным свойствам безопасности DAP с точки зрения баланса, невозможности внесения изменений и единообразия данных в реестре.

## 1 Введение

С момента появления протокола транзакций Bitcoin, обеспечивающего возможность построения транзакций и использования выходов из реестра, а также ограниченную, но полезную возможность написания сценариев, было проведено большое количество исследований в области протоколов распределённых цифровых активов. Тем не менее, протоколы на базе Bitcoin также имеют множество недостатков, связанных с обеспечением приватности: при проведении транзакций раскрывается информация об адресе источника и сумма, а при последующих тратах раскрывается и адрес получателя. Помимо этого, данные и метаданные, связанные с транзакциями, такие как содержание сценариев, могут оставить нежелательный след после проведения транзакций.

Более поздние исследования были направлены на обход или полное устранение таких ограничений с охранением уже существующего полезного функционала, например, операций с мультиподписями или возможности выборочного просмотра транзакций третьими сторонами. Технические решения и ориентированные на обеспечения приватности криптовалюты, такие как Beam, Firo, Grin, Monero и Zcash используют различные подходы для достижения этой цели, и данные подходы подразумевают наличие различных компромиссов. Основанный на RingCT протокол, используемый в настоящее время Monero, например, на практике обеспечивает ограниченную анонимность отправителя, благодаря масштабируемой по занимаемому пространству и времени схеме подписи [17, 9]. Протоколы Sprout и Sapling, поддерживаемый Zcash [11] (равно как и их реализованные на этот день обновления) требуют генерирования доверенного параметра, необходимого для самозагрузки их основанных на итерации систем доказательства, и взаимодействуют с прозрачными выходами типа Bitcoin таким образом, что появляется риск утечки информации [2, 4]. Схема на базе протокола Mimblewimble, используемая в качестве основы Grin, допускает утечку данных графов до того, как майнерами будет выполнена операция объединения [8]. Для решения проблемы связываемости, присущей Mimblewimble, разработчиками Beam была

* Электронный адрес автора: aram@firo.org

придумана и реализована в их системе адаптированная версия протокола Lelantus, подходящая для совместного использования с протоколом Mimblewimble и скрывающая графы транзакций [19]. Протокол Lelantus, который в настоящее время использует Firo, не обеспечивает приватности получателя; он поддерживает только «чеканку» монет и трату произвольных сумм с сокрытием подписанта, но это происходит с использованием прозрачных выходов типа Bitcoin, что может привести к утечке информации о личности получателя [12]. Seraphis [20] является похожим в плане технического решения протоколом транзакций, и разрабатывался в то же самое время.

В данной работе нами предлагается Spark, итерация протокола Lelantus, позволяющая строить приватные транзакции, не требующие доверенных настроек, и обеспечивающая приватность отправителя, получателя и суммы транзакции. Транзакции в Spark, как и транзакции Lelantus и Monero, используют заданные анонимные группы со стороны отправителя, которые состоят из ранее сгенерированных и защищённых выходов. Параллельная система доказательства, по сути, являющаяся адаптированной версией, предложенной Гротом, Бутлем и др. (Groth and Bootle et al.) [10, 3] (представляет собой отдельный интерес и используется в других модифицированных версиях Lelantus [12] и Triptych [16]), доказывает, что тот выход, который тратится. Действительно присутствует в анонимной группе; суммы шифруются и алгебраически скрываются в обязательствах Педерсена, а тег, который выводится из верифицируемой случайной функции [7, 13], не даёт многократно потратить тот же самый выход, что в контексте протокола транзакций считается попыткой двойной траты.

Транзакции Spark поддерживают эффективную групповую верификацию, в рамках которой доказательства диапазона и траты выгодно используют общие элементы и параметры доказательства, что снижает уровень предельных затрат, связанных с верификацией каждого доказательства в такой группе; в сочетании с правильно выбранной анонимной группой со стороны отправителя, сэкономленное путём групповой верификации время может быть значительным.

Spark реализует дополнительные полезные функции. Использование модифицированного доказательства дискретного логарифма Шаума-Педерсена, задающее полномочия со стороны траты и правильную конструкцию тега, позволяет эффективно производить операции подписания и операции с мультиподписью, подобные операциям, описанным в работе [14], где выполнение затратных с точки зрения необходимых вычислений доказательств может быть делегировано более производительным устройствам с учётом ограниченных требований к аспекту доверия. Кроме того, протокол добавляет два уровня возможной «видимости» транзакций без делегирования полномочий, связанных с тратой. Передаваемые ключи просмотра позволяют обозначенной третьей стороне идентифицировать транзакции, содержащие выходы, направленные по какому-либо адресу, равно как и соответствующие суммы и зашифрованные сопроводительные данные. Полные ключи просмотра позволяют обозначенной третьей стороне дополнительно определять, когда полученные выходы будут потрачены в дальнейшем (но без каких-либо данных получателя), в результате чего появляется возможность проверки баланса и подотчётности в приложениях, использующих схему пороговой мультиподписи, где такое свойство считается необходимым.

Все конструкции, используемые в Spark, требуют исключительно генерирования публичного параметра, что гарантирует отсутствие необходимости в доверенных третьих сторонах для инициализации протокола или обеспечения его целостности.

## 2 Предварительные криптографические значения

В данной работе в отношении групповых операций нами используется аддитивная нотация. Допустим, $N$ является множеством неотрицательных целых чисел $\{0,1,2,\ldots\}$.

### 2.1 Схема обязательства Педерсена

Гомоморфная схема обязательства является конструкцией, позволяющей производить односторонние алгебраические представления входных значений. Схема обязательства Педерсена является гомоморфной схемой обязательства, использующей чрезвычайно простую конструкцию линейной комбинации. Допустим, $pp_{com}=(G,F,G,H)$ являются публичными параметрами схемы обязательства Педерсена, где $G$ — группа порядка, равного простому числу, в которой задача дискретного логарифмирования является сложной, $F$ — её поле скалярных величин, а $G,H\in G$ являются равномерно выбранными случайными генераторами. Схема обязательства содержит алгоритм $Com:F^2\to G$, где $Com(v,r)=vG+rH$, являющийся гомоморфным в том смысле, что

$$Com(v_1,r_1)+Com(v_2,r_2)=Com(v_1+v_2,r_1+r_2)$$

для всех таких входных значений $v_1,v_2\in F$, и который маскирует $r_1,r_2\in F$. Кроме того, конструкция идеально скрывает входные значения и является вычислительно обязательной.

Данное определение естественным образом расширяется до схемы обязательства с двойной маской. Допустим, $pp_{d\,com}=(G,F,F,G,H)$ являются публичными параметрами схемы обязательства Педерсена с двойной маской, где $G$ — группа порядка, равного простому числу, в которой задача дискретного логарифмирования является сложной, $F$ — её поле скалярных величин, а $F,G,H\in G$ являются равномерно выбранными случайными генераторами. Схема обязательства содержит алгоритм $DCom:F^3\to G$, где $DCom(v,r,s)=vF+rG+sH$, являющийся гомоморфным в том смысле, что

$$DCom(v_1,r_1,s_1)+DCom(v_2,r_2,s_2)=DCom(v_1+v_2,r_1+r_2,s_1+s_2)$$

для всех таких входных значений $v_1,v_2\in F$, и который маскирует $r_1,r_2,s_1,s_2\in F$. Кроме того, конструкция идеально скрывает входные значения и является вычислительно обязательной.

### 2.2 Система доказательства представления

Доказательство представления используется для того, чтобы продемонстрировать знание дискретного логарифма в схеме с нулевым разглашением. Допустим, $pp_{rep}=(G,F)$ являются публичными параметрами такой конструкции, где $G$ — группа порядка, равного простому числу, в которой задачи дискретного логарифмирования являются сложными, а $F$ — её поле скалярных величин.

Сама система доказательства представляет собой кортеж алгоритмов (RepProve, RepVerify) для следующего отношения:

$$\{pp_{rep},G,X\in G;x\in F:X=xG\}$$

Для этого может использоваться хорошо известная система доказательства Шнорра.

### 2.3 Модифицированная система доказательства Шаума-Педерсена

Доказательство Шаума-Педерсена используется, чтобы продемонстрировать равенство дискретного логарифма в доказательстве с нулевым разглашением. В данном случае нам требуется модификация стандартной системы доказательства, которая будет использовать дополнительные генераторы группы. Допустим, $pp_{c\,haum}=(G,F,F,G,H,U)$ являются публичными параметрами такой конструкции, где $G$ — группа порядка, равного простому числу, в которой задача дискретного логарифмирования является сложной, $F$ — её поле скалярных величин, а $F,G,H,U\in G$ являются равномерно выбранными случайными генераторами.

Сама система доказательства представляет собой набор алгоритмов (ChaumProve, ChaumVerify) для следующего отношения:

$$\{pp_{c\,haum},S,T\in G;(x,y,z)\in F:S=xF+yG+zH,u=xT+yG\}$$

В Приложении А нами предлагается конкретный вариант реализации такой системы доказательства, а также доказательства безопасности.

### 2.4 Параллельная система доказательства по одному из множества

Нам необходима параллельная система доказательства по одному из множества, которая бы демонстрировала знание раскрытий обязательств по нулевой сумме в том же индексе между двумя наборами групповых элементов в доказательстве с нулевым разглашением. В контексте протокола Spark она бы использовалась для маскировки порядкового номера монеты, которая тратится, и обязательств по её ценности в целях обеспечения баланса, определения принадлежности и во избежание двойной траты. Мы показываем, как построить такую систему доказательства в качестве простой модификации конструкции, предложенной Гротом и Кёльвейссом (Groth and Kohlweiss) в работе [10] и обобщённой Бутлем и др. (Bootle et al.) в работе [3].

Допустим, $pp_{par}=\left(G,F,n,m,pp_{com}\right)$ являются публичными параметрами такой конструкции, где $G$ — группа порядка, равного простому числу, в которой задача дискретного логарифмирования является сложной, $F$ — её поле скалярных величин, $n>1$ и $m>1$ — целочисленные параметры разложения по размеру, а $pp_{com}$ — публичные параметры конструкции обязательства Педерсена (и матричного обязательства).

Сама система доказательства представляет собой набор алгоритмов (ParProve, ParVerify) для следующего отношения, где $N=n^m$:

$$\left\{pp_{par},\left\{S_i,V_i\right\}_{i=0}^{N-1}\subset G^2;l\in N,(s,v)\in F:0\le l<N,S_l=Com(0,v)\right\}$$

В Приложении В нами предлагается конкретный вариант реализации такой системы доказательства.

### 2.5 Схема аутентифицированного шифрования

Нам необходима схема аутентифицированного симметричного шифрования со связанными данными (AEAD). В контексте протокола Spark эта конструкция используется для шифрования значения и произвольных сопроводительных данных, которые будут использоваться отправителем и получателем транзакции.

Допустим, $pp_{sym}$ являются публичными параметрами такой конструкции. Сама конструкция представляет собой кортеж алгоритмов (AEADKeyGen, AEADEncrypt, AEADDecrypt). В данном случае AEADKeyGen является функцией выведения ключа, которая берёт в качестве входа произвольную строку и выдаёт ключ в соответствующем пространстве ключей. Алгоритм AEADEncrypt в качестве вводных данных берёт ключ, связанные с ним данные, произвольную строку сообщения и выдаёт шифротекст в соответствующем пространстве. Алгоритм AEADDecrypt в качестве вводных данных берёт ключ, связанные с ним данные и строку шифротекста и в случае успешной аутентификации выдаёт сообщение в соответствующем пространстве. С точки зрения данного протокола, мы можем допустить использование фиксированного нонса, так как ключи используются уникальным образом со связанными с ними данными.

Предположим, что такая конструкция будет неразличимой при атаке на основе подобранного простого текстом (IND-CPA), неразличимой при адаптивной атаке на основе подобранного зашифрованного текста (IND-CCA2), а также в этом контексте сохраняет приватность ключей при атаке на основе подобранного зашифрованного текста (IK-CCA).

### 2.6 Система доказательства диапазона

Нам необходима система доказательства диапазона с нулевым разглашением. Система доказательства диапазона демонстрирует, что обязательство даётся именно по сумме в указанном диапазоне. В контексте протокола Spark она позволяет избежать выхода за пределы, который бы в противном случае привёл к искажению определения баланса за счёт привязки к недействительным отрицательным значениям. Допустим, $pp_{rp}=\left(G,F,v_{max},pp_{com}\right)$ являются соответствующими публичными параметрами такой конструкции, где $pp_{com}$ — публичные параметры конструкции обязательства Педерсена.

Сама система доказательства представляет собой кортеж алгоритмов (RangeProve, RangeVerify) для следующего отношения:

$$\left\{ pp_{rp}, C \in G; (v, r) \in F : 0 \le v \le v_{max}, C = Com(v, r) \right\}$$

На практике для обеспечения соответствия данному требованию может использоваться такая эффективная система доказательства, как Bulletproofs [5] или Bulletproofs+ [6].

## 3 Концепции и алгоритмы

В данном разделе нами определяются основные концепции и алгоритмы, используемые в рамках протокола транзакций Spark.

**Адреса**. Пользователи генерируют адреса, позволяющие проводить транзакции. Адрес состоит из набора следующих элементов:

$$\left( addr_{pk}, addr_{¿}, addr_{full}, addr_{sk} \right)$$

В каждом адресе $addr_{pk}$ является публичным адресом, используемым для получения средств, $addr_{¿}$ — входящим ключом просмотра, используемым для идентификации полученных средств, $addr_{full}$ — полным ключом просмотра, используемым для идентификации исходящих средств и проведения вычислительно сложных операций, связанных с доказательством, а $addr_{sk}$ является ключом траты, необходимым для создания транзакций.

**Монеты**. В монете шифруется абстрактная ценность, передаваемая при проведении приватной транзакции. Каждая монета связана со следующими элементами:

- (секретным) порядковым номером, который уникальным образом определяет монету;
- обязательством по порядковому номеру;
- целочисленным значением монеты;
- зашифрованным значением, которое будет расшифровано получателем;
- обязательством по значению;
- доказательством диапазона для обязательства по значению или доказательством, в котором значение, выраженное простым текстом, будет представлено обязательством по значению;
- комментарии, содержащие произвольные данные получателя;
- зашифрованная сопроводительная информация, которая будет расшифрована получателем;
- ключ восстановления, используемый получателем для идентификации монеты и расшифровки приватных данных.

Кроме того, монеты связаны с адресами получателей неразличимым образом; это может оказаться полезным при доказательстве платежа по внешнему каналу, когда установление такой связи необходимо.

**Приватные транзакции**. В Spark существует два типа приватных транзакций:

- *Транзакции создания*. При совершении *транзакций создания* (*Mint Transaction*) генерируются новые монеты с публичным значением, которые направляются на публичный адрес получателя, и это происходит конфиденциальным образом: либо в процессе майнинга на базе консенсуса, либо путём использования прозрачных выходов с базового уровня, не охватываемого Spark. В этот тип транзакций включается доказательство представления, позволяющее продемонстрировать, что созданная монета имеет ожидаемое значение. При совершении *транзакции создания* генерируются данные $tx_{mint}$, которые будут включены в реестр.
- *Транзакции траты*. При совершении *транзакции траты* (*Spend Transaction*) используются уж существующие монеты и генерируются новые, которые конфиденциальным образом отправляются на один или несколько публичных

адресов получателя. В этот тип транзакций включается доказательство представления, позволяющее продемонстрировать, что скрытые значения входов и выходов равны. При совершении *транзакции траты* генерируются данные $tx_{spend}$, которые будут включены в реестр.

**Теги**. Теги используются для того, чтобы монеты не могли быть потрачены сразу во множестве транзакций. При создании *транзакции траты* отправитель производит тег для каждой расходуемой монеты и включает его в реестр. При верификации правильности транзакций, достаточно убедиться в том, что теги не появлялись в предыдущих транзакциях. Теги уникальным образом связаны с восстанавливаемыми действительными монетами посредством секретных данных, содержащихся в обязательстве по порядковому номеру, но не могут быть связаны с определёнными монетами при отсутствии соответствующего полного ключа просмотра.

**Алгоритмы**. Spark является системой децентрализованных анонимных платежей (DAP), определяемой следующими алгоритмами полиномиальной временной сложности:

- **Setup**. Этот алгоритм выдаёт все публичные параметры, используемые протоколом и лежащими в его основе компонентами. Процесс подготовки не требует генерирования каких-либо доверенных параметров.
- **CreateAddress**. Этот алгоритм выдаёт публичный адрес, входящий ключ просмотра, полный ключ просмотра и ключ траты.
- **CreateCoin**. Этот алгоритм в качестве вводных данных берёт публичный адрес, значение монеты и сопроводительные данные, а выдаёт монету, которая будет отправлена по адресу назначения.
- **Mint**. Этот алгоритм в качестве вводных данных берёт публичный адрес, значение монеты и (опционально) зависящие от варианта реализации данные, связанные с выходами базового уровня, и выдаёт транзакцию создания $tx_{mint}$.
- **Identify**. Этот алгоритм в качестве вводных данных берёт входящий ключ просмотра и выдаёт значение монеты и сопроводительную информацию.
- **Recover**. Этот алгоритм в качестве вводных данных берёт монету и полный ключ просмотра, и выдаёт значение монеты, сопроводительную информацию, порядковый номер и тег.
- **Spend**. Этот алгоритм в качестве вводных данных берёт полный ключ просмотра, ключ траты, набор входящих монет (включая монеты, которые использовались в большом наборе, обеспечивающем анонимность), индексы монет, которые будут потрачены, соответствующие порядковые номера, значение размера комиссии, набор исходящих монет, которые будут созданы, и выдаёт транзакцию траты $tx_{spend}$.
- **Verify**. Этот алгоритм берёт либо транзакцию создания, либо транзакцию траты и выдаёт бит, определяющий её действительность.

Ниже в приложениях нами приводится подробное описание, а также демонстрируется безопасность полученного протокола.

## 4 Алгоритмически конструкции

В этом разделе нами приводится подробное описание алгоритмов, построенных по схеме DAP.

### 4.1 Setup

В нашем случае публичные параметры *pp* состоят из соответствующих публичных параметров схемы обязательства Педерсена (и матричного обязательства), системы доказательства представления, модифицированной системы доказательства Шаума-Педерсена, параллельной системы доказательства по одному из множества, схемы симметричного шифрования и системы доказательства диапазона.

**Вводные данные**: параметр безопасности $\lambda$, параметры декомпозиции по размеру $n>1$ и $m>1$, параметр максимального значения $v_{max}$.

**Результат**: публичные параметры $pp$.

1 Выбрать группу $G$ порядка, равного простому числу, в которой задачи дискретного логарифмирования, принятия решения Диффи-Хеллмана и вычислительная задача Диффи-Хеллмана являются сложными. Допустим, $F$ является полем скалярных величин $G$.

2 Равномерно случайным образом выбрать $F, G, H, U \in G$. Фактически, эти генераторы могут быть выбраны путём применения подходящей криптографической хеш-функции к публичным вводным данным.

3 Равномерно случайным образом выбрать криптографические функции

$$H_{ser}, H_{val}, H_{ser'}, H_{val'}, H_{bind} : \{0,1\}^{\lambda} \to F$$

Фактически, эти хеш-функции могут быть выбраны путём доменного разделения одной подходящей криптографической хеш-функции.

4 Вычислить публичные параметры $pp_{com} = (G, F, G, H)$ схемы обязательства Педерсена.

5 Вычислить публичные параметры $pp_{d\,com} = (G, F, F, G, H)$ схемы обязательства Педерсена с двойной маской.

6 Вычислить публичные параметры $pp_{rep} = (G, F)$ системы доказательства представления.

7 Вычислить публичные параметры $pp_{c\,haum} = (G, F, F, G, H, U)$ модифицированной системы доказательства Шаума-Педерсена.

8 Вычислить публичные параметры $pp_{par} = (G, F, n, m, pp_{com})$ параллельной системы доказательства по одному из множества.

9 Вычислить публичные параметры $pp_{sym}$ схемы аутентифицированного симметричного шифрования.

10 Вычислить публичные параметры $pp_{rp} = (G, F, v_{max}, pp_{com})$ системы доказательства диапазона.

11 Выдать все сгенерированные публичные параметры и хеш-функции как $pp$.

### 4.2 CreateAddress

Нами описывается конструкция всех адресов и лежащих в их основе типов ключей, используемых в рамках данного протокола.

**Вводные данные**: параметр безопасности $\lambda$, публичные параметры $pp$.

**Результат**: набор ключей адресов $(addr_{pk}, addr_{\lambda}, addr_{full}, addr_{sk})$.

1 Равномерно случайным образом выбрать $s_1, s_2, r \in F$ и задать $D = DCom(0, r, 0)$.

2 Вычислить $Q_1 = DCom(s_1, 0, 0)$ и $Q_2 = DCom(s_2, r, 0)$.

3 Задать $addr_{pk} = (Q_1, Q_2)$.

4 Задать $addr_{\lambda} = s_1$.

5 Задать $addr_{full} = (s_1, s_2, D)$.

6 Задать $addr_{sk} = (s_1, s_2, r)$.

7 Выдать набор $(addr_{pk}, addr_{\lambda}, addr_{full}, addr_{sk})$.

### 4.3 CreateCoin

Этот алгоритм создаёт новую монету, которая будет отправлена по заданному публичному адресу. Следует отметить, что, несмотря на то, что данный алгоритм генерирует обязательство по порядковому номеру, он не может вычислить, лежащий в его основе порядковый номер.

**Вводные данные**: параметр безопасности $\lambda$, публичные параметры $pp$, публичный адрес получателя $addr_{pk}$, значение $v \in [0, v_{max})$, сопроводительная информация $m$, бит типа $b$.

**Результат**: публичный ключ монеты $S$, ключ восстановления $K$, обязательство по значению $C$, доказательство диапазона обязательства по значению $\Pi_{rp}$ (если $b=0$), зашифрованное значение $\overline{v}$ (если $b=0$) или значение $v$ (если $b=1$), зашифрованную сопроводительную информацию $\overline{m}$.

1 Разбить адрес получателя $addr_{pk} = (Q_1, Q_2)$.
2 Выбрать $k \in F$.
3 Вычислить ключ восстановления $K = DCom(k, 0, 0)$.
4 Вычислить обязательство по порядковому номеру

$$S = DCom(H_{ser}(k Q_1, Q_1, Q_2), 0, 0) + Q_2$$

5 Сгенерировать обязательство по значению $C = Com(v, H_{val}(k Q_1))$.
6 Если $b=0$, сгенерировать доказательство диапазона

$$\Pi_{rp} = RangeProve(pp_{rp}, C; (v, H_{val}(k Q_1)))$$

7 Сгенерировать ключ симметричного шифрования $k_{aead} = AEADKeyGen(k Q_1)$; зашифровать значение $\overline{v} = AEADEncrypt(k_{aead}, val, v)$ (если $b=0$) и сопроводительную информацию $\overline{m} = AEADEncrypt(k_{aead}, memo, m)$.
8 Если $b=0$ выдать набор элементов $(S, K, C, \Pi_{rp}, \overline{v}, \overline{m})$.

$b=0$ означает, что монета со скрытым значением была создана в транзакции траты, в то время как $b=0$ означает, что монета со значением, выраженным простым текстом, была сгенерирована в транзакции создания. Следует отметить возможность безопасного накопления доказательств диапазона в транзакции; это никак не влияет на безопасность протокола.

### 4.4 Mint

Этот алгоритм создаёт новые монеты либо в рамках определяемого консенсусом процесса майнинга, либо путём использования выходов, не принадлежащих Spark, из базового уровня с публичным значением. Несмотря на то, что для создания такой транзакции могут понадобиться и включаются зависящие от реализации вспомогательные данные, мы не рассматриваем их отдельно в данной работе. Важно то, что значение, используемое в рамках данного алгоритма, представляет собой сумму всех публичных входящих значений, что и указано в описании реализации.

**Вводные данные**: параметр безопасности $\lambda$, публичные параметры $pp$, публичный адрес получателя $addr_{pk}$, значение монеты $v \in [0, v_{max})$, сопроводительная информация $m$.

**Результат**: транзакция создания $tx_{mint}$.

1 Сгенерировать новую монету $CreateCoin(addr_{pk}, v, m, 1) \to Coin(S, K, C, v, \overline{m})$.
2 Сгенерировать доказательство представления значения для обязательства по значению:

$$\Pi_{bal} = RepProve(pp_{rep}, H, C - Com(v, 0); H_{val}(k Q_1))$$

3 Выдать $tx_{mint} = (Coin, \Pi_{bal})$.

### 4.5 Identify

Этот алгоритм позволяет получателю (или обозначенной стороне) вычислить значение и сопроводительные данные монеты, отправленной на его публичный адрес. Для этого необходим входящий ключ просмотра, соответствующий такому публичному адресу. Если

монета предназначена для получения на другой публичный адрес, алгоритм сообщит об ошибке.

**Вводные данные**: параметр безопасности $\lambda$, публичные параметры $pp$, входящий ключ просмотра $addr_{¿}$, монета $Coin$.

**Результат**: значение $v$, сопроводительная информация $m$.

1 Разбить входящий ключ просмотра $addr_{¿} = s_1$ и публичный адрес $addr_{pk} = (Q_1, Q_2)$.

2 Разбить обязательство по порядковому номеру $S$, обязательство по значению $C$, ключ восстановления $K$, зашифрованное значение $\overline{v}$ (если монета принадлежит к типу, где $b=0$) или значение $v$ (если монета принадлежит к типу, где $b=1$) и зашифрованную сопроводительную информацию $\overline{m}$ монеты $Coin$.

3 Если $DCom(H_{ser}(s_1 K, Q_1, Q_2), 0, 0) + Q_2 \neq S$, выдать ошибку.

4 Сгенерировать ключ симметричного шифрования $k_{aead} = AEADKeyGen(s_1 K)$; расшифровать значение $v = AEADDecrypt(k_{aead}, val, \overline{v})$ (если $b=0$) и сопроводительную информацию $m = AEADDecrypt(k_{aead}, memo, \overline{m})$.

5 Если $Com(v, H_{val}(s_1 K)) \neq C$, выдать ошибку.

6 Выдать набор элементов $(v, m)$.

### 4.6 Recover

Этот алгоритм позволяет получателю (или обозначенной стороне) вычислить порядковый номер, тег, значение и сопроводительную информацию монеты, отправленной на его публичный адрес. Для этого необходим полный ключ просмотра, соответствующий такому публичному адресу. Если монета предназначена для получения на другой публичный адрес, алгоритм сообщит об ошибке.

**Вводные данные**: параметр безопасности $\lambda$, публичные параметры $pp$, полный ключ просмотра $addr_{full}$, монета $Coin$.

**Результат**: порядковый номер монеты $s$, тег $T$, значение $v$, сопроводительная информация $m$.

1 Разбить полный ключ просмотра $addr_{full} = (s_1, s_2, D)$ и публичный адрес $addr_{pk} = (Q_1, Q_2)$.

2 Разбить обязательство по порядковому номеру $S$, обязательство по значению $C$, ключ восстановления $K$, зашифрованное значение $\overline{v}$ (если монета принадлежит к типу, где $b=0$) или значение $v$ (если монета принадлежит к типу, где $b=1$) и зашифрованную сопроводительную информацию $\overline{m}$ монеты $Coin$.

3 Если $DCom(H_{ser}(s_1 K, Q_1, Q_2), 0, 0) + Q_2 \neq S$, выдать ошибку.

4 Сгенерировать ключ симметричного шифрования $k_{aead} = AEADKeyGen(s_1 K)$; расшифровать значение $v = AEADDecrypt(k_{aead}, val, \overline{v})$ (если $b=1$) и сопроводительную информацию $m = AEADDecrypt(k_{aead}, memo, \overline{m})$.

5 Если $Com(v, H_{val}(s_1 K)) \neq C$, выдать ошибку.

6 Вычислить порядковый номер

$$s = H_{ser}(s_1 K, Q_1, Q_2) + s_2$$

и тег

$$T = (1/s)(U - D)$$

7 Если тег $T$ был построен при выполнении какой-либо другой действительной операции восстановления, выдать ошибку.

8 Выдать набор элементов $(s, T, v, m)$.

### 4.7 Spend

Этот алгоритм позволяет получателю сгенерировать транзакцию, использующую монеты, направленные на публичный адрес и создать новые монеты, предназначенные для произвольных публичных адресов. Схема процесса является модульной; в частности, для генерирования параллельного доказательства по одному из множества, что может оказаться довольно затратным с точки зрения необходимых вычислений, требуется только полный ключ просмотра. Ключи траты необходимы исключительно на последнем этапе выполнения доказательства Шаума-Педерсена, что менее сложно.

Предполагается, что получатель использует алгоритм Return применительно ко всем монетам, которые он желает использовать в такой транзакции.

**Вводные данные**:

- параметр безопасности $\lambda$ и публичные параметры $pp$;
- полный ключ просмотра $addr_{full}$;
- ключ траты $addr_{sk}$;
- набор из $N$ входящих монет $InCoins$, являющегося частью закрывающего набора;
- для каждой $u \in [0, w)$ монеты, которая будет потрачена, индекс в наборе $InCoins$, порядковый номер, тег, значение и ключ восстановления: $(l_u, s_u, T_u, v_u, K_u)$;
- целочисленное значение комиссии $f \in [0, v_{max})$;
- набор из $t$ публичных адресов исходящих монет, их значений и сопроводительной информации:

$$\{addr_{pk,j}, v_j, m_j\}_{j=0}^{t-1}$$

**Результат**: транзакция траты $tx_{spend}$.

1 Разбить необходимый компонент полного ключа просмотра $addr_{full} = D$.

2 Разбить ключ траты $addr_{sk} = (s_1, s_2, r)$.

3 Разбить обязательства по порядковому номеру и обязательства по значению закрывающего набора $InCoins = \{(S_i, C_i)\}_{i=0}^{N-1}$.

4 Для каждой $u \in [0, w)$:

a вычислить офсет обязательства по порядковому номеру:

$$S_u' = DCom(s_u, 0, -H_{ser'}(s_u, D)) + D$$

b вычислить офсет обязательства по значению:

$$C_u' = Com(v_u, H_{val'}(s_u, D))$$

c сгенерировать параллельное доказательство по одному из множества:

$$(\Pi_{par})_u = ParProve\left(pp_{par}, \{(S_i - S_u', C_i - C_u')\}_{i=0}^{N-1}; (l_u, H_{ser'}(s_u, D), H_{val}(s_1 K_u) - H_{val'}(s_u, D))\right)$$

5 Сгенерировать набор исходящих монет $OutCoins = \{CreateCoin(addr_{pk,j}, v_j, m_j, 0)\}_{j=0}^{t-1}$.

6 Разбить обязательства по значениям исходящих монет $OutCoins = \{\overline{C}_j\}_{j=0}^{t-1}$, где каждое $\overline{C}_j$ содержит прообраз ключа восстановления $k_j$ и компонента адреса получателя $(Q_1)_j$.

7 Сгенерировать доказательство представления для подтверждения баланса:

$$\Pi_{bal} = Rep\,Prove¿$$

8 Допустим, $\mu = H_{bind}(InCoins, OutCoins, f, \{S_u', C_u', T_u, (\Pi_{par})_u\}_{u=0}^{w-1}, \Pi_{bal})$.

9 Для каждой $u \in [0, w)$ сгенерировать модифицированное доказательство Шаума-Педерсена, в котором мы дополнительно связываем $\mu$ с начальным транскриптом:

$$(\Pi_{chaum})_u = ChaumProve(pp_{chaum}, S_u', T_u; (s_u, r, -H_{ser}(s_u, D)))$$

10 Выдать набор элементов:

$$tx_{spend} = (InCoins, OutCoins, f, \{S_u', C_u', T_u, (\Pi_{par})_u, (\Pi_{chaum})_u\}_{u=0}^{w-1}, \Pi_{bal})$$

*Примечание 1*. Нам бы хотелось отметить наличие возможности такого изменения доказательства баланса, которое бы учитывало другие значения входов и выходов, не представленные в обязательствах по значению монеты, и делается это подобно тому, как в случае с обработкой комиссий. Это позволяет переносить значение в новые монеты, не прибегая к транзакции создания, или же переносить значение на прозрачный базовый уровень. Такой перенос функций, вероятно, представляет собой практический риск, не охваченный моделью безопасности протокола, и требует тщательного анализа.

### 4.8 Verify

Этот алгоритм оценивает действительность транзакции.

**Вводные данные**: либо транзакция создания $tx_{mint}$, либо транзакция траты $tx_{spend}$.

**Результат**: бит, представляющий действительность транзакции. Если входящая транзакция является транзакцией создания:

1 Разбить транзакцию $tx_{mint} = (Coin, \Pi_{bal})$.

2 Разбить значение монеты и обязательство по значению $Coin = (v, C)$.

3 Проверить соответствие $v \in [0, v_{max})$ и выдать 0, если условие не соблюдено.

4 Проверить соответствие $RepVerify(pp_{rep}, \Pi_{bal}, H, C - Com(v, 0))$ и выдать 0, если условие не соблюдено.

5 Выдать значение 1.

Если входящая транзакция является транзакцией траты:

1 Разбить транзакцию:

$$tx_{spend} = (InCoins, OutCoins, f, \{S_u', C_u', T_u, (\Pi_{par})_u, (\Pi_{chaum})_u\}_{u=0}^{w-1}, \Pi_{bal})$$

2 Разбить обязательства по порядковому номеру закрывающего набора и обязательства по значению $InCoins = \{(S_i, C_i)\}_{i=0}^{N-1}$.

3 Разбить обязательства по значению исходящей монеты и доказательства диапазона $OutCoins ¿ \{\overline{C}_j, (\Pi_{rp})_j\}_{j=0}^{t-1}$.

4 Для каждой $u \in [0, w)$:

a Убедиться в том, что $T_u$ не встречается где-либо ещё в данной транзакции или любой ранее верифицированной транзакции, и выдать 0, если встречается.

b Проверить соответствие $ParVerify(pp_{par}, (\Pi_{par})_u, \{S_i - S_u', C_i - C_u'\}_{i=0}^{N-1})$ и выдать 0, если условие не соблюдено.

c Проверить соответствие $ChaumVerify(pp_{chaum}, (\Pi_{chaum})_u, S_u', T_u)$ и выдать 0, если условие не соблюдено.

5 Для каждого $j \in [0, t)$:

а Проверить соответствие $RangeVerify\left(pp_{rp},\left(\Pi_{rp}\right)_j,C\right)$ и выдать 0, если условие не соблюдено.

6 Проверить соответствие $f\in\left[0,v_{max}\right)$ и выдать 0, если условие не соблюдено.

7 Проверить соответствие

$$RepVerify\left(pp_{rep},\Pi_{bal},H,\sum_{u=0}^{w-1}C_u^{'}-\sum_{j=0}^{t-1}\overline{C}_j-Com\left(f,0\right)\right)$$

и выдать 0, если условие не соблюдено.

8 Выдать значение 1.

## 5 Операции с мультиподписью

Адреса и транзакции Spark поддерживают эффективные и безопасные операции с мультиподписью, когда для подтверждения транзакций требуется наличие группы подписантов. Нами описывается метод реализации алгоритмов CreateAddress и Spend группами подписантов для создания адресов и неотличимых от других транзакций траты, поддерживающих схему мультиподписи.

В рамках данного раздела предположим, что у нас есть группа, состоящая из $v$ подписантов, желающих совместно создать адрес или транзакцию. Кроме того, равномерно случайным образом мы выбираем криптографическую хеш-функцию $H_{agg}:\{0,1\}^{\zeta}\rightarrow F$.

### 5.1 CreateAddress

1 Каждый участник $\alpha\in\left[0,v\right)$ равномерно случайным образом выбирает $s_{1,\alpha}$, $s_{2,\alpha}$, $r_\alpha\in F$ и задаёт $D_\alpha=DCom\left(0,r_\alpha,0\right)$. Он отправляет значения $s_{1,\alpha}$, $s_{2,\alpha}$, $D_\alpha$ всем участникам.

2 Все участники вычисляют компоненты совокупного входящего ключ просмотра и полного ключа просмотра:

$$s_1=\sum_{\alpha=0}^{v-1}H_{agg}\left(\left\{s_{1,\beta}\right\}_{\beta=0}^{v-1},\alpha\right)s_{1,\alpha}$$

$$s_2=\sum_{\alpha=0}^{v-1}H_{agg}\left(\left\{s_{2,\beta}\right\}_{\beta=0}^{v-1},\alpha\right)s_{2,\alpha}$$

$$D=\sum_{\alpha=0}^{v-1}H_{agg}\left(\left\{D_\beta\right\}_{\beta=0}^{v-1},\alpha\right)D_\alpha$$

3 Все участники вычисляют компоненты совокупного публичного адреса:

$$Q_1=DCom\left(s_1,0,0\right)$$

$$Q_2=DCom\left(s_2,0,0\right)+D$$

Следует отметить, что каждый участник $\alpha$ сохраняет приватность своей доли ключа траты $r_\alpha$.

### 5.2 Spend

Поскольку у всех участников имеется совокупный полный ключ просмотра, соответствующий совокупному публичному адресу, любой участник может использовать его, чтобы построить все компоненты транзакции, кроме модифицированных доказательств Шаума-Педерсена. Мы описываем, как подписанты совместно создают такое доказательство,

подтверждающее трату монеты, используя следующие вводные данные доказательства (мы используем ту же систему обозначений, что и раньше):

$$\left\{pp_{chaum}, S_u^{'}, T_u; \left(s_u, r, -H_{ser'}\left(s_u, D\right)\right)\right\}$$

1 Каждый участник $\alpha \in [0, v)$ равномерно случайным образом выбирает $\bar{r}_\alpha$, $\bar{s}_\alpha$, $\bar{t}_\alpha \in F$. Он генерирует обязательство по набору элементов $\left(\bar{r}_\alpha, \bar{s}_\alpha G, \bar{t}_\alpha\right)$ и отправляет его всем участникам.

2 Каждый участник раскрывает свою часть обязательства всем остальным участникам, верифицирует такие же части обязательства остальных участников, и прерывает процесс, если какая-либо из них не будет действительной.

3 Все участники вычисляют начальные элементы доказательства:

$$A_1 = \left(\sum_{\beta=0}^{v-1} \bar{r}_\beta\right) F + \sum_{\beta=0}^{v-1} \left(\bar{s}_\alpha G\right)$$

$$A_2 = \left(\sum_{\beta=0}^{v-1} \bar{r}_\beta\right) T_u + \sum_{\beta=0}^{v-1} \left(\bar{s}_\alpha G\right)$$

$$A_3 = \left(\sum_{\beta=0}^{v-1} \bar{t}_\beta\right) H$$

Запрос $c$ вычисляется на основе начального транскрипта доказательства.

4 Каждый участник $\alpha \in [0, v)$ вычисляет следующее:

$$t_1 = \sum_{\beta=0}^{v-1} \bar{r}_\beta + c s_u$$

$$t_{2,\alpha} = \bar{s}_\alpha + c\, H_{agg}\left(\left\{D_\beta\right\}_{\beta=0}^{v-1}, \alpha\right) r_\alpha$$

$$t_3 = \sum_{\beta=0}^{v-1} \bar{t}_\beta - c\, H_{ser'}\left(s_u, D\right)$$

$t_{2,\alpha}$ рассылается всем участникам.

5 Все участники вычисляют последний компонент доказательства:

$$t_2 = \sum_{\beta=0}^{v-1} t_{2,\beta}$$

## 6 Применение структур ключей

Структура ключей в рамках протокола Spark обеспечивает гибкость и практичность функций, связанных со сканированием и построением транзакций.

Входящий ключ просмотра используется при проведении операций в рамках алгоритма Identify, чтобы определить, когда на связанный с ним публичный адрес будет отправлена монета, а также чтобы определить значение монеты и соответствующую сопроводительную информацию. Это даёт нам два известных варианта использования. В одном случае ответственность за процесс сканирования блокчейна может быть делегирована устройству или сервису без передачи полномочий, связанных с тратой обозначенных монет. В другом случае программное обеспечение кошелька, которому известен ключ траты, может сохранить этот ключ в зашифрованном или другом безопасном виде при проведении операции сканирования, что снижает риски, связанные с раскрытием такого ключа.

Полный ключ просмотра используется при проведении операций в соответствии с алгоритмом Recover, чтобы дополнительно вычислить порядковый номер и тег монет, передаваемых на соответствующий публичный адрес. Эти теги могут использоваться для идентификации транзакции, в которой тратится монета. Передача такого ключа позволяет третьей стороне идентифицировать входящие и обнаруживать исходящие транзакции, что в дополнение к этому даёт возможность вычисления баланса без делегирования полномочий, связанных с тратой. Такие пользователи, как общественные благотворительные организации, могут пожелать, чтобы за их средствами вёлся общественный надзор, и это будет возможно благодаря этой функции. Другие пользователи могут захотеть воспользоваться этой функцией и передать ключ аудитору или бухгалтеру для ведения бухгалтерского учёта. В тех случаях, когда публичный адрес используется для проведения операций с применением схемы пороговой мультиподписи, кто-нибудь из подписантов может захотеть узнать, создали ли остальные подписанты или когда они создали транзакцию, в которой были потрачены средства с его адреса.

Кроме того, полный ключ просмотра используется в рамках реализации алгоритма Spend, чтобы сгенерировать доказательства по одному из множества. Поскольку использование параллельного доказательства по одному из множества в соответствии с протоколом Spark может оказаться затратным с точки зрения вычислений, оно может не подойти в случае такими устройствами, как аппаратный кошелёк, вычислительные ресурсы которых ограничены. Использование такого ключа с более мощным устройством позволяет без труда сгенерировать такое доказательство (равно как и другие компоненты транзакции, такие как доказательство диапазона). При этом сохраняется гарантия того, что только то устройство, которому известен ключ траты, сможет провести транзакцию, сгенерировав простые модифицированные доказательства Шаума-Педерсена.

## 7 Эффективность

Было бы полезно изучить эффективность транзакций Spark с точки зрения их размера, сложности создания и верификации. В дополнение к ранее обозначенным нами параметрам добавим $v_{max}=2^{64}$, чтобы значения монет и комиссий можно было представить 8-байтовыми целыми числами без знака. Далее предположим, что сопроводительная информация монет имеет фиксированную длину $M$ байт и 16-байтовый тег аутентификации; в этом случае используется схема аутентифицированного сквозного шифрования ChaCha20-Poly1305, подобная той, что представлена в работе [15], например. Данные по размеру транзакций для определённого варианта реализации компонента приводятся в Таблице 1, в которой размер рассматривается относительно групповых элементов, элементов поля и других данных. Следует отметить, что мы не включаем в эти данные ссылки на наборы обеспечения неоднозначности входов, так как это целиком зависит от выбора варианта реализации и критериев представления.

Чтобы оценить сложность верификации транзакций Spend, используя эти компоненты, мы принимаем во внимание, что верификация таких конструкций, как система доказательства по одному из множества, которая рассматривается в данной работе, система доказательства диапазона Bulletproof+, система доказательства представления Шнорра и модифицированная система доказательства Шаума-Педерсена, о которой также говорится в этой работе, сокращает одиночную оценку линейных комбинаций в $G$. Поэтому доказательства можно оценивать группами при условии, что верификатор сначала взвесит каждое доказательство, используя случайное значение в $F$, чтобы такие отдельные элементы группы появились лишь единожды в полученной взвешенной линейной комбинации. Примечательно то, что техники, подобные той, что описана в работе [18], можно использовать для снижения уровня сложности такой оценки вплоть до логарифмического множителя. Предположим, нам нужно верифицировать группу, в которую входит $B$ транзакций, в каждой из которых тратится $w$ монет и генерируется $t$ монет. В Таблице 2

приводится уровень сложности групповой верификации относительно общего количества отдельных элементов $G$, которые должны быть включены при оценке линейной комбинации.

*Таблица 1. Размер транзакции Spend по компонентам*

| **Компонент** | **Вариант реализации** | **Размер ($G$)** | **Размер ($F$)** | **Размер (в байтах)** |
|---|---|---|---|---|
| $f$<br>$\Pi_{rp}$<br>$\Pi_{bal}$ | Bulletproofs+<br>Доказательство Шнорра | $2\lceil lg(64t)\rceil+3$ | 3<br>2 | 8 |
| Входящие данные ($w$ монет) | | | | |
| $(S',C')$<br>$\Pi_{par}$<br>$\Pi_{chaum}$ | В соответствии с данной работой<br>В соответствии с данной работой | $2w$<br>$(2m+4)w$<br>$3w$ | $[m(n-1)+4]w$<br>$3w$ | |
| Исходящие данные ($t$ монет) | | | | |
| $(S,K,C)$<br>$(\overline{v},\overline{m})$ | ChaCha20-Poly1305 | $3t$ | | $(8+M+16)$t |

*Таблица 2. Сложность верификации группы, состоящей из $B$ транзакций, при проведении которых тратится $w$ монет и создаётся $t$ монет*

| **Компонент** | **Сложность** |
|---|---|
| Параллельное доказательство по одному из множества<br>Bulletproofs+<br>Модифицированное доказательство Шаума-Педерсена<br>Доказательство Шнорра | $B[w(2m+6)+2m^m]+m^n+1$<br>$B(t+2lg(64t)+3)+128T+2$<br>$B(5w)+4$<br>$B(w+t+1)+2$ |

Кроме того, мы отмечаем, что параллельная система доказательства по одному из множества, представленная в данной работе, может быть дополнительно оптимизирована с точки зрения верификации. Поскольку соответствующие элементы входящих наборов $\{S_i\}$ и $\{V_i\}$ взвешиваются идентичным образом в соответствии с уравнениями верификации, предусмотренными протоколом, возможно, мы могли бы повысить эффективность (в зависимости от варианта реализации) путём объединения элементов с достаточным взвешенным значением до применения зависящего от типа доказательства взвешивания (как указано выше) для групповой верификации. Начальное тестирование с использованием библиотеки кривых зависимости дисперсии от времени свидетельствует о значительном сокращении времени верификации при использовании такой техники.

**Благодарность**

Авторы выражают свою благодарность koe за постоянное участие в обсуждениях в ходе написания данной работы. Также авторы благодарят Николаса Кратцшмара (Nikolas Krätzschmar) за то, что он обратил наше внимание на недочёт в ранней версии протокола, связанный с генерированием тегов.

**Список использованной литературы**

1 И. Бен Сэссон (*Ben Sasson, E.*), А. Чьеза (*Chiesa, A.*), К. Гарман (*Garman, C.*), М. Грин (*Green, M.*), И. Миерс (*Miers, I.*), И. Тромер (*Tromer, E.*), М. Вирца (*Virza, M.*), «Zerocash: децентрализованные анонимные платежи из блокчейна Bitcoin» (*Zerocash: Decentralized anonymous payments from Bitcoin*), материалы Симпозиума IEEE по безопасности и конфиденциальности данных 2014 г. (*2014 IEEE*

*Symposium on Security and Privacy*), стр. 459-474 (2014). https://doi.org/10.1109/SP.2014.36.

2 И. Бен Сэссон (*Ben Sasson, E.*), А. Чьеза (*Chiesa, A.*), И. Тромер (*Tromer, E.*), М. Вирца (*Virza, M.*), «Сжатое неинтерактивное доказательство с нулевым разглашением, применимое в рамках архитектуры фон Неймана» (*Succinct non-interactive zero knowledge for a von Neumann architecture*), Материалы 23-й конференции USENIX по безопасности данных (*Proceedings of the 23rd USENIX Conference on Security Symposium*), стр. 781-796, SEC'14, USENIX Association, USA (2014).

3 Дж. Бутль (*Bootle, J.*), А. Церулли (*Cerulli, A.*), П. Чайдос (*Chaidos, P.*), И. Гадафи (*Ghadafi, E.*), Дж. Грот (*Groth, J.*), К. Пти (*Petit, C.*), «Краткие проверяемые кольцевые подписи на базе DDH» (*Short accountable ring signatures based on DDH*), под редакцией Дж. Пернуля (*Pernul, G.*), П. И А Райана (*Y A Ryan, P.*), И. Вейпла (*Weippl, E.*), «Компьютерная безопасность — ESORICS 2015» (*Computer Security — ESORICS 2015*), стр. 243-265, Springer International Publishing, Cham (2015).

4 С. Боуи (*Bowe, S.*), А. Гейбизон (*Gabizon, A.*), И. Миерс (*Miers, I.*), «Масштабируемое многостороннее вычисление параметров zk-SNARK в рамках модели децентрализованного генератора случайности Random Beacon» (*Scalable multi-party computation for zk-SNARK parameters in the random beacon model*), Архив электронных документов по криптологии (*Cryptology ePrint Archive*), Документ 2017/1050 (2017), https://ia.cr/2017/1050

5 Б. Бюнц (*Bünz, B.)*, Дж. Бутль (*Bootle, J.*), Д. Боне (*Boneh, D.*), А. Поэлстра (*Poelstra, A.*), П. Вуйль (*Wuille, P.*), Г. Максвелл (*Maxwell, G.*), «Bulletproofs: краткие доказательства для проведения конфиденциальных транзакций и многого другого» (*Bulletproofs: Short proofs for confidential transactions and more*), материалы Симпозиума IEEE по безопасности и конфиденциальности данных 2018 г. (*2018 IEEE Symposium on Security and Privacy* (*SP*)). стр. 315-334 (2018), https://doi.org/10.1109/SP.2018.00020

6 Х. Чанг (*Chung, H.*), К. Хан (*Han, K.*), К. Джу (*Ju, C.*), М. Ким (*Kim, M.*), Дж. Х. Сео (*Seo, J. H.*), «Bulletproofs+: использование кратких доказательств в усовершенствованном распределённом реестре» (*Bulletproofs+: Shorter proofs for privacy-enhanced distributed ledger*), Архив электронных документов по криптологии (*Cryptology ePrint Archive*), Документ 2020/735 (2020), https://ia.cr/2020/735

7 И. Додис (*Dodis, Y.*), А. Ямпольский (*Yampolskiy, A.*), «Верифицируемая случайная функция с краткими доказательствами и ключами» (*A verifiable random function with short proofs and keys*), под редакцией С. Воденая (Vaudenay, S.), «Криптография публичных ключей — PKC 2005» (*Public Key Cryptography — PKC 2005*), стр. 416-431, Springer Berlin Heidelberg, Berlin, Heidelberg (2005).

8 Дж. Фушбауэр (*Fuchsbauer, G.*), М. Орру (*Orrù, M.*), И. Сюрин (*Seurin, Y.*), «Агрегированные Mimblewimble» (*Aggregate cash systems: A cryptographic investigation of Mimblewimble*), под редакцией И. Ишая (*Ishai, Y.*), В. Риджмана (*Rijmen, V.*), «Прогресс в криптологии — EUROCRYPT 2019» (*Advances in Cryptology — EUROCRYPT 2019*), стр. 657-689, Springer International Publishing, Cham (2019)

9 Б. Гуделл (*Goodell, B.*), С. Нёзер (*Noether, S.*), RandomRun, «Сжатые связываемые кольцевые подписи и возможность подделки с использованием вредоносных ключей» (*Concise linkable ring signatures and forgery against adversarial keys*), Архив электронных документов по криптологии (*Cryptology ePrint Archive*), Документ 2019/654 (2019), https://ia.cr/2019/654

10 Дж. Грот (*Groth, J.*), М. Кёльвейс (*Kohlweiss, M.*). «Доказательства по одному из множества: или как раскрыть секрет и потратить монету» (*One-out-of-many proofs: Or how to leak a secret and spend a coin*), под редакцией И. Освальда (*Oswald, E.*), М. Фишлин (*Fischlin, M.*), «Компьютерная безопасность — ESORICS 2015» (*Computer Security — ESORICS 2015*), стр. 253-280, Springer Berlin Heidelberg, Berlin, Heidelberg (2015)

11 Д. Хопвуд (*Hopwood, D.*), С. Боуи (*Bowe, S.*), Т. Хорнби (*Hornby, T.*), Н. Уилкокс (*Wilcox, N.*). «Спецификация протокола Zcash» (*Zcash protocol specification*) (2021), https://github.com/zcash/zips/blob/master/protocol/protocol.pdf

12 А. Дживанян (*Jivanyan, A.*), «Lelantus: новое решение для анонимных и конфиденциальных криптовалют» (*Lelantus: A new design for anonymous and confidential cryptocurrencies*), Архив электронных документов по криптологии (*Cryptology ePrint Archive*), Документ 2019/373 (2019), https://ia.cr/2019/373

13 Р. В. Ф. Лай (*Lai, R. W. F.*), В. Ронже (*Ronge, V.*), Т. Руффинг (*Ruffing, T.*), Д. Шрёдер (*Schröder, D.*), С. А. К. Тягаранян (*Thyagarajan, S. A. K.*), Дж. Вонг (*Wang, J.*), «Omniring: масштабирование приватных платежей без доверенных настроек» (*Omniring: Scaling private payments without trusted setup*), материалы Конференции ACM SIGSAC по компьютерной безопасности и защите каналов связи 2019 (*2019 ACM SIGSAC Conference on Computer and Communications Security*), стр. 31-48. CCS '19, Ассоциация вычислительной техники (*Association for Computing Machinery*), Нью-Йорк, США (2019). https://doi.org/10.1145/3319535.3345655, https://doi.org/10.1145/3319535.3345655

14 Г. Максвелл (*Maxwell, G.*), А. Поэлстра (*Poelstra, A.*), И. Сюрин (*Seurin, Y.*), П. Вуйль (*Wuille, P.*), «Простая схема мультиподписей Шнорра и её применение в Bitcoin» (*Simple Schnorr multi-signatures with applications to Bitcoin*), Разработки, код и криптография (*Designs, Codes and Cryptography*) 87(9), 2139-2164 (2019)

15 И. Нир (*Nir, Y.*), А. Лэнгли (*Langley, A.*), «Применение шифров ChaCha20 и Poly1305 в протоколах IETF» (*ChaCha20 and Poly1305 for IETF protocols*), RFC 7539, RFC Editor (май 2015), http://www.rfc-editor.org/rfc/rfc7539.txt, http://www.rfceditor.org/rfc/rfc7539.txt

16 С. Нёзер (*Noether, S.*), Б. Гуделл (*Goodell, B.*), «Triptych: логарифмически масштабируемые связываемые кольцевые подписи и их применение» (*Triptych: Logarithmic-sized linkable ring signatures with applications*), под редакцией Дж. Гарсия-Альфаро (*Garcia-Alfaro, J.*), Г. Наварро-Аррибаса, (*Navarro-Arribas, G.*), Дж. Геррера-Хоанкомарти (*Herrera-Joancomarti, J.*), Управление конфиденциальностью данных (*Data Privacy Management*), Криптовалюты и блокчейн-технология (*Cryptocurrencies and Blockchain Technology*), стр. 337-354, Springer International Publishing, Cham (2020)

17 С. Нёзер (*Noether, S.*), А. Маккензи (*Mackenzie, A.*) и др., «Кольцевые конфиденциальные транзакции» (*Ring confidential transactions*), Ledger 1, 1-18 (2016).

18 Н. Пиппенгер (*Pippenger, N.*), «По вопросу оценки степеней и одночленов» (*On the evaluation of powers and monomials*), Журнал по вычислительной технике SIAM (*SIAM Journal on Computing*), 9(2), 230-250 (1980)

19 В. Г. Пиррос Чайдос (*Pyrros Chaidos, V. G.*), «Lelantus-CLA» (*Lelantus-CLA*), Архив электронных документов по криптологии (*Cryptology ePrint Archive*), Документ 2021/1036 (2021), https://ia.cr/2021/1036

20 koe, «Seraphis: идея протокола приватных транзакций» (*Seraphis: A privacy-preserving transaction protocol abstraction*), репозиторий GitHub, DRAFT-v0.0.11 (2021), https://github.com/UkoeHB/Seraphis/releases/tag/DRAFT-v0.0.11

## A Модифицированная система доказательства Шаума-Педерсена

Система доказательства представляет собой набор алгоритмов (ChaumProve, ChaumVerify) для следующего отношения:

$$\left\{ pp_{chaum}, S, T \in G; (x, y, z) \in F : S = xF + yG + zH, u = xT + yG \right\}$$

Протокол выполняется следующим образом:

a.1 Доказывающая сторона выбирает случайные $r, s, t \in F$ и вычисляет

$$A_1, A_2, A_3 := (rF + sG, rT + sG, tH)$$

и отправляет полученные значения верификатору.

a.2 Верификатор выбирает случайный запрос $c \in F$ и отправляет его доказывающей стороне.

a.3 Доказывающая сторона вычисляет ответ

$$t_1, t_2, t_3 := (r + cx, s + cy, t + cz)$$

a.4 Верификатор принимает доказательство только в том случае, если

$$A_1 + A_3 + cS = t_1 F + t_2 G + t_3 H$$

и

$$A_2 + cU = t_1 T + t_2 G$$

Теперь докажем, что протокол является полным, надёжным и обеспечивает возможность доказательства с нулевым разглашением при условии наличия честного верификатора.

*Доказательство*. Полнота протокола доказывается простой проверкой.

Чтобы продемонстрировать, что доказательство в рамках протокола предполагает нулевое разглашение при наличии честного верификатора мы строим действительный симулятор, создающий транскрипты, распределяемые идентично тому, как это происходит в случае с действительными доказательствами. Симулятор выбирает случайный запрос $c \in F$, случайные скалярные величины $t_1, t_2, t_3 \in F$, и случайное значение $A_1 \in G$. Затем симулятор задаёт $A_2 := t_1 T + t_2 H - cU$ и $A_3 := t_1 F + t_2 G + t_3 H - cS - A_1$. Такой транскрипт будет принят честным верификатором. Следует отметить, что все элементы транскрипта в действительном доказательстве независимо и равномерно распределяются случайным образом, если генераторы $F, G, H, U$ являются независимыми, как и элементы транскрипта, созданные симулятором.

Чтобы продемонстрировать надёжность протокола, рассмотрим два транскрипта подтверждения с отдельными элемента ми запроса $c \neq c' \in F$:

$$\left( A_1, A_2, A_3, c, t_1, t_2, t_3 \right)$$

и

$$\left( A_1, A_2, A_3, c', t_1', t_2', t_3' \right)$$

Первое уравнение верификации, применяемое к двум транскриптам, подразумевает, что:

$$(c - c')S = (t_1 - t_1')F + (t_2 - t_2')G + (t_3 - t_3')H,$$

и мы можем выделить подтверждающие значения $x := (t_1 - t_1')/(c - c')$, $y := (t_2 - t_2')/(c - c')$ и $z := (t_3 - t_3')/(c - c')$ или нетривиальное дискретное логарифмическое отношение $F, G, H$ (контрадикцию в случае независимости этих генераторов). Подобным образом второе уравнение верификации подразумевает, что:

$$(c-c')U=(t_1-t_1')T+(t_2-t_2')G,$$

что даёт нам те значения $x$ и $y$, что и требуется.

Это делает доказательство полным.

## B Параллельная система доказательства по одному из множества

Сама система доказательства представляет собой набор алгоритмов (ParProve, ParVerify) для следующего отношения, где, допустим, $N=n^m$:

$$\left\{pp_{par},\{S_i,V_i\}_{i=0}^{N-1}\subset G^2;l\in N,(s,v)\in F:0\le l<N,S_l=Com(0,s)V_l=Com(0,v)\right\}$$

Протокол представлен на рисунке 1, где мы используем систему обозначений, предложенную в работе [12].

Протокол является полным, надёжным и обеспечивает возможность доказательства с нулевым разглашением при условии наличия честного верификатора. Доказательство является практически таким же, что и в случае с оригинальной схемой, за исключением минимальных и простых изменений.

Compute – Вычислить
Accept only and only if – Принимается только и если только
computing $p_{i,j}$ as in the orig. paper – $p_{i,j}$ вычисляется точно так же, как и в оригинальной работе

***Рисунок 1**. Протокол параллельного доказательства по одному из множества*

## C Безопасность платёжной системы

Zerocash [1] была создана устойчивая концепция безопасности схемы децентрализованных анонимных платежей (DAP), охватывающая реалистичную модель угрозы, предусматривающую наличие обладающего достаточно мощными ресурсами злоумышленника, способного добавлять «вредоносные» монеты в анонимные группы входов транзакций, контролировать выбор входов транзакций и создавать произвольные транзакции, которые будут добавлены в реестр. В данном приложении мы формально доказываем безопасность протокола Spark в рамках соответствующей (но модифицированной) модели безопасности; доказательства некоторым образом схожи с теми, что были представлены в работе [1].

Напомним определение безопасности схемы DAP:

$$П=(Setup,CreateAddress,Mint,Spend,Recover,Verify),$$

которая безопасна, если соответствует определениям неотличимости данных в реестре, невозможности изменения транзакций, а также свойствам безопасности баланса, о которых говорится ниже.

Каждое из свойств безопасности формализуется как игра между злоумышленником $A$, решающим свою задачу за полиномиальное время, и запросчиком $C$, где в рамках каждой такой игры поведение честных сторон моделируется посредством оракула $O^{DAP}$. Оракул $O^{DAP}$ поддерживает реестр транзакций $L$ и обеспечивает интерфейс для выполнения операций алгоритмов CreateAddress, Mint и Spend честными сторонами. Чтобы смоделировать поведение честных сторон $A$ передаёт запрос $C$, который производит проверки правильности, а затем передаёт полномочия оракулу $O^{DAP}$, по необходимости возвращающему результаты $A$. В случае с запросами CreateAddress $C$ запускает алгоритм протокола CreateAddress и выдаёт $A$ публичный адрес $addr_{pk}$. В случае с запросами Mint злоумышленник указывает значение суммы и публичный адрес получателя транзакции, а соответствующая транзакция создаётся и передаётся $C$, если является действительной. В случае с запросами Spend злоумышленник указывает количество входящих монет, которые

будут использованы, а также значения сумм и публичные адреса получателя транзакции, а соответствующая транзакция создаётся (после того, как $C$ восстановит использованные монеты) и передаётся $C$, если является действительной. Оракул $O^{DAP}$ также передаёт запрос Insert, который позволяет злоумышленнику вставить в реестр $L$ произвольную и потенциально вредоносную транзакцию $tx_{mint}$ или $tx_{spend}$, которые, как при этом предполагается, являются действительными.

В случае с каждым свойством безопасности мы можем утверждать, что схема DAP соответствует ему, если злоумышленник может выиграть в относящейся к нему игре лишь с ничтожной вероятностью.

*Примечание 2*. Нам также требуется, чтобы схема DAP была полной, а это подразумевает, что каждую непотраченную монету (с уникальным частичным раскрытием обязательства по порядковому номеру ¿) в реестре можно потратить. Это свойство означает, что если монета присутствует в реестре $L$ в качестве выхода транзакции, но её соответствующие тег не раскрывается в какой-либо действительной транзакции, то пользователь, обладающий ключом траты для соответствующего адреса может сгенерировать действительную транзакцию траты, в которой она будет потрачена.

Следует отметить, что если владелец ключа траты не может создать такую действительную транзакцию, в соответствии с конструкцией тег монеты должен появиться уже в предыдущей действительной транзакции. В такой предыдущей действительной транзакции должно присутствовать модифицированное доказательство Шаума-Педерсена, которое частично позволяет выделить приватного ключа траты $r$ соответствующего адреса, что является противоречием, поскольку у злоумышленника имеется лишь незначительное преимущество при выведении дискретного логарифма.

### C.1 Баланс

Баланс требует, чтобы у злоумышленника $A$ не было возможности контролировать большее количество монет, чем было создано или потрачено. Это формализуется в рамках игры *BAL*. Злоумышленник $A$ адаптивно взаимодействует с $C$ и оракулом, отправляя запросы, и в конце взаимодействия выдаёт набор монет AdvCoins. Допустим, ADDR является набором всех адресов честных пользователей, сгенерированных запросами CreateAddress. $A$ побеждает в игре, если

$$v_{unspent} + v_{A \to ADDR} > v_{mint} + v_{ADDR \to A},$$

а это подразумевает, что общая сумма, которую злоумышленник может потратить или уже потратил, больше, чем та сумма, которую он создал или получил. В данном случае:

- $v_{unspent}$ является общим значением непотраченных монет в AdvCoins;
- $v_{mint}$ является общим значением монет, созданных $A$ для себя путём запросов Mint или Insert;
- $v_{ADDR \to A}$ является общим значением монет, полученных $A$ с адресов из набора ADDR;
- $v_{A \to ADDR}$ является общим значением монет, отправленных злоумышленником на адреса из набора ADDR.

Мы можем утверждать, что DAP-схема $\Pi$ является безопасной в рамках *BAL*, если злоумышленник $A$ может победить в игре *BAL* лишь с ничтожной вероятностью:

$$Pr\left[BAL(\Pi, A, \lambda) = 1\right] \leq negl(\lambda)$$

Предположим, что запрашивающая сторона поддерживает дополнительный дополненный реестр $(L, \vec{a})$, где каждая $a_i$ содержит секретные данные из транзакции $tx_i$ из $L$. В таком случае, если транзакция $tx_i$ была создана в результате запроса, переданного $A$

запросчику $C$, $a_i$ содержит все данные свидетельства, выведенные из доказательств, имеющихся в транзакции. Полученный дополненный реестр $(L, \vec{a})$ будет сбалансирован при соблюдении следующих условий:

1 В каждой транзакции траты $tx_{spend,k}$ из реестра $(L, \vec{a})$ должны быть использованы отдельные, не совпадающие монеты, и каждая из монет должна являться выходом действительной транзакции $tx_{mint,i}$ или $tx_{spend,j}$ для некоторого $i<k$ или $j<k$. Это требование подразумевает, что во всех транзакциях будут тратиться только действительные монеты, и что ни одна из монет не будет потрачена более одного раза в рамках одной и той же действительной транзакции.

2 Две действительные транзакции траты в реестре $(L, \vec{a})$ не должны использовать одну и ту же монету. Это подразумевает. Что ни одна монета не может быть потрачена в двух разных транзакциях. В сочетании с первым требованием это означает, что каждая из монет тратится лишь единожды.

3 Для каждой $(tx_{spend}, a)$ в реестре $(L, \vec{a})$, использующей входящие монеты с обязательствами по значению $\{C_u\}_{u=0}^{w-1}$ для каждого $u \in [0, w)$:

– если $C_u$ является выходом действительной транзакции создания со свидетельством дополненного реестра $a'$, значение $C_u$, содержащееся в $a'$, будет таким же, как и соответствующее значение, которое содержится в $a$ для офсета обязательства по значению $C_u'$;

– если $C_u$ является выходом действительной транзакции траты со свидетельством дополненного реестра $a'$, значение $C_u$, содержащееся в $a'$, будет таким же, как и соответствующее значение, которое содержится в $a$ для офсета обязательства по значению $C_u'$.

Это подразумевает, что значения сохраняются между транзакциями.

4 Для каждой $(tx_{spend}, a)$ в реестре $(L, \vec{a})$ с комиссией $f$, использующей входящие монеты с обязательствами по значению $\{C_u\}_{u=0}^{w-1}$ и генерирующей монеты с обязательствами по значению $\{\overline{C}_j\}_{j=0}^{t-1}$, $a$ содержит значения $\{v_u\}_{u=0}^{w-1}$ и $\{\overline{v}_j\}_{j=0}^{t-1}$, соответствующие обязательствам таким образом, что уравнение баланса

$$\sum_{u=0}^{w-1} v_u = \sum_{j=0}^{t-1} \overline{v}_j + f$$

остаётся действительным. Для каждой $(tx_{mint}, a)$ в реестре $(L, \vec{a})$ с публичным значением $v$, генерирующей монету с обязательством по значению $C$, $a$ содержит значение $v'$, соответствующее обязательству таким образом, что $v = v'$. Это означает, что значения не могут быть получены произвольным образом.

5 Для каждой $tx_{spend}$ в реестре $(L, \vec{a})$, вставленной $A$ посредством запроса Insert, каждая монета, использованная в $tx_{spend}$, не может быть восстановлена по какому-либо адресу из набора ADDR. Это означает, что злоумышленник не в состоянии сгенерировать транзакцию, использующую монеты, которые он не контролирует.

Если эти пять условий соблюдены, значит, злоумышленник $A$ не смог потратить и не контролирует больше монет, чем было ранее создано или получено им, а неравенство

$$v_{mint} + v_{ADDR \to A} \leq v_{unspent} + v_{A \to ADDR}$$

сохраняется. Теперь докажем, что протокол Spark является безопасным в рамках *BAL*, согласно настоящему определению.

*Доказательство*. Методом противоречия предположим, что злоумышленник $A$ взаимодействует с $C$, в результате чего с не ничтожно малой вероятностью создаётся не

сбалансированный дополненный реестр $(L,\vec{a})$; в этом случае с не ничтожно малой вероятностью нарушается по крайней мере одно из пяти условий, описанных выше:

$A$ **нарушает Условие 1**. Предположим, что вероятность того, что $A$ победит в игре, нарушив Условие 1, не является ничтожно малой. Каждая транзакция $tx_{spend}$, сгенерированная запросом оракула, не при помощи Insert, уже соответствует этому условию, поэтому, в $(L,\vec{a})$ должна существовать транзакция $(tx_{spend},a)$, вставленная $A$.

Предположим, что существуют входы $u_1,u_2\in[0,w)$ транзакции $tx_{spend}$, в соответствии с которыми используется одна и та же монета, то есть, демонстрируется одно и то же частичное раскрытие ¿ обязательств по порядковому номеру монеты. Действительность модифицированных доказательств Шаума-Педерсена $(\Pi_{chaum})_{u_1}$ и $(\Pi_{chaum})_{u_2}$ для этих входов даёт выделенные раскрытия $S'_{u_1}=s_{u_1}F+r_{u_1}G+y_{u_1}H$ и $S'_{u_2}=s_{u_2}F+r_{u_2}G+y_{u_2}H$ и такие представления тегов, что $U=s_{u_1}T_{u_1}+r_{u_1}G$ и $U=s_{u_2}T_{u_2}+r_{u_2}G$. Поскольку действительность транзакции подразумевает, что $T_{u_1}\neq T_{u_2}$, у нас должно быть $(s_{u_1},r_{u_1})\neq(s_{u_2},r_{u_2})$. Действительность соответствующего параллельных доказательств по одному из множества $(\Pi_{par})_{u_1}$ и $(\Pi_{par})_{u_2}$ даёт нам индексы (соответствующие элементам входящей группы $S_1$ и $S_2$) и такие результаты выведения дискретного логарифма, что $S_1-S'_{u_1}=x_{u_1}H$ и $S_2-S'_{u_2}=x_{u_2}H$. Это означает, что

$$S_1=DCom(s_{u_1},r_{u_1},x_{u_1}+y_{u_1})$$

и

$$S_2=DCom(s_{u_2},r_{u_2},x_{u_2}+y_{u_2})$$

что противоречит допуску наличия уникальных частичных раскрытий для использованных обязательств по порядковому номеру монеты.

Второй вариант нарушения условия состоит в том, чтобы в транзакции $tx_{spend}$ была потрачена монета, не созданная в какой-либо предыдущей действительной транзакции. Действительное модифицированное обязательство Шаума-Педерсена при таком входе даёт представление тега $U=sT+rG$ и офсет обязательства по порядковому номеру $S'=sF+rG+yH$. Действительное параллельное доказательство по одному из множества для входа даёт такой индекс $l$, что $S_l-S'=xH$, а это означает, что $S_l=sF+rG+(x+y)H$ является раскрытием данного обязательства. Поскольку действительность транзакции требует, чтобы все элементы набора, обеспечивающего неопределённость входов, были созданы в предшествующих транзакциях как действительные обязательства, злоумышленнику будет известно раскрытие такого обязательства, что является противоречием.

$A$ **нарушает Условие 2**. Предположим, что вероятность того, что $A$ победит в игре, нарушив Условие 2, не является ничтожно малой. Это означает, что в дополненном реестре $(L,\vec{a})$ содержится две действительные транзакции Spend, в которых была использована одна и та же монета, но при этом были созданы отдельные теги. Подобно тому, как и в прошлом случае, это означает наличие отдельных раскрытий обязательства по порядковому номеру монеты, что является противоречием.

$A$ **нарушает Условие 3**. Предположим, что вероятность того, что $A$ победит в игре, нарушив Условие 3, не является ничтожно малой. Допустим, $C$ является обязательством по значению монеты, использованной со входом транзакции $tx_{spend}$ и сгенерированной в предыдущей транзакции (любого типа) в реестре $(L,\vec{a})$. Поскольку сгенерированная транзакция является действительной, у нас имеется выделенное раскрытие $C=vG+aH$ либо из доказательства баланса (в случае с транзакцией Mint), либо из доказательства диапазона (в случае с транзакцией Spend). Действительность соответствующего параллельного доказательства по одному из множества из транзакции $tx_{spend}$ даёт нам выделенный

дискретный логарифм $C - C' = xH$, где $C'$ является офсетом обязательства по значению входа. Но это также даёт нам и $C' = vG + (a - x)H$, что является противоречием, поскольку схема обязательства является обязательной для реализации.

$A$ **нарушает Условие 4**. Предположим, что вероятность того, что $A$ победит в игре, нарушив Условие 4, не является ничтожно малой. Если дополненный реестр $(L, \vec{a})$ содержит транзакцию Spend, нарушающую уравнение баланса, это также нарушает свойство обязательной реализации схемы обязательства, поскольку соответствующее доказательство баланса $\Pi_{bal}$ является действительным, а это уже будет противоречием. Если же дополненный реестр $(L, \vec{a})$ будет содержать транзакцию Mint, нарушающую требование к балансу, это также будет означать нарушение свойства обязательной реализации схемы обязательства, поскольку соответствующее доказательство баланса $\Pi_{bal}$ является действительным, а это опять же будет противоречием.

$A$ **нарушает Условие 5**. Предположим, что вероятность того, что $A$ победит в игре, нарушив Условие 5, не является ничтожно малой. То есть, $A$ создаст транзакцию Spend, $tx_{spend}$, воспользовавшись запросом Insert, являющимся действительным для дополненного реестра $(L, \vec{a})$, и использует монету, соответствующую обязательству по порядковому номеру монеты $S$, которое можно восстановить по публичному адресу $(Q_1, Q_2) \in ADDR$.

Действительность обязательства Шаума-Педерсена, соответствующего этому входу транзакции $tx_{spend}$ даёт нам выделенное представление $S' = s'F + r'G + yH$. Действительность соответствующего параллельного обязательства по одному из множества даёт нам обязательство по порядковому номеру $S$ и такой результат выведения, что $S - S' = xH$, и таким образом $S = s'F + r'G + (x + y)H$.

Теперь допустим, что $(s_1, s_2, r)$ является секретным ключом, соответствующим адресу $(Q_1, Q_2)$. Поскольку в транзакции $tx_{spend}$ используется монета, которую можно восстановить по этому адресу, обязательством по порядковому номеру для восстановленной монеты будет

$$\overline{S} = H_{ser}(s_1 K, Q_1, Q_2)F + Q_2 = \left(H_{ser}(s_1 K, Q_1, Q_2) + s_2\right)F + rG$$

для ключа восстановления $K$.

Поскольку схема обязательства является обязательной для реализации, мы должны получить $r' = r$, что представляет собой противоречие, поскольку $A$ не может выделить этот дискретный логарифм на основе публичного адреса.

Это делает наше доказательство полным.

### C.2 Отсутствие гибкости транзакции

Это свойство требует, чтобы никакой злоумышленник не смог в значительной степени изменить действительную транзакцию. В частности, отсутствие гибкости не позволяет злоумышленникам модифицировать транзакции честных пользователей путём изменения данных или перенаправления выходов действительных транзакций до того, как транзакция будет добавлена в реестр. Поскольку свойство отсутствия гибкости транзакций Mint зависит от полномочий, предусмотренных правилами консенсуса или операциями базового уровня, нами рассматриваются транзакции Spend.

Это свойство формализовано при помощи экспериментальной игры **TR-NM**, в рамках которой злоумышленник $A$ адаптивно взаимодействует с оракулом $O^{DAP}$, а затем выдаёт транзакцию траты $tx'$. Если мы обозначим как $T$ набор всех транзакций, производимых путём запросов Spend, передаваемых $O^{DAP}$, а как $L$ обозначим окончательный вариант реестра, $A$ победит в игре в том случае, если будет создана такая $tx \in T$, что:

- $tx' \neq tx$;
- $tx'$ будет раскрывать тот же тег, что и $tx$;

– как $tx'$, так и $tx$ будут действительными транзакциями относительно реестра $L'$, содержащего все транзакции, предшествующие $tx$ из $L$.

Мы можем утверждать, что DAP-схема $\Pi$ является безопасной в рамках **TR-NM**, если злоумышленник $A$ может победить в игре **TR-NM** лишь с незначительной вероятностью:

$$Pr\left[TR-NM\left(\Pi,A,\lambda\right)=1\right]\leq negl\left(\lambda\right)$$

Допустим, $T$ является набором всех транзакций $tx_{spend}$, сгенерированных $O^{DAP}$ в ответ на запросы Spend. Поскольку эти транзакции генерируются только при помощи таких запросов, передаваемых оракулу, $A$ не может узнать каких-либо секретных данных, используемых для создания этих транзакций.

*Доказательство*. Предположим, что вероятность того, что $A$ победит в игре, не является ничтожно малой. То есть, при этом $A$ создаст транзакцию $tx'$, раскрывающую тег $T$, который также раскрывается в транзакции $tx$. Без ущерба для общности предположим, что в каждой транзакции используется одна монета.

Отметим, что действительная транзакция Spend связывает все элементы транзакции, за исключением модифицированных доказательств Шаума-Педерсена, посредством хеша $H_{bind}$ и транскриптов доказательства. Следовательно, чтобы создать действительную транзакцию $tx'\neq tx$, существует два пути:

– модифицированные доказательства Шаума-Педерсена должны быть идентичными, но $tx'$ и $tx$ будут отличаться другим элементом в структуре транзакции;
– модифицированное доказательство Шаума-Педерсена в транзакции $tx'$ будет отличаться от доказательства в $tx$.

В первом случае, по крайней мере, один вход связующего хеша $H_{bind}$, используемый для инициализации транскриптов доказательства Шаума-Педерсена, должен отличаться между доказательствами. Поскольку мы моделируем эту хеш-функцию как случайный оракул, и результаты будут отличаться за исключением ничтожной вероятности, что является противоречием, так как полученные структуры доказательств должны быть идентичными.

Во втором случае предположим, что модифицированное доказательство Шаума-Педерсена $\Pi'_{chaum}$, которое содержится в транзакции $tx'$, даёт нам выделение $\left(s',r',y'\right)$, и что доказательство $\Pi_{chaum}$, которое содержится в транзакции $tx$, даёт нам $\left(s,r,y\right)$. Поскольку соответствующие теги будут идентичными, получаем $s'=s$ и $r'=r$. Любое обязательство по порядковому номеру $S$ с частичным раскрытием ¿, используемое в транзакции $tx$, генерируется таким образом, что $r$ является компонентом ключа траты контролирующего публичного адреса $\left(Q_1,Q_2\right)$. Поскольку злоумышленник $A$ не контролирует данный адрес, он не может получить $r$, не выделив его из $S$ или из $\left(Q_1,Q_2\right)$, что означает наличие не являющегося ничтожно малым преимущества с точки зрения получения дискретного логарифма, то есть, противоречие.

### C.3 Неразличимость данных реестра

Это свойство подразумевает, что никакой злоумышленник $A$ не может получить из реестра какой-либо информации за исключением той, что уже была публично раскрыта, даже если он может повлиять на выполнение операций с действительным реестром честными пользователями.

Свойство неразличимости данных реестра формализовано при помощи экспериментальной игры **L-IND**, в которой участвуют злоумышленник $A$ и запросчик $C$, и которая заканчивается получением двоичного выхода $b'$ злоумышленником $A$. В начале эксперимента $C$ выбирает $Setup\left(1^{\lambda}\right)\rightarrow pp$ и отправляет параметры $A$. Затем он выбирает

случайный бит $b \in \{0,1\}$ и запускает два отдельных оракула DAP $O_0^{DAP}$ и $O_1^{DAP}$, каждый из которых имеет собственный отдельный реестр и своё внутреннее состояние. При выполнении каждого последующего шага эксперимента:

1 $C$ предлагает $A$ два реестра ¿, где $L_b$ и $L_{1-b}$ являются текущими реестрами оракулов $O_b^{DAP}$ и $O_{1-b}^{DAP}$, соответственно.

2 $A$ отправляет $C$ два запроса, $Q$ и $Q'$, одного и того же типа (один из следующих: CreateAddress, Mint, Spend, Recover или Insert).

- Если типом запроса является Insert или Mint, $C$ отправляет запрос $Q$ реестру $L_b$ и запрос $Q'$ реестру $L_{1-b}$, позволяя $A$ вставить собственные транзакции или создать новые монеты в $L_{¿¿}$ и $L_{¿}$.
- В случае со всеми запросами типов CreateAddress, Spend или Recover $C$ сначала проверяет, являются ли запросы $Q$ и $Q'$ публично согласованными, а затем направляет запрос $Q$ оракулу $O_b^{DAP}$ и запрос $Q'$ оракулу $O_{1-b}^{DAP}$. Он получает два ответа оракула $(a_0, a_1)$, но возвращает $(a_b, a_{1-b})$ злоумышленнику $A$.

Поскольку злоумышленнику не известен бит $b$ и распределение ¿ и $(L_0, L_1)$, он не может узнать, влияет ли оно на поведение честных сторон в $(L_0, L_1)$ или $(L_1, L_0)$. В конце эксперимента $A$ отправляет $C$ бит $b' \in \{0,1\}$. Запросчик $C$ выдаёт 1, если $b = b'$, и 0 в противном случае.

Запросы $Q$ и $Q'$ должны быть публично согласованы следующим образом: если типом запроса $Q$ и $Q'$ является Recover, они являются публично согласованными по конструкции. Если типом запроса $Q$ и $Q'$ является CreateAddress, оба оракула генерируют одинаковый адрес. Если типом запроса $Q$ и $Q'$ является Mint, созданные значения обоих запросов должны быть равными. Если типом запроса $Q$ и $Q'$ является Send, тогда:

- Оба запроса, $Q$ и $Q'$, должны быть правильно оформлены и должны быть действительными, чтобы входящие монеты были сгенерированы в предшествующей транзакции, находящейся в реестре, и не были потрачены. Кроме того, транзакция должна быть сбалансированной.
- Количество потраченных монет и выходящих должно быть одним и тем же в $Q$ и $Q'$.
- Если использованная в $Q$ монета ссылается на монету в $L_0$, указанном $A$ в запросе Insert, соответствующий индекс в $Q'$ также должен ссылаться на монету в $L_1$, указанном $A$ в запросе Insert, и значения этих двух монет также должны быть равными (и наоборот в случае с $Q'$).
- Если выходящая монета, на которую ссылается $Q$, не ссылается на адрес получателя в списке ADDR оракула (а следовательно, контролируется $A$), то соответствующее значение должно быть равным значению соответствующей монеты, на которую ссылается $Q$ по тому же самому индексу (и наоборот в случае с $Q'$).

Мы можем утверждать, что DAP-схема $\Pi$ является безопасной в рамках **L-IND**, если злоумышленник $A$ может победить в игре **L-IND** лишь с незначительно большей вероятностью, чем по чистой случайности:

$$Pr\left[L\text{-}IND(\Pi, A, \lambda) = 1\right] - \frac{1}{2} \leq negl(\lambda)$$

*Доказательство*. Чтобы доказать, что преимущество злоумышленника $A$ в рамках игры **L-IND** является ничтожным, сначала рассмотрим моделирующий эксперимент $D^{¿}$, в рамках

которого $A$ взаимодействует с $C$, как в случае с экспериментом L-IND, но с некоторыми изменениями.

**Моделирующий эксперимент** $D^{¿}$. Поскольку параллельная система доказательства по одному из множества, системы доказательства Шаума-Педерсена, представления и диапазона подразумевают наличие честного верификатора при нулевом разглашении, мы можем воспользоваться преимуществами модели в отношении каждой из них. Если имеются заявленные данные по входам и запросы верификатора, модель каждой из систем доказательства создаёт транскрипты, неотличимые от честных доказательств. Помимо этого, мы определяем поведение полной модели.

**Моделирование**. Моделирование $D^{¿}$ работает следующим образом. Как и в случае с оригинальным экспериментом $C$ выбирает системные параметры $Setup(1^{\lambda}) \rightarrow pp$ и случайный бит $b$ и запускает два оракула DAP $O_0^{DAP}$ и $O_1^{DAP}$. Затем $D^{¿}$ реализуется этапами. На каждом из этапов $A$ предлагаются реестры $L_{¿=L_b¿}$ и $L_{¿}=L_{1-b}$, после чего $A$ отправляет два публично согласованных запроса $(Q, Q')$ одного и того же типа. Мы помним о том, что запросы $Q$ и $Q'$ согласованы относительно общедоступных данных и информации, связанной с адресами, контролируемыми $A$. В зависимости от типа запроса запросчик действует следующим образом:

- В ответ на запросы Recover и Insert запросчик действует, как в случае с оригинальным экспериментом **L-IND**.
- В ответ на запросы CreateAddress запросчик заменяет компоненты публичного адреса $Q_1$ и $Q_2$ случайными строками соответствующей длины, создавая таким образом $addr_{pk}$, который отправляется $A$.
- В ответ на запросы Mint запросчик, чтобы ответить $Q$ и $Q'$ по-отдельности, делает следующее:
  1. Если $A$ предоставит публичный адрес $addr_{pk}$, созданный не запросчиком, он создаст монету, используя CreateCoin, как обычно.
  2. В противном случае он симулирует создание монеты:
     a. Равномерно случайным образом выберет ключ восстановления $K$.
     b. Равномерно случайным образом выберет обязательство по порядковому номеру $S$.
     c. Равномерно случайным образом выберет обязательство по значению $C$.
     d. Выберет случайный вход, использованный для создания ключа симметричного шифрования $AEAD\ KeyGen \rightarrow k_{enc}$.
     e. Симулирует шифрование сопроводительных данных, выбрав случайное значение $\widetilde{m}$ соответствующей длины, зашифровав его, чтобы получить

$$AEAD\ Encrypt(k_{enc}, memo, \widetilde{m}) \rightarrow \overline{m}.$$

  3. Симулирует доказательство баланса $\Pi_{bal}$ для утверждения $(C - Com(v, 0))$.
  4. Соберёт транзакцию и надлежащим образом добавит её в реестр.
- В ответ на запросы Spend запросчик, чтобы ответить $Q$ и $Q'$ по-отдельности, делает следующее (при этом, $w$ указывает на количество использованных монет, а $t$ — на количество созданных монет, указанных $A$ в его запросах):
  1. Разобьёт обязательства по порядковому номеру и обязательства по значению набора выходов как $InCoins = \{(S_i, C_i)\}_{i=0}^{N-1}$.
  2. Для каждого $u \in [0, w)$, где $l_u$ представляет индекс использованной в $InCoins$ монеты:
     a. Равномерно случайным образом выберет тег $T_u$.

b Равномерно случайным образом выберет офсет обязательства по порядковому номеру $S_u^{'}$ и офсет обязательства по значению $C_u^{'}$.

c Симулирует параллельное доказательство по одному из множества $\left(\Pi_{par}\right)_u$ для утверждения $\left(\left\{S_i - S_u^{'}, C_i - C_u^{'}\right\}_{i=0}^{N-1}\right)$.

3 Для каждого $j \in [0, t)$:

a Если $A$ предоставит публичный адрес $addr_{pk}$, созданный не запросчиком, создаст монету, используя CreateCoin, как обычно.

b В противном случае симулирует создание монеты:

i Равномерно случайным образом выберет ключ восстановления $K_j$.

ii Равномерно случайным образом выберет обязательство по порядковому номеру $S_j$.

iii Равномерно случайным образом выберет обязательство по значению $\overline{C}_j$.

iv Выберет случайный вход, использованный для создания ключа симметричного шифрования $AEAD\ KeyGen \rightarrow k_{enc}$.

v Симулирует шифрование сопроводительных данных, выбрав случайное значение $\tilde{v}$ соответствующей длины, зашифровав его, чтобы получить

$$AEAD\ Encrypt\left(k_{enc}, val, \tilde{v}\right) \rightarrow \overline{v}_j.$$

vi Симулирует шифрование сопроводительных данных, выбрав случайное значение $\tilde{m}$ соответствующей длины, зашифровав его, чтобы получить

$$AEAD\ Encrypt\left(k_{enc}, memo, \tilde{m}\right) \rightarrow \overline{m}_j.$$

vii Симулирует доказательство диапазона $\left(\Pi_{rp}\right)_j$ для утверждения $\left(\overline{C}_j\right)$.

4 Симулирует доказательство баланса $\Pi_{bal}$ для утверждения

$$\left(\sum_{u=0}^{w-1} C_u^{'} - \sum_{j=0}^{t-1} \overline{C}_j - Com(f, 0)\right)$$

5 Для каждого $u \in [0, w)$ согласно определению вычислит связующий хеш $\mu$ и симулирует модифицированное доказательство Шаума-Педерсена $\left(\Pi_{chaum}\right)_u$ для утверждения $\left(S_u^{'}, T_u\right)$.

6 Соберёт транзакцию и надлежащим образом добавит её в реестр.

В случае с экспериментами, о которых говорится ниже, мы определяем $Adv^D$ как преимущество $A$ в рамках некоторого эксперимента $D$ над оригинальной игрой **L-IND**. По определению все ответы, отправляемые $A$ в $D^{¿}$ вычисляются независимо от бита $b$, поэтому, $Adv^{D^{¿}} = 0$. Мы докажем, что преимущество $A$ в рамках реального эксперимента L-IND $D^{real}$ практически ничтожно отличаются от преимущества $A$ в рамках $D^{¿}$. Чтобы продемонстрировать это, построим промежуточные эксперименты, в которых $C$ участвует в определённом модифицированном варианте $D^{real}$ против $A$.

**Эксперимент** $D_1$. В рамках данного эксперимента $D^{real}$ изменяется путём моделирования всех доказательств по одному из множества, доказательств диапазона, доказательств представления и модифицированных доказательств Шаума-Педерсена. Поскольку все эти протоколы подразумевают наличие честного верификатора при нулевом

разглашении, смоделированные доказательства будут неотличимы от реальных доказательств, сгенерированных при $D^{real}$. Следовательно, $Adv^{D_1}=0$.

**Эксперимент** $D_2$. В рамках данного эксперимента $D_1$ изменяется путём замены всех зашифрованных значений и сопроводительных данных в транзакциях на сгенерированные запросчиком публичные адреса получателя с шифрованием случайных значений надлежащей длины в соответствии с ключами, выбранными равномерно и случайным образом, а также путём замены ключей восстановления равномерно выбранными случайными значениями. Поскольку лежащая в основе схема аутентифицированного симметричного шифрования является безопасной в рамках IND-CCA и IK-CCA, и мы допускаем, что задача принятия решения Диффи-Хеллмана является сложной, преимущество злоумышленника при распознавании выхода в реестре в рамках эксперимента $D_2$ будет ничтожно отличаться от его преимущества в рамках эксперимента $D_1$. Следовательно, $\left|Adv^{D_2}-Adv^{D_1}\right|$ также будет ничтожным.

**Эксперимент** $D^{¿}$. Эксперимент $D^{¿}$ формально описан выше. В частности, он отличается от $D_2$ заменой тегов использованных монет, офсета обязательства по порядковому номеру и офсетов обязательства по значению равномерно выбранными случайными значениями, а также заменой обязательств по порядковому номеру и обязательств по значению исходящих монет случайными значениями. В рамках предшествующих экспериментов (включая $D^{real}$) теги генерировались при помощи псевдослучайной функции [7], а другие заданные значения генерировались как обязательства с масками, выведенными на основе хеш-функций в качестве независимых случайных оракулов, поэтому, преимущество злоумышленника при распознавании выхода в реестре при реализации эксперимента $D^{¿}$ будет ничтожно отличаться от его преимущества при реализации эксперимента $D_2$. Следовательно, $\left|Adv^{D^{¿}}-Adv^{D_2}\right|$ также будет ничтожным.

Это демонстрирует, что у злоумышленника имеется лишь ничтожное преимущество в реальной игре **L-IND** относительно моделирования, где мы можем победить лишь с незначительно большей вероятностью, чем по чистой случайности, что делает наше доказательство полным.