Эффективная реализация подадресов Monero

Саранг Ноезер (Sarang Noether)* и Брэндон Гуделл (Brandon Goodell)†
Исследовательская лаборатория Monero (Monero Research Lab)
03 Октября 2017

Аннотация

Пользователям криптовалюты Monero, желающим многократно использовать адреса своих кошельков и делать это без какой-либо привязки, приходится поддерживать несколько отдельных кошельков, каждый из которых требует сканирования входящих транзакций. Мы предлагаем новую адресную схему, которая позволит пользователю поддерживать один главный адрес кошелька и генерировать произвольное количество несвязанных подадресов. Каждая транзакция требует всего одного сканирования, чтобы определить, предназначена ли она для какого-либо из подадресов пользователя. Помимо этого, схема поддерживает множество выходов на другие подадреса и работает так же эффективно, как и при проведении традиционных транзакций с использованием кошелька.

1 Введение

Конфиденциальность транзакций Monero обеспечивается тремя основными конструкциями: кольцевыми подписями, одноразовыми ключами и обязательствами по сумме. Использование кольцевых подписей гарантирует, что злоумышленник не сможет определить фактический публичный ключ ввода, используемый для совершения транзакции, так как этот ключ скрывается наличием произвольно выбираемых публичных ключей ввода [3]. Обязательства по сумме используют гомоморфные свойства, гарантирующие: в то время как третья сторона не сможет определить сумму на выходе транзакции, она сможет доказать, что суммы на входе и на выходе транзакции будут сбалансированы. В сочетании с доказательством диапозона, который гарантирует, что выход транзакции будет находиться в пределах предопределенной и действительной цепи, обязательства маскируют суммы транзакции и позволяют избежать каких-либо нарушений злоумышленником. Одноразовые ключи генерируются с использованием параметров транзакции и опубликованного получателем адреса кошелька. Такие ключи гарантируют, что никто, кроме получателя, не сможет идентифицировать направление транзакции или потратить полученные средства.

Проблемой, которую не решают эти средства обеспечения конфиденциальности, являются адреса получателя. Боб может захотеть положить Monero в кошелек для личных пожертвований, но также он может захотеть получить Monero для покупки чего-либо на своем рабочем сайте. И если Боба волнует вопрос собственной анонимности, он может не захотеть использовать один и тот же адрес кошелька, так как это свяжет его присутствие в сети с личной и рабочей целью с одним человеком. Очевидным решением для Боба было бы создание двух кошельков и раздельная публикация адресов: одного — для его личного блога и второго — для его рабочего сайта. Тем не менее это бы означало, что Бобу придется сканировать каждую входящую транзакцию дважды, чтобы убедиться в том, что

*sarang.noether@protonmail.com
†surae.noether@protonmail.com

она прошла на один из его кошельков. Данная проблема последовательно разрастается по мере того, как Боб создает дополнительные адреса для каждого отдельного случая.

Таким образом, возникает необходимость в методе, который позволил бы Бобу публиковать различные и несвязанные адреса так, чтобы это не сказалось отрицательно на вычислениях, касающихся входящих транзакций. В этом докладе нами предлагается эффективное решение проблемы — схема *подадресов* [1]. Такая схема позволит Бобу создать столько адресов, сколько он захочет, и распоряжаться ими любым способом, который он сочтет уместным. Эти подадреса не могут быть связаны ни друг с другом, ни с оригинальным адресом кошелька Боба. При сканировании входящих транзакций вычисления потребовали масштабирования в определенных временных рамках в соответствии с количеством подадресов, что означало отсутствие дополнительной сложности.

2 Реализация схемы

Предположим, что Алиса хочет послать несколько заработанных нелегким трудом Monero Бобу. Главным адресом кошелька Боба, который он нигде не публиковал, является $(A, B) = (aG, bG)$, где a и b являются секретными ключами (скалярами), а G является общей базовой точкой на эллиптической кривой. Боб хочет использовать свой главный адрес для того, чтобы сгенерировать подадреса, которые он сможет дать Алисе или опубликовать каким-либо другим образом.

Предполагается, что Боб поддерживает список на компьютере, который должен управлять его кошельками. Этот список состоит из скаляров, используемых предыдущими подадресами. Для того чтобы создать новый подадрес, Боб выбирает скаляр i, которого уже нет в списке. Это необязательно делается в произвольном порядке. Происходит следующее вычисление:

$$\begin{aligned} D_i &\equiv B + H_s(a, i)G \\ C_i &\equiv aD_i \end{aligned}$$

В данном случае H_s является криптографической скалярной хеш-функцией. Подадреса определяются как пара точек (C_i, D_i). Предполагается, что Боб также хранит в компьютере хеш-таблицу, которая увязывает $D_i \mapsto i$.

Чтобы отправить Monero на подадрес Боба (C_i, D_i), Алиса выбирает произвольный скаляр транзакции s и вычисляет публичный ключ транзакции:

$$R \equiv sD_i$$

Затем Алиса вычисляет публичный ключ выхода:

$$P \equiv H_s(sC_i)G + D_i$$

Использование отдельных ключей транзакции для каждого выхода позволяет Алисе использовать множество выходов, направленных на подадреса.

Если Алиса захочет отправить сдачу на свой собственный главный адрес кошелька $(X, Y) = (xG, yG)$, то она может сделать это, сгенерировав публичный ключ сдачи:

$$P_{\text{change}} \equiv H_s(xR)G + Y$$

Если же вместо этого у Алисы есть собственный подадрес (Z_j, W_j), она может направить сдачу на него:

$$P_{\text{change}} \equiv H_s(xR)G + W_j$$

Когда Боб сканирует входящие транзакции, он проверяет каждый публичный ключ выхода P (вместе со связанным публичным ключом транзакции R) следующим образом:

$$D' \equiv P - H_s(aR)G$$

Если Боб увидит, что значение D' совпадает со скаляром i в его локальной хеш-таблице, то он будет уверен в том, что выход был отправлен на подадрес (C_i, D_i), поскольку:

$$\begin{aligned} P - H_s(aR)G &= H_s(sC_i)G + D_i - H_s(a(sD_i))G \\ &= H_s(sC_i)G + D_i - H_s(s(aD_i))G \\ &= H_s(sC_i)G + D_i - H_s(sC_i)G \\ &= D_i. \end{aligned}$$

Чтобы использовать свои средства в качестве входа при проведении последующей транзакции, Боб должен иметь возможность определить приватный ключ, связанный с P. Он легко может сделать это, используя индекс, возвращенный в результате поиска по хеш-таблице:

$$p \equiv H_s(aR) + b + H_s(a, i)$$

И это получится, поскольку:

$$\begin{aligned} pG &= (H_s(aR) + b + H_s(a, i))G \\ &= (H_s(sC_i) + b + H_s(a, i))G \\ &= H_s(sC_i)G + D_i \\ &= P. \end{aligned}$$

3 Интеграция в транзакции

Представленная нами схема использования подадресов требует внесения изменений в процесс обработки транзакций и не может рассматриваться в качестве «дополнения» или замены для стандартных кошельков. Если Алиса захочет сделать перевод на стандартный кошелек Боба, ей понадобится построить публичный ключ транзакции $R = rG$, используя общую базовую точку. Одним из бесхитростных способов, при помощи которого Алиса потом смогла бы доказать, что она являлась автором транзакции, является доказательство знания r (или, что еще проще, простое обнародование r третьей стороне. Что примечательно, она могла бы сделать это, не указывая, что средства были переданы на адрес кошелька Боба.

В случае применения подадресов Алиса должна построить публичный ключ транзакции $R = sD$, , используя публичный ключ подадреса Боба в качестве базовой точки. Так как секретный ключ s выбирается равномерно произвольно, публичный ключ транзакции с подадресом также по-прежнему распространяется равномерно. Тем не менее, Алиса больше не может просто доказать, что она знает секретный ключ общей базовой точки, не открыв подадреса Боба получателю средств, так как он привязан к подадресу Боба в рамках протокола обмена Диффи-Хеллмана. Вдобавок к s, ей также придется открыть и D. Несмотря на то, что существуют и другие методы доказательства авторства транзакции при помощи обязательств, это изменение стоит отметить.

При проведении стандартных транзакций Боб может попросить Чарли просмотреть входящую для него транзакцию или же пожелать, чтобы Чарли каким-либо другим образом проверил его кошелек.

Для этого Бобу придется открыть Чарли секретный ключ просмотра a и компонент адреса кошелька B. Схема использования подадресов предполагает, что Боб также может открыть ключ a Чарли. Затем Чарли должен построить хеш-таблицу, либо используя список индексов, которые известны ему от Боба, либо установив цепочку, которую он предварительно вычислит.

4 Счета подадресов

Так как выход сдачи может быть направлен либо на главный адрес кошелька получателя, либо на подадрес, подадреса могут быть естественным образом сгруппированы, что позволит распараллелить функцию отдельных балансов кошельков. Одним из способов реализации этой задачи является замена индекса подадреса i на упорядоченную пару (i, j). Для любого фиксированного iв качестве *счета* владелец кошелька определяет ряд подадресов $\{i, j\}_j$, где каждый подадрес имеет *старший индекс* i и *младший индекс* j.

Если Боб получает средства на подадрес (i, j) в пределах счета i, программное обеспечение его кошелька суммирует средства, которые содержатся на всех подадресах счета, в один баланс. Затем, после того как средства будут потрачены, сдача может быть перенаправлена на подадрес $(i, 0)$. Мы подчеркиваем, что это просто удобная особенность протокола и она никак не влияет на шифрование схемы подадресов.

5 Анализ

5.1 Связывание и определение кошелька

Поскольку криптографическая скалярная хеш-функция H_s имеет равномерно распределенный выход, набор компонентов возможных подадресов Боба

$$\{D_i\}_i = \{B + H_s(a, i)G\}_i$$

также распределяется равномерно [2]. Это означает, что злоумышленник, владеющий произвольным набором подадресов $\{(C_i, D_i)\}_i = \{aD_i, D_i\}$, не сможет ни решить дискретный логарифм $a = \log_{C_i} D_i$, ни инвертировать хеш-функцию (которая также требует знания B). А это означает, что главный адрес кошелька Боба будет защищен даже в том случае, если злоумышленник убедит его сгенерировать новые подадреса с выбранными индексами.

5.2 Восстановление кошелька

В том случае, если Боб утратил доступ к программному обеспечению своего локального кошелька и восстанавливает все, начиная с сида, он не сможет сразу идентифицировать транзакции, предназначенные для его подадресов, так как для этого ему нужен доступ к хеш-таблице. Чтобы восстановить таблицу, не потеряв каких-либо подадресов, Бобу необходимо выбрать предварительные значения L_M и L_m. Для этого, используя старшие индексы $i \leq L_M$ и младшие индексы $j \leq L_m$(для каждого старшего индекса), ему придется сгенерировать хеш-таблицу. После сканирования транзакций при помощи начальной таблицы Боб должен повторить этот процесс, что гарантирует, что он сгенерировал хеш-вводную таблицы L_M сверх самого высокого старшего индекса любой транзакции и L_m сверх самого высокого младшего индекса в пределах каждого старшего индекса. При этом, если предположить, что используемые подадреса не находятся в пределах этих предварительных значений, Боб восстановит необходимые хеш-вводные.

5.3 Транзакции смешанного типа

Предположим, что в программном обеспечении кошелька Алисы слишком много бокалов криптоконьяка и форматирование транзакции происходит некорректно. В частности, предположим, что Алиса создает публичный ключ транзакции $R = sG$ (вместо $R = sD$), но продолжает вычислять публичный ключ выхода как $P = H_s(sC_i) + D_i$. То есть теперь транзакция будет «смешанного типа», а не корректной транзакцией с использованием подадресов.

В этом случае публичный ключ транзакции уже более не будет содержать информации о месте назначения подадресов и Боб не сможет обойтись одним сканированием для обнаружения это транзакции. Несомненно, программное обеспечение его кошелька не сможет распознать, что такая транзакция смешанного типа направлена ему. Тем не менее, если Боб заподозрит, что возникла такая ситуация, он может повторно пройтись по индексам своих подадресов и вычислить:

$$P - H_s(a[b + H_s(a, j)]R)$$

для каждого j, который он использовал для генерирования подадреса. В том случае, если $j = i$, гарантированно результатом будет D_i. Это означает, что ошибка Алисы будет обнаружена, если Боб выполнит такое линейное сканирование, а также, что Боб сможет восстановить приватный ключ, который необходим ему, чтобы потратить деньги. В результате Боб теряет возможность одиночного сканирования транзакции. Поэтому, если его беспокоит возможность транзакций смешанного типа, он может предпочесть, чтобы программное обеспечение его кошелька часто производило линейное сканирование.

5.4 Секретный ключ просмотра

Невозможно обеспечить третьей стороне выборочный доступ к просмотру, чтобы она могла видеть только определенные подадреса при равномерно распределяемых секретных ключах. Если Чарли получит ключ просмотра a Боба и одну вводную из хеш-таблицы $D_i \mapsto i$, он сможет вычислить компонент главного адреса кошелька $B = D_i - H_s(a, i)G$. Если Боб выберет индексы своего подадреса непроизвольно, Чарли легко сможет восстановить другие вводные хеш-таблицы, используя нижние индексы, которые, вероятно, так же будут подадресами, которые использовал Боб:

$$D_j = B + H_s(a, j)G$$

Чтобы другие транзакции оставались конфиденциальными и чтобы Чарли их не видел, Боб должен создать новый главный кошелек и убедиться в том, что у Чарли нет доступа к секретному ключу просмотра.

5.5 Секретный ключ траты

Подобным образом, если секретные ключи распространяются единообразно, третьей стороне нельзя дать возможность выборочно тратить средства с определенного подадреса. Чтобы тратить средства с какого-либо подадреса, Бобу необходимы оба секретных ключа главного кошелька, a и b, а также индекс подадреса. Раскрытие такой информации Чарли позволит ему тратить средства, отправленные на любой подадрес, индекс которого ему известен (или который он может определить каким-либо образом). Безусловно, правильнее будет, если Боб отправит средства на кошелек или подадрес Чарли, как в случае со стандартными настройками кошелька.

5.6 Эффективность

Построение новых подадресов требует незначительного количества операций на эллиптической кривой и является несущественным, так как эта операция выполняется только по необходимости.

Отправка на подадреса требует того же количества операций, что и эквивалентная традиционная транзакция со стандартным адресом кошелька. Тем не менее, вместо вычисления ключа транзакции $R = rG$ при помощи общей базовой точки, как при традиционной транзакции, Алиса вычисляет ключ как $R = sD$, используя базовую точку подадреса. В действительности в этом случае Алиса даже не знает секретного ключа! Это означает, что она не может предварительно вычислить ключ транзакции, не зная подадреса желаемого получателя. На практике это не является проблемой. Вычисление публичного ключа выхода в точности аналогично вычислению в традиционном случае.

Используя стандартный кошелек, Боб проверил бы транзакцию относительно своего секретного ключа просмотра главного кошелька, применив одну операцию присвоения хеша скаляру, два умножения скаляра эллиптической кривой и добавление одной точки на эллиптическую кривую (где время хеширования незначительно). При реализации настоящей схемы Боб должен применить то же количество эквивалентных операций, заменив добавление точки на удаление точки. Для каждой транзакции существует дополнительный просмотр хеша, но это масштабируется в определенных (постоянных) временных рамках в соответствии с количеством сгенерированных подадресов. В этом заключается абсолютное отличие от использования множества адресов кошельков, где каждая транзакция проверяется отдельно относительно каждого секретного ключа просмотра.

Поэтому мы можем сделать вывод, что сканирование входящих (правильно отформатированных) транзакций на их принадлежность с использованием произвольного количества подадресов займет не больше времени, чем сканирование одного стандартного адреса кошелька на предмет его принадлежности. В случае использования подадресов объем линейного масштабирования сканирования множества стандартных адресов кошельков сокращается до постоянного времени.

6 Заключение

Нами была кратко представлена схема использования подадресов, которая является эффективной альтернативой поддержке множества адресов кошельков. При стандартной реализации кошелька CryptoNote для Monero Боб должен публиковать отдельные адреса кошельков, если он не хочет, чтобы они были связаны с точки зрения возможности их использования третьей стороной или злоумышленниками. Тем не менее данный подход требует от Боба (или другого владельца его секретного ключа просмотра), чтобы он проверял каждую входящую транзакцию относительно каждого из его адресов.

Предлагаемая схема позволяет Бобу использовать единственный главный адрес кошелька, чтобы генерировать столько подадресов, сколько он пожелает. Злоумышленник не сможет определить, были ли два подадреса сгенерированы на основе одного главного адреса кошелька, а также не сможет идентифицировать «родительский» адрес. От Боба требуется только поддерживать локальную приватную хеш-таблицу, которая будет связывать подадреса со скалярным индексом, используемым для их создания.

Проведение транзакций с использованием подадресов требует внесения минимальных изменений в существующий протокол проведения транзакций. Поддерживается множество выходов, а выход сдачи транзакции от Алисы к Бобу может быть направлен как на главный адрес кошелька Алисы, так и на любой подадрес по ее выбору. Изучая входящие транзакции, Боб производит по одной проверке для каждой транзакции, независимо от количества поддерживаемых им подадресов, используя свою

локальную хеш-таблицу для восстановления соответствующего приватного ключа транзакции.

Несвязываемые подадреса являются эффективным и безопасным решением проблемы использования множества адресов кошельков и позволяют пользователю определить, как их кошельки видят другие, а также установить желаемый уровень конфиденциальности.

Ссылки

[1] Monero project. Subaddresses. `https://github.com/monero-project/monero/pull/2056`. GitHub pull request #2056.

[2] Паола Скозафава (Paola Scozzafava). Равномерное распределение и сумма по m-модулю независимых случайных величин. *Журнал "Statistics & Probability Letters"*, 18(4):313–314, 1993.

[3] Николас ван Саберхаген (Nicolas van Saberhagen). Cryptonote v2.0, 2013.