ИССЛЕДОВАТЕЛЬСКИЙ БЮЛЛЕТЕНЬ MRL-0003

Monero вовсе не так загадочна

Шен Ноезер (Shen Noether)* и Саранг Ноезер (Sarang Noether)†
Исследовательская лаборатория Monero (Monero Research Lab)
25 сентября 2014

Аннотация

Недавно мы стали свидетелями смутных страхов, распространившихся по сети интернет и связанных с исходным кодом и протоколом CryptoNote. Эти страхи были основаны на том, что протокол CryptoNote сложнее протокола, например, лежащего в основе Bitcoin. Цель настоящего бюллетеня состоит в том, чтобы попытаться развеять некоторое недопонимание, а также снять завесу таинственности с кольцевых подписей Monero. Я начну со сравнения математики, лежащей в основе кольцевых подписей Monero (как это описано в [CN]), и математики, о которой говорится в [FS], и на которой основан сам протокол CryptoNote. После этого я сравню математику кольцевых подписей с тем, что фактически заложено в кодовую базу CryptoNote.

1 Происхождение CryptoNote

Как было отмечено в работе ([CN], 4.1), написанной Саберхагеном (Saberhagen), история групповых кольцевых подписей начинается в 1991 г. [CH], и в течение последних двух десятилетий различные исследовали изучали самые разные схемы построения кольцевых подписей. Как было заявлено в ([CN] 4.1), основная схема кольцевой подписи, используемая CryptoNote, основана на [FS] с некоторыми изменениями, необходимыми для адаптации под блокчейн технологию.

1.1 Отслеживаемые кольцевые подписи

В [FS] Фуджисаки (Fujisaki) и Сузуки (Suzuki) была предложена схема кольцевой подписи, позволяющая «анонимно делиться секретными данными без риска доверительной передачи личных данных». Отсутствие необходимости в такой доверительной передачи личных данных позволяет пользователям кольцевой подписи скрываться в составе группы на более высоком уровне отсутствия необходимости в доверии, если сравнивать со схемами, подразумевающими наличие менеджера группы.

В случае со схемами, подразумевающими наличие менеджера группы, такими как оригинальная схема кольцевых подписей, описанная в [CH], назначенное доверенное лицо отвечает за безопасность секретных данных участников группы. Несмотря на анонимность, такие схемы полагаются, безусловно, на честность менеджера. В результате наличие группового менеджера с точки зрения криптовалют в значительной мере означает то же, что и наличие доверенной организации или узла, смешивающего ваши монеты.

И наоборот, схема отслеживаемых кольцевых подписей, предложенная в [FS], не требует наличия группового менеджера. Таким образом, по своей сути она более ненадежна, и потенциально имеет дру-

*lab@monero.cc
†sarang.noether@protonmail.com

гие уязвимости, которые в свою очередь, [CN] пытается смягчить с помощью технологии реализуемых в блокчейне.

Согласно [FS] в случае со схемой отслеживаемых кольцевых подписей существует четыре формальных требования к безопасности:

- *Публичная отслеживаемость.* Любой, создающий две подписи для различных сообщений с одним тегом, может быть отслежен. (В случае с протоколом CryptoNote, если пользователь решает не использовать одноразовый ключ для каждой транзакции, они могут быть отслежены. Тем не менее, если пользователь решает сохранить анонимность, он станет использовать одноразовый ключ. Таким образом, как было указано на странице 5 [CN], свойство отслеживаемости ослаблено в CryptoNote).

- *Связываемость тегов.* Каждые две подписи, сгенерированные одним и тем же подписантом с одним тегом, могут быть связаны. (Этот аспект CryptoNote касается каждой транзакции, использующей образ ключа, исключающий возможность двойной траты).

- *Анонимность.* Так как подписант не подписывает два различных сообщения одним и тем же тегом, личность подписанта ничем не отличается от других возможных участников кольца. Кроме того, любые две подписи, сгенерированные с двумя различными тегами, никогда не получится связать. (С точки зрения CryptoNote, если подписант попытается использовать один и тот же образ ключа более одного раза, то его можно будет выделить среди участников группы. Аспект несвязываемости сохраняется как ключевая часть CryptoNote).

- *Исключаемость.* Честный участник кольца не может быть обвинён в том, что дважды подписывался, используя один тег. Другими словами, должна обеспечиваться невозможность подделки тега, соответствующего секретному ключу другого человека. (С точки зрения CryptoNote это означает, что образы ключей подделать невозможно).

Кроме того, на странице [FS] предлагается протокол построения кольцевых подписей, эквивалентный алгоритму построения кольцевых подписей CryptoNote, описанному на страницах 9-10 [CN]. Тем не менее стоит отметить, что [FS] является публичной информацией, прошедшей независимый анализ и встречающейся в Lecture Notes издания Computer Science, в отличие от типичных описаний криптовалютных протоколов, в случае с которыми невозможно понять, прошли они независимую экспертизу или нет.

1.2 Отслеживаемость и CryptoNote

Оригинальный алгоритм построения отслеживаемых кольцевых подписей, описанный в [FS], допускает многократное использование тега L, соответствующего подписи. Тем не менее многократное использование тега позволяет отследить пользователя; другими словами, индекс подписанта можно будет выделить среди группы других подписывающихся пользователей. Тут следует отметить, что из-за свойства исключаемости протокола ([FS] 5.6, [CN], A2) украсть ключи подобным образом не получится, если только злоумышленник не сможет решить задачу дискретного логарифмирования в группе точек эллиптической кривой (ECDLP), на которой основана значительная часть современной криптографии ([Si] XI.4).

Процесс отслеживания тега L, использованного более одного раза, описан в ([FS] (страница 10)). Тем не менее, в случае с протоколом CryptoNote, образы ключей (теги), использованные более одного раза, отклоняются блокчейном как попытка двойной траты, а следовательно, отслеживаемость не является аспектом протокола CryptoNote.

1.3 Связываемость тегов в случае с CryptoNote

По сути, аспект связываемости тегов протокола отслеживаемых кольцевых подписей является тем, что не позволяет совершать двойные траты в случае с транзакциями CryptoNote. В соответствующих протоколах это обозначается как «Trace» ([FS] и «LNK» в работе по CryptoNote. При этом важно, что всё, что необходимо для того, чтобы отследить образы ключей, которые уже использовались ранее, это верификация того, что образ ключа более не будет использован.

Если один образ ключа будет обнаружен в блокчейне до другого образа ключа, то второй образ ключа будет считаться попыткой двойной траты при проведении транзакции. Образ ключа невозможно подделать, он исключителен, и двойную трату фактически будет совершать тот же человек, а не кто-то другой, пытающийся украсть кошелёк.

2 Одноразовые кольцевые подписи (математика)

В основе безопасности схемы кольцевой подписи, описанной в ([FS] 10, [CN] 10) и реализованной в исходнике CryptoNote, лежат известные свойства обеспечения безопасности кривой Curve25519. Следует отметить, что это именно та кривая, которая используется OpenSHH 6.5, Tor, Apple iOS и многими другими [‡] системами безопасности.

2.1 Скрученные кривые Эдвардса

Базовая безопасность алгоритма построения кольцевых подписей CryptoNote обеспечивается ECDLP ([Si], XI.4) по скрученной кривой Эдвардса ed25519. Свойства безопасности кривой ed25519 описаны в работе [Bern] выдающимся криптографом Дэниелом Бернштейном (Daniel Bernstein), а также в ([BCPM]) командой Microsoft Research. Бернштейн отмечает, что, в случае с кривой ed25519, «любой из известных видов атаки требует больших затрат, чем поиск методом перебора применительно к типовому зашифрованному 128-битному секретному ключу».

Кривая ed25519 является сингулярной кривой первого типа с групповым законом и описывается уравнением $-x^2 + y^2 = 1 + \left(\frac{-121665}{121666}\right) x^2y^2$. Эта кривая рассматривается через конечное поле $\mathbb{F}_q$, $q = 2^{255} - 19$. Для тех, кто незнаком с алгебраической геометрией, поясняем: алгебраическая кривая считается неким одномерным пространством, состоящим из всех точек (x, y), удовлетворяющих условия приведённого выше уравнения. Все точки также рассматриваются по модулю q. В силу своего типа кривая ed25519 имеет «групповую структуру», которая с точки зрения нашей темы означает, что если $P = (x_1, y_1)$ является точкой на кривой, а $Q = (x_2, y_2)$ является другой точкой на кривой, то эти точки могут быть сложены (или вычтены), а их сумма (или разница), $P + Q$ (or $P - Q$) также будут на кривой. Сложение **не является** простым $x_1 + x_2$ и $y_1 + y_2$, а вместо этого точки складываются по следующему правилу:

$$P + Q = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 + x_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

где $d = \left(\frac{-121665}{121666}\right)$ ([BBJLP] 6, [BCPM]). Для заинтересованного читателя математическая основа кривых первого типа подробно рассматривается в работе [Si].

На основе приведённого выше примера, мы можем вычислить $P+P$ для любой такой точки. Чтобы сократить систему представления, мы полагаемся на нашу алгебраическую интуицию и обозначаем

[‡] `http://ianix.com/pub/curve25519-deployment.html`

$2P = P + P$. Если $n \in \mathbb{Z}$, то nP обозначает повторную сумму

$$\underbrace{P + P + \cdots + P}_{n\ times}$$

с применением вышеуказанного закона нелинейного сложения. В качестве примера, чем это отличается от обычного сложения, рассмотрим следующую систему уравнений:

$$\begin{aligned} aP + bQ &= X \\ aP' + bQ' &= Y \end{aligned}$$

где a, b, c, d являются целыми числами, а P, Q, X являются точками. Если бы это была стандартная система линейных уравнений, то можно было бы использовать линейные алгебраические методы, чтобы без каких-либо затруднений вычислить a и b, допустив, что P, Q, X, Y, P' и Q' известны. Тем не менее, если a и b окажутся слишком малыми, то вышеуказанная система окажется крайне сложной для решения с использованием закона сложения ed25519. Например, если $a = 1$ и $b = 1$, то у нас получится:

$$\begin{aligned} \left(\frac{x_P y_Q + y_P x_Q}{1 + d x_P x_Q y_P y_Q}, \frac{y_P y_Q + x_P x_Q}{1 - d x_P x_Q y_P y_Q}\right) &= (x_X, y_X) \\ \left(\frac{x_{P'} y_{Q'} + y_{P'} x_{Q'}}{1 + d x_{P'} x_{Q'} y_{P'} y_{Q'}}, \frac{y_{P'} y_{Q'} + x_{P'} x_{Q'}}{1 - d x_{P'} x_{Q'} y_{P'} y_{Q'}}\right) &= (x_Y, y_Y) \end{aligned}$$

Таким образом в реальности эта система состоит из 4 нелинейных уравнений. Чтобы убедиться в том, что вычислить a и b довольно сложно, попытайтесь записать приведённые выше системы, допустив, что $a = 2, b = 1$. Сразу станет ясно, что задача очень сложна, если выбираются очень большие значения a и b. Способы эффективного решения этой системы для больших значений a и b пока отсутствуют.

Рассмотрим следующую задачу. Предположим, у вашего друга есть случайное целое число q, и он вычисляет qP, используя показанную выше форму сложения. Затем ваш друг сообщает вам значение координат x и y, $qP = (x, y)$, но не само значение q. Не спросив его, как вы сможете узнать значение q? Простой подход подразумевает, что вы начнёте с P и будете складывать $P + P + P...$ до тех пор, пока не достигните qP (о чём вы узнаете, поскольку вы остановитесь на (x, y)). Но если значение q будет слишком большим, то этот простой подход займёт миллиарды лет, даже если использовать современные суперкомпьютеры. Отталкиваясь от того, что на сегодняшний день известно математикам о задаче и количестве возможных q, как правило, ни один из известных видов атаки в практическом смысле не сможет дать лучших результатов, чем метод прямого подбора.

В случае с CryptoNote ваш секретный ключ в значительной мере является просто очень-очень большим числом x (в других случаях, см. раздел 3.3.3, мы берём значение x как кратное 8). На кривой ed25519 есть специальная точка G, называемая «базовой точкой», которая используется в качестве стартовой точки для получения xG. Ваш публичный ключ будет просто xG, а описанная выше задача защитит вас от того, что кто-то использует известную информацию для определения приватного ключа.

2.2 Связь с протоколом Диффи-Хеллмана

В кольцевую подпись входят следующие уравнения, включающие в себя ваш секретный ключ x:

$$\begin{aligned} P &= xG \\ I &= x\mathcal{H}_p(P) \\ r_s &= q_s - c_s x. \end{aligned}$$

В данном случае s является числом, присваивающим индекс групповой подписи вашему публичному ключу, а $\mathcal{H}_p(P)$ является хеш-функцией, которая детерминировано выбирает точку P для другой точки $P' = x'G$, где x' является другим очень большим равномерно выбираемым числом. Значение q_s выбирается единообразно в случайном порядке, а c_s вычисляется при помощи другого уравнения, использующего случайные значения. CryptoNote использует определённую хеш-функцию, Keccak1600, которую также использует, например, SHA-3; на данный момент эта функция считается безопасной ([FIPS]). Использование CryptoNote одной хеш-функции соответствует стандартной процедуре объединения отдельных случайных оракулов (в доказательствах безопасности в [FS], например) в одиночную сильную хеш-функцию.

Приведённые выше уравнения могут быть записаны следующим образом:

$$\begin{aligned} P &= xG \\ P' &= xx'G' \\ r_s &= q_s - c_s x \end{aligned}$$

Решение двух верхних уравнений эквивалентно ECDH (о чём говорилось в предыдущей технической записке ([SN])) и имеет ту же практическую сложность, что и ECDLP. Несмотря на то, что уравнения выглядят линейно, на самом деле они далеко не линейны, так как используют сложение, описанное в пункте 2.1 выше. Третье уравнение (с неизвестными q_s и x) имеет сложность нахождения случайного числа (либо q_s либо x) в $\mathbb{F}_q$, очень большом конечном поле; это невозможно сделать. Следует отметить, что так как в третьем уравнении присутствуют два неизвестных, его объединение с двумя предыдущими уравнениями не поможет. Злоумышленнику придётся вычислить хотя бы одно из случайных чисел q_s или x.

2.3 Время, необходимое для нахождения q или x

Поскольку q и x являются случайными очень большими числами в поле $\mathbb{F}_q$, при этом $q = 2^{255} - 19$ (сгенерированным 32-байтным целым числом), это эквивалентно 128-битному уровню безопасности ([BCPM]), предполагающему, что для вычисления с применением современных суперкомпьютеров понадобятся миллиарды лет.

2.4 Обзор доказательств в приложении

В приложении CryptoNote приводится четыре доказательства четырёх основных свойств, необходимых для обеспечения безопасности схемы одноразовой кольцевой подписи:

- Связываемость (защита от двойной траты);
- Исключительность (защита секретного ключа);
- Невозможность подделки (защита от создания фальшивых кольцевых подписей);
- Анонимность (возможность спрятать транзакцию среди других транзакций).

Эти теоремы в значительной степени идентичны тем, что описаны в [FS], и демонстрируют, что протокол создания кольцевых подписей соответствует вышеуказанным признакам. Первая теорема демонстрирует, что только те секретные ключи, которые соответствуют публичным ключам в группе, могут использоваться для создания подписи для такой группы. Для решения двух одновременных (нелинейных) уравнений на эллиптической кривой, которые, как объяснялось в пункте 2.2, практически

нерешаемы, используется ECDLP. Вторая теорема использует то же обоснование, но демонстрирует, что для создания поддельной подписи, которая пройдёт верификацию, понадобится решить ECDLP. Третья и четвёртая теоремы взяты непосредственно из [FS].

3 Одноразовые кольцевые подписи (применение)

Чтобы понять, как CryptoNote реализует одноразовые кольцевые подписи, мною была построена модель на языке Python на базе Crypto-ops.cpp и Crypto.cpp из исходного кода Monero при помощи простых операций на скрученной кривой Эдвардса (взятых из кода, написанного Бернштейном), вместо очевидно разумно оптимизированных операций, существующих в коде CryptoNote. Функции описаны в комментариях к коду ниже. Использование модели позволяет получить рабочую кольцевую подпись, которая немного отличается от кольцевых подписей Monero только небольшой разницей в хешировании и компоновкой используемых библиотек. Полная версия кода выложена по следующему адресу:

`https://github.com/ShenNoether/MiniNero/blob/master/MiniNero.py`

Следует отметить, что большинство важных вспомогательных функций в Crypto-ops.cpp в исходнике CryptoNote взяты из эталонного варианта реализации Curve25519. Этот эталонный вариант реализации был написан Мэтью Демпски (Matthew Dempsky) (Mochi Media, а сейчас Google)[§].

Кроме того, после сравнения кода, написанного на Python, со статьёй и, в свою очередь, сравнения кода, написанного на Python, с фактическим исходником Monero стало очевидно, что функции, вроде `generate_ring_sig`, делают то, что и должны в соответствии с протоколом, описанным в «белой книге». Например, вот алгоритм создания кольцевой подписи, используемый в исходнике CryptoNote:

Алгоритм 1 Кольцевые подписи

$i \leftarrow 0$
while $i < numkeys$ **do**
  if $i = s$ **then**
    $k \leftarrow random\ \mathbb{F}_q\ element$
    $L_i \leftarrow k \cdot G$
    $R_i \leftarrow k \cdot \mathcal{H}_p(P_i)$
  else
    $k1 \leftarrow random\ \mathbb{F}_q\ element$
    $k2 \leftarrow random\ \mathbb{F}_q\ element$
    $L_i \leftarrow k_1 P_i + k_2 G$
    $R_i \leftarrow k_1 I + k_2 \mathcal{H}_p(P_i)$
    $c_i \leftarrow k_1$
    $r_i \leftarrow k_2$
  end if
  $i \leftarrow i + 1$
end while
$h \leftarrow \mathcal{H}_s(prefix + L'_i s + R'_i s))$
$c_s \leftarrow h - \sum_{i \neq s} c_i$
$r_s \leftarrow k - x c_s$
return $(I, \{c_i\}, \{r_i\})$

[§] `http://nacl.cr.yp.to/`

Сравнение этого алгоритма с [CN] показывает, что он соответствует белой книге. Подобным образом вот алгоритм, используемый в исходнике CryptoNote для верификации кольцевых подписей:

Алгоритм 2 VER

$i = 0$
while $i < numkeys$ **do**
  $L'_i \leftarrow c_i P_i + r_i G$
  $R'_i \leftarrow r_i \mathcal{H}_p(P_i) + c_i I$
  $i \leftarrow i + 1$
end while
$h \leftarrow \mathcal{H}_s(prefix + L'_i s + R'_i s))$
$h \leftarrow h - \sum_{i \neq s} c_i$
return $(h == 0 (mod\ q)) == 0$

3.1 Важные функции Crypto-ops

Описание важных функций в Crypto-ops.cpp. Ещё больше ссылок и информации приводится в комментариях к коду MiniNero.py, ссылка на который давалась выше.

3.1.1 ge_frombytes_vartime

Берёт в качестве входа некоторые данные и преобразует их в точку на кривой ed25519. Для ссылки на используемое уравнение $\beta = uv^3 \left(uv^7\right)^{(q-5)/8}$, см. [BBJLP], раздел 5.

3.1.2 ge_fromfe_frombytesvartime

Подобна функции выше, но представлена в сжатой форме.

3.1.3 ge_double_scalarmult_base_vartime

Берёт в качестве входов два целых числа a и b и точку A на кривой ed25519 и возвращает точку $aA + bG$, где G является базовой точкой ed25519. Используется с кольцевыми подписями, например, при вычислении L_i, где $i \neq s$, как указано в работе ([CN], 4.4)

3.1.4 ge_double_scalarmult_vartime

Берёт в качестве входов два целых числа a и b и две точки A и B на кривой ed25519 и возвращает $aA + bB$. Используется, например, при вычислении R_i в кольцевых подписях, где $i \neq s$ ([CN], 4.4)

3.1.5 ge_scalarmult

При наличии заданной точки A на кривой ed25519 и целого числа a используется для вычисления точки aA. Применяется, например, для вычисления L_i and R_i, если $i = s$.

3.1.6 ge_scalarmult_base

В качестве входа берёт целое число a и вычисляет aG, где Gявляется базовой точкой ed25519.

3.1.7 ge_p1p1_to_p2

Существуют различные варианты представления точек кривой ed25519. Данная функция позволяет осуществлять преобразование между ними. Дополнительную информацию можно найти в MiniNero.

3.1.8 ge_p2_dbl

Берёт точку в представлении "p2" и удваивает её.

3.1.9 ge_p3_to_p2

Берёт точку в представлении"p3" на кривой ed25519 и преобразует её в точку в представлении "p2".

3.1.10 ge_mul8

Берёт точку A на кривой ed25519 и возвращает $8A$.

3.1.11 sc_reduce

Берёт 64-байтное целое число и выводит нижние 32 байта по модулю простого числа q. Данная функция не является специфической функцией CryptoNote, но она идёт со стандартной библиотекой ed25519.

3.1.12 sc_reduce32

Берёт 32-байтное целое число и выводит целое число по модулю q. Тот же код, что и у функции выше, за исключением того, что пропускается этап преобразования из 64 в 32 байта.

3.1.13 sc_mulsub

Берёт три целых числа a, b и c в поле $\mathbb{F}_q$ и возвращает $c - ab$ по модулю q.

3.2 Важные функции хеширования

3.2.1 cn_fast_hash

Берёт данные и выдаёт хеш данных Keccak1600.

3.3 Crypto.cpp Functions

3.3.1 random_scalar

Генерирует 64-байтное целое число, а затем сокращает его до 32-байтного целого числа по модулю q для обеспечения 128-битного уровня безопасности, как было описано в пункте 2.3.

3.3.2 hash_to_scalar

Вводит данные (например, точку P на кривой ed25519) и выводит $\mathcal{H}_s(P)$, что является хешем данных Keccak1600. Затем функция преобразует хешированные данные в 32-байтное целое число по модулю q.

3.3.3 generate_keys

Выдаёт пару, состоящую из секретного ключа и приватного ключа, используя `random_scalar` (как описано выше) для получения секретного ключа. Следует отметить, что, как было описано в работе [Bern], набор ключей для кривой ed25519 фактически просто умножает на 8 в поле $\mathbb{F}_q$, а следовательно, `ge_scalarmult_base` включает `ge_mul8`, чтобы гарантировать, что секретный ключ будет в наборе ключей. Это защищает от атак на транзакции, описанных в работе ([Bern], см. раздел по «атакам на небольшие подгруппы»). Это часть алгоритма **GEN** , описанного в работе ([CN], 4.4).

3.3.4 check_key

Вводит публичный ключ и выводит данные, подтверждающие, находится ли точка на кривой.

3.3.5 secret_key_to_public_key

Вводит секретный ключ, проверяет его на соответствие некоторым условиям соблюдения единообразия и выводит соответствующий публичный ключ, который в основном в 8 раз больше, чем базовая точка от точки.

3.3.6 hash_to_ec

Вводит ключ, хеширует его, а затем создаёт эквивалент путём поразрядных операций, умножая полученное целое число на базовую точку, а затем на 8.

3.3.7 generate_key_image

Берёт в качестве входа секретный ключ x и публичный ключ P и выдаёт $I = x\mathcal{H}_p(P)$, образ ключа. Эта часть алгоритма **GEN** описана в работе ([CN], 4.4).

3.3.8 generate_ring_signature

Вычисляет кольцевую подпись, реализуя **SIG**, как описано в работе ([CN], 4.4), при наличии образа ключа I, списка n публичных ключей P_i и секретного индекса. По существу, по i есть цикл, и если доходит до секретного индекса, то оператор «если» контролирует специальное вычисление L_i, R_i если значение i равно секретному индексу. Значения c_i и r_i для подписи вычисляются по всему циклу и выдаются вместе с образом с целью создания полной подписи $(I, c_1, ..., c_n, r_1, ..., r_n)$.

3.3.9 check_ring_signature

Запускает алгоритм **VER**, описанный в работе ([CN], 4.4). Верификатор использует заданную кольцевую подпись, чтобы вычислить $L'_i = r_i G_i$, $R'_i = r_i \mathcal{H}_p(P_i) + c_i I$ и, наконец, чтобы проверить, что $\sum_{i=0}^{n} c_i = \mathcal{H}_s(m, L'_0, ..., L'_n, R'_0, ..., R'_n) \ mod\ l$.

3.3.10 generate_key_derivation

Берёт секретный ключ b и публичный ключ P и выдаёт $8 \cdot bP$. (8 берётся для набора секретного ключа, как описано в пункте 3.3.3). Это используется функцией derive_public_key в рамках создания одноразовых адресов.

3.3.11 derivation_to_scalar

Производит $\mathcal{H}_s(-)$ в рамках процесса создания ключей, как описано в работе ([CN], 4.3, 4.4). Хеширует индекс выхода вместе с точкой.

3.3.12 derive_public_key

Берёт вывод rA (вычисленный посредством `generate_key_derivation`), точку B и индекс выхода, вычисляет скалярную величину при помощи `derivation_to_scalar`, а затем вычисляет $\mathcal{H}_s(rA) + B$.

3.3.13 generate_signature

Берёт префикс, публичный ключ и секретный ключ и генерирует стандартную (не кольцевую) подпись транзакции (подобную подписи транзакции Bitcoin).

3.3.14 check_signature

Проверяет, является ли стандартная (не кольцевая) подпись действительной подписью.

4 Заключение

Несмотря на то, что функции кольцевых подписей в оригинальном исходнике CryptoNote были прокомментированы плохо, код можно проследить, чтобы установить и использовать исходные данные, и сам код довольно прост. Вариант реализации на языке Python, предлагаемый в данной работе, подтверждает правильность кода. Кроме того, математика эллиптических кривых, лежащая в основе схемы создания подписей, была изучена достаточно хорошо; концепция кольцевых подписей не нова, даже несмотря на то, что их применение в области криптовалют и является новым.

Список литературы

[BBJLP] Дениэл Дж. Бернштейн (Bernstein, Daniel J.) и др., «Скрученные кривые Эдвардса» (Twisted edwards curves) Прогресс в криптографии (Progress in Cryptology) – –AFRICACRYPT 2008. Springer Berlin Heidelberg, 2008. 389-405.

[BCPM] Джоппи В. Бос (Bos, Joppe W.) и др., «Выбор эллиптических кривых для криптографии: анализ эффективности и безопасности» (Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis) Архив электронных документов по криптографии IACR (IACR Cryptology ePrint Archive) 2014 (2014): 130.

[Bern] Дениэл Дж. Бернштейн (Bernstein, Daniel J.), «Curve25519: новые скоростные рекорды Диффи-Хеллмана» (Curve25519: new Diffie-Hellman speed records), Криптография на базе публичных ключей (Public Key Cryptography – PKC) 2006. Springer Berlin Heidelberg, 2006. 207-228.

[CH] Дэвид Чаум (Chaum, David) и Юджин Ван Хейст (Eugene Van Heyst), «Групповые подписи» (Group signatures), Прогресс в криптографии (Advances in Cryptology - —EUROCRYPT'91. Springer Berlin Heidelberg, 1991.

[CN] Николас Ван Саберхаген (van Saberhagen, Nicolas). CryptoNote v 2.0. (2013).

[FIPS] SHA, NIST DRAFT. «Стандарт: функции расширяемых выходов на базе перестановок» (standard: Permutation-based hash and extendable-output functions), DRAFT FIPS 202 (2014).

[Fu] Еиширо Фуджисаки (Fujisaki, Eiichiro), «Сублинейные отслеживаемые по размеру кольцевые подписи без случайных оракулов» (Sub-linear size traceable ring signatures without random oracles), Материалы IEICE по фундаментальным основам электроники, систем коммуникаций и компьютерной техники (IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences) 95.1 (2012): 151-166.

[FS] Еиширо Фуджисаки (Fujisaki, Eiichiro) и Кутару Сузуки (Koutarou Suzuki), «Отслеживаемая кольцевая подпись» (Traceable ring signature), Криптография на базе публичных ключей (Public Key Cryptography – –PKC 2007). Springer Berlin Heidelberg, 2007. 181-200.

[IAN] IANIX http://ianix.com/pub/curve25519-deployment.html

[Si] Джозеф Х. Сильверман (Silverman, Joseph H.), «Арифметика эллиптических кривых» (The arithmetic of elliptic curves), Vol. 106. Dordrecht: Springer, 2009.

[SN] http://lab.monero.cc/pubs/multiple_equations_attack.pdf