Краткие связываемые кольцевые подписи и фальсификация по ключам злоумышленника

Брэндон Гуделл (Brandon Goodell)[1], Саранг Ноезер (Sarang Noether)[1] и Артур Блю (Arthur Blue)[2]

[1] Исследовательская лаборатория Monero (Monero Research Lab), {surae,sarang}@getmonero.org
[2] Независимый исследователь, randomrun@protonmail.com

Аннотация Мы демонстрируем, что наша версия защиты от оговора (выдача себя за другого пользователя) является естественным определением невозможности подделки связываемых кольцевых подписей. Нами предлагается схема построения связываемой кольцевой подписи с использованием кратких подписей и многомерных ключей, являющуюся связно анонимной при наличии сложного варианта решения задачи Диффи-Хеллмана со случайными оракулами, связываемой, если агрегация ключей является односторонней функцией, а также защищающей от оговора, если ещё одна версия задачи дискретного логарифмирования также является сложной. Нами приводятся некоторые варианты применения схемы в моделях конфиденциальных транзакций, скрывающих подписанта и не требующих доверенных настроек.

1 Введение

Будучи впервые представленной в работе [20] в настройках RSA и в работе [14] в рамках дискретного логарифма, схема построения кольцевой подписи позволяет подписывать сообщения при помощи набора публичных ключей, а не одного единственного публичного ключа. Кольцевые подписи можно применять в самых различных целях, начиная с лёгкой анонимной аутентификации, описанной в работе [25], и заканчивая протоколами проведения транзакций, такими как Monero [16] и CryptoNote [24]. Верификатор получает уверенность в том, что подписавшей стороне известен приватный ключ, по крайней мере, к одному из этих публичных ключей, которые называются участниками кольца. Кольцевые подписи являются анонимными или скрывают подписанта в том смысле, что верификатор не может узнать на основе подписи информацию о том, какой из ключей принадлежит подписанту. Мы подчёркиваем, что методы практического анализа, подобные методам, описанным в работах [15,19], могут использовать метаданные при использовании протоколов анонимной аутентификации в реальных условиях для снижения уровня анонимности.

Схемы групповой подписи, предшествующие представленным в работах [20,14], требуют некоторой интерактивности, фиксированного набора участ-

ников, наличия доверенного администратора группы или других доверенных настроек, или применения допусков сложности, не основанных на решении задачи дискретного логарифмирования. После публикации работы [20], кольцевые подписи множество раз улучшались, становились предметом всевозможных расширений и модификаций. Например, кольцевые подписи, которые строятся на основе билинейного спаривания, описаны в работе [26], структуры ключей обобщаются в работе [1], определения безопасности были улучшены в работе [5], возможность сокращения размера подписей была рассмотрена в работах [7,11], а возможность отслеживания стала темой работы [8].

Схемы связываемой кольцевой подписи (LRS) были впервые представлены в работе [14]; в контексте протоколов проведения транзакций связываемые кольцевые подписи являются основой аутентификации анонимной транзакции. Связываемые кольцевые подписи гарантируют, что две подписи с одним и тем же кольцом в произвольных сообщениях могут быть публично связаны, если они были подписаны с использованием одного и того же ключа. В основе одного из вариантов реализации, предложенного в работе [14], лежит задача дискретного логарифмирования; данный вариант работает так же, как и подписи Шнорра, описанные в работе [21].

Связующие теги подписи, о которых говорится в работе [14], не подходят для применения в тех случаях, когда подписи должны быть связаны соответствующими ключами, а не кольцами (например, при использовании защиты от «двойного подписания», когда пользователи генерируют новые ключи с течением времени и выбирают *подходящих* для этого участников кольца). Защита от двойного подписания обеспечивается с помощью связующих тегов, описанных в работе [24] (где авторы называют их образами ключей). В более поздней работе [16] подход, представленный в работе [14], расширяется до анонимной модели проведения конфиденциальных транзакций с особым *подбором* участников кольца. В работе [16] суммы транзакций заменяются обязательствами Педерсена по суммам и доказательствами диапазона. Подписи строятся на основе векторов ключей, а также разницы обязательств по сумме, используемой в качестве одного из ключей. Тем не менее доказательства, описанные в работе [16], являются неформальными и не базируются на строгих моделях безопасности.

Альтернативные кольцевым подписям решения, такие как более общие системы доказательства с нулевым разглашением, как правило, требуют наличия доверенной стороны, которая бы честно выполняла процесс настройки (как в работах [9,4,10]), или предполагают отсутствие практической эффективности при использовании больших схем (как в работе [6]), а это означает, что такие системы не подходят для применения в случае с распределённым реестром. Однако более поздние подходы, подобные описанному в работе [12], демонстрируют улучшения как с точки зрения необходимости в доверии, так и в плане повышения эффективности.

Определения экзистенциальной невозможности подделки схем кольцевой подписи, приводимые в работе [5], как правило, неприменимы к LRS. Эти

определения гарантируют, что подделки можно будет вычислить на основе неповреждённых ключей запросов при ограничении выбора участников анонимной группы и запросов оракула. При наличии связываемости мы можем предложить лучший вариант.

Идея экзистенциальной невозможности подделки обычных цифровых подписей заключается в том, чтобы поставить перед алгоритмом такую задачу, чтобы он выводил любую действительную, не связанную с использованием оракулов подпись для любого сообщения, и чтобы она была подписана неповреждённым ключом запроса. В случае со схемами LRS аналогичным образом перед алгоритмом ставится задача, в результате выполнения которой мы бы получали любую действительную, не связанную с использованием оракула подпись для любого сообщения, и при этом связи подписи с подписью запроса должны вычисляться на основе неповреждённого ключа запроса. Сравните это с определением невозможности подделки обычных кольцевых подписей в плане внутреннего повреждения, о котором говорится в работе [5], где действительная подпись σ с анонимной группой Q не считается поделкой, если какой-либо ключ в Q повреждён или не является ключом запроса, или же подписывающий оракул был запрошен той же анонимной группой Q и сообщением. В результате возникают две проблемы.

Первая проблема связана с признанием недействительной предполагаемой подделки, если в анонимной группе присутствуют какие-либо выбранные злоумышленником ключи или повреждённые ключи запроса. Такая подпись, связанная с неповреждённым ключом запроса, не считается подделкой. В схемах построения кольцевой подписи, не обладающих свойством связываемости, данное ограничение необходимо, чтобы подпись гарантировано была подписана неповреждённым ключом запроса. В случае применения, например, с протоколами транзакций с сокрытием подписанта, использующими *специально выбираемые* анонимные группы, предполагается, что участники анонимной могут быть созданы злоумышленником. Это следует учитывать при формулировке определений невозможности подделки.

Вторая проблема связана с ограничением применения подписывающего оракула в рамках этого определения. Злоумышленник может сгенерировать действительную подпись, не связанную с использованием оракула, повторно используя те же сообщение и анонимную группу, что и некоторая подпись запроса, сгенерированная оракулом. Верификатор, обнаруживший более одной действительной подписи, не связанной с использованием оракула, в одном и том же сообщении с одной и той же анонимной группой, согласно этому определению, может только прийти к выводу, что, по крайней мере, один из членов анонимной группы поставил хотя бы одну из подписей. Это позволяет очевидному злоумышленнику собрать исходящие честные подписи из некоторых честных сообщений и попытаться создать клоны таких честных сообщений. Отправитель, который затем попытается заявить о том, что является автором подписи, может оказаться в опасности, так как не сможет восстановить подпись, отправленную какому-либо получателю.

Эти проблемы противоречат духу концепции экзистенциальной невозможности подделки с возможностью внутреннего повреждения, описанной в работе [5]: подпись является действительной, не создаётся оракулом, вычисляется на основе неповреждённого ключа запроса и всё же не считается подделкой.

1.1 Наш вклад

Мы смягчаем понятие успешной подделки, подразумевая, что для этого требуется только определение связи с неповреждёнными ключами запроса, и получаем таким образом понятие невозможности подделки связываемых кольцевых подписей, которое эквивалентно определению защиты от оговора, представленному в работе [2]. Защита от оговора допускает возможность наличия злоумышленников, способных сделать запрос подписывающего оракула с любой анонимной группой, и даже выбираемой такими злоумышленниками анонимной группой. Это позволяет смоделировать чрезвычайно убедительного злоумышленника, который сможет сделать так, что пользователи будут подписывать сообщения с помощью выбранных им анонимных групп. Мы демонстрируем, что защита от оговора предполагает невозможность подделки в плане внутреннего повреждения, о котором говорится в работе [5].

Нами также предлагается схема связываемой кольцевой подписи, подобная схеме подписи Шнорра, которую мы называем d-CLSAG и которая использует методы агрегирования ключей с целью увеличения размера подписи. Мы описываем способ применения схемы d-CLSAG в рамках протоколов кольцевых конфиденциальных транзакций, позволяющий создавать транзакции, состоящие из одновременно $d-1$ различных объектов. Мы доказываем, что d-CLSAG является связываемой, что предполагает отсутствие конфликтов при агрегации ключей и связываемую анонимность при условии сложности решения алгоритма Диффи-Хеллмана. Напомним, что схемы подписи, подобные схеме Шнорра, не сводятся чисто к допуску дискретного логарифмирования (*см. работу* [17]) и, как правило, безопасны только при наличии (относительно ненадёжных) дополнительных допусков сложности решения, а методы доказательства, как правило, используют программируемые случайные оракулы и леммы разветвления, что приводит к ослаблению связей. Нами используется общая лемма разветвления, позволяющая доказать, что d-CLSAG обеспечивает защиту от оговора при условии наличия κ-дополнительного варианта решения задачи дискретного логарифмирования, что само по себе является сложным.

2 Предварительные условия

Допустим, λ является параметром безопасности, а $1 \leq q, \eta, \kappa \in poly(\lambda)$. Допустим, $\mathtt{G} = \langle G \rangle$ обозначает группу над полем $\mathtt{F}_p$ для некоторых главных p, с генератором G. Допустим, $\mathcal{H}^s : \{0,1\}^* \to \mathtt{F}_p$ и $\mathcal{H}^p : \{0,1\}^* \to \mathtt{G}$ являются

двумя независимыми криптографическими хеш-функциями, смоделированными в качестве случайных оракулов.

Если существует функция f, являющаяся незначительной по некоторому параметру λ, и существуют такие зависимые от λ события A, B, что $|\mathbb{P}[A] - \mathbb{P}[B]| \leq f(\lambda)$, мы обозначаем это как $\mathbb{P}[A] \approx \mathbb{P}[B]$. Событие A является таким, что $\mathbb{P}[A] \approx 0$ считается незначительным в λ (или может произойти с минимальной вероятностью в λ). Дополнение события A обозначается как $\overline{A}$. Для поля $\mathtt{F}_p$ обозначим ненулевые элементы $\mathtt{F}_p^* := \mathtt{F}_p \setminus \{0\}$. Для конечного множества S определяем S^*, интерпретируя S как список символов, образующих свободный моноид S^*. Например, $\{0,1\}^*$ состоит из строк битов конечной длины.

Для набора S определяем $\mathcal{P}(S)$ как степенное множество S, то есть, множество всех подмножеств S.

Обозначаем векторы жирным шрифтом, то есть для последовательности $x_i \in \mathtt{F}_p$ при $i = 0, \ldots, n-1$ обозначаем кортеж $(x_0, \ldots, x_{n-1})$ как $\mathbf{x}$. Обозначаем произведение Адамара между векторами как $\mathbf{x} \circ \mathbf{y} = (x_i y_i)_i$. Во избежание какой-либо двусмысленности начинаем все индексы с 0. Мы используем обозначение $[n]$ для набора $\{0, \ldots, n-1\}$.

3 Определения

3.1 Допуски сложности решения

Начнём с того, что если $\left\{\mu_i : \mathtt{F}_p^d \to \mathtt{F}_p\right\}_{i=0}^{d-1}$ являются функциями, выбираемыми равномерно и случайным образом, то сложная функция $\mu : \mathtt{F}_p^d \to \mathtt{F}_p$, определяемая равномерным распределением $\mathbf{x} \mapsto \sum_i \mu_i(\mathbf{x}) x_i$, имеет равномерно распределяемый выход и устойчива к возникновению конфликтов и использованию предварительно созданных образов. Мы используем это свойство далее при рассмотрении возможности обеспечения связываемости и связываемой анонимности.

Следующее решение задачи эквивалентно[3] обычному решению κ-дополнительной задачи дискретного логарифмирования согласно модели случайного оракула.

Определение 1 (Решение κ-дополнительной задачи дискретного логарифмирования в линейных комбинациях).

Допустим, $\kappa \geq 1$. Мы утверждаем, что любой PPT-алгоритм $\mathtt{A}$, который может успешно решить следующую задачу самое большее за время t

[3] Обычный алгоритм решения κ-дополнительной задачи дискретного логарифмирования может преуспеть в решении задачи в соответствии с Определением 1 с той же вероятностью и в течение некоторого дополнительного времени путём простого суммирования решений. Алгоритм, решающий задачу в соответствии с Определением 1, может использовать знание дискретного логарифма, чтобы найти другой неповреждённый ключ запроса с той же вероятностью и в течение некоторого дополнительного времени.

и с вероятностью такого успешного решения по крайней мере ϵ, является (t, ϵ, q)-алгоритмом решения к-дополнительной задачи дискретного логарифмирования в линейных комбинациях в G (где $k < q$).

1. *Запросчик случайным образом выбирает $\{sk_i\}_{i=0}^{q-1} \subseteq \mathsf{F}_p$, вычисляет $pk_i := sk_i \cdot G$ и отправляет $S := \{pk_i\}_{i=0}^{q-1}$ A.*
2. *A получает доступ к повреждённому оракулу CO, который берёт в качестве входа публичный ключ $pk_i \in S$, а в качестве выхода выдаёт соответствующий приватный ключ sk_i. Повреждённые ключи записываются в таблицу C.*
3. *A выдаёт некоторое $w \in \mathsf{F}_p^*$, последовательность элементов поля $\{h_j\}_{j=0}^{\kappa} \subseteq \mathsf{F}_p \backslash \{0\}$ и подмножество ключей запроса $\{pk_j^*\}_{j=0}^{\kappa} \subseteq \{pk_i\}_{i=0}^{q-1}$, успешно решая таким образом задачу, если и только если $w \cdot G = \sum_{j=0}^{\kappa} h_j \cdot pk_j^*$ и A запросили CO не более к раз.*

Мы утверждаем, что задача дискретного логарифмирования в линейных комбинациях сложна для решения в G, если какой-либо (t, ϵ, q)-алгоритм решения этой задачи имеет преимущество, которое является ϵ незначительным в q.

В случае с алгоритмом $\mathtt{A}$, который в качестве входа берёт случайный ряд ρ, последовательность запросов случайного оракула $\mathbf{h}$ и некоторый вход inp и выдаёт $(out, idx) \leftarrow \mathtt{A}(\rho, \mathbf{h}, inp)$ в качестве выхода, общий алгоритм разветвления работает следующим образом.

(1) Выбираем случайный ряд ρ для $\mathtt{A}$ и две последовательности запросов случайного оракула $\mathbf{h} = \{h_0, h_1, \ldots\}$ и $\mathbf{h}' = \{h'_0, h'_1, \ldots\}$.
(2) Выполняем $(out, idx) \leftarrow \mathtt{A}(\rho, \mathbf{h}, inp)$.
(3) Задаём значение $j := idx$ и связываем запросы оракула $\mathbf{h}$, $\mathbf{h}'$ вместе

$$\mathbf{h}^* = \left\{h_0, h_1, \ldots, h_{idx-1}, h'_{idx}, h'_{idx+1}, \ldots, \right\}.$$

(4) Выполняем $(out^*, idx^*) \leftarrow \mathtt{A}(\rho, \mathbf{h}^*, inp)$.
(5) Если $idx \neq idx^*$ или же $h_{idx} = h'_{idx}$, выводится $\perp$ и происходит завершение процесса. В ином случае выводится (out, out', idx).

Данный алгоритм является алгоритмом $\mathcal{F}^{\mathtt{A}}$, используемым в следующей лемме.

Лемма 1 (Общая лемма разветвления). *Допустим, $1 \leq \eta \in poly(\lambda)$. Допустим, A является любым PPT-алгоритмом, который берёт в качестве входа некоторый кортеж $x_{\mathsf{A}} = (x, \boldsymbol{h})$, где $\boldsymbol{h} = (h_0, h_1, \ldots, h_{q-1})$ является последовательностью ответов на запросы оракула (строками из η-бит), и выдаёт в качестве выхода y_{A} либо выделенный символ сбоя $\perp$, либо пару (idx, y), где $idx \in [q]^2$ и y являются каким-то выходом. Допустим, ϵ_{A} обозначает вероятность того, что A не выдаёт $\perp_{\mathsf{A}}$ (при этом такая вероятность относится ко всем случайным монетам A, распределению x,*

и всем вариантам $\boldsymbol{h}$*). Допустим,* $\mathcal{F} = \mathcal{F}^{\mathtt{A}}$ *является алгоритмом разветвления для* $\mathtt{A}$*. Допустимая вероятность приемлемости* $\mathcal{F}$ *соответствует* $\epsilon_{\mathcal{F}} \geq \epsilon_{\mathtt{A}} \left(\frac{\epsilon_{\mathtt{A}}}{q} - \frac{1}{2^{\eta}} \right)$.

Мы используем вариант решения задачи, чтобы связать классический допуск сложности решения Диффи-Хеллмана со схемой использования случайного оракула, которая пригодится далее при введении определения связываемой анонимности. Свойства конструкции такого типа, связанные с псевдослучайностью и обратимостью, более подробно рассматриваются в работе [13].

Определение 2 (Решение задачи Диффи-Хеллмана с использованием случайного оракула). *Мы утверждаем, что любой PPT-алгоритм* $\mathtt{A}$*, который может успешно решить следующую задачу самое большее за время* t *и с вероятностью такого успешного решения, по крайней мере,* $\epsilon > 0$*, является* (t, ϵ, q)*-алгоритмом решения задачи Диффи-Хеллмана с использованием случайного оракула.*

- *Запросчик случайным образом выбирает* $b \in \{0, 1\}$.
- *Если* $b = 0$*, запросчик случайным образом и единообразно выбирает* $\{r_i\}_{i=0}^{q-1}$ *из* $\mathtt{F}_p$ *и задаёт*

$$S := \{(R_i, R_i', R_i'')\}_{i=0}^{q-1} = \{(r_i G, \mathcal{H}^p(r_i G), r_i \mathcal{H}^p(r_i G))\}_{i=0}^{q-1},$$

  и отправляет S *алгоритму* $\mathtt{A}$.
- *Если же* $b = 1$*, запросчик случайным образом и единообразно выбирает* $\{(r_i, r_i'')\}_{i=0}^{q-1}$ *из* $\mathtt{F}_p^2$ *и задаёт*

$$S := \{(R_i, R_i', R_i'')\}_{i=0}^{q-1} = \{(r_i G, \mathcal{H}^p(r_i G), r_i'' G)\}_{i=0}^{q-1},$$

  и отправляет S *алгоритму* $\mathtt{A}$.
- $\mathtt{A}$ *предоставляется доступ к случайному оракулу* $\mathcal{H}^p$.
- $\mathtt{A}$ *выдаёт бит* $b' \in \{0, 1\}$*; считается, что* $\mathtt{A}$ *успешно решает задачу, если и только если* $b' = b$.

Мы утверждаем, что нахождение групповых элементов в соответствии с алгоритмом Диффи-Хеллмана с использованием случайного оракула является сложным в $\mathtt{G}$*, если какой-либо* (t, ϵ, q)*-алгоритм решения имеет преимущество* ϵ.

Примечание 1. Мы допускаем, что если решение классической задачи в соответствии с алгоритмом Диффи-Хеллмана (DDH) в $\mathtt{G}$ является сложным, то и решение задачи RO-DDH тоже. Следует помнить о том, что DDH просит злоумышленника отличить распределения кортежей в форме $(rG, r'G, rr'G)$ и $(rG, r'G, r''G)$. Если решение DDH представляется сложным в $\mathtt{G}$, то распределение кортежей в таких формах отличить невозможно. Поскольку $\mathcal{H}^p$ является случайным оракулом, выход которого не зависит от входа, распределение кортежей в форме $(rG, \mathcal{H}^p(rG), r''G)$ и $(rG, r'G, r''G)$

будет идентичным. Подобным образом, и распределение кортежей в форме $(rG, \mathcal{H}^p(rG), r\mathcal{H}^p(rG))$ и $(rG, \mathcal{H}^p(rG), r''G)$ так же будет идентичным. Наконец, случайное свойство самовосстановления классического алгоритма DDH означает, что решение одной версии задачи будет не менее сложным, чем решение последовательности случайных версий задачи.

3.2 Связываемые кольцевые подписи

Определение 3. *Схема связываемой кольцевой подписи (LRS) Π_{LRS} представляет собой набор алгоритмов (`SETUP`, `KEYGEN`, `SIGN`, `VERIFY`, `LINK`), соответствующих следующим ограничениям:*

- *`SETUP` берёт в качестве входа параметр безопасности λ и выдаёт некоторые общедоступные параметры настройки ρ в качестве выхода.*
- *`KEYGEN` является случайным алгоритмом и берёт в качестве входа (λ, ρ), и выдаёт пару, состоящую из приватного и публичного ключа (sk, pk).*
- *`SIGN` является случайным алгоритмом и берёт в качестве входа (λ, ρ) и три значения (m, Q, sk), и выдаёт подпись σ или символ сбоя $\perp$. Здесь m обозначает сообщение, Q - анонимный набор публичных ключей $Q = \{pk_0, pk_1, ..., pk_{n-1}\}$, а sk обозначает приватный ключ.*
- *`VERIFY` берёт в качестве входа (λ, ρ) и тройное значение (m, Q, σ) и выдаёт бит b. Здесь m обозначает сообщение, Q — анонимный набор публичных ключей $Q = \{pk_0, pk_1, ..., pk_{n-1}\}$, а σ является подписью.*
- *`LINK` берёт в качестве входа (λ, ρ), пару тройных значений (m, Q, σ), (m^*, Q^*, σ^*) и выдаёт бит b. В данном случае m, m^* обозначают сообщения, Q, Q^* — анонимные наборы публичных ключей, σ, σ^* являются подписями.*

Мы называем тройку значений в форме (m, Q, σ) (то есть тройного значения, подходящего для использования в качестве входа для `VERIFY` и `LINK`) тройным значением подписи. Мы говорим, что Π_{LRS} является d-LRS, если размер приватного и публичного ключей составляет $d \geq 1$, а анонимные группы можно описать как матрицы $d \times n$.

В целях дальнейшей конкретизации, мы допускаем, что ρ включает (неявно или явно) описания пространства приватных ключей $\mathcal{SK}$, пространства публичных ключей $\mathcal{PK}$, пространства подписи $\mathcal{SIG}$, преобразования $\phi : \mathcal{SK} \to \mathcal{PK}$ и семейства хеш-функций (смоделированных в качестве случайных оракулов) $\mathcal{H}$, из которых мы можем построить $\mathcal{H}^s$ и $\mathcal{H}^p$. Например, в случае с группой эллиптической кривой $\mathtt{G}$ в поле $\mathtt{F}_p$ для двух хеш-функций $H^s : \{0,1\}^* \to \mathtt{F}_p$ и $H^p : \{0,1\}^* \to \mathtt{G}$ схема кольцевой подписи, подобная схеме Шнорра, как в работе [14], будет упакована в описания ρ, $\mathcal{SK} = \mathtt{F}_p^*$, $\mathcal{PK} = \mathtt{G}$, $\mathcal{SIG} = \mathtt{F}_p^n$ и генератор $G \in \mathtt{G}$ (которого достаточно, чтобы указать преобразование $\phi : \mathtt{F}_p^* \to \mathtt{G}$, определяемое $x \mapsto xG$). Следует отметить, что каждый алгоритм в схеме LRS берёт (λ, ρ) в качестве входа; впоследствии

мы исключаем их из обозначений схем LRS, поскольку их использование подразумевается изначально.

Также обратите внимание, что мы не допускаем, что схема LRS является правильной, если она позволяет использовать анонимные мультимножества Q.

Определение 4. *Допустим, $b \in \{0,1\}$, $sk, sk^* \in \mathcal{SK}$, Q, $Q^* \subset \mathcal{PK}$ будут множествами, а m, m^*, m' будут сообщениями; допустим, $\sigma, \sigma^*, \sigma' \in \mathcal{SIG}$ будут какими-то предполагаемыми подписями. Определим следующие события.*

- $E_1(sk, pk)$ *является таким событием, что некоторые* $(sk, pk) \leftarrow \texttt{KEYGEN}$.
- $E_2(sk, Q)$ *является таким событием, что* $\phi(sk) \in Q$.
- $E_3(m, Q, \sigma)$ *является таким событием, что* $\texttt{VERIFY}(m, Q, \sigma) = 1$.
- $E_4(b, m, m^*, m', Q, Q^*, Q', \sigma, \sigma^*, \sigma')$ *является таким событием, что*

$$b = \texttt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = \texttt{LINK}((m^*, Q^*, \sigma^*), (m', Q', \sigma')).$$

- $E_5(m, m^*, Q, Q^*, \sigma, \sigma^*, sk)$ *является таким событием, что*

$$\phi(sk) \in Q \cap Q^*,\ \sigma \leftarrow \texttt{SIGN}(m, Q, sk),\ \text{и}\ \sigma^* \leftarrow \texttt{SIGN}(m^*, Q^*, sk).$$

- $E_6(m, m^*, Q, Q^*, \sigma, \sigma^*, sk, sk^*)$ *является таким событием, что*

$$\sigma \leftarrow \texttt{SIGN}(m, Q, sk),\ \sigma^* \leftarrow \texttt{SIGN}(m^*, Q^*, sk^*),\ \text{и}\ sk \neq sk^*.$$

Мы говорим, что схема Π_{LRS} является верной, если наблюдается соответствие всем следующим свойствам, где эти вероятности вычисляются для всех случайных монет и всех вариантов хеш-функций.

(i) Действительные ключи распределяются надлежащим образом:

$$\mathbb{P}\left[\phi(sk) = pk \mid E_1(sk, pk)\right] \approx 1$$

(ii) Подписание ключом, который отсутствовал в кольце, не состоялось:

$$\mathbb{P}\left[\texttt{SIGN}(m, Q, sk) = \perp \mid \overline{E_2}(sk, Q)\right] \approx 1$$

(iii) Верификация с использованием действительной подписи прошла успешно:

$$\mathbb{P}\left[\texttt{VERIFY}(m, Q, \texttt{SIGN}(m, Q, sk)) = 1 \mid E_2(sk, Q)\right] \approx 1$$

(iv) Действительная подпись связывается сама с собой:

$$\mathbb{P}\left[\texttt{LINK}((m, Q, \sigma), (m, Q, \sigma)) = 1 \mid E_3(m, Q, \sigma)\right] \approx 1$$

(v) Связывание является коммутационным:

$$\mathbb{P}\left[\texttt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = \texttt{LINK}((m^*, Q^*, \sigma^*), (m, Q, \sigma))\right] \approx 1$$

(vi) Связывание является транзитивным:

$$\mathbb{P}\left[b = \texttt{LINK}((m, Q, \sigma), (m', Q', \sigma')) \mid E_4(b, m, m^*, m', Q, Q^*, Q', \sigma, \sigma^*, \sigma')\right] \approx 1$$

(vii) Повторное использование подписывающего ключа является связующим:

$$\mathbb{P}\left[\texttt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = 1 \mid E_5(m, m^*, Q, Q^*, \sigma, \sigma^*, sk)\right] \approx 1$$

(viii) Использование отдельного подписывающего ключа подразумевает отсутствие связывания:

$$\mathbb{P}\left[\texttt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = 0 \mid E_6(m, m^*, Q, Q^*, \sigma, \sigma^*, sk, sk^*)\right] \approx 1$$

3.3 Связываемость

Мы используем два отдельных, но связанных между собой определения связываемости. Первым определением является Определение 5, взятое из работы [2], где мы используем термин *связываемость ACST* (по инициалам автора). Данное определение допускает использование запросов подписывающего оракула с любыми анонимными группами Q, состоящими по крайней мере из одного ключа запроса (который может быть смоделирован путём внесения обратных правок). Это определение также позволяет запросчику успешно решить задачу связываемости, используя тройные значения подписи (m, Q, σ), (m^*, Q^*, σ^*), в которых присутствуют «полуповреждённые» анонимные группы Q, Q^* (при условии, что ключи в $Q \cup Q^*$. Вторым определением является Определение 8, взятое из работы [3], в отношении которого мы используем термин *связываемости по голубиным отверстиям* в соответствии с известным принципом голубиных отверстий. Это определение допускает наличие у злоумышленника полного контроля над выбором ключей, когда считается, что злоумышленник добился успеха, если ему удалось создать больше несвязываемых кольцевых подписей, чем составляет общее количество участников кольца. Такому злоумышленнику не требуется генерировать ключи, повреждать их или обладать доступом к подписывающему оракулу.

Определение 5 (Связываемость ACST с использованием ключа злоумышленника). *Мы утверждаем, что любой PPT-алгоритм* `A`*, который может успешно решить эту задачу самое большее за время t и с вероятностью успешного выполнения по крайней мере ϵ, является (t, ϵ, q)-алгоритмом решения задачи связываемости ACST.*

1. *Запросчик выводит $\{(sk_i, pk_i)\}_{i=0}^{q-1}$ из* `KEYGEN` *и отправляет публичные ключи запроса $S = \{pk_i\}_{i=0}^{q-1}$ алгоритму* `A`*.*
2. *Алгоритм* `A` *получает доступ к* `SO` *и* `CO`*.*
   - `CO` *берёт в качестве входа публичный ключ pk. Если $pk \in S$,* `CO` *выдаёт соответствующий приватный ключ sk. В противном случае* `CO` *выдаёт символ отказа, $\perp$. Повреждённые ключи отслеживаются через таблицу C.*

– $\texttt{SO}$ *выполняется следующим образом:*
  (i) В качестве входа берётся сообщение m*, анонимная группа* $Q = \{pk'_i\}_{i=0}^{n-1}$ *и индекс* l *(обратите внимание, что мы специально не запрашиваем* $Q \subset S$*).*
  (ii) Если $0 > l$ *или* $l \geq n$*, или же если* $pk'_l \notin S$*,* $\texttt{SO}$ *выдаёт символ отказа* $\perp$*.*
  (iii) В ином случае существует такое значение i*, что* $pk'_l = pk_i \in S$*.* $\texttt{SO}$ *выбирает приватный ключ* sk_i *и выводит такое действительное тройное значение подписи* (m, Q, σ)*, что* $\sigma \leftarrow \texttt{SIGN}(m, Q, sk_i)$*.*

3. $\texttt{A}$ *выдаёт пару тройных значений подписи* (m, Q, σ)*,* (m^*, Q^*, σ^*) *и решение является успешным, если и только если соблюдены все следующие условия (следует отметить, что в данном случае мы не ставим чёткого требования, чтобы* $Q \subset S$ *или* $Q^* \subset S$*).*
  (i) $\texttt{VERIFY}(m, Q, \sigma) = \texttt{VERIFY}(m^*, Q^*, \sigma^*) = 1$
  (ii) $\texttt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = 0$
  (iii) σ, σ^* *не выводятся в результате запроса* $\texttt{SO}$*.*
  (iv) $\left|(Q \cup Q^*) \cap (C \cup \overline{S})\right| \leq 1$

Мы говорим, что схема является связываемой по ACST, если каждый PPT-алгоритм $\texttt{A}$*, являющийся* (t, ϵ, q)*-алгоритмом решения задачи связываемости ACST, имеет ничтожную вероятность приемлемости* ϵ*.*

Определение 6 (Связываемость по q голубиных отверстий). *Мы утверждаем, что любой PPT-алгоритм* $\texttt{A}$*, который может успешно выдать* q *публичных ключей* $\{pk_i\}_{i=0}^{q-1}$ *и* $q+1$ *действительных, несвязанных тройных значений подписи* $\{(m_j, Q_j, \sigma_j)\}_{j=0}^{q}$ *так, чтобы* $\cup_j Q_j \subseteq \{pk_i\}_{i=0}^{q-1}$*, самое большее за время* t *и с вероятностью, по крайней мере,* ϵ*, является* (t, ϵ, q)*-алгоритмом решения задачи связываемости по* q *голубиных отверстий. Мы говорим, что схема является связываемой по* q*-классам, если каждый PPT-алгоритм* $\texttt{A}$*, являющийся* (t, ϵ, q)*-алгоритмом решения задачи связываемости по* q*-классам имеет ничтожную вероятность приемлемости* ϵ*. (Следует отметить, что в данном случае злоумышленник не имеет доступа к подписывающему или повреждённому оракулу.)*

Эти определения существуют по отдельности. Связываемость по голубиным отверстиям совсем не обязательно предполагает наличие связываемости ACST. На самом деле алгоритм, который успешно решит задачу в соответствии с Определением 5, может сделать это только при наличии достаточно больших колец, и в таком случае это может быть успешно сделано уже с использованием Определения 6 с достаточно большим значения q. Подобным образом связываемость ACST может и не предполагать связываемости по голубиным отверстиям. Алгоритм решения задачи, в соответствии с Определением 6, требующий знания множества приватных ключей, может потребовать большого количества повреждений, чтобы успешно решить задачу в соответствии с Определением 5.

В случае со схемой связываемости по голубиным отверстиям, в Определении 6 не сказано ничего о злоумышленнике, способном подписываться множество раз, используя один и тот же ключ, чтобы строить новые несвязанные подписи до тех пор, пока будут использоваться различные анонимные группы.

В случае со схемой связываемости ACST пользователь, предоставляющий честные подписи злоумышленнику, действует в качестве подписывающего оракула, поэтому злоумышленник может попытаться построить плохую/клонированную подпись, которая будет связана с честной подписью (возможно, с выбранными злоумышленником участниками кольца), и всегда выдавать только одну подпись. В Определении 5 ничего не сказано о возможности подобной атаки. Такой злоумышленник может находиться между Элис и Бобом и выдавать Бобу клонированные подписи, как бы созданные Элис, тем самым обманывая его. Когда в конечном счёте Элис свяжется с Бобом и между ними уже не будет никакого злоумышленника, она не сможет доказать свою невиновность в связи с клонированными подписями. Боб придёт к выводу, что σ^* и σ были подписаны одним и тем же приватным ключом, несмотря на то, что злоумышленнику не был известен приватный ключ, при помощи которого была сгенерирована σ.

Тем не менее, Определение 6 предполагает возможность решения по Определению 5 при определённых обстоятельствах. Вот один пример (из множества): если существует $(t, \epsilon, 1)$-алгоритм решения задачи, в соответствии с Определением 5, который выдаёт такие два тройных значения подписи (m, Q, σ), (m', Q', σ'), $Q = Q'$, что $|Q| = 1$, значит, $\mathtt{A}$ может успешно решить задачу в соответствии с Определением 6.

Данная работа не предполагает более подробного рассмотрения отношений между определениями связываемости. Поскольку ни одно из определений само по себе не является достаточным в нашем случае, а также, поскольку эти определения в целом являются самостоятельными, мы используем оба.

3.4 Невозможность подделки и защита от оговора

Идея экзистенциальной невозможности подделки обычных цифровых подписей состоит в том, чтобы алгоритм выдавал любую действительную не использующую оракул подпись для любого сообщения и чтобы такая подпись создавалась с использованием неповреждённого ключа запроса. Очевидным аналогом схем LRS является задача, согласно которой алгоритм должен выдавать любую действительную не использующую оракул подпись для любого сообщения, и чтобы такая подпись была связана с подписью запроса, вычисленной на основе неповреждённого ключа запроса. Фактически вариант реализации, предлагаемый в Разделе 4, предполагает сравнение связующих тегов, как если бы они были ключами верификации; в этом контексте перед злоумышленником, создающим поделку, стоит задач по созданию действительной подписи с использованием некоторых ключей верификации, приватный ключ к которым ему не известен. Мы считаем такую подделку

хрестоматийной, и поэтому учитываем такую вероятность в рамках нашего определения. В отличие от определений невозможности подделки, приводимых в работах [2] и [3], мы однозначно учитываем свойство связываемости, а также то, что злоумышленник может использовать повреждённые ключи. Далее мы связываем данное определение с определением защиты от оговора.

Определение 7 (Обеспечение экзистенциальной невозможности подделки связываемых кольцевых подписей при наличии выбираемых злоумышленником ключей и внутреннего повреждения). *Мы утверждаем, что любой PPT-алгоритм* `A`*, который может успешно решить данную задачу самое большее за время* t *и с вероятностью по крайней мере* ϵ*, является* (t, ϵ, q)*-алгоритмом подделки.*

1. *Запросчик выводит* $\{(sk_i, pk_i)\}_{i=0}^{q-1}$ *из* `KEYGEN` *и задаёт значение* $S := \{pk_i\}_{i=0}^{q-1}$*. Для каждого* $0 \leq i < q$ *запросчик единообразно выбирает секрет* $\emptyset \neq \widetilde{Q}_i \in \mathcal{P}(S)$*, определяет* $Q_i := \widetilde{Q}_i \cup \{pk_i\}$*, выбирает такой индекс* l_i*, что* $pk_i \in Q_i \cap S$ *имеет индекс* l_i *в* Q_i*, выбирает случайные секретные сообщения* $\{m_i\}_{i=0}^{q-1}$ *и вычисляет секретные подписи запроса* $\sigma_i \leftarrow$ `SIGN`(m_i, Q_i, sk_i)*. Запросчик отправляет* S *алгоритму* `A`*.*
2. *Запросчик даёт* `SO` *и* `CO` *доступ к* `A`*:*
   - *В качестве входа* `CO` *берёт публичный ключ запроса* $pk_i \in S$*, а в качестве выхода выдаёт соответствующий приватный ключ* sk_i*, сохраняя список всех повреждённых публичных ключей во внутренней таблице* C*.*
   - `SO` *выполняется следующим образом:*
     (i) В качестве входа берётся сообщение m*, анонимная группа* Q *и индекс* l*.*
     (ii) Если ключ с индексом l *в* Q *не является публичным ключом вызова* S*,* `SO` *выдаёт символ отказа* $\perp$*.*
     (iii) В ином случае `SO` *выбирает такой индекс* i*, что* $pk_i \in Q \cap S$ *имеет индекс* l *в* Q *и выбирает соответствующий приватный ключ* sk_i*, и выводит подпись* $\sigma \leftarrow$ `SIGN`(m, Q, sk_i)*.*
3. `A` *выдаёт сообщение* m*, кольцо* Q*, подпись* σ*, и решение является успешным, если и только если:*
   (i) `VERIFY`$(m, Q, \sigma) = 1$*,*
   (ii) Не существует никакого запроса, полученного `SO`*, выходом которого является* σ*,*
   (iii) Существует такое $i \in [0, q)$ *что* `LINK`$((m, Q, \sigma), (m_i, Q_i, \sigma_i)) = 1$*, а* $pk_i \in S \cap Q \setminus C$*.*

Более того, мы говорим, что схему связываемой кольцевой подписи невозможно подделать, если каждый (t, ϵ, q)*-алгоритм решения данной задачи имеет ничтожную вероятность приемлемости* ϵ*.*

Примечание 2. Доступ к повреждённому оракулу в соответстви с данным определением невозможности подделки является одной из причин (небольшого) ослабления безопасности до решения задачи κ-дополнительного дискретного логарифма, а не обычной задачи дискретного логарифмирования:

алгоритм, реализующий алгоритм подделки в чёрном ящике, не может смоделировать повреждённый оракул для осуществления подделки без получения доступа к повреждённому оракулу или без использования общей групповой модели.

Примечание 3. Если A является алгоритмом, создающим подписи в соответствии со схемой LRS, соответствующей Определению 7, значит, (за исключением незначительной вероятности) эти подписи не являются подделками, и, таким образом, некоторое свойство 3(i)-3(iii) должно быть нарушено. Если подпись является действительной и не была получена в результате запроса оракула, значит, свойство 3(iii) было нарушено. В частности, в случае со схемой LRS, соответствующей Определению 7, если алгоритм выдаёт действительное полученное без использования оракула тройное значение (m, Q, σ), связанное с ключом запроса в Q, значит, такой ключ запроса был повреждён.

Схема защиты от оговора была представлена в работе [23]. Это определение обновлялось дважды в работе [2], допуская, что A может успешно решить задачу всякий раз, когда публикует любую подпись, связанную с любой подписью из запроса A, сделанного SO, за исключением определённых условий. Мы изменяем определение, приведённое в работе [2], допуская наличие запросов подписывающего оракула с анонимными группами, содержащими выбранных злоумышленников участников.

Определение 8 (Защита от оговора выбранной цели при наличии выбираемых злоумышленником ключей и внутренних повреждений). *Мы утверждаем, что любой PPT-алгоритм* $\mathtt{A}$*, который может успешно решить следующую задачу самое большее за время* t *и с вероятностью по крайней мере* ϵ*, является* (t, ϵ, q)*-алгоритмом решения задачи защиты от оговора при наличии выбранных злоумышленником ключей.*

1. *Запросчик выводит* $\{(sk_i, pk_i)\}_{i=0}^{q-1}$ *из* $\mathtt{KEYGEN}$ *и отправляет публичные ключи запроса* $S = \{pk_i\}_{i=0}^{q-1}$ $\mathtt{A}$*.*
2. *A получает доступ к* $\mathtt{SO}$ *и* $\mathtt{CO}$ *точно так же, как в случае с Определением 7.*
3. *A выдаёт тройное значение* (m, Q, σ)*, и решение является успешным, если и только если:*
   - *(i)* $\mathtt{VERIFY}(m, Q, \sigma) = 1$*, и*
   - *(ii) не существует никакого запроса, полученного* $\mathtt{SO}$*, выходом которого является* σ*, и*
   - *(iii) существует такой запрос, отправленный* $\mathtt{SO}$*, допустим,* $\sigma^* \leftarrow \mathtt{SO}(m^*, Q^*, l^*)$*, что*
     - *(a)* $\mathtt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = 1$*, и*
     - *(b)* $pk_{l^*}^* \in S \cap Q^* \cap Q \setminus C$*.*

Более того, мы говорим, что схема связываемой кольцевой подписи защищена от оговора, если каждый (t, ϵ, q)*-алгоритм решения данной задачи имеет ничтожную вероятность приемлемости* ϵ*.*

Теорема 1 (Защита от оговора равноценна невозможности подделки). *Правильную схему LRS невозможно подделать согласно Определению 7, если и только если она защищена от оговора в соответствии с Определением 8.*

Доказательство. Допустим, A является (t, ϵ, q)-алгоритмом решения задачи в соответствии с Определением 7. Мы демонстрируем, как построить алгоритм B, который будет выполнять A в чёрном ящике и будет успешно решать задачу защиты от оговора в соответствии с Определением 8. Отметим, что подписывающий и повреждённый оракул идентичны в обоих определениях, поэтому такие запросы могут бесшовным способом передаваться между участниками, как это описано ниже. Формально B работает следующим образом:

- Алгоритм B получает от своего запросчика набор публичных ключей $S = \{pk_i\}_{i=0}^{q-1}$. Он выбирает сообщения и кольца (так же как это делается в соответствии с Определением 7) и генерирует набор кортежей $\{(m_i, Q_i, \sigma_i)\}_{i=0}^{q-1}$ при помощи запросов в форме $\mathtt{SO}(m_i, Q_i, l_i) \to \sigma_i$, где каждая l_i является индексом pk_i в Q_i. Затем алгоритм передаёт набор публичных ключей S алгоритму A
- Алгоритм B принимает запросы оракула SO и CO от алгоритма A, передаёт их запросчику и возвращает полученный результат A.
- Алгоритм A возвращает кортеж (m, Q, σ), соответствующий условиям Определения 7.
- Алгоритм B выводит (m, Q, σ).

Поскольку алгоритм A является алгоритмом решения задачи невозможности подделки, то при наличии преимущества ϵ существует такой индекс $i \in [0, q)$, что

$$\mathtt{LINK}((m, Q, \sigma), (m_i, Q_i, \sigma_i)) = 1$$

и $pk_i \in S \cap Q \setminus C$. Кроме того, алгоритм B при помощи запроса оракула $\mathtt{SO}(m_i, Q_i, l_i)$ получает σ_i, поэтому по схеме также $pk_i \in Q_i$. Так как алгоритм B использует дополнительное время t' для своих начальных q запросов подписывающего оракула и просмотров транскриптов, а также имеет то же преимущество ϵ, что и алгоритм A, можно утверждать, что нами построен $(t + t', \epsilon, q)$-алгоритм решения задачи защиты от оговора, соответствующий Определению 8.

Теперь мы приводим обратную формулировку, согласно которой предполагаем, что A является (t, ϵ, q)-алгоритмом решения задачи защиты от оговора в соответствии с Определением 8. Мы строим алгоритм B, который выполняет алгоритм A в чёрном ящике и является алгоритмом решения задачи невозможности подделки в соответствии с Определением 7.

- Алгоритм B получает от своего запросчика набор публичных ключей $S = \{pk_i\}_{i=0}^{q-1}$. Он передаёт набор публичных ключей S алгоритму A
- Алгоритм B принимает запросы оракула SO и CO от алгоритма A, передаёт их запросчику и возвращает полученный результат A.

– Алгоритм A возвращает кортеж (m, Q, σ) , соответствующий условиям Определения 8.
– Алгоритм B выводит (m, Q, σ).

Поскольку алгоритм A является алгоритмом решения задачи защиты от оговора, то при наличии преимущества ϵ существует такой запрос подписывающего оракула $\mathtt{SO}(m^*, Q^*, l^*) \to \sigma^*$, что

$$\mathtt{LINK}((m, Q, \sigma), (m^*, Q^*, \sigma^*)) = 1$$

и $pk^*_{l^*} \in S \cap Q^* \cap Q \setminus C$. Но так как $pk^*_{l^*} = pk_i \in S$ для некоторого индекса $i \in [0, q)$, запросчик решения задачи невозможности подделки создаёт действительную подпись σ_i для некоторого сообщения m_i и кольца Q_i, где $pk_i \in Q_i$. Таким образом, это должно выглядеть как

$$\mathtt{LINK}((m^*, Q^*, \sigma^*), (m_i, Q_i, \sigma_i)) = 1$$

и в силу свойства транзитивности мы также получаем

$$\mathtt{LINK}((m, Q, \sigma), (m_i, Q_i, \sigma_i)) = 1$$

Таким образом, мы продемонстрировали, что B является (t, ϵ, q)-алгоритмом решения задачи невозможности подделки, что делает наше доказательство полным.

3.5 Связываемая анонимность

Нами используется модифицированный вариант определения связываемой анонимности из работы [3]. Определение, представленное нами в данной работе, отличается от того, что предлагается в работе [3], поскольку не предполагает наличия у злоумышленника доступа к поднабору повреждённых ключей до того, как он получит доступ к подписывающему оракулу; это необходимо для того, чтобы позднее свести всё к сложности решения задачи в соответствии с Определением 2.

Определение 9 (Обеспечение связываемой анонимности выбранной цели при наличии выбранных злоумышленником ключей). *Мы утверждаем, что любой PPT-алгоритм* ***A****, который может успешно решить следующую задачу самое большее за время t и с вероятностью по крайней мере ϵ, является (t, ϵ, q)-алгоритмом решения задачи обеспечения связываемой анонимности.*

1. *Запросчик выбирает секретный случайный бит $b \in \{0, 1\}$, выводит $\{(sk_i, pk_i)\}_{i=0}^{q-1}$ из* ***KEYGEN*** *и отправляет публичные ключи запроса $S := \{pk_i\}_{i=0}^{q-1}$ алгоритму* ***A****.*
2. *Алгоритм* ***A*** *выдаёт пару индексов $0 \leq i_0, i_1 < q$, что $pk_{i_0}, pk_{i_1} \in S$, указывая ключи цели.*

3. *Алгоритм* $\mathtt{A}$ *получает доступ к подписывающему оракулу* $\mathtt{SO}$ *(который заметно отличается от подписывающего оракула, который использовался в предыдущих определениях):*
   (i) *В качестве входа берётся сообщение* m*, анонимная группа* Q *и публичный ключ* $pk \in Q$*.*
   (ii) *Если* $\{pk_{i_0}, pk_{i_1}\} \not\subseteq Q$ *или* $pk \notin \{pk_{i_0}, pk_{i_1}\}$*,* $\mathtt{SO}$ *выдаёт действительную подпись* σ*, связанную с* pk*.*
   (iii) *В ином случае* $pk = pk_{i_c}$ *для бита* c*. Вычисляется бит* $c' = (1-c)b + c(1-b)$ *и оракул* $\sigma \leftarrow \mathtt{Sign}(m, Q, sk_{i_{c'}})$ *(то есть* $c' = c \oplus b$*).*
4. *Алгоритм* $\mathtt{A}$ *выдаёт бит* b'*, и задача считается успешно решённой, если* $b' = b$*.*

Более того, мы говорим, что схема является связываемо анонимной, если каждый (t, ϵ, q)*-алгоритм решения задачи связываемой анонимности имеет преимущество с ничтожной вероятностью приемлемости* ϵ *более* $1/2$*.*

4 Структура

В этом разделе нами описывается d, наш вариант реализации сжатой схемы d-LRS.

Определение 10 (d-CLSAG). *Схемой подписи d-LRS является следующий кортеж* ($\mathtt{Setup}$, $\mathtt{KeyGen}$, $\mathtt{Sign}$, $\mathtt{Verify}$, $\mathtt{Link}$)*.*

- $\mathtt{Setup} \to par$*.* $\mathtt{Setup}$ *выбирает простое число* p*, группу* $\mathtt{G}$ *порядка простого числа* p*, равномерно случайным образом выбирает генератор* $G \in \mathtt{G}$*, выбирает* d *криптографических хеш-функций* $\mathcal{H}^s_0, \ldots, \mathcal{H}^s_{d-1}$ *(смоделированы как случайные оракулы) с кодоменом* $\mathtt{F}_p$*, выбирает криптографическую хеш-функцию* $\mathcal{H}^p$ *с кодоменом* $\mathtt{G}$*.* $\mathtt{Setup}$ *выдаёт кортеж параметров группы и хеш-функции,* $par := \left(p, \mathtt{G}, d, G, \left\{\mathcal{H}^s_j\right\}_{j=0}^{d-1}, \mathcal{H}^p\right)$.[4]
- $\mathtt{KeyGen} \to (\boldsymbol{sk}, \boldsymbol{pk})$*. При наличии запроса нового ключа* $\mathtt{KeyGen}$ *выбирает свежий секретный ключ и вычисляет соответствующий ему публичный ключ:*

$$\begin{aligned} \boldsymbol{sk} &= (z_0, z_1, \ldots, z_{d-1}) \leftarrow (\mathtt{F}_p^*)^d \\ \boldsymbol{pk} &:= \boldsymbol{sk} \circ \boldsymbol{G} = (Z_0, Z_1, \ldots, Z_{d-1}) \in \mathtt{G}^d \end{aligned}$$

  где $\boldsymbol{G} = (G, \ldots, G) \in \mathtt{G}^d$*.* $\mathtt{KeyGen}$ *выдаёт* $(\boldsymbol{sk}, \boldsymbol{pk})$*. Мы называем* z_0 *связующим ключом, остальные ключи* $\{z_j\}_{j=1}^{d-1}$ *вспомогательными ключами и обозначаем связующий ключ как* x*.*

[4] Следует отметить, что в данном случае разделение доменов может использоваться для того, чтобы разделить хеш-функцию $\mathcal{H}^s$ и построить каждую $\mathcal{H}^s_j$, определив $\mathcal{H}^s_j(x) := \mathcal{H}^s(j \mid\mid x)$.

– $\texttt{Sign}(m, Q, \boldsymbol{sk}) \to \{\perp_{\texttt{Sign}}, \sigma\}$. *В качестве входа* `Sign` *берёт сообщение* $m \in \{0,1\}^*$, *кольцо* $Q = (\boldsymbol{pk}_0, \ldots, \boldsymbol{pk}_{n-1})$ *для участников кольца* $\boldsymbol{pk}_i = (X_i, Z_{i,1}, \ldots, Z_{i,d-1}) \in \mathcal{G}^d$ *и секретный ключ* $\boldsymbol{sk} = (x, z_1, \ldots, z_{d-1}) \in (F_p^*)^d$. `Sign` *выполняет следующее:*

1. *Если* $Q \not\subseteq \mathcal{G}^{d\times n}$ *для некоторого* n, `Sign` *выводит* $\perp_{\texttt{Sign}}$ *и завершает процесс.*
2. *В ином случае* `Sign` *разбирает*[5] Q, *чтобы получить каждый* $\boldsymbol{pk}_i$. *Если публичный ключ, связанный со входом* $\boldsymbol{sk}$, *не является участником кольца* Q, `Sign` *выводит* $\perp_{\texttt{Sign}}$ *и завершает процесс.*
3. *В ином случае* `Sign` *находит такой подписывающий индекс* ℓ *чтобы* $\boldsymbol{pk}_\ell = \boldsymbol{sk} \circ (G, \ldots, G)$. `Sign` *единнобразно случайным образом выбирает* $\alpha \in F_p$, *выбирает* $\{s_i\}_{i\neq\ell} \in (F_p)^{n-1}$ *и вычисляет точки* $H_i = \mathcal{H}^p(X_i)$ *для каждого* i. `Sign` *вычисляет коэффициенты агрегирования* μ_X *и* $\{\mu_j\}_{j=1}^{d-1}$, *связующий тег* $\mathfrak{T}$, *вспомогательные элементы группы* $\{\mathfrak{D}_j\}_{j=1}^{d-1}$ *и агрегированные публичные ключи:*

$$\begin{aligned} \mathfrak{T} &:= xH_\ell & \{\mathfrak{D}_j\} &:= \{z_j H_\ell\} \\ \mu_X &:= \mathcal{H}_0^s(Q \mid\mid \mathfrak{T} \mid\mid \{\mathfrak{D}_j\}_{j=1}^{d-1}) & \mu_j &:= \mathcal{H}_j^s(Q \mid\mid \mathfrak{T} \mid\mid \{\mathfrak{D}_j\}_{j=1}^{d-1}) \\ W_i &:= \mu_X X_i + \sum_{j=1}^{d-1} \mu_j Z_{i,j} & \mathfrak{W} &:= \mu_X \mathfrak{T} + \sum_{j=1}^{d-1} \mu_j \mathfrak{D}_j \end{aligned}$$

а также агрегированный секретный ключ $w_\ell := \mu_X x + \sum_{j=1}^{d-1} \mu_j z_j$. *Для* $i = \ell, \ell+1, \ldots, \ell-1$ *(по модулю* n*)* `Sign` *вычисляет*

$$\begin{aligned} L_\ell &= \alpha G & R_\ell &= \alpha H_\ell & c_{\ell+1} &= \mathcal{H}_0^s(Q \mid\mid m \mid\mid L_\ell \mid\mid R_\ell) \\ L_i &= s_i G + c_i W_i & R_i &= s_i H_i + c_i \mathfrak{W} & c_{i+1} &= \mathcal{H}_0^s(Q \mid\mid m \mid\mid L_i \mid\mid R_i) \end{aligned}$$

и наконец вычисляет $s_\ell = \alpha - c_\ell w_\ell$.

4. `Sign` *выдаёт подпись* $\sigma = (c_0, s_0, s_1, \ldots, s_{n-1}, \mathfrak{T}, \{\mathfrak{D}_j\}_{j=1}^{d-1})$.

– $\texttt{Verify}(m, Q, \sigma) \to \{0,1\}$. `Verify` *берёт в качестве входа сообщение* m, *матрицу* $Q = (\boldsymbol{pk}_0, \ldots, \boldsymbol{pk}_{n-1})$ *и подпись* σ.

1. *Если* $Q \not\subseteq \mathcal{G}^{d\times n}$ *для некоторого* n *или если* $\sigma \notin F_p^{n'+1} \times \mathcal{G}^d$ *для некоторого* n', `Verify` *выводит* 0 *и завершает процесс. В ином случае, если* $n' \neq n$, `Verify` *выводит* 0 *и завершает процесс.*
2. `Verify` *разбивает*[6] $(\boldsymbol{pk}_0, \ldots, \boldsymbol{pk}_{n-1}) \leftarrow Q$ *для ключей* $\boldsymbol{pk}_i \in \mathcal{G}^d$ *в* $i \in [0, n-1]$ *и разбивает каждый публичный ключ* $(X_i, Z_{i,1}, \ldots, Z_{i,d-1}) \leftarrow \boldsymbol{pk}_i$. `Verify` *также разбивает* $(c_0, s_0, \ldots, s_{n-1}, \mathfrak{T}, \mathfrak{D}_1, \ldots, \mathfrak{D}_{d-1}) \leftarrow \sigma$. `Verify` *вычисляет каждый* $H_i = \mathcal{H}^p(X_i)$, *вычисляет коэффициенты*

[5] Следует отметить, что такой разбор всегда происходит успешно, если `Sign` проходит предыдущий этап без ошибок.

[6] Разбор всегда происходит успешно, если на предыдущем этапе выполнение `Verify` не завершается.

агрегирования и вычисляет агрегированные публичные ключи:

$$\mu_X := \mathcal{H}_0^s(Q \,||\, \mathfrak{T} \,||\, \{\mathfrak{D}_j\}_{j=1}^{d-1}) \qquad \mu_j := \mathcal{H}_j^s(Q \,||\, \mathfrak{T} \,||\, \{\mathfrak{D}_j\}_{j=1}^{d-1})$$
$$W_i := \mu_X X_i + \sum_{j=1}^{d-1} \mu_j Z_{i,j} \qquad \mathfrak{W} := \mu_X \mathfrak{T} + \sum_{j=1}^{d-1} \mu_j \mathfrak{D}_j$$

3. **`Verify`** *задаёт* $c'_0 := c_0$ *и для* $i = 1, 2, \ldots, n-1$, *вычисляет следующее:*

$$L_i := s_i G + c'_i W_i, \quad R_i := s_i H_i + c'_i \mathfrak{W}, \quad c'_{i+1} := \mathcal{H}_0^s\left(Q \,||\, m \,||\, L_i \,||\, R_i\right)$$

4. *Если* $c'_n = c_0$, **`Verify`** *выводит* 1, *в ином случае выводит* 0.

– **`Link`**$((m, Q, \sigma), (m', Q', \sigma')) \to \{0, 1\}$. *В качестве входа* **`Link`** *берёт два тройных значения сообщения-кольца-подписи.*
1. *Если* **`Verify`**$(m, Q, \sigma) = 0$ *или* **`Verify`**$(m', Q', \sigma') = 0$, **`Link`** *выводит* 0 *и завершает процесс.*
2. *В ином случае* **`Link`** *разбирает*[7] *подписи, чтобы получить отдельные связующие теги* $(\mathfrak{T}, \{\mathfrak{D}_j\}_j), (\mathfrak{T}', \{\mathfrak{D}'_j\}_j) \leftarrow \sigma, \sigma'$. **`Link`** *выводит* 1, *если* $\mathfrak{W} = \mathfrak{W}'$, *и* 0 *в противном случае.*

Данный вариант реализации предполагает наличие связываемости с *полным ключом* с использованием связующих тегов $\mathfrak{W}$: две подписи будут связаны не только в том случае, если они будут подписаны при помощи одних и тех же связующих и вспомогательных ключей, но также и с использованием одного и того же кольца. Мы можем заменить алгоритм `Link` связываемостью с *использованием одного ключа*:

– `Link`$((m, Q, \sigma), (m', Q', \sigma')) \to \{0, 1\}$. В качестве входа `Link` берёт два тройных значения сообщения-кольца-подписи.
1. Если `Verify`$(m, Q, \sigma) = 0$ или `Verify`$(m', Q', \sigma') = 0$, `Link` выводит 0 и завершает процесс.
2. В ином случае `Link` разбирает[8] подписи, чтобы получить отдельные связующие теги $(\mathfrak{T}, \{\mathfrak{D}_j\}_j), (\mathfrak{T}', \{\mathfrak{D}'_j\}_j) \leftarrow \sigma, \sigma'$. `Link` выводит 1, если $\mathfrak{T} = \mathfrak{T}'$, и 0 в противном случае.

5 Доказательства безопасности

Следующая лемма вытекает непосредственно из модели случайного оракула, которую мы использовали для $\mathcal{H}^s$.

Лемма 2. *Для любого* $Q \subseteq \mathcal{PK}$, *для любого приватного ключа* $sk = (x, \{z_j\}_j) \in Q$, *преобразование* $sk \mapsto \mu_X x + \sum_j \mu_j z_j$, *где* μ_X, μ_j, *вычисляются так же, как в Определении 10, и является устойчивым к конфликтам функцией.*

[7] Как и ранее в случае с `Verify`, такой разбор всегда будет успешным, если на предыдущем этапе выполнен `Link`.

[8] Как и ранее в случае с `Verify`, такой разбор всегда будет успешным, если на предыдущем этапе выполнение `Link` не завершается.

В Теореме 2 мы доказываем, что схему d-CLSAG нельзя подделать, демонстрируя, что если какой-либо PPT-алгоритм выдаёт некоторое действительное не использующее оракул тройное значение (m, Q, σ), связанное с участником анонимной группы в Q (злоумышленником или другим), алгоритм может быть выполнен заново с целью вычисления дискретного логарифма такого участника анонимной группы. Эта теорема является стандартной для подписей, подобных подписям Шнорра, в рамках программируемой модели случайного оракула.

Теорема 2 (Сложность задач дискретного логарифмирования в линейных комбинациях подразумевает невозможность подделки). *Если существует (t, ϵ, q)-алгоритм решения задачи невозможности подделки для схемы, соответствующей Определению 10, создающая запросы повреждённого оракула κ', то существует и $(2(t + t_0) + t_1, \epsilon \left(\frac{\epsilon}{q} - \frac{1}{2^\eta}\right) - \mu, \lfloor \frac{q}{d} \rfloor)$-алгоритм решения задачи $2d\kappa'$-дополнительного дискретного логарифма в линейных комбинациях в $\mathtt{G}$ для некоторого незначительного μ и некоторых t_0, t_1.*

Доказательство. Допустим, $\mathtt{A}$ является (t, ϵ, q)-алгоритмом решения задачи защиты от оговора в соответствии с Определением 8. Мы заворачиваем алгоритм $\mathtt{A}$ в алгоритм $\mathtt{B}$. Алгоритм $\mathtt{B}$ выполняет $\mathtt{A}$ в чёрном ящике, обрабатывая запросы оракула для $\mathtt{A}$. Затем алгоритм $\mathtt{B}$ выдаёт результат выполнения $\mathtt{A}$ с индексом idx. Таким образом, $\mathtt{B}$ подходит для применения в рамках леммы разветвления. Мы заворачиваем $\mathcal{F}^{\mathtt{B}}$ в главный алгоритм $\mathtt{M}$, то есть $(2(t + t_0) + t_1, \epsilon \left(\frac{\epsilon}{q} - \frac{1}{2^\eta}\right) - \mu, \lfloor \frac{q}{d} \rfloor)$-алгоритм решения задачи κ-дополнительного дискретного логарифма в линейных комбинациях в $\mathtt{G}$, где η определяется так же, как в Лемме 1.

Если $\mathtt{A}$ создаёт удачную подделку, каждый запрос верификации в форме $c_{\ell+1} = \mathcal{H}^s(m \mid\mid Q \mid\mid R_\ell \mid\mid L_\ell)$ происходит в транскрипте между $\mathtt{A}$ и случайным оракулом $\mathcal{H}^s$. На самом деле тройное значение подписи, создаваемое $\mathtt{A}$, проходит верификацию, поэтому каждый запрос $c_{\ell+1}$, независимо от того, сделан он вместе с запросами оракула в транскрипте или нет, должен быть сопоставлен с запросами случайного оракула, которые делает верификатор. Доказывающая сторона не может угадать результат такого запроса, пока не сделает его, за минимально вероятным исключением. Следовательно, если $\mathtt{A}$ выдаёт действительную подпись, все запросы верификации вычисляются фактическим запросом оракула. Формальное доказательство этого факта изложено в работе [14]. Поскольку все запросы верификации находятся посредством оригинальных запросов оракула, которые правильно упорядочены, существует первый запрос $\mathcal{H}^s$, сделанный $\mathtt{A}$ для вычисления запросов верификации, скажем, $c = \mathcal{H}^s(m \mid\mid Q \mid\mid R^* \mid\mid L^*)$. Это совсем не обязательно первый запрос, который делается $\mathcal{H}^s$ в целом. Допустим, это запрос k^{th}. Несмотря на то, что индекс кольца может быть не определён на момент первой выдачи этого запроса $\mathtt{A}$, к концу транскрипта этот индекс будет определён.

Мы строим B следующим образом. Мы даём B доступ к тем же оракулам, что и A. Любые запросы оракула, которые делаются A, пропускаются B к оракулам. Ответы записываются, а затем передаются обратно A. Алгоритм B работает путем нахождения двух индексов для увеличения выхода A. Во-первых, B находит индекс k запроса $\mathcal{H}^s$, соответствующий первому запросу верификации, вычисленному A, используемому для верификации предполагаемой подделки. Во-вторых, B проверяет транскрипт A, чтобы в транскрипте такой индекс соответствовал анонимной группе ℓ, чтобы $c = c_{\ell+1}$, $R^* = R_\ell$ и $L^* = L_\ell$. Теперь B выводит $idx = (k, \ell)$ вместе с тем, что выдаст A. Очевидно, B делает то же количество запросов повреждённого оракула, что и A.

Следует отметить, что B выполняется успешно всякий раз, когда выполняется A, за время, составляющее самое большее t, как и в случае A, за исключением некоторого дополнительного времени t_0, необходимого для поиска транскрипта для idx. Поскольку B подходит для использования в рамках леммы разветвления, мы можем использовать $\mathcal{F}^{\mathtt{B}}$ для построения M.

Алгоритм $\mathcal{F}^{\mathtt{B}}$ получает доступ к тем же оракулам, что и B, за исключением $\mathcal{H}^s$ и SO. Алгоритм $\mathcal{F}^{\mathtt{B}}$ моделирует запросы SO, которые делаются B путём внесения простых обратных поправок в $\mathcal{H}^s$, и моделирует другие запросы $\mathcal{H}^s$, сделанные B с использованием случайных последовательностей $\mathbf{h}, \mathbf{h}^*$, как было сказано в подпункте 3.1. Все другие запросы оракула, сделанные B, передаются вместе с $\mathcal{F}^{\mathtt{B}}$ фактическим оракулам, а затем отправляются обратно B.

Следует отметить, что $\mathcal{F}^{\mathtt{B}}$ выполняется за время $2(t + t_0)$ и (с вероятностью, равной по крайней мере $\epsilon\left(\frac{\epsilon}{q} - \frac{1}{2^\eta}\right)$) выдаёт пару действительных тройных значений (m, Q, σ), (m', Q', σ'). Сообщения и анонимные группы выбираются до момента разветвления в транскриптах, поэтому $m = m'$ и $Q = Q'$. Кроме того, $\mathcal{F}^{\mathtt{B}}$ делает самое большее $2\kappa'$ повреждённых запросов. Запросы в двух транскриптах являются самостоятельными, поскольку алгоритм разветвления выдаёт символ отказа $\perp$ и завершает процесс, если запросы $c_{\ell+1}$ одинаковы в обоих транскриптах.

$$c_{\ell+1} \leftarrow \mathcal{H}_0^s(m \mid\mid Q \mid\mid R_\ell \mid\mid L_\ell) \rightarrow c'_{\ell+1}.$$

Мы заворачиваем $\mathcal{F}^{\mathtt{B}}$ в алгоритм M, который решает задачу κ-дополнительного дискретного логарифма в соответствии с Определением 1 для $\kappa = 2 \cdot d \cdot \kappa'$. Алгоритм M имеет доступ к повреждённому оракулу и выполняет $\mathcal{F}^{\mathtt{B}}$ в чёрном ящике, пропуская запросы повреждённого оракула вместе с $\mathcal{F}^{\mathtt{B}}$. Алгоритм M находит следующую систему уравнений в транскриптах, проверяя запросы верификации.

$$R_\ell = s_\ell G + c_\ell X_\ell = s'_\ell G + c'_\ell X_\ell,$$
$$L_\ell = s_\ell H_\ell + c_\ell \mathfrak{W} = s'_\ell H_\ell + c'_\ell \mathfrak{W}.$$

Данный алгоритм M имеет достаточно информации для вычисления

$$W_\ell = \frac{s_\ell - s'_\ell}{c'_\ell - c_\ell} G, \mathfrak{W} = \frac{s_\ell - s'_\ell}{c'_\ell - c_\ell} H_\ell.$$

а следовательно, и приватный ключ $w = \frac{s_\ell - s'_\ell}{c'_\ell - c_\ell}$. Формально $\mathtt{M}$ выполняется следующим образом.

(1) $\mathtt{M}$ вводит набор публичных ключей запроса дискретного логарифма $S = \left\{\widetilde{pk}_i\right\}_{i=0}^{q-1}$.
(2) $\mathtt{M}$ разделяет ключи запроса на списки из d ключей

$$\begin{aligned} pk_0 =& (X_0, Z_{0,1}, \ldots, Z_{0,d-1}) := (\widetilde{pk}_0, \ldots, \widetilde{pk}_{d-1}) \\ pk_1 =& (X_1, Z_{1,1}, \ldots, Z_{1,d-1}) := (\widetilde{pk}_d, \ldots, \widetilde{pk}_{2d-1}) \\ & \vdots \end{aligned}$$

что позволяет получить $S := \{pk_i\}_{i=0}^{\lfloor \frac{q}{d} \rfloor}$.
(3) $\mathtt{M}$ выбирает две случайные последовательности $\mathbf{h}, \mathbf{h}'$, чтобы смоделировать ответы на запросы оракула для $\mathcal{F}^{\mathtt{B}}$.
(4) $\mathtt{M}$ выполняет $\mathcal{F}^{\mathtt{B}}$ в чёрном ящике, используя S в качестве входа. По получении повреждённого запроса от $\mathcal{F}^{\mathtt{B}}$ по некоторому pk_i $\mathtt{M}$ делает запрос $\mathtt{CO}$ по X_i и каждому $Z_{j,i}$, пропуская sk_i обратно $\mathcal{F}^{\mathtt{B}}$. Каждый повреждённый запрос, сделанный $\mathcal{F}^{\mathtt{B}}$, состоит из d повреждённых запросов, сделанных $\mathtt{CO}$ алгоритмом $\mathtt{M}$.
(5) Если выполнение $\mathcal{F}^{\mathtt{B}}$ проходит неудачно или же выполнение $\mathcal{F}^{\mathtt{B}}$ успешно со всеми нулевыми коэффициентами μ_X и μ_j, $\mathtt{M}$ выбирает случайный $w \in \mathtt{F}_p$, выбирает случайный поднабор ключей запроса $\left\{pk_j^*\right\}_j \subseteq S$, выбирает случайные коэффициенты $\{h_j\}_j$, выводит $w, \left\{pk_j^*\right\}_j, \{h_j\}_j$ и завершает процесс.
(6) В ином случае $\mathtt{M}$ получает два тройных значения с одним и тем же сообщением и кольцом, $(m, Q, \sigma), (m, Q, \sigma')$, по крайней мере, с одним ненулевым коэффициентом агрегирования. $\mathtt{M}$ вычисляет дискретный логарифм запроса $w = (c'_\ell - c_\ell)^{-1}(s_\ell - s'_\ell)$. $\mathtt{M}$ выводит $w, \left\{\mu_X, \{\mu_j\}_j\right\}$ и $\left\{X_\ell, \{Z_{\ell,j}\}_j\right\}$.

Обозначим как t_1 время, необходимое $\mathcal{M}$, чтобы проверить транскрипт, выполнить операции с полем и обработать запросы для $\mathcal{F}^{\mathtt{B}}$. Затем алгоритм $\mathtt{M}$ выполняется за время, составляющее самое большее $2(t + t_0) + t_1$.

Чтобы закончить доказательство, рассмотрим общую вероятность успеха и время выполнения $\mathtt{M}$. Поскольку $\mathtt{A}$ является (t, ϵ, q)-алгоритмом решения задачи невозможности подделки и это успешные подписи, должен быть по крайней мере один запрос, сделанный $\mathtt{SO}$, соответствующий неповреждённому ключу запроса, связанному с этими подписями. В частности, $w \cdot G = W_\ell = \mu_X X_\ell + \sum_j \mu_j Z_{\ell,j}$ для некоторого $(X_\ell, \{Z_{\ell,j}\}_j) \in Q$. Алгоритм $\mathtt{M}$ успешно решает задачу дискретного логарифмирования в линейных комбинациях всякий раз, когда $\mathcal{F}^{\mathtt{B}}$ успешно выполняет разветвление $\mathtt{B}$, и, по крайней мере, один коэффициент μ_X и μ_j является не нулём; мы обозначаем вероятность получения любого нулевого коэффициента как μ. Следует отметить, что вероятность получения μ в рамках модели случайного оракула

является незначительной. Таким образом, M выполняется в течение времени, составляющего самое большее $2(t+t_0)+t_1$ и с вероятностью успешного выполнения выше $\epsilon\left(\frac{\epsilon}{q}-\frac{1}{2^\eta}\right)-\mu$.

Доказательство Теоремы 2 демонстрирует, что действительность тройного значения подразумевает, что агрегированный приватный ключ w является дискретным логарифмом агрегированного связующего тега $\mathfrak{W}$ относительно H_ℓ, а также дискретным логарифмом агрегированного ключа W_ℓ относительно G. Таким образом, связующий тег действительной подписи должен быть связующим тегом, соответствующим по крайней мере одному участнику кольца за исключением ничтожной вероятности.

Следствие 1 (Отсутствие чужих связующих тегов). *Если существует PPT-алгоритм* $\mathtt{A}$*, выдающий тройное значение действительной подписи* (m, Q, σ) *в рамках схемы, соответствующей Определению 10, значит, существует участник кольца в* Q*, чей агрегированный ключ* W_ℓ *имеет тот же дискретный логарифм* w *относительно* G*, что* $\mathfrak{W}$ *имеет относительно* H_ℓ*, и этот* w *известен* $\mathtt{A}$ *(за исключением минимально возможной вероятности).*

Теорема 3. *Схема, представленная в Определении 10, является связываемой в соответствии с Определением 5 и Определением 6.*

Доказательство. Мы демонстрируем, что тройное значение действительной не использующей оракула подписи, соответствующей Определению 10 и условиям задачи повреждённого ключа из Определения 5, всегда связывается. Следовательно, любой алгоритм не сможет решить эту задачу, за исключением незначительной вероятности.

Предположим, что A, решая задачу связываемости ACST из Определения 5, выдаёт такую пару тройных значений действительной, не использующей оракула подписи (m, Q, σ), (m^*, Q^*, σ^*), что по крайней мере один ключ в $Q \cup Q^*$ является повреждённым или находится за пределами S. Этот алгоритм может быть разветвлён и выполнен заново, как описано выше, чтобы вычислить агрегированный приватный ключ, используемый при вычислении каждой подписи, скажем, w, w^*. Самое большее один ключ в $Q \cup Q^*$ является повреждённым или находится за пределами S. Поскольку A известен w, w является повреждённым или находится за пределами S, и подобным образом w^* также является повреждённым или находится за пределами S. Поскольку самое большее один ключ в $Q \cup Q^*$ может быть поврежденным или находиться за пределами S, мы делаем вывод, что $w = w^*$.

Поскольку агрегация ключей устойчива к конфликтам, а wG является агрегированным публичным ключом для некоторого публичного ключа $(X_\ell, \{Z_{\ell,j}\}_j) \in Q \cap Q^*$, w должен быть агрегирован из приватного ключа $(x_\ell, \{z_{\ell,j}\}_j)$ посредством устойчивой к конфликтам функции агрегирования. В обоих случаях связываемости, предполагающей использование одного ключа, и связываемости, предполагающей использование полного ключа,

связующие теги являются совершено одинаковыми. Следовательно, с вероятностью, равной 1, пара тройных значений (m, Q, σ), (m^*, Q^*, σ^*) являются связанными, и A не может решить задачу связываемости ACST, за исключением незначительной вероятности.

Подобным образом алгоритм, выдающий $q+1$ несвязанных подписей, может быть перезапущен для вычисления $2(q+1)$ подписей, на основе которых можно вычислить $q+1$ агрегированных ключей. Более того, если эти подписи являются несвязанными, значит, $q+1$ агрегированных ключей является раздельным, что нарушает схему связываемости по q голубиных отверстий.

Теорема 4. *Если существует (t, ϵ, q)-алгоритм решения задачи связываемой анонимности в соответствии с Определением 9 и конструкцией по Определению 10, то существует и $(t+t', \epsilon/2, q)$-алгоритм решения задачи RO-DDH в соответствии с Определением 2 для некоторого t'.*

Доказательство. Допустим, A является таким алгоритмом решения задачи связываемой анонимности. Мы строим алгоритм B, который будет выполнять алгоритм A в чёрном ящике и будет алгоритмом решения задачи RO-DDH, являясь при этом запросчиком для A; алгоритм передаёт запросы случайного оракула $\mathcal{H}^p$ собственному запросчику, выбирает между запросами случайного оракула $\mathcal{H}_0^s$ и $\{\mathcal{H}_j^s\}$ и моделирует запросы подписывающего оракула путём обратной правки. Мы допускаем, что алгоритм B ведёт внутренние таблицы с целью обеспечения соответствия между запросами случайного оракула, необходимыми для моделирования запросов подписывающего оракула.

Алгоритм B выполняется следующим образом:

- Алгоритм B получает набор кортежей $\{(R_i, R_i', R_i'')\}_{i=0}^{q-1}$ от своего запросчика и единообразно случайным образом выбирает бит $b' \in \{0,1\}$. Следует отметить, что алгоритму B не известно, являются ли кортежи тройными значениями RO-DDH или нет, так как его запросчик, чтобы определить это, выбирает секретный бит $b \in \{0,1\}$ единообразно и случайным образом.
- Для всех $i \in [0, q)$ B определяет $X_i := R_i$ и записывает распределение оракула $\mathcal{H}^p$ как $\mathcal{H}^p(X_i) = R_i'$. Алгоритм единообразно случайным образом выбирает $\{z_{i,j}\}_{j=1}^{d-1}$ из $\mathtt{F}_p$ и строит набор публичных ключей $S := \{(X_i, z_{i,1}G, \ldots, z_{i,d-1}G)\}_{i=0}^{q-1}$. B передаёт набор S алгоритму A.
- Алгоритм A возвращает индексы $0 \le i_0, i_1 < q$ алгоритму B.
- Алгоритм B получает запросы подписывающего оракула в форме $\mathtt{SO}(m, Q, pk)$, где $0 \le \ell < q$ является индексом $pk \in Q$, $pk \in S$, а $|Q| = n$. Есть два случая, в которых алгоритм B моделирует ответ оракула, выбирая между запросами оракула $\mathcal{H}_0^s$ и $\{\mathcal{H}_j^s\}$:
  - В том случае, если $\{pk_{i_0}, pk_{i_1}\} \not\subset Q$ или $pk \notin \{pk_{i_0}, pk_{i_1}\}$, алгоритм B моделирует подписывающий оракул при помощи ключа pk.

- В ином случае существует такой бит $c \in \{0,1\}$, что $pk = pk_{i_c}$. В этом случае B задаёт значение $c' := c \oplus b'$ и моделирует подписывающий оракул, используя ключ $pk_{i_{c'}}$. То есть, если $b' = 0$, алгоритм B моделирует подпись при помощи запрошенного ключа из набора индексов, предоставленных участником. Если же $b' = 1$, B моделирует подпись при помощи другого ключа.

В любом из случаев B разбирает набор публичных ключей Q, предоставленный алгоритмом A. Для любого ключа $pk_i := (X'_i, Z'_{i,1}, \ldots, Z'_{i,d-1}) \in Q \setminus S$ он передаёт запросы оракула своему запросчику, чтобы получить $\mathcal{H}^p(X'_i)$. Затем B моделирует подпись:

- Определяет распределение $\pi : [0,n) \to [0,q) \cup \{\perp\}$ для индексов элементов Q по соответствующим элементам S (или возвращает выделенный символ сбоя $\perp$ для индексов, не распределённых по элементам S), делает $0 \le \ell < n$ индексом $pk \in Q$.
- Единообразно случайным образом выбирает $c_\ell, \{s_i\}_{i=0}^{n-1} \in \mathtt{F}_p$.
- Поскольку в соответствии с конструкцией $pk \in S$, $\pi(\ell) \neq \perp$. Алгоритм задаёт такие значения $\mathfrak{T} := R''_{\pi(\ell)}$ и $\{\mathfrak{D}_j\}_{j=1}^{d-1}$, что каждый $\mathfrak{D}_j := z_{\pi(\ell),j}\mathcal{H}^p(X_{\pi(\ell)})$.
- Определяет следующее:

$$\begin{aligned}
\mu_X &\leftarrow \mathcal{H}^s_0(Q, \mathfrak{T}, \{\mathfrak{D}_j\}) \\
\mu_j &\leftarrow \mathcal{H}^s_j(Q, \mathfrak{T}, \{\mathfrak{D}_j\}) \text{ for } j \in (0,d) \\
\mathfrak{W}_i &:= \begin{cases} \mu_X X_{\pi(i)} + \sum_j \mu_j Z_{\pi(i),j} & (\pi(i) \neq \perp) \\ \mu_X X'_i + \sum_j \mu_j Z'_{i,j} & (\pi(i) = \perp) \end{cases} \\
W &:= \mu_X \mathfrak{T} + \sum_j \mu_j \mathfrak{D}_j
\end{aligned}$$

- Для каждого $i = \ell, \ell+1, \ldots, n-1, 0, \ldots, \ell-1$ (то есть индекса по модулю n) определяет следующее:

$$\begin{aligned}
L_i &:= s_i G + c_i \mathfrak{W}_i \\
R_i &:= \begin{cases} s_i \mathcal{H}^p(X_{\pi(i)}) + c_i W & (\pi(i) \neq \perp) \\ s_i \mathcal{H}^p(X'_i) + c_i W & (\pi(i) = \perp) \end{cases} \\
c_{i+1} &\leftarrow \mathcal{H}^s_0(Q, m, L_i, R_i)
\end{aligned}$$

- Алгоритм B возвращает алгоритму A кортеж $(c_0, \{s_i\}, \mathfrak{T}, \{\mathfrak{D}_j\})$.

– Алгоритм A возвращает алгоритму B бит b^*.
– Если $b^* = b'$, B возвращает своему запросчику 0. В ином случае возвращает 1.

В этом случае алгоритм B точно решает задачу RO-DDH, когда он правильно угадывает бит b, выбранный его запросчиком. Следовательно, $\mathbb{P}[\mathtt{B}\text{ wins}] = \frac{1}{2}\mathbb{P}[\mathtt{B} \to 0 | b = 0] + \frac{1}{2}\mathbb{P}[\mathtt{B} \to 1 | b = 1]$.

Если $b = 1$, значит, запросчик RO-DDH предоставил случайные точки $\{R''_i\}$, которые B использовал в своих смоделированных подписях, и, таким

образом, алгоритму A не остаётся ничего, кроме как случайно определить b'. Так как однозначно $\mathtt{B} \to 1$, если A не решает задачи анонимной связываемости, то мы получаем $\mathbb{P}[\mathtt{B} \to 1 | b = 1] = \frac{1}{2}$.

С другой стороны, если $b = 0$, значит, запросчик RO-DDH предоставил структурированные кортежи, которые B использовал в своих смоделированных подписях, а алгоритм A успешно решил задачу связываемой анонимности с совсем не ничтожным преимуществом ϵ по случайному оракулу. Если однозначно $\mathtt{B} \to 0$, когда A решает задачу анонимной связываемости, мы получаем $\mathbb{P}[\mathtt{B} \to 0 | b = 0] = \frac{1}{2} + \epsilon$.

Это означает, что алгоритм B решает задачу RO-DDH с вероятностью $\mathbb{P}[\mathtt{B}\ \text{wins}] = \frac{1}{2} + \frac{\epsilon}{2}$ и имеет совсем не ничтожное преимущество $\frac{\epsilon}{2}$. Далее, алгоритм B завершает выполнение за дополнительное время t', необходимое для моделирования запросов оракула и выполнения поиска. Следовательно, алгоритм B является $(t + t', \epsilon/2, q)$-алгоритмом решения задачи RO-DDH.

6 Эффективность

Рассмотрим эффективность по необходимому пространству и времени в соответствии с Определением 10. Мы не учитываем какую-либо дополнительную информацию, которая обычно передаётся вместе с подписью, такую как представление участников кольца.

Подпись d-CLSAG с размером кольца, равным n, содержит $n + 1$ поле элементов и d элементов группы. Размером подписи является $k_s(n+1)+k_p d$, где k_s описывает размер элементов поля, а k_p описывает размер элементов группы.

Чтобы проверить сложность по времени верификации, допустим, что t_s и t_p обозначают временную сложность оценки функций хеширования до скалярных величин $\mathcal{H}^s$ и оценки функции хеширования до точки $\mathcal{H}^p$, соответственно. Допустим, $t^{(i)}$ является временной сложностью оценки линейной комбинации i членов; при использовании специальных алгоритмов для мультискалярного умножения [22,18] такая линейная комбинация может быть оценена гораздо быстрее, чем путём простого почленного вычисления. Следует отметить, что также можно кешировать множество групповых элементов, которые будут использоваться повторно при верификации, для более быстрой оценки линейной комбинации, но мы не выделяем эту возможность в данной работе. В итоге сложность верификации d-CLSAG составляет $(n + d)t_s + nt_p + 2nt^{(d+1)}$.

Чтобы провести сравнение с эффективностью варианта реализации MLSAG, предложенного в работе [16], отметим, что 2-CLSAG обладает той же функциональностью, что и подпись MLSAG (которая является 2-LRS). Подпись MLSAG, используемая таким образом, создаёт $2n + 1$ элементов поля и 1 элемент группы.

Нами был написан тест на C++ с использованием подгруппы `ed25519` порядка, равного простому числу, который мы использовали для оценки

процесса подписания и верификации MLSAG и 2-CLSAG на 2,1 ГГц процессоре Opteron. В Таблице 1 приводятся результаты тестирования при различном размере кольца. В частности, следует отметить, что при меньшем размере анонимной группы 2-CLSAG работала быстрее, чем MLSAG. Тем не менее при очень больших размерах колец MLSAG была быстрее из-за дополнительных вычислений, необходимых для вычисления коэффициентов агрегирования и добавления ключей. Несмотря на конечную неэффективность, хотелось бы отметить, что требования к линейному пространству в целом исключают возможность использования на практике колец большого размера, что делает схему 2-CLSAG эффективным усовершенствованием MLSAG как с точки зрения пространства, так и с точки зрения времени.

	Верификация		Подписание	
Анонимная группа	MLSAG	CLSAG	MLSAG	CLSAG
2	2.4	2.0	2.3	2.7
4	4.7	4.0	4.6	4.6
8	9.5	7.8	9.4	8.5
16	18.9	15.9	18.9	16.5
32	37.8	32.3	37.8	33.0
64	75.4	67.5	75.9	68.3
128	150	147	151	148
256	301	344	303	346

Таблица 1. Время подписания и верификации (мсек) при использовании схем MLSAG и 2-CLSAG

Список литературы

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys (Подписи, построенные по схеме «1 из n» на основе множества ключей). In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 415–432. Springer (2002)
2. Au, M.H., Chow, S.S., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited (Пересмотр схемы построения коротких связываемых кольцевых подписей). In: European Public Key Infrastructure Workshop. pp. 101–115. Springer (2006)
3. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: Logarithmic-size, no setup—from standard assumptions (Кольцевые подписи: логарифмический размер, отсутствие настроек на основе стандартных допусков). In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 281–311. Springer (2019)
4. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von Neumann architecture (Компактные неинтерактивные доказательства с нулевым разглашением для архитектуры Вон Ньюмана). In: 23rd {USENIX} Security Symposium ({USENIX} Security 14). pp. 781–796 (2014)

5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles (Кольцевые подписи: более строгие определения и конструкции без случайных оракулов). In: Theory of Cryptography Conference. pp. 60–79. Springer (2006)
6. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more (Bulletproofs: короткие доказательства для конфиденциальных транзакций и другого применения). In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE (2018)
7. Fujisaki, E.: Sub-linear size traceable ring signatures without random oracles (Отслеживаемые кольцевые подписи сублинейного размера без случайных оракулов). In: Cryptographers' Track at the RSA Conference. pp. 393–415. Springer (2011)
8. Fujisaki, E., Suzuki, K.: Traceable ring signature (Отслеживаемая кольцевая подпись). In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
9. Groth, J.: On the size of pairing-based non-interactive arguments (По вопросу размера неинтерактивных аргументов на базе попарного объединения). In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 305–326. Springer (2016)
10. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs (Обновляемые и универсальные общие последовательности, применимые в zk-SNARKs). In: Annual International Cryptology Conference. pp. 698–728. Springer (2018)
11. Gu, K., Wu, N.: Constant size traceable ring signature scheme without random oracles (Схема отслеживаемой кольцевой подписи с постоянным размером, не использующая случайных оракулов). IACR Cryptology ePrint Archive **2018**, 288 (2018)
12. Hoffmann, M., Klooß, M., Rupp, A.: Efficient zero-knowledge arguments in the discrete log setting, revisited (Эффективные аргументы с нулевым разглашением в рамках задачи дискретного логарифмирования, новая редакция). In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2093–2110 (2019)
13. Lai, R.W.F., Ronge, V., Ruffing, T., Schröder, D., Thyagarajan, S.A.K., Wang, J.: Omniring: Scaling private payments without trusted setup (Omniring: масштабирование анонимных платежей при отсутствии доверенных настроек). In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 31–48. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3345655, https://doi.org/10.1145/3319535.3345655
14. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups (Применение связываемой подписи спонтанной анонимной группы в специально создаваемых группах). In: Australasian Conference on Information Security and Privacy. pp. 325–335. Springer (2004)
15. Möser, M., Soska, K., Heilman, E., Lee, K., Heffan, H., Srivastava, S., Hogan, K., Hennessey, J., Miller, A., Narayanan, A., et al.: An empirical analysis of traceability in the Monero blockchain (Эмпирический анализ отслеживаемости в блокчейне Monero). Proceedings on Privacy Enhancing Technologies **2018**(3), 143–163 (2018)
16. Noether, S., Mackenzie, A., et al.: Ring confidential transactions (Кольцевые конфиденциальные транзакции). Ledger **1**, 1–18 (2016)

17. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log (Подписи на основе дискретного логарифма не могут быть эквивалентны дискретному логарифму). In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 1–20. Springer (2005)
18. Pippenger, N.: On the evaluation of powers and monomials (По вопросу оценки степеней и мономиалов). SIAM Journal on Computing **9**(2), 230–250 (1980)
19. Quesnelle, J.: On the linkability of Zcash transactions (По вопросу связываемости транзакций Zcash). arXiv preprint arXiv:1712.01210 (2017)
20. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret (Как раскрыть секрет). In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 552–565. Springer (2001)
21. Schnorr, C.P.: Efficient signature generation by smart cards (Создание эффективных подписей с использованием смарт-карт). Journal of cryptology **4**(3), 161–174 (1991)
22. Straus, E.G.: Addition chains of vectors (problem 5125) (Дополнительные цепочки векторов (задача 5125)). American Mathematical Monthly **70**(806-808), 16 (1964)
23. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures (Отделимые связываемые пороговые кольцевые подписи). In: International Conference on Cryptology in India. pp. 384–398. Springer (2004)
24. Van Saberhagen, N.: CryptoNote v 2.0 (2013)
25. Yang, X., Wu, W., Liu, J.K., Chen, X.: Lightweight anonymous authentication for ad hoc group: A ring signature approach (Лёгкая анонимная аутентификация специально создаваемых групп: подход к созданию кольцевых подписей). In: International Conference on Provable Security. pp. 215–226. Springer (2015)
26. Zhang, F., Kim, K.: ID-based blind signature and ring signature from pairings (Слепая подпись на базе ID и кольцевая подпись на базе попарного объединения). In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 533–547. Springer (2002)