Кольцевые конфиденциальные транзакции

Шен Ноезер (Shen Noether)*, Адам Маккензи (Adam Mackenzie) и Monero Core Team
Исследовательская лаборатория Monero (Monero Research Lab)
Февраль 2016

Аннотация

В данной статье предлагается метод сокрытия сумм транзакций децентрализованной анонимной криптовалюты Monero. Подобно Bitcoin, Monero является криптовалютой, распределяемой с использованием доказательства работы в процессе «майнинга». В основе оригинального протокола Monero лежит CryptoNote, протокол, предполагающий использование кольцевых подписей и одноразовых ключей с целью сокрытия адресата и источника транзакций. Недавно ведущим разработчиком Bitcoin Грегори Максвеллом (Gregory Maxwell) была рассмотрена и реализована технология сокрытия суммы транзакций при помощи схемы обязательства. В этой статье описан новый тип кольцевой подписи, многоуровневая связываемая подпись спонтанной анонимной группы (Multi-layered Linkable Spontaneous Anonymous Group signature), позволяющая скрывать суммы, источники и адресатов транзакций с разумной эффективностью, обеспечивая при этом верифицируемое и не требующее доверия создание монет. Предлагаются некоторые расширения протокола, такие как агрегированные доказательства диапазона Шнорра (Aggregate Schnorr Range Proofs) и кольцевые мультиподписи (Ring Multisignature). Автору хотелось бы отметить, что ранние версии этой работы публиковались на исследовательском IRC канале сообществ Monero и Bitcoin. Укороченная черновая версия блокчейна рассматривается в работе [14], и это свидетельствует о том, что работа началась летом 2015 года, а завершена была в октябре 2015. Электронная версия также доступна по ссылке `http://eprint.iacr.org/2015/1098`.

1 Введение

1.1 Применение специализированных (ad hoc) спонтанных кольцевых подписей в криптовалютах

Вспомним, что каждая транзакция Bitcoin подписывается владельцем передаваемых монет, и такие подписи подтверждают, что владелец имеет право отправлять эти монеты. Это полностью аналогично подписанию банковского чека.

Работы по CryptoNote [16] и Ring Coin [2] развивают идею использования «кольцевых подписей», которая изначально была описана в работе [15] как «цифровая подпись, указывающая на группу вероятных подписантов таким образом, что верификатор не может точно сказать, кто из членов этой группы является создателем подписи». Следовательно, идея заключается в обладании оригинальным публичным ключом транзакции, который будет скрыт внутри группы публичных ключей, каждый из которых будет соответствовать одной и той же сумме монет, чтобы никто не мог точно сказать, который из пользователей отправил монеты.

Оригинальный протокол CryptoNote, описанный в работе [16], включает в себя небольшое изменение этой схемы, направленное на предотвращение двойной траты. Это изменение, предлагаемое в

*lab@getmonero.org

работе [16], представляет собой небольшую модификацию «отслеживаемой кольцевой подписи», описанной в работе [6]. Этот тип кольцевой подписи обеспечивает преимущество, которое заключается в том, что владелец монеты не сможет создать две различные кольцевые подписи, используя один и тот же публичный ключ так, чтобы этого нельзя было увидеть в блокчейне. Очевидной причиной этого изменения является предотвращение «двойной траты», которая в случае с Bitcoin означает, что одна и та же монета будет потрачена дважды. Ring Coin [2, 3] использует более эффективную схему связываемой кольцевой подписи, которая является небольшой модификацией связываемой подписи спонтанной анонимной группы (Linkable Spontaneous Anonymous Group signature), описанной в работе [8].

Одним из преимуществ, указанных выше типов кольцевых подписей относительно других методов обеспечения анонимности, таких как CoinJoin [10] или сервисы для смешивания монет (миксеры), является то, что они дают возможность «спонтанного» смешивания. В случае с CoinJoin или миксерами также можно скрыть источник отдельно взятой транзакции, но на практике эти методы требуют привлечения своего рода менеджера централизованной группы, например, использования централизованного сервера CoinJoin, чтобы такая доверенная сторона смогла комбинировать соответствующие транзакции. В случае, если доверенная сторона будет взломана, анонимность транзакции также будет подвергнута риску.

Некоторые монеты, такие как [5] (которая изначально называлась Darkcoin), пытаются избежать этого, используя большое количество доверенных миксеров (в этом случае именуемых masternode), но всё же это количество по-прежнему не так велико, как количество пользователей монеты. При использовании спонтанной кольцевой подписи, транзакции, напротив, создаются владельцем соответствующего публичного ключа (это спонтанное или «специальное» свойство) без привлечения какого-либо доверенного сервера, что гораздо безопасней с точки зрения анонимности.

Одним из возможных вариантов атаки на оригинальный протокол CryptoNote или Ring Coin [16, 2] является анализ блокчейна, в основе которого лежат данные по сумме, отправляемой в определённой транзакции. Например, если злоумышленнику известно, что в определённое время было отправлено .9 монет, ему необходимо сузить список возможных отправителей, и для этого он ищет транзакции с .9 монет. Этого можно некоторым образом избежать при помощи одноразовых ключей [16], так как у отправителя есть возможность включить ряд адресов сдачи в транзакцию, скрыв таким образом сумму, которая была отправлена с применением такого «рюкзачного смешивания». Тем не менее, этот метод имеет оборотную сторону: в блокчейне создаётся большое количество транзакций с «пылью», то есть, транзакций с небольшими суммами, которые занимают непропорционально много места относительно своей важности. Кроме того, получатель монет может «отмести» пыль, когда захочет отправить их, что, вероятно, позволит злоумышленнику отследить, какие ключи были некоторым образом связаны. Более того, довольно просто установить верхнюю и нижнюю границы отправляемых сумм.

Другой оборотной стороной оригинального решения протокола CryptoNote является то, что оно требует использования в кольцевой подписи пары (P, A), состоящей из публичного ключа P и суммы A, наряду с другими публичными ключами с той же суммой. В случае с необычными суммами это будет означать наличие меньшего количества потенциальных пар (P', A') в блокчейне при $A' = A$ для соответствующей кольцевой подписи. Таким образом, в оригинальной версии протокола CryptoNote размер потенциальной анонимной группы будет меньше желаемого. Анализ вышеуказанного недостатка содержится в работе [9].

1.2 Применение Ring CT в Monero

Очевидным способом избежать недостатков протокола CryptoNote, описанных в предыдущем подразделе, является сокрытие суммы любой транзакции. В этой работе мною описывается изменение протокола Monero, криптовалюты, использующей алгоритм доказательства работы, расширяющий оригинальную версию CryptoNote, которое позволит скрывать суммы в проводимых транзакциях. В основе этого изменения лежат конфиденциальные транзакции, описанные в работе [11] и используемые в сайдчейне Elements криптовалюты Bitcoin. Отличие состоит в возможности их применения с кольцевыми подписями. Поэтому, применительно к Monero изменение носит название «кольцевые конфиденциальные транзакции» (Ring Confidential Transactions).

В целях сохранения свойства, не позволяющего тратить монеты дважды, мною приводится обобщение LSAG из работы [8], многоуровневая связываемая подпись спонтанной анонимной группы (MLSAG), позволяющая объединить схему конфиденциальных транзакций с кольцевой подписью таким образом, что становится возможным использование множества входов и выходов, при этом сохраняется анонимность, а двойная трата остаётся невозможной. Автор хотел бы отметить, что примерно через месяц после публикации второго черновика этой работы подобный протокол был предложен Коннором Френкенехтом (Connor Frenkenecht).

1.3 Строго децентрализованные схемы анонимных платежей

Протокол Ring CT позволяет скрывать суммы, источник и адресатов транзакций, подобно тому, как это делается в случае Zerocash [4]. Разница состоит в том, что в отличие от ZeroCash протокол Ring CT при создании монет позволяет использовать доказательство работы. ZeroCash предполагает предварительное генерирование всех монет доверенной группой.

Следует отметить, что самой большой инновацией Bitcoin [13] стала модель децентрализованного распределения, позволяющая любому желающему использовать свою вычислительную мощь для создания криптовалюты. Некоторые преимущества этого типа доказательства работы включают в себя не требующий доверия стимул, обеспечивающий безопасность и более высокий уровень децентрализации сети (например, защиту от атак методом «отравленной таблетки»).

Последним очевидным преимуществом создания монет с использованием доказательства работы является защита Ring CT от сильного злоумышленника, способного каким-то образом завладеть всеми частями главного ключа, необходимого для создания монет. Поскольку существует очевидный значительный стимул (способность генерировать свободные деньги [†]) собирать все части доверенного ключа генерации, это, безусловно, важно.

1.4 Благодарность

Мне хотелось бы поблагодарить команду Monero за помощь и обсуждение проблемы при создании этой работы, а также сообщества Monero и Bitcoin за поддержку и рассмотрение вопроса. Автором также было получено несколько пожертвований от сообщества Monero на общую сумму от 2 до 3 Bitcoin в благодарность за его работу над этим исследованием.

[†] Автор ранее допустил возможность «демаскировки» транзакций, но в более позднем документе ZeroCash утверждается, что это невозможно.

2 Многоуровневые связываемые подписи спонтанной анонимной группы

В этом разделе мною даётся определение многоуровневой связываемой подписи спонтанной анонимной группы (MLSAG), используемой протоколом Ring CT. Следует отметить, что я определяю эту подпись как общую, а не только относительно её применения в кольцевых конфиденциальных транзакциях. Подпись MLSAG во многом схожа с подписью LSAG, описанной в работе [8], но скорее является не подписью по набору, включающему в себя n ключей, а а подписью по набору, состоящему из n векторов ключей.

Определение 2.1. Вектор ключа представляет собой простой набор из $\overline{y} = (y_1, ..., y_r)$ публичных ключей с соответствующими приватными ключами $\overline{x} = (x_1, ..., x_r)$.

2.1 Сравнение LWW и FS подписей

Кольцевые подписи, используемые Monero и оригинальным протоколом CryptoNote, получены на основе отслеживаемых кольцевых подписей, предложенных в работе [6]. Кольцевые подписи CryptoNote [16] предполагают наличие «образа ключа», а это означает, что подписант сможет оставить в блокчейне только одну подпись с одной отдельно взятой парой, состоящей из публичного и приватного ключа, в противном случае его транзакция будет помечена как недействительная. Поэтому, CryptoNote использует одноразовые ключи, что также способствует анонимности.

В работе [3] Адам Бэк (Adam Back) отметил, что связываемые подписи спонтанной анонимной группы (LSAG), описанные в работе [8], могут быть изменены так, чтобы получилась более эффективная связываемая кольцевая подпись, которая бы работала так же, как кольцевые подписи, предложенные в работе [6]. Такое изменение позволило бы сократить место, необходимое для хранения в блокчейне вдвое.

Сначала я почти дословно приведу изменение, предложенное в работе [3]:

Keygen: Найти ряд публичных ключей $P_i, i = 0, 1, ..., n$ и такой секретный индекс j чтобы $xG = P_j$ где G является базовой точкой ed25519, а x является ключом траты подписанта. Допустим, $I = xH_p(P_j)$ где H_p является хеш-функцией, возвращающей точку [‡]

Допустим, $\mathfrak{m}$ является определённым сообщением.

SIGN: Допустим, $\alpha, s_i,\ i \neq j,\ i \in \{1, ..., n\}$ являются случайными значениями в $\mathbb{Z}_q$ (базовом поле ed25519).

Вычисляем

$$L_j = \alpha G$$

$$R_j = \alpha H_p(P_j)$$

$$c_{j+1} = h(\mathfrak{m}, L_j, R_j)$$

где h является хеш-функцией, возвращающей значение в $\mathbb{Z}_q$. Теперь, действуя последовательно в j по модулю n определяем

$$L_{j+1} = s_{j+1}G + c_{j+1}P_{j+1}$$

$$R_{j+1} = s_{j+1}H_p(P_{j+1}) + c_{j+1} \cdot I$$

[‡]На практике $H_p(P) = Keccak(P) \cdot G$ где G является базовой точкой ed25519, вместе с тем означает, что в схеме обязательства мною используется $toPoint(Keccak(P))$, производящая последовательное хеширование до тех пор, пока $Keccak(P)$ не возвратит кратное значение базовой точки.

$$c_{j+2} = h\left(\mathfrak{m},\ L_{j+1},\ R_{j+1}\right)$$

$$\dots$$

$$L_{j-1} = s_{j-1}G + c_{j-1}P_{j-1}$$

$$R_{j-1} = s_{j-1}H_p\left(P_{j-1}\right) + c_{j-1} \cdot I$$

$$c_j = h\left(\mathfrak{m}, L_{j-1},\ R_{j-1}\right)$$

таким образом, что определяются $c_1, ..., c_n$.

Допустим, $s_j = \alpha - c_j \cdot x_j \ mod\ l$, (где l имеет значение порядка кривой ed25519), следовательно $\alpha = s_j + c_j x_j \ mod\ l$ так, что

$$L_j = \alpha G = s_j G + c_j x_j G = s_j G + c_j P_j$$

$$R_j = \alpha H_p\left(P_j\right) = s_j H_p\left(P_j\right) + c_j I$$

и

$$c_{j+1} = h\left(\mathfrak{m},\ L_j,\ R_j\right)$$

и таким образом, при одном известном значении c_i и значениях P_j, образ ключа I и все значения s_j, все остальные $c_k,\ k \neq i$, могут быть восстановлены наблюдателем. Следовательно, подпись приобретает следующую форму:

$$\sigma = (I, c_1, s_1, ..., s_n)$$

что даёт большую экономию места, чем в [16, 4.4], где кольцевая подпись выглядела бы так:

$$\sigma = (I, c_1, ..., c_n, s_1, ..., s_n)$$

Верификация выполняется следующим образом. Наблюдатель вычисляет L_i, R_i, и c_i для всех i и проверяет равенство $c_{n+1} = c_1$. Затем верификатор проверяет, чтобы

$$c_{i+1} = h\left(\mathfrak{m}, L_i, R_i\right)$$

для всех i по модулю n

LINK: Подписи с дублирующими друг друга образами ключей I отклоняются.

Следует отметить, что доказательства невозможности подделки, анонимности и связываемости, используемые вышеуказанным протоколом, являются лишь незначительными модификациями доказательств, описанных в работе [8]. Мною будет приведена более обобщённая версия этих доказательств для MLSAG.

2.2 Описание MLSAG

В случае с протоколом Ring CT, о котором пойдёт речь в Разделе 4, мне потребовалась генерализация обратных LSAG-подписей, описанная в предыдущем разделе и позволяющая использовать векторы ключей (Определение 2.1), а не просто ключи.

Предположим, что каждый подписант (обобщённого) кольца, включающего в себя n участников, имеет точное количество m ключей $\left\{P_i^j\right\}_{j=1,...,m}^{i=1,...,n}$. Задача кольцевой MLSAG-подписи состоит в следующем:

- доказать, что одному из n подписантов известны секретные ключи ко всему вектору ключей;

- гарантировать, что в случае использования подписантом одного из m подписывающих ключей в другой MLSAG-подписи, два кольца будут связаны, и вторая такая MLSAG-подпись (прописанная в блокчейне) будет забракована.

Алгоритм выполняется следующим образом: допустим, $\mathfrak{m}$ является определённым сообщением. Допустим, π является секретным индексом, соответствующим подписанту созданного кольца. Для $j = 1, ..., m$, допустим, что $I_j = x_j H\left(P_\pi^j\right)$, а для $j = 1, ..., m$, $i = 1, ..., \hat{\pi}, ...n$ (где $\hat{\pi}$ означает опущение π) допустим, что s_i^j является некоторыми случайными скалярными величинами (элементами $\mathbb{Z}_q$). Теперь, аналогично тому, как это делалось в подразделе 2.1, определяем

$$L_\pi^j = \alpha_j G$$

$$R_\pi^j = \alpha_j H\left(P_\pi^j\right)$$

для случайных скалярных величин α_j и $j = 1, ..., m$. Теперь, также аналогично тому, как это делалось в подразделе 2.1, задаём:

$$c_{\pi+1} = H\left(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m\right).$$

$$L_{\pi+1}^j = s_{\pi+1}^j G + c_{\pi+1} P_{\pi+1}^j$$

$$R_{\pi+1}^j = s_{\pi+1}^j H\left(P_{\pi+1}^j\right) + c_{\pi+1} I_j$$

и повторяем это, увеличивая i по модулю n, пока не получим

$$L_{\pi-1}^j = s_{i-1}^j G + c_{i-1} P_{i-1}^j$$

$$R_{\pi-1}^j = s_{i-1}^j H\left(P_{i-1}^j\right) + c_{i-1} \cdot I_j$$

$$c_\pi = H\left(\mathfrak{m}, L_{\pi-1}^1, R_{\pi-1}^1, ..., L_{\pi-1}^m, R_{\pi-1}^m\right).$$

Наконец, находим решение для каждого s_π^j, используя $\alpha_j = s_\pi^j + c_\pi x_j \mod \ell$. Получаем следующую подпись: $\left(I_1, ..., I_m, c_1, s_1^1, ..., s_1^m, s_2^1, ..., s_2^m, ..., s_n^1, ..., s_n^m\right)$. Таким образом, сложность будет следующей: $O\left(m\left(n+1\right)\right)$. Верификация продолжается путём повторного генерирования L_i^j, R_i^j, начиная с $i = 1$ как это делалось в подразделе 2.1 (что является особым случаем при $m = 1$) и верифицируя хеш $c_{n+1} = c_1$. сли они использовались в блокчейне подобном Monero, подписи с образами ключей I_j, которые уже появлялись в нём, будут отклоняться. Это легко демонстрируется подобно тому, как это сделано в работе [8]:

- вероятность того, что подписант сгенерирует действительную подпись, не зная всё количество "m" приватных ключей, принадлежащее вектору ключей для индекса π ничтожна;
- вероятность того, что подписант не подпишется по какому-либо ключу с индексом π ничтожна (другими словами, образы ключей в подписи, все будут основаны на индексе π);
- если подписант подписывает два кольца, используя, по крайней мере, один из одинаковых публичных ключей, то два кольца связаны.

Я расширяю эти определения ниже, приводя доказательства безопасности.

2.3 Модель безопасности MLSAG

MLSAG соответствуют следующим трём свойствам: невозможности поделки, связываемости и неопределённости подписанта, что очень схоже с определениями, приводимыми в работе [8].

Определение 2.2. (невозможность подделки) Схему MLSAG-подписи невозможно подделать, даже в случае с вероятностным алгоритмом полиномиальной временной сложности (PPT) $\mathcal{A}$ с подписывающим оракулом $\mathcal{SO}$, выдающим действительные подписи, при наличии определённого списка n векторов публичных ключей, выбираемых $\mathcal{A}$, такой алгоритм $\mathcal{A}$ лишь с ничтожной вероятностью сможет создать действительную подпись, если $\mathcal{A}$ не будет известен один из соответствующих векторов приватных ключей.

Примечание 2.3. Следует отметить, что в следующее определение мною было включено отрицание возможности использования дублирующих друг друга образов ключей в качестве части критерия верификации MLSAG, что несколько отличает моё определение связываемости от того, что приводится в работе [8].

Определение 2.4. (связываемость) Допустим, L является набором публичных ключей в текущей структуре (например, в блокчейне). Схема MLSAG-подписи с L будет связной по образам ключей, если вероятность того, что злоумышленник, использующий PPT-алгоритм $\mathcal{A}$, сможет создать две подписи σ, σ' подписанные соответствующими векторами ключей $\overline{y}$ и $\overline{y}'$, каждый из которых будет содержать один и тот же публичный ключ $y_i = y'_i$ в L, и каждый будет верифицирован и не помечен как дублирующий, будет ничтожной.

Определение 2.5. (неопределённость подписанта) Считается, что схема MLSAG-подписи обеспечивает неопределённость подписанта, если в случае с любой верифицируемой подписью σ по векторам ключей $(\overline{y}_1, ..., \overline{y}_n)$ и любому набору из t приватных ключей, ни один из них не имеет ни одного одинакового индекса, и ни одного одинакового секретного индекса, и тогда вероятность угадывания секретного ключа составляет менее $\frac{1}{n-t} + \frac{1}{Q(k)}$.

Доказательства приведённых выше определений MLSAG-подписей содержатся в приложении.

3 История конфиденциальных транзакций

3.1 Конфиденциальные транзакции в Bitcoin

В своей работе [11] Грег Максвелл (Greg Maxwell) говорит о конфиденциальных транзакциях, как о способе проведения транзакций Bitcoin с сокрытием суммы. Основная идея состоит в использовании обязательства Педерсена в качестве, и сам метод хорошо описан в указанном источнике. В этой работе я немного изменяю рабочий механизм конфиденциальных транзакций и вместо того, чтобы использовать обязательства по сумме с нулевым разглашением, я подписываю обязательство, чтобы доказать, что мне известен приватный ключ. Более подробное описание приводится в следующем разделе.

3.2 Модификация для кольцевых подписей

Допустим, G является базовой точкой ed25519. Допустим [§]

$$H = toPoint\left(cn\_fast\_hash\left(G\right)\right)$$

[§] $H = MiniNero.getHForCT()$ с точки зрения кода рассматривается в работе [14]

Следует отметить, что не каждый хеш даёт точку в группе базовой точки (то есть, $H = \psi G$ для некоторой неизвестной ψ) (в этом состоит отличие от того, что происходит в случае с secp256k1, кривой, используемой Bitcoin). Тем не менее, кажется, выбор самой базовой точки сам по себе работает (ранее я использовал H(123456G), что касалось мне более безопасным, но использовать базовую точку более естественно). Выбор $H = \gamma G$ для некоторой неизвестной γ олжен обязательно быть таким, чтобы сохранялась вся обычная математика эллиптической кривой. Учитывая допуск дискретного логарифмирования ed25519, вероятность того, что злоумышленник найдёт γ, ничтожна. Находим $C(a, x) = xG + aH$, обязательство по значению a с маской x. Следует отметить, что поскольку $log_G H$ неизвестен, и если $a \neq 0$, то и $log_G C(a, x)$ будет неизвестен. С другой стороны, если $a = 0$, то $log_G C(a, x) = x$, и можно будет создать подпись, используя sk-pk пару ключей $(x, C(0, x))$.

В работе [11] используются обязательства по входу, обязательства по выходу, и сеть проверяет, чтобы

$$\sum Inputs = \sum Outputs.$$

Тем не менее, этого недостаточно в случае с Monero: так как определённая транзакция содержит множество возможных входов $P_i, i = 1, ..., n$, только один из которых принадлежит отправителю (см. [16, 4.4]), если мы сможем проверить приведённое выше равенство, сеть также сможет увидеть, какой P_i принадлежит отправителю транзакции. Это нежелательно, так как нейтрализуется анонимность, обеспечиваемая кольцевыми подписями. Поэтому, вместо этого используются обязательства по входам и выходам, которые создаются следующим образом (сначала предположим, что имеется только один вход):

$$C_{in} = x_c G + aH$$
$$C_{out-1} = y_1 G + b_1 H$$
$$C_{out-2} = y_2 G + b_2 H$$

так, чтобы $x_c = y_1 + y_2 + z$, $x_c - y_1 - y_2 = z$, y_i являлись значениями маски, $z > 0$ и $a = b_1 + b_2$. В данном случае x_c является специальным приватным ключом, «ключом суммы», известным только отправителю, а также лицу, которому отправляются монеты, и он должен отличаться от обычного приватного ключа. В этом случае

$$C_{in} - \sum_{i=1}^{2} C_{out-i}$$
$$= x_c G + aH - y_1 G - b_1 H - y_2 G - b_2 H$$
$$= zG.$$

Таким образом, приведённое выше суммирование становится обязательством по 0, где $sk = z$, а $pk = zG$, а не фактическим уравнением с суммой, равной нулю. Следует отметить, что z не может быть вычислено источником монет x_c, если только ему не будут известны y_1, y_2, но даже этого можно избежать, просто включив дополнительный адрес для сдачи (обычно второе обязательство, использующее y_2 в качестве маски, отправляется себе как сдача).

Так как нежелательно демонстрировать принадлежность конкретного входа отправителю, создаётся кольцевая подпись, состоящая из всех обязательств по входам $C_i, i = 1, ..., s, ..., n$ (где s является секретным индексом обязательства отправителя), с добавлением соответствующего публичного ключа (чтобы обязательства и публичные ключи были объединены в пары (C_i, P_i) и их можно было бы отправить только вместе), а также вычитания $\sum C_{out}$:

$$\left\{ P_1 + C_{1,in} - \sum_j C_{j,out}, ..., P_s + C_{s,in} - \sum_j C_{j,out}, ..., P_n + C_{n,in} - \sum_j C_{j,out} \right\}.$$

Эта кольцевая подпись может быть подписана, так как нам известен один из приватных ключей (а именно $z + x'$ с z, как указано выше, и $x'G = P_s$). По сути, поскольку нам для каждого i известен как приватный ключ для P_i, так и приватный ключ для $P_i + C_{i,in} - \sum_j C_{j,out}$, мы можем создать подпись так же, как это было сделано в подразделе 2.2. Подробности содержатся в Определении 4.1.

Как было отмечено в работе [11], важно доказать, что все суммы выходов[¶] $b_1, ...b_n$ находятся в диапазоне положительных значений, например, $(0, 2^{16})$. Это делается в значительной степени так же, как описано в работе [11] и более подробно в Разделе 5.

Наконец, следует отметить, что выше мною никак не упоминалось свойство связываемости по тегам, которое используется Monero и CryptoNote во избежание двойной траты. Свойство связываемости по тегам в нашем случае приводит к объединению тому, что обсуждалось выше, с MLSAG-подписями, как они описаны в Определении 4.1.

4 Ring CT и протокол Monero

4.1 Описание протокола

Определение 4.1. (Связываемые по тегам Ring-CT со множеством входов и одноразовыми ключами)

- Допустим, $\left\{\left(P_\pi^1, C_\pi^1\right), ..., \left(P_\pi^m, C_\pi^m\right)\right\}$ является набором адресов/обязательств с соответствующими секретными ключами $x_j,\ j = 1, ..., m$.
- Находим $q + 1$ наборов $\left\{\left(P_i^1, C_i^1\right), ..., \left(P_i^m, C_i^m\right)\right\}, i = 1, ..., q + 1$, которые уже не связываемы по тегам так, как это описано в работе [6, стр. 6].
- Выбираем набор адресов выходов $(Q_i, C_{i,out})$ так, чтобы $\sum_{j=1}^m C_\pi^j - \sum_i C_{i,out}$ была обязательством по нулю.
- Допустим
$$\mathfrak{R} := \left\{\left\{\left(P_1^1, C_1^1\right), ..., \left(P_1^m, C_1^m\right), \left(\sum_j P_1^j + \sum_{j=1}^m C_1^j - \sum_i C_{i,out}\right)\right\},\right.$$
$$...,$$
$$\left.\left\{\left(P_{q+1}^1, C_{q+1}^1\right), ..., \left(P_{q+1}^m, C_{q+1}^m\right), \left(\sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_i C_{i,out}\right)\right\}\right\}.$$
является обобщённым кольцом, которое мы хотим подписать. Следует отметить, что последний столбец является кольцом Ring-CT, как это описано в Разделе 4.
- Вычисляем MLSAG-подпись Σ по $\mathfrak{R}$.

В этом случае, согласно Теореме A.2, $P_\pi^j, j = 1, ..., m$ не может являться подписантом какой-либо дополнительной несвязанной кольцевой подписи в данном поднаборе $\mathcal{P}$ всех таких пар $\mathcal{P} = \{(P, C)\}$ после подписания Σ.

Примечание 4.2. Пространственная сложность указанного выше протокола. Необходимо отметить, что размер подписи Σ по $\mathfrak{R}$ в соответствии с Определением 4.1 при $m > 1$ 1 фактически меньше, чем в случае с используемой в настоящее время кольцевой подписью CryptoNote [16], основанной на

[¶]Так как обязательства по входам потенциально могут быть просто унаследованы из предшествующей транзакции, этого достаточно, чтобы учитывать суммы выходов.

транзакции, включающей в себя множество входов. Причина состоит в улучшении размера, предлагаемого в работе [8] для каждого столбца. Также следует отметить вероятное отсутствие необходимости во включении образа ключа вводимых данных обязательства в случае с вышеуказанной подписью. Кроме того, вполне возможна дальнейшая оптимизация размера.

4.2 Преобразование видимого номинала в обязательства

Поскольку в настоящее время Monero использует в блокчейне видимые скалярные величины, важная задача состоит в преобразовании видимых сумм в обязательства с сохранением анонимности. По сути, это сделать несложно. При наличии пары (P, a), где P является публичным ключом, а a обозначает сумму, её можно использовать в качестве входа транзакции в виде (P, aH), а верификатор должен проверить, чтобы сумма входа a, умноженная на точку маскировки H, на самом деле составляла aH. Таким образом, на первом этапе суммы входов не будут скрыты, но выходы транзакций скрыть можно, и все необходимые отношения, о которых говорится в Разделе 4, сохраняются. Необходимо отметить, что необходимость в доказательстве диапазона в случае с таким входом отсутствует.

Примечание 4.3. Очевидным преимуществом данного метода преобразования видимых сумм в обязательства является то, что сумма монет, созданных в процессе майнинга, может быть верифицирована на доверительной основе. Это преимущество протокола Ring CT относительно платёжных схем, подобных описанной в работе [4], в основе которых лежит этап доверительной настройки.

4.3 Комиссии за проведение транзакций

Поскольку Monero является строго децентрализованной криптовалютой (то есть, использует алгоритм доказательства работы), майнеры должны получать комиссию с каждой транзакции. Это способствует безопасности сети, предотвращая раздувание блокчейна. Эти комиссии должны выплачиваться «в открытую», то есть, просто как bH, а не как $xG + bH$, и с какой-то стандартной суммы b, чтобы майнер мог проверить, что $b \cdot H = bH$, а следовательно, имеется достаточное количество денег для уплаты такой комиссии, но с сохранением уравнений, касающихся H, чтобы необходимые отношения, описанные в Разделе 4, также сохранялись.

4.4 Кольцевая мультиподпись

Необходимо отметить, что реализация простой версии кольцевой мультиподписи t из m из n также возможна и в случае с MLSAG-подписями. Это позволяет группе, состоящей из m участников, создавать мультиподпись, которую должно будет подписать t из m участников, чтобы она была принята как действительная, и кроме того, подписант остаётся неопределённым, в силу сокрытия того, какое количество m из n ключей принадлежит участникам.

- n участников мультиподписи создаёт общий секрет x_e и публичный ключ P_e и совместно использует образы ключей мультиподписи
$$I_j = x_e H(P_e|P_j)$$
где P_j является публичным ключом участника.

- Любой участник мультиподписи выбирает $n - m$ ополнительных публичных ключей в блокчейне и создаёт MLSAG-подпись с первым рядом, состоящим из n ключей, и вторым рядом, в котором каждый элемент содержит общий ключ P_e.

- Каждый подписант, передающий часть мультиподписи, обеспечивает начальное $I_j,\ j = 1, ..., m$ и подпись принимается после того, как t верифицированных подписей будет передано в блокчейн, и каждая из них будет соответствовать образам ключей I_j.

5 Агрегированные доказательства диапазона Шнорра

В работе [11] конфиденциальные транзакции без кольцевых подписей всё же используют определённый тип кольцевой подписи, основанный на том, что было описано в работе [1], под названием кольцевой подписи Борромео. Этот тип подписи помогает доказать, что значение суммы, по которой даётся обязательство, находится в пределах определённого диапазона. В этой статье мною предлагается альтернативный метод, вдохновением для которого послужила работа [7], позволяющий в той же мере сэкономить место, но, возможно, имеет более простые доказательства безопасности. В данном случае имелась следующая мотивация: предположим, что в определённой транзакции имеются обязательства по входам

$$C_{in} = a_{in}G + 10H$$

и обязательства по выходам

$$C_{out,1} = a_{out,1}G + 5H,\ C_{out,2} = a_{out,2}G + 5H$$

и этот сценарий является действительным, так как есть возможность подписания по

$$C_{in} - C_{out,1} - C_{out,2} = (a_{in} - a_{out,1} - a_{out,2})\,G$$

Тем не менее, следует отметить, что (при отсутствии доказательств диапазона) существует возможность альтернативного определения обязательств по выходам

$$C_{out,1} = a_{out,1}G - H,\ C_{out,2} = a_{out,2}G + 11H$$

так как -1 является довольно большим модулем числа порядка группы кривой, создаются свободные деньги. Следовательно, необходимо доказать, что $C_{out,i}$ являются обязательствами по положительным значениям, лежащим в пределах ограниченного диапазона $[0, 2^n]$ для некоторого n. Для этого необходимо разбить каждое значение выхода на двоичные значения:

$$b = b_0 2^0 + b_1 2^1 + b_2 2^2 + \cdots + b_n 2^n$$

и вычислить обязательства $C^j_{out,i}$ для $b_j \cdot 2^j$ и так, чтобы

$$C^1_{out,i} + C^2_{out,i} + \cdots + C^n_{out,i} = C_{out,i}$$

Наконец, используя секретный ключ b_j можно вычислить кольцевую подпись по

$$(C^j_{out,i}, C^j_{out,i} - 2^j H)$$

для всех j и предоставить $C^j_{out,i}$ верифицирующим сторонам (в данном случае майнерам).

В целях экономия места мы можем либо использовать кольцевую подпись Борромео (как предлагается в работе [11]), чтобы объединить все эти простые кольцевые подписи, либо некую агрегированную кольцевую подпись, определяемую следующим образом:

5.0.1 Создание агрегированной несвязываемой кольцевой подписи Шнорра (ASNL)

Допустим, (x_i^j, P_1^j, P_2^j) является набором ключей, $j = 1, ..., n$, где x_i^j является секретным ключом P_i^j

- Для каждого j допустим, что $i' := i + 1 \ mod \ 2$, присваиваем α_j случайную скалярную величину и вычисляем $L_i^j = \alpha_j G$.
- Задаём $c_{i'}^j = H_s(L_i^j)$, где H_s является криптографической хеш-функцией, возвращающей скалярную величину, и, случайным образом выбрав $s_{i'}^j$, вычисляем

$$L_{i'}^j = s_{i'} G + c_{i'} H.$$

- Задаём $c_i = H_s(L_{i'}^j)$ и вычисляем

$$s_i^j = \alpha - c_i^j x_i \ mod \ \ell$$

- Получаем (L_1^j, s_2^j) для всех j и $s = \sum_j s_1^j$.

5.0.2 Верификация агрегированной несвязываемой кольцевой подписи Шнорра (ASNL)

Берём $(P_1^j, P_2^j, L_1^j, s_2^j)$ для $j = 1, ..., n$ и s.

- Для всех j вычисляем $c_2^j = H_s(L_1^j)$, $L_2^j = s_2^j G + c_2^j H$ и $c_1^j = H_s(L_2^j)$.
- Если $\sum_{j=1}^n L_1^j = sG + (c_1^j + \cdots + c_n^j)H$, в случае действительной подписи получаем 0. В противном случае получаем -1.

Теорема 5.1. Агрегированную несвязываемую кольцевую подпись Шнорра нельзя поделать согласно допуску дискретного логарифмирования.

Доказательство. Мною приводится доказательство для случая, в котором $n = 2$. Общий случай будет подобным. Предположим, злоумышленник $\mathcal{A}$ может подделать ASNL-подпись по

$$\left\{(x_i^1, P_1^1, P_2^1), (x_i^2, P_1^2, P_2^2)\right\}$$

с незначительной вероятностью, если ему известен, по крайней мере, один из x_i^j (не в ущерб общности предположим, что $\mathcal{A}$ известен x_i^1). Для любой такой подделки:

$$\{s, (P_1^j, P_2^j, L_1^j, s_2^j)\}, \ j = 1, 2,$$

я с незначительной вероятностью решаю дискретный логарифм P_1^2. В соответствии с алгоритмом верификации, допустим, $c_1^j = H_s\left(s_2^j G + H_s(L_1^j)H\right)$. В таком случае, действительным будет

$$L_1^1 + L_1^2 = sG + \left(c_1^1 P_1^1 + c_1^2 P_1^2\right).$$

Предположим, что $L_1^1 = aG$ and $L_1^2 = bG$, где a и b известны $\mathcal{A}$, тогда

$$aG + bG - sG - c_1^1 P_1^1 = c_1^2 P_1^2$$

Таким образом, поскольку c_1^2 определяется протоколом верификации, это тот случай, когда $\mathcal{A}$ известен приватный ключ P_1^2,

$$x_1^2 := \frac{a + b - s - c_1^1 x_1^1}{c_1^2} \mod \ell$$

Это противоречит допуску дискретного логарифмирования для данной группы.

5.1 Представление суммы

При использовании протокола Ring CT представление суммы происходит подобно тому, как это описано в работе [11], но с небольшим изменением, необходимым в случае с Monero. Монеты CryptoNote организуют монеты с определённым номиналом в базы данных индексов, чтобы их можно было использовать в порядке их появления в качестве входов в кольцевых подписях. Поскольку все значения представлены в виде 64-битных целых чисел без знака, мы можем использовать доказательство диапазона по всему реестру для шифровки суммы в выходе, передаваемом получателю. Благоприятно то, что это также позволяет смешивать все выходы друг с другом. Таким образом, для представления выходов требуется только один индекс. Размер доказательства диапазона в этом случае составляет приблизительно 5 килобайт. Кольцевые подписи со входами при этом становятся более устойчивыми к анализу, так как анонимная группа значительно расширяется.

Несмотря на большой размер доказательств диапазона, применение схемы хеширования транзакций, используемой для подписи «префикса» транзакции (индексов входов и выходов) и сохранения доказательства диапазона, наряду с кольцевыми подписями также позволяет сэкономить довольно много места при синхронизации или при выполнении роли SPV-узла. Подобно тому, как это делается в случае с элементами сайдчейна [12], мы сохраняем $H(H(prefix)||H(signatures))$ в дереве Меркла. Затем, узлам, синхронизирующимся и доверяющим контрольной точке, необходимо только загрузить префикс и хеш подписей, что значительно сокращает объём пропускной способности, необходимой для синхронизации. Так как префикс транзакции использует варинт для выходов, на которые он ссылается в кольцевых подписях и выводит средства по отдельным публичным ключам, они предельно малы, если сравнивать с данными кольцевых подписей во входах и доказательствах диапазона. В дальнейшем некоторые узлы смогут даже вырезать кольцевые подписи из баз данных в целях экономии места, а вместо них хранить сравнительно небольшой набор непотраченных выходов транзакций (UTXO).

5.1.1 Передача суммы получателю

Теперь, когда есть сумма в выходе, $b = b_0 2^0 + b_1 2^1 + \cdots b_n 2^n$, отправитель вычисляет новую пару приватных/публичных ключей и соответствующий общий ECDH секрет ss и создаёт следующую информацию, которая будет содержаться в его транзакции:

- $C_j = a_j G + (b_j 2^j)H$, где a_i является некоторыми случайными числами для $j = 0, ..., n$.
- Данные $\left\{(L_1^i, s_2^j), s\right\}$.
- Публичный ключ ECDH и $a + ss \ mod \ \ell$, где $a = a_0 + \cdots + a_n$.

Затем, получатель:

- Вычисляет общий секрет ss и на основе a вычисляет $a + ss \ mod \ \ell$.
- Вычисляет $C = \sum C_i$, вычисляет $C - aG = bH$ и путём сравнения со всеми bH в заданном диапазоне $[0, 2^n]$ находит b. (На практике это будет быстрый поиск на 500 килобайт при $n = 14$, как в предыдущем разделе. Если же в качестве верхнего предела будет выбрано значение 2^{32}, как в работе [11], поиск станет сложным с вычислительной точки зрения).

6 Заключение

Протокол кольцевых конфиденциальных транзакций обеспечивает строгую децентрализацию криптовалюты (то есть, отсутствие привилегированных сторон), что даёт возможность доказуемой оценки

безопасности с точки зрения сокрытия сумм, источника и адресатов транзакций. Кроме того, создание монет согласно протоколу кольцевых конфиденциальных транзакций не требует доверия и является верифицируемо безопасным. Соблюдение этих пяти факторов необходимо в случае с подобными наличным деньгам криптовалютам, таким как Monero.

Список литературы

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys (Подписи, построенные по схеме «1 из n», на основе ряда ключей). *Advances in Cryptology?Asiacrypt 2002*, pages 415–432, 2002.

[2] Adam Back. Bitcoins with homomorphic value (validatable but encrypted) (Bitcoin с гомоморфного достоинства (с возможностью валидации, но зашифрованный)). `https://bitcointalk.org/index.php?topic=305791.0`, 2013. [Online; accessed 1-May-2015].

[3] Adam Back. Ring signature efficiency (Эффективность кольцевых подписей). `https://bitcointalk.org/index.php?topic=972541.msg10619684#msg10619684`, 2015. [Online; accessed 1-May-2015].

[4] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin (zerocash: децентрализованные анонимные платежи на базе bitcoin). In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014.

[5] Evan Duffield and Kyle Hagan. Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system (darkcoin: Одноранговая криптовалюта с возможностью проведения анонимных транзакций в блокчейне и улучшенной системой доказательства работы). 2014.

[6] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature (Отслеживание кольцевых подписей). In *Public Key Cryptography–PKC 2007*, pages 181–200. Springer, 2007.

[7] Javier Herranz. Aggregate signatures (Агрегированные подписи). `http://www.iiia.csic.es/~jherranz/papers/Nijmegen_seminar_aggregate.pdf`, oct 2005.

[8] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups (Связываемая подпись спонтанной анонимной группы для специальных групп). In *Information Security and Privacy*, pages 325–335. Springer, 2004.

[9] Adam Mackenzie, Surae Noether, and Monero Core Team (Повышение уровня обфускации протокола CryptoNote). Improving obfuscation in the cryptonote protocol. `"https://lab.getmonero.org/pubs/MRL-0004.pdf"`, January 2015.

[10] Greg Maxwell. Coinjoin: Bitcoin privacy for the real world (Coinjoin: анонимность bitcoin в реальном мире), Август 2013. Bitcoin Forum. `https://bitcointalk.org/index.php?topic=279249.0`, 2013. [Online; accessed 1-July-2015].

[11] Greg Maxwell. Confidential Transactions (Конфиденциальные транзакции). `https://people.xiph.org/~greg/confidential_values.txt`, 2015. [Online; accessed 1-June-2015].

[12] Greg Maxwell. Elements project (Проект elements). `https://github.com/ShenNoether/MiniNero`, 2015.

[13] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system (Bitcoin: одноранговая электронная денежная система). *Consulted*, 1(2012):28, 2008.

[14] Shen Noether. Mininero. `https://github.com/ShenNoether/MiniNero`, 2015.

[15] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret (Как украсть секрет). In *Advances in Cryptology (Прогресс в криптологии)???ASIACRYPT 2001*, pages 552–565. Springer, 2001.

[16] Nicolas van Saberhagen. Cryptonote v 2. 0. *HYPERLINK https:// cryptonote. org/ whitepaper. pdf* , 2013.

A Приложение. Доказательства безопасности

A.1 Невозможность подделки MLSAG

Невозможность подделки доказывается подобно тому, как это делается в работе [8, Теорема 1]. Допустим, H_1 и H_2 являются случайными оракулами, а $\mathcal{SO}$ является подписывающим оракулом, который выдаёт действительные MLSAG-подписи. Предположим, существует вредоносный алгоритм полиномиальной временной сложности (PPT) $\mathcal{A}$, с не ничтожной вероятностью способный подделать MLSAG на основе списка векторов ключей L

$$Pr\left(\mathcal{A}\left(\mathcal{L}\right) \rightarrow (m,\sigma) : Ver\left(L,m,\sigma\right) = True\right) > \frac{1}{Q_1\left(k\right)}$$

где Q_1 является полиномиальным вводом параметра безопасности k, и где (m,σ) не является одной из подписей, возвращаемой $\mathcal{SO}$. Предположим, что $\mathcal{A}$ делает не более $q_H + nq_S$ (где n является количеством ключей в $\mathcal{L}$) запросов подписывающим оракулам H_1, H_2 и $\mathcal{SO}$ соответственно. Предполагается, что оракулы H_1 и H_2 являются независимыми и случайными, а также соответствуют дублирующим запросам. Подписывающий оракул $\mathcal{SO}$ также может запрашивать H_1 и H_2. При наличии $\mathcal{A}$ я продемонстрирую возможность создания вредоносного PPT-алгоритма $\mathcal{M}$, который будет использовать $\mathcal{A}$ для вычисления дискретного логарифма одного из ключей в $\mathcal{L}$.

Если L является набором векторов $\{\overline{y_1}, ..., \overline{y_n}\}$, каждый из которых имеет размер r, (то есть, $\overline{y_i} = \left(y_1^i, ..., y_r^i\right)$ где $y_1, ..., y_r$ являются публичными ключами), то подделанная подпись

$$\sigma = (c_1, s_1, ..., s_n, y_0)$$

созданная $\mathcal{A}$, должна соответствовать

$$c_{i+1} = H\left(\mathfrak{m}, L_i^1, R_i^1, ..., L_i^m, R_i^m\right)$$

где i выбираются по модулю n, а L_i^j и R_i^j определяются так же, как описано в подразделе 2.2. Новый злоумышленник $\mathcal{M}$ может запросить $\mathcal{A}$ подделать подписи полиномиальное количество раз, используя Turing script $\mathcal{T}$, независимо от того, была или нет попытка подделки успешной.

Лемма A.1. [8, Лемма 1] Допустим, $\mathcal{M}$ задействует $\mathcal{A}$ для получения транскрипта $\mathcal{T}$. Если получение $\mathcal{T}$ будет успешным, $\mathcal{M}$, $\mathcal{T}$ перейдёт к заголовку H и перемоделирует $\mathcal{A}$, чтобы получить $\mathcal{T}'$. Если $Pr\left(\mathcal{T}\text{ успешен}\right) = \epsilon$, значит и $Pr\left(\mathcal{T}'\text{успешен}\right) = \epsilon$.

Доказательство вытекает из указанной Теоремы.

Теорема А.2. При наличии допуска дискретного логарифмирования вероятность того, что $\mathcal{A}$ создаст верифицируемую поддельную MLSAG-подпись, ничтожна.

Доказательство. Я следую системе обозначений, использованной выше. Подобно тому, как это изложено в работе [8, Теорема 1], поскольку вероятность того, что выход случайного оракула будет угадан, ничтожна, для каждой успешной подделки $\mathcal{A}$ получает транскрипт $\mathcal{T}$, H_1 делается $m_{\mathcal{T}}$ запросов, соответствующих n запросам, используемым для верификации подписи. Следовательно, допустим, что $X_{i_1}, ..., X_{i_m}$ обозначают эти запросы, используемые при верификации для такой i подделки, и допустим, что π является индексом, соответствующим такому последнему запросу верификации для данной поделки

$$X_{i_m} = H_1\left(m, L_{\pi-1}^1, R_{\pi-1}^1, ..., L_{\pi-1}^{m_{\mathcal{T}}}, R_{\pi-1}^{m_{\mathcal{T}}}\right).$$

(Интуитивно, π соответствует секретному индексу подделанной подписи, так как соответствует последнему запросу, переданному случайному оракулу по данной подписи).

Попытка подделки σ, сделанная $\mathcal{A}$, является (ℓ, π)-подделкой если $i_1 = \ell$ и π такие же, как указано выше (таким образом, эта подделка соответствует запросам ℓ через $\ell + \pi$). Согласно допуску, существует такая пара (ℓ, π), что вероятность создания транскриптом $\mathcal{T}$ удачной подделки, $\epsilon_{\ell,\pi}(\mathcal{T})$, будет соответствовать условию

$$\epsilon_{\ell,\pi} \geq \frac{1}{m_{\mathcal{T}}\left(q_H + m_{\mathcal{T}} q_S\right)} \cdot \frac{1}{Q_1(k)} \geq \frac{1}{n\left(q_H + n q_S\right)} \cdot \frac{1}{Q_1(k)}.$$

Теперь возвращаемся к началу $\mathcal{T}$ в место перед ℓ^{th}запросом и снова пытаемся создать подделку по тому же набору ключей (и позволяем H_1 вычислить новые случайные варианты для всех последующих запросов), следовательно, согласно Лемме А.1, вероятность того, что $\mathcal{T}'$ также приведёт к удачной подделке, соответствует условию

$$\epsilon_{\ell,\pi}(\mathcal{T}') \geq \frac{1}{n\left(q_H + n q_S\right)} \cdot \frac{1}{Q_1(k)}.$$

Следовательно, вероятность того, что $\mathcal{T}$, так и $\mathcal{T}'$ соответствуют верифицируемым поделкам σ и σ' не ничтожна:

$$\epsilon_{l,\pi}(\mathcal{T} \; and \; \mathcal{T}') \geq \left(\epsilon_{l,\pi}(\mathcal{T})\right)^2.$$

Поскольку новые случайные варианты были вычислены для выходов случайного оракула для H_1, , существует огромная вероятность того, что есть такое j, что $s_\pi^j \neq s_\pi'^j$ и $c_\pi \neq c_{\pi+1}$. Таким образом, мы можем найти решение для приватного ключа по индексу π:

$$x_\pi^j = \frac{s_\pi'^j - s_\pi^j}{c_\pi - c_\pi'} \; mod \; q$$

что противоречит допуску дискретного логарифмирования.

А.2 Связываемость MLSAG

Теорема А.3. (Связываемость образов ключей). Вероятность того, что PPT-алгоритм злоумышленника $\mathcal{A}$ сможет создать такие две верифицируемые (и несвязываемые при заданных параметрах) подписи σ, σ', подписанные согласно векторов ключей $\overline{y}$ и $\overline{y}'$, соответственно, и что в обоих $\overline{y}$ и $\overline{y}'$ будет присутствовать публичный ключ y, ничтожна.

Доказательство. Предположим, наоборот, что $\mathcal{A}$ создал две верифицируемые подписи σ и σ', каждая из которых подписана по векторам ключей $\overline{y}$ и $\overline{y}'$, соответственно, таким образом, что существует публичный ключ y, как в $\overline{y}$, так и в $\overline{y}'$. Допустим, y появляется в качестве элемента j из $\overline{y}$, а также j', в качестве элемента $\overline{y}'$. Согласно Теореме A.2, существует большая вероятность существования таких индексов π и π' для публичных ключей в σ и σ', соответственно, что

$$L_\pi^j = s_\pi^j G + c_\pi y_\pi^j$$

$$R_\pi^j = s_\pi^j H\left(y_\pi^j\right) + c_\pi I_j$$

и

$$L_{\pi'}^{j'} = s_{\pi'}^{j'} G + c_{\pi'} y_{\pi'}^{j'}$$

$$R_{\pi'}^{j'} = s_{\pi'}^{j'} H\left(y_{\pi'}^{j'}\right) + c_{\pi'} I_{j'}$$

с

$$log_G L_\pi^j = log_{H\left(y_\pi^j\right)} R_\pi^j$$

и

$$log_G L_{\pi'}^{j'} = log_{H\left(y_{\pi'}^{j'}\right)} R_{\pi'}^{j'}$$

При допуске, что x обозначает приватный ключ y, $y = xG$, после решения вышеуказанных уравнений для I_j и $I_{j'}$, следует, что $I_j = xH\left(y_\pi^j\right) = xH\left(y\right)$ и подобным образом $I_{j'} = xH\left(y\right)$. Таким образом, две подписи включают в себя $I_j = I_{j'}$, а следовательно, так как дублирующие друг друга образы ключей отклоняются, один из них не будет верифицирован.

A.3 Анонимность MLSAG

Чтобы доказать анонимность, обеспечиваемую вышеуказанным протоколом в рамках модели случайного оракула, допустим, что H_1 и H_2 являются случайными оракулами, моделирующими дискретные хеш-функции. Допустим, $\mathcal{A}$ направлен на нейтрализацию анонимности. Я создаю злоумышленника $\mathcal{M}$ против допуска задачи решения Диффи-Хеллмана (DDH) следующим образом. Допуск DDH предполагает, что при наличии набора элементов $(G, aG, bG, \gamma G)$ вероятность определения того, что $\gamma G = abG$, ничтожна.

Теорема A.4. Протокол Ring CT обеспечивает сокрытие подписанта согласно допуску задачи решения Диффи-Хеллмана.

Доказательство. (Подобно доказательству, приведённому в работе [8, Теорема 2]). Предположим, что задача решения Дифии-Хеллмана сложна в случае циклической группой, созданной G, и предположим, что существует некоторый PPT-алгоритм, используемый злоумышленником $\mathcal{A}$, направленный на раскрытие подписанта. Таким образом, при наличии списка L, в который входит n векторов публичных ключей длиной m, набора, состоящего из t приватных ключей $\mathcal{D}_t = \{x_1, ..., x_t\}$, действительная подпись σ по L, подписанная пользователем по вектору ключа $\overline{y}$ так, что соответствующий вектор приватного ключа $\overline{x} = (x_1^\pi, ..., x_m^\pi)$ соответствует $x_j^\pi \notin \mathcal{D}_t$, $\mathcal{A}$ сможет вычислить π с вероятностью

$$Pr\left(\mathcal{A} \to \pi\right) > \frac{1}{n-t} + \frac{1}{Q\left(k\right)}$$

для некоторого полиномиального $Q\left(k\right)$. Мною строится PPT-алгоритм $\mathcal{M}$, который в качестве входа берёт набор элементов (G, aG, bG, c_iG), в котором значение $i \in \{0, 1\}$ выбирается случайным образом

(и не известно $\mathcal{M}$ заранее), $c_1 = ab$ и c_0 является случайной скалярной величиной, и выводит i с вероятностью

$$Pr\left(\mathcal{M}\left(G, aG, bG, c_iG\right) \to i\right) \geq \frac{1}{2} + \frac{1}{Q_2\left(k\right)}$$

для некоторого полиномиального $Q_2\left(k\right)$.

Рассмотрим алгоритм SIMNIZKP (подобный тому, определение которому даётся в работе [6]), который в качестве входа берёт скалярные величины a и c, вектор приватного ключа $\overline{x}$, набор векторов публичных ключей $\overline{y}_i, i = 1, ..., m$, индекс π, сообщение $\mathfrak{m}$ и использует их следующим образом:

1. Генерирует случайные скалярные величины $s_1, ..., s_m$ и случайную скалярную величину $c_\pi \leftarrow H$.

2. Для j индексирование $\overline{x}$ задаёт

$$L_\pi^1 = aG$$

$$R_\pi^1 = cG$$

и для всех остальных j

$$L_\pi^j = s_\pi^j G + c_\pi y_\pi^j$$

$$R_\pi^j = s_\pi^j H\left(y_\pi^j\right) + c_\pi x^j H\left(y_\pi^j\right)$$

3. На основе случайного оракула вычисляет случайный выход

$$c_{\pi+1} \leftarrow H\left(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m\right).$$

4. Для каждого i, по модулю m, вычисляет

$$L_i^j = s_i^j G + c_i y_\pi^j$$

$$R_i^j = s_i^j H\left(y_i^j\right) + c_i x^j H\left(y_i^j\right)$$

$$c_{i+1} \leftarrow H\left(\mathfrak{m}, L_i^1, R_i^1, ..., L_i^m + R_i^m\right).$$

и следует отметить, что, по крайней мере, на последнем этапе, где $i = \pi - 1$, значение c_{i+1} уже определено, что обеспечивает соответствие выходу случайного оракула.

Также необходимо отметить, что независимо от того, является ли $\overline{x}$ фактическим приватным ключом, соответствующим $\overline{y}$, из-за того, что в последующих вызовах соответствие поддерживается случайными оракулами, вышеуказанная подпись является верифицируемой. Если $\overline{x}$ действительно является вектором приватного ключа $\overline{y}$, нет никакой разницы между SIMNIZKP и подлинной подписью.

Наконец, при наличии набора элементов (G, aG, bG, c_iG), в котором a и b являются случайно выбранными скалярными величинами, $c_1 = ab$, c_0 является случайным элементом, $i \in \{0, 1\}$, $\mathcal{M}$ для решения задачи Диффи-Хеллмана с не незначительной вероятностью выполняет следующие шаги. $\mathcal{M}$ берёт у случайного оракула случайное $\gamma \leftarrow H$ и берёт пару векторов приватного/публичного ключа $(\overline{x}, \overline{y})$, а затем вычисляет s так, чтобы $a = s + \gamma x$. Затем, $\mathcal{M}$ выполняет алгоритм SIMNIZKP, используя произвольно выбранные векторы ключей $\{\overline{y_i}\}_{i=1,...,n}$ так, чтобы $\overline{y} = \overline{y}_\pi$, $a \to a$, $c_i \to c$ некоторое сообщение $\mathfrak{m}$ и $\overline{x} \to \overline{x}$.

В том случае, если $i = 1$, то $c = ab$ и

$$log_G aG = log_{bG} cG = a$$

и поскольку предполагается, что $\mathcal{A}$ с не ничтожной вероятностью способен найти π , существует и не ничтожная вероятность более $\frac{1}{2}$, что $\mathcal{A}$ выдаст 1 (по которой $\mathcal{M}$ выдаст 1). Если $i = 0$, то $\mathcal{A}$ выдаёт 1 только с вероятностью $\frac{1}{2}$, и с не незначительной вероятностью более $\frac{1}{2}$, $\mathcal{M}$выдаёт то же значение, что и $\mathcal{A}$, и таким образом решает задачу Диффи-Хеллмана для случайно выбранных скалярных величин с не незначительной вероятностью более $\frac{1}{2}$, что является противоречием.

B Пример кода Ring CT

В репозитории [14] мною был опубликован простой пример работы протокола кольцевых конфиденциальных транзакций с применением MLSAG-подписей, описанных в Разделе 2, а также ASNL-подписей, о которых говорится в Разделе 5:

```
H_ct = RingCT.getHForCT()
print("H", H_ct)
sr, Pr = PaperWallet.skpkGen() #receivers private/ public
se, pe, ss = ecdh.ecdhgen(Pr) #compute shared secret ss
digits = 14 #in practice it will be 14
print("inputs")
Cia, L1a, s2a, sa, ska = RingCT.genRangeProof(10000, digits)
print("outputs")
Cib, L1b, s2b, sb, skb = RingCT.genRangeProof(7000, digits)
Cic, L1c, s2c, sc, skc = RingCT.genRangeProof(3000, digits)
print("verifying range proofs of outputs")
RingCT.verRangeProof(Cib, L1b, s2b, sb)
RingCT.verRangeProof(Cic, L1c, s2c, sc)
x, P1 = PaperWallet.skpkGen()
P2 = PaperWallet.pkGen()
C2 = PaperWallet.pkGen()
#some random commitment grabbed from the blockchain
ind = 0
Ca = RingCT.sumCi(Cia)
Cb = RingCT.sumCi(Cib)
Cc = RingCT.sumCi(Cic)
sk = [x, MiniNero.sc_sub_keys(ska, MiniNero.sc_add_keys(skb, skc))]
pk = [[P1, P2], [MiniNero.subKeys(Ca, MiniNero.addKeys(Cb, Cc)), \
MiniNero.subKeys(C2, MiniNero.addKeys(Cb, Cc)) ] ]
II, cc, ssVal = MLSAG.MLSAG_Sign(pk, sk, ind)
print("Sig verified?", MLSAG.MLSAG_Ver(pk, II, cc, ssVal) )
print("Finding received amount corresponding to Cib")
RingCT.ComputeReceivedAmount(pe, sr, MiniNero.addScalars(ss, skb), Cib)
print("Finding received amount corresponding to Cic")
RingCT.ComputeReceivedAmount(pe, sr, MiniNero.addScalars(ss, skc), Cic)
```

Вот пример транзакции со входом номиналом $10,000$ и выходами номиналом $3,000$ и $7,000$.

```
('H', '61fe7f0f5a607a33427d01dd1fded5ffa03fae2e9df9ebccf2e0a2f5bd77a204')
inputs
('b, b in binary', 10000, [0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1])
Generating Aggregate Schnorr Non-linkable Ring Signature
outputs
('b, b in binary', 7000, [0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0])
Generating Aggregate Schnorr Non-linkable Ring Signature
('b, b in binary', 3000, [0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0])
```

```
Generating Aggregate Schnorr Non-linkable Ring Signature
verifying range proofs of outputs
Verifying Aggregate Schnorr Non-linkable Ring Signature
Verified
Verifying Aggregate Schnorr Non-linkable Ring Signature
Verified
('Generating MLSAG sig of dimensions ', 2, 'x ', 2)
('verifying MLSAG sig of dimensions ', 2, 'x ', 2)
('c',
['80a3cfd06dd2862307cd75c2a1566f20cd743dbb0b9feb22d79dcbecb9023f42',
'a9b7342ba7bf2f102505ca19dab734fde638916c0a29f5b30e49833ab51393ea',
'80a3cfd06dd2862307cd75c2a1566f20cd743dbb0b9feb22d79dcbecb9023f42'])
('sig verifies?', True)
('Sig verified?', True)
Finding received amount corresponding to Cib
('received ', 7000,
'a488ec68732fb551841c2c6dcc7ffac895d98ec7e9378275ed20ea12805fc18e')
Finding received amount corresponding to Cic
('received ',3000,
'1b46626858e130a0f3884c74c9fdeabc4d812c519103ea16a35a3f82a3d0ed6d')
```