

Техники генерации одноразовых чисел в мультиподписях Шнорра: исследование подходов, вдохновлённых EdDSA

Наурас Х. Сабри¹, Алла Б. Левина¹

¹Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»,
Россия, 197022, г. Санкт-Петербург

Аннотация. В работе представлена детерминированный способ генерации одноразовых чисел (nonce) для решения проблем, связанных с повторным использованием одноразовых чисел при подписании сообщений, и повышения эффективности схем мультиподписей Шнорра. Кроме того, исследование направлено на уменьшение вычислительной сложности и требований к пропускной способности в цифровых и мультиподписных схемах с сохранением надёжной защиты от распространённых атак. Предложенный метод был вдохновлён подходом EdDSA. Методология включает математический анализ алгоритмов цифровой подписи и исследование их уязвимостей к известным криптографическим атакам. Анализ оценивает эффективность и надёжность предлагаемой техники генерации одноразовых чисел в рамках схем цифровой подписи Шнорра и двухраундовых схем MuSig. В рамках исследования использовались техники и инструменты для детерминированной генерации одноразовых чисел путём хеширования закрытого ключа и последующего хеширования результата с сообщением. Кроме того, предложено исключить открытое одноразовое число R из расчётов вызова и позволить подписантам непосредственно доказывать владение своими секретными ключами через агрегированный закрытый ключ, тем самым устраняя необходимость в доказательствах с нулевым разглашением (NIZK). Результаты демонстрируют значительное снижение вычислительной сложности и требований к вычислительным ресурсам, что улучшает эффективность пропускной способности и делает предложенный метод подходящим для устройств с ограниченными ресурсами. Подход также демонстрирует высокую устойчивость к различным атакам, включая повторное использование одноразовых чисел, аннулирование ключей, атаки с использованием ложных ключей и откаты виртуальных машин.

Ключевые слова: мультиподписи Шнорра; схемы MuSig; EdDSA; техника генерации одноразовых чисел.

Abstract. This paper proposes a deterministic nonce generation technique to address the catastrophic issues associated with nonce reuse in message signing and to enhance the efficiency of Schnorr multi-signature schemes. Additionally, this research aims to reduce computational complexity and bandwidth requirements in digital and multi-signature schemes while maintaining robust security against common attacks. The proposed method was inspired by the EdDSA approach. The methodology includes a comprehensive mathematical analysis of digital signature algorithms and a rigorous examination of their vulnerabilities to well-known cryptographic attacks. This analysis evaluates the effectiveness and robustness of the proposed nonce generation technique within the frameworks of the Schnorr digital signature and the two-round MuSig schemes. Techniques and tools employed in this research involve deterministically generating

nonces by hashing the private key and subsequently hashing the result with the message. Furthermore, it is proposed to exclude the public nonce R from the challenge calculations and to allow signers to directly prove possession of their secret keys through the aggregated public key, thereby eliminating the need for non-interactive zero-knowledge (NIZK) proofs. The findings demonstrate significant reductions in computational complexity and operational requirements, thereby improving bandwidth efficiency and making this method well-suited for resource-constrained devices. The approach also exhibits strong resistance to various attacks, including nonce reuse, key cancellation, rogue keys, and virtual machine rewinding.

Keywords: Schnorr multi-signatures; MuSig schemes; EdDSA; nonce generation technique.

Введение

Цифровые подписи служат основой современных криптографических практик, обеспечивая безопасность и надёжность цифровых коммуникаций, финансовых транзакций и процессов аутентификации.

Концепция цифровых подписей восходит к 1970-м годам, когда были заложены теоретические основы криптографии с открытым ключом. Уитфилд Диффи и Мартин Хеллман впервые представили эту идею в своей знаковой работе 1976 года [1], что проложило путь к разработке различных схем цифровой подписи. Первая практическая реализация, алгоритм RSA, была разработана в 1977 году Рональдом Ривестом, Ади Шамиром и Леонардом Адлеманом [2]. Он использует вычислительную сложность факторизации больших целых чисел в качестве основы для обеспечения безопасности. Затем в начале 1990-х годов был введён алгоритм цифровой подписи (DSA), который опирается на задачу дискретного логарифма [3]. Оба этих алгоритма стали основой для последующих усовершенствований и адаптаций технологий цифровых подписей. Схема подписи Шнорра, представленная Клаусом Шнорром [4], отличается своей простотой и эффективностью, особенно с точки зрения скорости проверки и более коротких подписей. Позднее был разработан алгоритм цифровой подписи на кривых Эдвардса (EdDSA), обеспечивающий более сильные гарантии безопасности и лучшую производительность благодаря использованию скрученных кривых Эдвардса [5]. Все эти разработки внесли вклад в создание надёжной основы, на которой сегодня работают цифровые подписи, решая различные аспекты безопасности и эффективности.

Эволюция схем цифровых подписей характеризуется постоянным стремлением к повышению безопасности, эффективности и практической применимости. В данном контексте одноразовое число (nonce) – случайное или псевдослучайное число, которое не должно повторяться с тем же закрытым ключом. Однако задача безопасной генерации одноразовых чисел (важный компонент в процессе подписания) остаётся ключевой проблемой, которая напрямую влияет на безопасность и операционную пригодность схемы. Например, в ECDSA и классическом DSA уникальность и секретность одноразового числа имеют решающее значение; если одноразовое число раскрыто, то это может привести к простым атакам, которые восстановят закрытый ключ. Тем не менее, генерация случайных одноразовых чисел сопряжена с множеством проблем. Основная уязвимость заключается в качестве случайности. Эта проблема была явно использована при взломе прошивки PlayStation 3, когда одноразовые числа ECDSA генерировались предсказуемо

из-за недостаточной случайности, что в конечном итоге привело к компрометации закрытого ключа для подписи.

В схемах цифровой подписи алгоритм цифровой подписи на эллиптических кривых (ECDSA) [6] и алгоритм цифровой подписи на кривых Эдвардса (EdDSA) [7] установили стандарты безопасности и производительности. Особенно подход EdDSA к детерминированной генерации одноразовых чисел привлёк внимание благодаря своей способности обходить проблемы, связанные с генерацией случайных одноразовых чисел, в частности, риски отказа случайности и последующее воздействие на атаки, которые могут компрометировать закрытый ключ подписанта. Таким образом, интеграция детерминированной генерации одноразовых чисел, используемой в EdDSA, в подписи Шнорра представляет собой перспективный путь для повышения надёжности и эффективности этих протоколов.

Новизна исследования заключается в трёх аспектах: во-первых, предложенный подход объединяет детерминированную генерацию одноразовых чисел, вдохновлённую EdDSA, в мультиподписи Шнорра, эффективно сочетая надёжность EdDSA с гибкостью и эффективностью агрегации подписей Шнорра. Во-вторых, исключает открытое одноразовое число (R) из расчётов вызова (c), чтобы обеспечить уникальность каждой подписи. В-третьих, позволяет подписантам непосредственно проверять владение секретными ключами с помощью агрегированного открытого ключа, тем самым повышая эффективность использования пропускной способности за счёт исключения необходимости в доказательствах с нулевым разглашением (NIZK). Подобный упрощённый подход значительно улучшает безопасность и снижает вычислительную сложность, особенно для устройств с ограниченными ресурсами.

Цель данного исследования заключается в устранении уязвимостей безопасности, присущих мультиподписям Шнорра, особенно тех, которые возникают из-за случайной генерации одноразовых чисел, таких как атаки с аннулированием ключа, атаки с использованием ложных ключей и атаки с повторным использованием одноразовых чисел. Кроме того, исследование направлено на повышение эффективности и практической применимости подписей Шнорра в сценариях, где вычислительные ресурсы, память и пропускная способность ограничены.

Методология исследования включала математический анализ алгоритмов цифровой подписи (ECDSA, EdDSA, Шнорра и MuSig) и известных атак (таких как атаки с повторным использованием одноразовых чисел, атаки с откатом виртуальной машины, атаки с аннулированием ключей и атаки с использованием ложных ключей) для понимания и оценки эффективности предложенного метода в решении различных проблем и определения его эффективности. Анализируя сильные и слабые стороны этих схем цифровой подписи и исследуя новые техники, исследование представляет новый подход, который предлагает уровень безопасности, сопоставимый с EdDSA. Таким образом, данная работа вносит вклад в продолжающиеся усилия по повышению эффективности и безопасности цифровых подписей. Рассматриваемое исследование достигло поставленных целей, решив критические проблемы, связанные с генерацией случайных одноразовых чисел, генерируя одноразовые числа детерминированным образом, снижая вычислительную сложность и уменьшая объем передаваемых данных в сочетании с доказательством NIZK для лучшего использования пропускной способности.

1. Обзор литературы

Учитывая, что текущее исследование представляет новый подход к генерации одноразовых чисел в схеме Шнорра, используя методологию, применяемую в схеме EdDSA, цель данного обзора – консолидировать и оценить предыдущие исследования, оценивающие эффективность генерации случайных чисел в различных алгоритмах цифровой подписи, включая те, которые используют криптографию на эллиптических кривых (ECDSA, EdDSA и Шнорр).

В статье [8] Бенардо и др. обсуждают проблему повторного использования одноразовых чисел в подписях Шнорра, что может привести к раскрытию секретного ключа для подписи и компрометации безопасности схемы подписи. Для преодоления этой проблемы авторы предлагают вариант схемы подписи Шнорра, который позволяет повторное использование одноразовых чисел в нескольких подписях, сохраняя при этом безопасность. Их вариант использует простое модульное число p таким образом, что $p - 1$ имеет несколько различных факторов q_i , достаточно больших, чтобы противостоять атакам дня рождения, и объединяет усилия по возведению в степень для достижения более быстрых и ресурсно-эффективных подписей. Авторы также сравнивают производительность своей схемы с классическими подписями Шнорра и представляют несколько методов предварительных вычислений для ускорения модульного возведения в степень, которое часто является узким местом в реализации криптографических систем.

Стоит отметить, что выявлено отсутствие специфической литературы, касающейся атак на повторное использование одноразовых чисел в цифровых подписях EdDSA, главным образом потому, что EdDSA устойчива к таким атакам при выборе подходящей хеш-функции. Тем не менее, важно признать, что безопасность любой схемы цифровой подписи, включая EdDSA, не является непроницаемой для потенциальных уязвимостей. Например, атаки с использованием сбоев, которые эксплуатируют вычислительные ошибки в процессе вычисления подписи [9], а также другие типы атак, не связанные с повторным использованием одноразовых чисел, могут представлять значительные риски.

В статье [10] авторы описывают подробное исследование методов неинтерактивной агрегации для подписей EdDSA и Шнорра, сосредотачиваясь на оптимизации агрегации подписей для повышения эффективности криптографических приложений. Исследование углубляется в тонкости компактных схем подписей, обсуждая преимущества и проблемы уменьшения размеров подписей. Проведя тщательный анализ внедрения и сравнительных тестов, исследователи демонстрируют практичность и производительность предложенных конструкций агрегации. Результаты подчёркивают осуществимость схемы полугагрегации и дают представление о вычислительных затратах, необходимых для достижения доказуемых гарантий безопасности в агрегации подписей. Авторы описывают методологию, которая не только улучшает компактность данных, но и сохраняет целостность и безопасность подписей от различных криптографических атак, что делает её актуальной для исследований по эффективности и безопасности мультиподписей Шнорра.

В работе [11] Конди Й., Орланди К. и Рой Л. представили новый протокол для двухсторонних пороговых подписей Шнорра, который не только обеспечивает подпись без хранения состояния и детерминированную подпись, но и решает неэффективности, обнаруженные в предыдущих пороговых протоколах. Используя псевдослучайные корреляционные функции (PCFs), исследование повышает как безопасность, так и эффективность распределённой подписи Шнорра. Данные PCFs позволяют

распределение одноразовых чисел для подписи в распределённой, детерминированной манере без сохранения состояния между сеансами, снижая затраты на пропускную способность для каждого участника и делая систему более подходящей для практических приложений.

Исследование включает анализ различных методов распределения подписей Шнорра, оценивая ключевые факторы, такие как количество раундов, требования к пропускной способности, основные предположения и гарантии безопасности. Предложенные протоколы разработаны для достижения как скрытой, так и полной активной безопасности, предлагая надёжную защиту от злонамеренных участников при общих криптографических предположениях. Данное усовершенствование демонстрирует значительные улучшения в области распределённых протоколов подписей, особенно в том, как подписи Шнорра могут быть эффективно и безопасно реализованы в пороговой настройке.

Научная работа «MuSig-DN: Мультиподписи Шнорра с проверяемыми детерминированными одноразовыми числами», написанная Йонасом Ником, Тимом Руффингом, Янником Сёриным и Питером Вуиллем (2020) [12], позволяют оценить генерацию одноразовых чисел и их влияние на алгоритмы цифровой подписи. Авторы представляют новую схему мультиподписей под названием MuSig-DN. Данная схема использует детерминированные одноразовые числа, которые генерируются таким образом, чтобы эффективно защищать от атак с откатом виртуальной машины и атак с повторным использованием одноразовых чисел.

Одной из ключевых проблем, существующих в MuSig-DN, является сложность, связанная с внедрением доказательств с нулевым разглашением (NIZK) для проверки детерминированной генерации одноразовых чисел. Эти доказательства, хотя и необходимы для обеспечения безопасности протокола от определённых атак и обеспечения детерминизма генерации одноразовых чисел, вводят значительный уровень вычислительной и концептуальной сложности. В частности, протокол требует построения и проверки доказательств NIZK для каждого одноразового числа, чтобы удостовериться, что оно было сгенерировано в соответствии с детерминированным протоколом. Подобное требование не только увеличивает вычислительную нагрузку, но и усложняет реализацию, делая протокол менее пригодным для сред с ограниченными ресурсами, таких как аппаратные кошельки, которые распространены в криптовалютных приложениях.

Статья «MuSig2: Простые двухраундовые мультиподписи Шнорра» (2023) [13], написанная Йонасом Ником, Тимом Руффингом и Янником Сёриным, представляет новый подход к мультиподписям, предлагая схему MuSig2, которая оптимизирует процесс цифровой подписи. Исследование, проведённое в данной статье, сосредоточено на повышении эффективности и безопасности схем мультиподписей, особенно в контексте подписей Шнорра. Используя концепцию множественных одноразовых чисел в процессе подписания, MuSig2 достигает значительных улучшений в гарантиях безопасности и показателях производительности.

Основные вклады статьи включают:

- Введение MuSig2 как двухраундового варианта схемы MuSig, исключающего предварительную фазу обязательства и позволяющего подписантам начинать процесс подписания непосредственно с одноразовых чисел.

– Использование разного количества одноразовых чисел (1, 2 или 4+), чтобы удовлетворить различные требования безопасности и криптографические предположения, такие как модель алгебраической группы (AGM) и предположение об одной дополнительной дискретной логарифме (OMDL) в модели случайного оракула (ROM).

Результаты, представленные в статье, подчёркивают повышение эффективности и улучшение безопасности, достигнутые благодаря реализации MuSig2, что делает её ценным вкладом в область цифровых подписей и мультиподписей.

2. Основы цифровых подписей

2.1. Алгоритм цифровой подписи на эллиптических кривых ECDSA

Для понимания предлагаемого метода в данном исследовании требуется подробное объяснение механизма ECDSA, так как он является фундаментальной основой нашего подхода.

Алгоритмы генерации и проверки подписи в ECDSA следующие [6]:

Предположим, отправитель намеревается отправить сообщение с цифровой подписью получателю. Сначала отправитель и получатель согласуют параметры кривой (E, G, n) и хеш-функцию H , где E представляет уравнение эллиптической кривой, G представляет базовую точку эллиптической кривой (генераторную точку), точку на кривой, которая генерирует подгруппу большого простого порядка n , и n представляет порядок G . Это означает, что $nG = O$, где O – тождественный элемент.

В дополнение к общим параметрам, процесс генерации подписи включает следующие переменные: d_A , Q_A , и m . Здесь, d_A представляет собой закрытый ключ, который случайным образом выбирается из интервала $[1, n - 1]$; Q_A представляет собой открытый ключ, вычисляемый как $d_A G$; и m представляет сообщение, которое нужно подписать и отправить. Обратите внимание, что буква «A» в переменных d_A и Q_A обозначает отправителя, которого мы будем называть Алиса.

Для того чтобы Алиса подписала сообщение (m), выполняются следующие шаги [6]:

- Вычислить $e = H(m)$;
- Случайно выбрать целое число « k » (одноразовое число) из диапазона $[1, n - 1]$;
- Вычислить точку $(x_1, y_1) = kG$;
- Вычислить $r \equiv x_1 \pmod{n}$;
- Вычислить $s = k^{-1}(e + rd_A)$;
- Подписью является пара (r, s) . Таким образом, Алиса отправит сообщение (m), Q_A , r и s Бобу.

Обратите внимание, что важно не только держать « k » в секрете, но и выбирать разное « k » для каждой подписи.

Боб выполняет следующие шаги для проверки подписи Алисы:

- Вычислить $e = H(m)$;
- Вычислить $u_1 \equiv es^{-1} \pmod{n}$ и $u_2 \equiv rs^{-1} \pmod{n}$;
- Вычислить точку $(x_1, y_1) = u_1 G + u_2 Q_A$;
- Подпись Алисы действительна, если $(x_1, y_1) \equiv kG$.

Доказательство корректности алгоритма:

$(x_1, y_1) = u_1 G + u_2 Q_A$, поскольку $u_1 \equiv es^{-1} \pmod{n}$ и $u_2 \equiv rs^{-1} \pmod{n}$ тогда:

$$(x_1, y_1) = es^{-1}G + rs^{-1}Q_A = s^{-1}(eG + rQ_A) = \frac{eG + rQ_A}{s},$$

Поскольку $s = k^{-1}(e + rd_A)$, тогда $(x_1, y_1) = \frac{eG + rQ_A}{K^{-1}(e + rd_A)} = \frac{k(eG + rQ_A)}{e + rd_A}$,

Так как $Q_A = d_A G$, то $(x_1, y_1) = \frac{k(eG + rd_A G)}{e + rd_A} = \frac{kG(e + rd_A)}{(e + rd_A)} = kG$.

Важно заметить, что ECDSA уязвим к атаке повторного использования одноразовых чисел – если Алиса использует одно и то же одноразовое число при подписании различных сообщений, злоумышленник – или противник (которого мы будем называть Оскар) – может потенциально обнаружить закрытый ключ Алисы (d_A). Если Алиса повторно использует одно и то же одноразовое число для подписания различных сообщений, то Оскар может выполнить следующие шаги для раскрытия закрытого ключа Алисы (d_A):

Алиса подписывает сообщение m_1	Алиса снова подписывает сообщение m_1
1. Вычислить $e_1 = H(m_1)$	1. Вычислить $e_2 = H(m_2)$
2. Выбрать одноразовое число « k »	2. Значение « k » остаётся тем же
3. Вычислить $(x_1, y_1) = kG$	3. Вычислить $(x_1, y_1) = (x_2, y_2) = kG$
4. Вычислить точку $r \equiv x_1 \pmod n$	4. « r » остаётся тем же $r \equiv x_1 \pmod n$
5. Вычислить $s_1 = k^{-1}(e_1 + rd_A)$	5. Вычислить $s_2 = k^{-1}(e_2 + rd_A)$
6. Подпись сообщения (m_1) это (r, s_1)	6. Подпись сообщения (m_2) это (r, s_2)

Если Оскар, который слушал этот разговор, выполнит операцию вычитания между (s_1) и (s_2) , результат раскроет закрытый ключ Алисы (d_A) следующим образом

$$s_1 - s_2 = k^{-1}(e_1 + rd_A) - (k^{-1}(e_2 + rd_A)) = k^{-1}e_1 + k^{-1}rd_A - k^{-1}e_2 - k^{-1}rd_A = k^{-1}e_1 - k^{-1}e_2,$$

$$s_1 - s_2 = k^{-1}(e_1 - e_2) = \frac{(e_1 - e_2)}{k}, k = \frac{e_1 - e_2}{s_1 - s_2} = (e_1 - e_2)(s_1 - s_2)^{-1}.$$

Зная одноразовое число « k », Оскар теперь знает $(s_1, s_2, e_1 = H(m_1)$ и $e_2 = H(m_2))$. Подставив их в уравнения для (s_1) или (s_2) , Оскар может получить закрытый ключ Алисы (d_A) следующим образом

$$s_1 = k^{-1}(e_1 + rd_A), s_1 = \frac{(e_1 + rd_A)}{k}, ks_1 = e_1 + rd_A, rd_A = ks_1 - e_1, d_A = \frac{ks_1 - e_1}{r},$$

$$d_A = (ks_1 - e_1)r^{-1}.$$

Атака с повторным использованием одноразового числа, вызванная сбоем генератора случайных чисел, была использована для извлечения ключа подписи игровой консоли PlayStation 3 [14]. Кроме того, сбой в генерации случайных чисел привёл к потере средств пользователями Android Bitcoin Wallet в августе 2013 года [15]. Данные инциденты подчёркивают важность надёжной и безопасной генерации случайных чисел в криптографических системах. Тщательное тестирование, оценка и реализация надёжных генераторов случайных чисел имеют решающее значение для предотвращения таких сбоев и связанных с ними рисков.

2.2. Алгоритм цифровой подписи на эллиптических кривых EdDSA

Для понимания предлагаемого метода в данном исследовании требуется подробное объяснение механизма EdDSA, так как он является одним из краеугольных элементов нашего подхода. Алгоритм генерации и проверки подписи в EdDSA следующий [5].

Для того чтобы Алиса подписала сообщение (m), она выполняет следующие шаги:

- Выбрать закрытый ключ (секретный ключ « S_k ») и вычислить открытый ключ $P_k = S_k G$;
- Вычислить $h = H(S_k)$;
- Вычислить одноразовое число $r = H(h||m)$, где $||$ означает конкатенацию h с сообщением m ;
- Вычислить R как x -координату точки (rG);
- Вычислить $s = r + H(R||P_k||m)S_k$;

Подписью является пара (R, s) . Таким образом, Алиса отправит сообщение (m), P_k , R и s Бобу.

Боб выполняет следующие шаги для проверки подписи Алисы:

- Вычислить $\tilde{s} = H(R||P_k||m)$;
- Вычислить $V_1 = sG$ и $V_2 = R + P_k \tilde{s}$;

Подпись действительна, если $V_1 = V_2$.

Доказательство корректности алгоритма:

$V_2 = R + P_k \tilde{s}$, $R = rG$ и $\tilde{s} = H(R||P_k||m)$.

$V_2 = rG + P_k H(R||P_k||m)$, так как $P_k = S_k G$, тогда:

$V_2 = rG + S_k G H(R||P_k||m) = G(r + S_k H(R||P_k||m))$.

Так как $s = r + H(R||P_k||m)S_k$, то $V_2 = sG = V_1$.

EdDSA выбирает одноразовое число (nonce) детерминированным образом как хеш от части закрытого ключа и сообщения. Таким образом, после генерации закрытого ключа EdDSA больше не нуждается в генераторе случайных чисел для создания подписей. Другими словами, EdDSA устойчива к атакам с повторным использованием одноразовых чисел, если выбрана подходящая хеш-функция.

Однако важно учитывать последствия многократного подписания одного сообщения, что приводит к повторному использованию одного и того же одноразового числа [7]. Повторение процесса подписания одного и того же сообщения несколько раз использует то же одноразовое число, так как ни закрытый ключ, ни сообщение не изменяются, и их многократное хеширование приводит к одному и тому же значению хеша. Тем не менее, подобное повторение не ставит под угрозу безопасность системы, так как сгенерированные подписи остаются идентичными. Следовательно, если Оскар попытается сравнить эти подписи, результат будет равен нулю, что указывает на то, что они неразличимы.

Для более полного понимания рассмотрим шаги, которые Оскар может выполнить для реализации атаки с повторным использованием одноразового числа:

Алиса подписывает сообщение m_1	Алиса подписывает сообщение m_1
1. Выбрать « S_k » и вычислить $P_k = S_k G$	1. « S_k » и P_k остаются теми же
2. Вычислить $h_1 = H(S_k)$	2. $h_2 = h_1 = H(S_k)$
3. Вычислить $r_1 = H(h_1 \parallel m_1)$	3. $r_2 = r_1 = H(h_1 \parallel m_1)$
4. Вычислить R_1 как x-координату ($r_1 G$)	4. $R_2 = R_1$
5. Вычислить $s_1 = r_1 + H(R_1 \parallel P_k \parallel m_1)S_k$	5. $s_2 = s_1 = r_2 + H(R_2 \parallel P_k \parallel m_1)S_k$
6. Подпись – пара (R_1, s_1)	6. Подпись – пара $(R_2, s_2) = (R_1, s_1)$

Поскольку две подписи идентичны, когда Оскар вычитает одну подпись (s_1) из другой (s_2), результат будет равен нулю ($s_1 - s_2 = 0$). Следовательно, Оскар не получает никакой пользы или информации от этих дублирующихся подписей.

2.3. Алгоритм цифровой подписи Шнорра

Для понимания предлагаемого метода в данном исследовании требуется подробное объяснение алгоритма цифровой подписи Шнорра, так как он является одним из краеугольных элементов нашего подхода. Интересный аспект подписей Шнорра заключается в их линейности, что предоставляет несколько полезных свойств. Подписи Шнорра следуют определённой структуре ($s = r + ck$), где (s) представляет подпись, (r) представляет случайное число, (c) представляет вызов, а (k) представляет секретный ключ. Эта конструкция также проявляет линейность, таким образом гармонично сочетается с линейной природой математики эллиптических кривых. Линейность, проявляемая подписями Шнорра, увеличивает их привлекательность и пригодность для агрегации подписей.

Алгоритм генерации и проверки подписи Шнорра следующий [16]:

Для того чтобы Алиса подписала сообщение (m), она выполняет следующие шаги:

1. Дана циклическая группа G простого порядка p с генератором g . Выбрать закрытый ключ x , такой что $0 < x < p$;
2. Вычислить открытый ключ $X = g^x$;
3. Случайным образом выбрать закрытое одноразовое число r , такое что $0 < r < p$;
4. Вычислить открытое одноразовое число $R = g^r$;
5. Вычислить $c = H(X, R, m)$;
6. Вычислить $s \equiv (r + cx) \bmod p$;
7. Подписью является пара (R, s) . Алиса отправляет сообщение (m), открытый ключ (X), R и s Бобу.

Боб может вычислить (c), так как он знает открытый ключ Алисы (X), (R) и сообщение (m).

Боб считает подпись Алисы действительной, если $g^s = RX^c$. Доказательство корректности алгоритма: $g^s = RX^c$ поскольку $R = g^r$ и $X = g^x$, тогда

$$g^s = g^r (g^x)^c = g^{r+xc} = g^s, \text{ так как } s = r + xc.$$

Важно отметить, что алгоритм Шнорра уязвим для атаки с повторным использованием одноразовых чисел. Оскар может выполнить следующие шаги, чтобы

раскрыть закрытый ключ Алисы (x), если она повторно использовала одноразовое число для различных подписей сообщений:

Алиса подписывает сообщение m_1	Алиса подписывает сообщение m_2
1. Выбрать закрытый ключ (x)	1. Закрытый ключ (x) остаётся тем же
2. Вычислить открытый ключ $X = g^x$	2. Открытый ключ $X = g^x$ остаётся тем же
3. Случайно выбрать одноразовое число r	3. Одноразовое число r остаётся тем же
4. Вычислить $R = g^r$	4. $R = g^r$ остаётся тем же
5. Вычислить $c_1 = H(X, R, m_1)$	5. Вычислить $c_2 = H(X, R, m_2)$
6. Вычислить $s_1 = r + c_1x$	6. Вычислить $s_2 = r + c_2x$

Если Оскар выполнит операцию вычитания между s_1 и s_2 , результат раскроет закрытый ключ Алисы (x) следующим образом:

$$s_1 - s_2 = r + c_1x - (r + c_2x) = c_1x - c_2x = x(c_1 - c_2).$$

$$x = \frac{s_1 - s_2}{c_1 - c_2} = (s_1 - s_2)(c_1 - c_2)^{-1}.$$

Зная все параметры ($s_1, s_2, c_1 = H(X, R, m_1)$ и $c_2 = H(X, R, m_2)$), Оскар может получить закрытый ключ Алисы (x).

2.4. Агрегация подписей

Для понимания причины существования трех раундов в схемах мультиподписей и как предложенная методология в настоящем исследовании может устранить второй раунд путём демонстрации знания закрытых ключей, необходимо объяснить основные принципы, связанные с агрегацией подписей, присущие слабости и потенциальные возможности для эксплуатации.

Процесс создания подписей последовательно следует данным шагам:

- Сгенерировать секретное одноразовое число (r);
- Создать открытое одноразовое число $R = rG$;
- Алиса отправляет сообщение (m), R и открытый ключ ($P = kG$) Бобу, где

k – закрытый ключ.

Подпись создаётся хешированием всей приведённой публичной информации: $c = H(R \parallel P \parallel m)$. Хеш-функция выбирается так, чтобы (e) имело тот же диапазон, что и закрытые ключи. Теперь подпись конструируется с использованием приватной информации $s = r + ck$.

Боб может вычислить (c), так как он уже знает (m), (R) и (P). Однако он не знает закрытый ключ или одноразовое число.

Учитывая уравнение $sG = (r + ck)G$, получаем:

$$sG = rG + ckG = rG + (kG)c = R + Pc.$$

Таким образом, Бобу нужно вычислить (sG) и проверить, совпадает ли это вычисленное значение с правой частью уравнения ($R + Pc$). Боб уже имеет всю необходимую информацию для выполнения этих шагов.

Рассмотрим применение линейного свойства подписей Шнорра для создания мультиподписи «два из двух». В данном сценарии Алиса и Боб хотят совместно подписать сообщение, не полностью доверяя друг другу. Им нужен способ доказать владение своими

индивидуальными ключами, и комбинированная подпись будет действительна только в том случае, если и Алиса, и Боб внесут свой вклад в её создание.

Пусть закрытые ключи обозначены как k_i , а открытые ключи – как P_i . Если попросить Алису и Боба предоставить по одноразовому числу, то возможно

$$P_{agg} = P_a + P_b, \quad (1)$$

$$c = H(R_a \parallel R_b \parallel P_a \parallel P_b \parallel m), \quad (2)$$

$$s_{agg} = r_a + r_b + (k_a + k_b)c, \quad (3)$$

$$s_{agg} = (r_a + k_a c) + (r_b + k_b c) = s_a + s_b. \quad (4)$$

Оказывается, что и Алиса, и Боб могут внести свои собственные значения (R). Кроме того, любой может создать мультиподпись «два из двух», комбинируя эти значения (R) с соответствующими открытыми ключами. Однако данная схема страдает от уязвимости безопасности, известной как атака с аннулированием ключей (Key Cancellation Attack).

Чтобы понять атаку с аннулированием ключей, рассмотрим предыдущий сценарий с другой точки зрения. На этот раз Боб заранее знает открытый ключ и открытое одноразовое число Алисы. Он ждёт, пока она их раскроет, а затем утверждает, что его открытый ключ $P'_b = P_b - P_a$, а его открытое одноразовое число $R'_b = R_b - R_a$. У Боба нет закрытых ключей, соответствующих этим ложным значениям.

Несмотря на это, предполагается, что выражение для проверки агрегированной подписи остаётся $s_{agg}G = R_a + R'_b + c(P_a + P'_b)$. Однако Боб может создать эту подпись самостоятельно. Используем уравнение ($sG = R + Pc$) для объяснения уязвимости:

$$s_{agg}G = R_a + R'_b + c(P_a + P'_b) = R_a + (R_b - R_a) + c(P_a + P_b - P_a) = R_b + cP_b,$$

$$s_{agg}G = r_bG + ck_bG = G(r_b + ck_b),$$

$$s_{agg} = r_b + ck_b. \text{ Таким образом } s_{agg} = s_b.$$

Боба можно заставить пройти аутентификацию, подписав сообщение, доказывая его знание закрытых ключей. Данный метод работает, но он требует дополнительного раунда обмена сообщениями между сторонами. Чтобы предотвратить эту атаку, метод мультиподписи включает три раунда.

Проектирование схемы мультиподписи Шнорра

Рассмотрим сценарий, в котором группа из n подписантов намеревается совместно подписать заданное сообщение m . Изначально каждый подписант случайным образом вычисляет и передаёт уникальное значение $R_i = g^{r_i}$, где g – генератор основной группы, а r_i – случайно выбранный показатель, связанный с каждым подписантом.

Затем каждый подписант вычисляет два значения: $R = \prod_{i=1}^n R_i$, $c = H(\tilde{X}, R, m)$.

Здесь $\tilde{X}$ представляет собой произведение индивидуальных открытых ключей всех подписантов (агрегированный открытый ключ): $\tilde{X} = \prod_{i=1}^n X_i$.

Далее частичная подпись для каждого подписанта независимо вычисляется как $s_i = r_i + cx_i$, где x_i – секретный ключ подписанта i . Наконец, все частичные подписи объединяются в одну: $s \equiv \sum_{i=1}^n s_i \pmod{p}$.

Процесс проверки подписи (R, s) для сообщения m с учётом набора открытых ключей $\{X_1, \dots, X_n\}$ выражается уравнением $g^s = R\tilde{X}$.

Данное уравнение проверки соответствует уравнению проверки подписи Шнорра относительно открытого ключа $\tilde{X}$. Это свойство обычно называется агрегацией ключей.

Однако такие протоколы уязвимы для атаки с подменой ключа [17]. Злоумышленник намеренно устанавливает свой открытый ключ как $X_1 = g^{x_1}(\prod_{i=2}^n X_i)^{-1}$, что позволяет ему самостоятельно создавать подписи для набора индивидуальных открытых ключей $\{X_1, \dots, X_n\}$.

Схема подписей MuSig

MuSig – основанная на Шнорре схема мультиподписей, которая позволяет группе подписантов создать короткую, совместную подпись на общем сообщении. MuSig доказано безопасна в модели открытых ключей.

Алгоритм генерации и проверки подписи в MuSig следующий [18]:

MuSig параметризуется групповыми параметрами (G, p, g) и тремя хеш-функциями $(H_{com}, H_{agg}, H_{sig})$ – эти три хеш-функции могут быть идентичными или разными в зависимости от желаемого уровня безопасности. Использование различных хеш-функций повышает уровень безопасности.

– Раунд 1

Группа из (n) подписантов хочет совместно подписать сообщение (m) .

Пусть X_1 и x_1 – открытый и закрытый ключи конкретного подписанта, где $X_1 = g^{x_1}$.

Пусть $X_2, \dots, X_n$ – открытые ключи других подписантов, а L – мультимножество всех открытых ключей, участвующих в процессе подписания.

Для $i \in \{1, \dots, n\}$ подписант вычисляет $a_i = H_{agg}(L, X_i)$, а также агрегированный открытый ключ: $\tilde{X} = \prod_{i=1}^n X_i^{a_i}$.

– Раунд 2

Подписант генерирует случайное закрытое одноразовое число r_1 , вычисляет открытое одноразовое число $R_1 = g^{r_1}$ и коммитмент $t_1 = H_{com}(R_1)$, затем отправляет t_1 всем другим подписантам.

После получения коммитментов $t_2, \dots, t_n$ от других подписантов он отправляет R_1 всем участникам – это гарантирует, что открытое одноразовое число не будет раскрыто до получения всех коммитментов. После получения $R_2, \dots, R_n$ подписант проверяет, что $t_i = H_{com}(R_i)$ для всех $i \in \{2, \dots, n\}$.

Отправляя сначала коммитменты, подписанты могут гарантировать, что никто не сможет изменить своё открытое одноразовое число после просмотра других, что предотвращает атаку с подменой ключа, так как злоумышленнику нужно было бы знать открытые одноразовые числа других подписантов заранее для создания ложного открытого ключа. При обнаружении несоответствия подписант прерывает протокол – это гарантирует, что подписанты используют один и тот же набор открытых одноразовых чисел для вычисления агрегированной подписи, что важно для безопасности и правильности MuSig.

– Раунд 3

Если все пары коммитментов и открытых одноразовых чисел прошли проверку, следующим шагом является вычисление:

$$R = \prod_{i=1}^n R_i, \quad c = H_{sig}(\tilde{X}, R, m), \\ s_1 \equiv (r_1 + ca_1x_1) \bmod p.$$

Подпись s_1 отправляется всем другим подписантам. После получения $s_2, \dots, s_n$ подписант вычисляет $s \equiv \sum_{i=1}^n s_i \bmod p$. Подписью является пара (R, s) .

Для проверки агрегированной подписи сообщение (m) , мультимножество открытых ключей (L) , (R) и (s) отправляются получателю.

Получатель следует следующим шагам для проверки агрегированной подписи:

1. Вычислить $a_i = H_{agg}(L, X_i)$ для $i \in \{1, \dots, n\}$;
2. Вычислить $\tilde{X} = \prod_{i=1}^n X_i^{a_i}$;
3. Вычислить $c = H_{sig}(\tilde{X}, R, m)$;
4. Принять подпись, если $g^s = R \prod_{i=1}^n X_i^{c a_i} = R \tilde{X}^c$.

Доказательство корректности алгоритма:

$g^s = R \tilde{X}^c$ поскольку $R = \prod_{i=1}^n R_i = \prod_{i=1}^n g^{r_i}$ и так как $\tilde{X} = \prod_{i=1}^n X_i^{a_i} = \prod_{i=1}^n (g^{x_i})^{a_i}$, тогда $g^s = (\prod_{i=1}^n g^{r_i})(\prod_{i=1}^n g^{x_i a_i})^c = \prod_{i=1}^n g^{r_i + x_i a_i c} = g^{\sum_{i=1}^n s_i} = g^s$, так как $s \equiv \sum_{i=1}^n s_i$.

Этот метод устойчив к атакам с аннулированием ключей и атакам с подменой ключа, но уязвим для атаки с повторным использованием одноразового числа. Чтобы понять, как MuSig уязвима для описанной атаки, рассмотрим сценарий, в котором Алиса, среди группы совподписантов, повторно использует одно и то же одноразовое число для подписания различных сообщений. В этом случае Оскар может выполнить следующие шаги, чтобы раскрыть закрытый ключ Алисы (x):

Алиса подписывает сообщение m_1	Алиса подписывает сообщение m_2
1. Выбрать x_1 ; вычислить $X_1 = g^{x_1}$	1. Ключи x_1 и X_1 остаются теми же
Раунд 1. Вычислить $a_1 = H_{agg}(L, X_1)$ и $\tilde{X} = \prod_{i=1}^n X_i^{a_i}$	Раунд 1. Коэффициент a_1 и ключ $\tilde{X}$ остаются теми же
Раунд 2. Случайно выбрать r_1 ; вычислить $R_1 = g^{r_1}$ и $t_1 = H_{com}(R_1)$	Раунд 2. r_1, R_1 и t_1 остаются теми же
Отправить t_1 и получить $t_2, \dots, t_n$; затем отправить R_1 , получить $R_2, \dots, R_n$ и проверить t_i	Повторить обмен коммитментами и открытыми одноразовыми числами; проверить t_i
Раунд 3. Вычислить $R = \prod_{i=1}^n R_i$, $c = H_{sig}(\tilde{X}, R, m_1)$ и $s_1 = r_1 + c a_1 x_1$	Раунд 3. Вычислить $R = \prod_{i=1}^n R_i$, $c' = H_{sig}(\tilde{X}, R, m_2)$ и $s'_1 = r_1 + c' a_1 x_1$
Отправить s_1 , получить частичные подписи остальных участников и вычислить $s \equiv \sum_{i=1}^n s_i$	Отправить s'_1 , получить частичные подписи остальных участников и вычислить $s \equiv \sum_{i=1}^n s_i$

Если Оскар слушает этот разговор, он может вычесть две частичные подписи Алисы и раскрыть её закрытый ключ:

$$s_1 - s'_1 = r_1 + c a_1 x_1 - (r_1 + c' a_1 x_1) = a_1 x_1 (c - c'),$$

$$x_1 = (s_1 - s'_1) (a_1 (c - c'))^{-1}.$$

Оскар знает $(s_1, s'_1, R, m_1$ и $m_2)$ и может вычислить $c = H_{sig}(\tilde{X}, R, m_1)$, $c' = H_{sig}(\tilde{X}, R, m_2)$ и $a_1 = H_{agg}(L, X_1)$. Следовательно, можно вычислить закрытый ключ Алисы x_1 .

Рассмотрим сценарий, в котором Алиса и Боб пытаются совместно подписать сообщение (m_1):

Алиса подписывает сообщение m_1 Выбрать x_1, вычислить $X_1 = g^{x_1}$, случайно выбрать r_1 и вычислить $R_1 = g^{r_1}$ Отправить R_1 Бобу и получить R_2 Вычислить агрегированное $R = R_1 + R_2$, $c = H_{sig}(\tilde{X}, R, m_1)$ и $s_1 = r_1 + ca_1x_1$; Отправить s_1 Бобу По какой-то причине подписание не завершилось При возобновлении снова отправить R_1 Бобу и получить R'_2 Вычислить новое агрегированное R', $c' = H_{sig}(\tilde{X}, R', m_1)$ и $s'_1 = r_1 + c'a_1x_1$; Отправить s'_1 Бобу ... и т.д.	Боб подписывает сообщение m_1 Выбрать x_2, вычислить $X_2 = g^{x_2}$, случайно выбрать r_2 и вычислить $R_2 = g^{r_2}$ Отправить R_2 Алисе и получить R_1 Вычислить то же R, $c = H_{sig}(\tilde{X}, R, m_1)$ и $s_2 = r_2 + ca_2x_2$ По какой-то причине подписание не завершилось При возобновлении случайно выбрать новое r'_2, вычислить $R'_2 = g^{r'_2}$ и отправить Алисе Продолжить подписание с новым одноразовым числом
---	---

Если Оскар слушает этот разговор, он может выполнить операцию вычитания между двумя подписями Алисы. Результат раскроет закрытый ключ Алисы (x_1) следующим образом:

$$s_1 - s'_1 = r_1 + ca_1x_1 - r_1 - c'a_1x_1 = x_1a_1(c - c'),$$

$$x_1 = \frac{s_1 - s'_1}{a_1(c - c')} = (s_1 - s'_1)(a_1(c - c'))^{-1}.$$

Это подчёркивает уязвимость протокола MuSig к атакам с повторным использованием одноразовых чисел, когда одно и то же одноразовое число используется в нескольких операциях подписания.

Исходя из предыдущего объяснения, новшество в схеме подписей MuSig заключается в добавлении компонента a_i для решения проблемы атак с аннулированием ключей в агрегации ключей для схем мультиподписей. Кроме того, схема подписей MuSig использует трёхраундовый процесс подписания для противодействия атакам с подменой ключа, что является основным недостатком этого метода. В результате были предложены альтернативные модели для схем мультиподписей, которые требуют всего два раунда, такие как MuSig-DN и MuSig2.

Как упомянуто в разделе обзора литературы [12], сложность доказательства NIZK в протоколе MuSig-DN делает его непрактичным для использования на специализированных устройствах для подписания, таких как аппаратные кошельки, которые обычно используются для хранения биткойнов.

Согласно разделу обзора литературы этого исследования [13], MuSig2 может использовать разное количество одноразовых чисел в зависимости от конкретной конфигурации:

- MuSig2 может использовать 1 одноразовое число, но эта конфигурация может иметь ограничения в отношении гарантий безопасности.

– MuSig2 может использовать 2 одноразовых числа, чтобы быть безопасной при предположении алгебраического одно-дополнительного дискретного логарифма (AOMDL) в модели случайного оракула (ROM), при условии дополнительного предположения AGM.

– MuSig2 может использовать 4 или более одноразовых числа, чтобы быть доказанно безопасной при предположении одно-дополнительного дискретного логарифма (OMDL) в модели случайного оракула (ROM).

Таким образом, каждый подписант в MuSig2 предоставляет список как минимум из 2 одноразовых чисел, которые затем объединяются, чтобы сформировать агрегированное одноразовое число для операции подписания.

В модели случайного оракула (ROM) каждый подписант использует четыре или более одноразовых чисел $(R'_i, R''_i, R'''_i, R''''_i, \dots, R^n_i)$. Затем используется случайная комбинация $R_i = R'_i(R''_i)^b(R'''_i)^{b^2}(R''''_i)^{b^3}$. Показатель b устанавливается хешированием практически всех входных данных протокола и стенограммы после обмена одноразовыми числами: агрегированного открытого ключа, сообщения и одноразовых чисел всех подписантов:

$$b = H\left(\tilde{X}, m, (\prod_{i=1}^n R'_i, \prod_{i=1}^n R''_i, \prod_{i=1}^n R'''_i, \prod_{i=1}^n R''''_i, \dots, \prod_{i=1}^n R^n_i)\right). \quad (5)$$

В модели AGM каждый подписант использует два одноразовых числа (R'_i, R''_i) и их случайную комбинацию $R_i = R'_i(R''_i)^b$, где $b = H(\tilde{X}, m, (R'_1 R'_2, R''_1 R''_2))$.

При использовании 2 или 4 одноразовых чисел в схеме MuSig2 по сравнению с использованием одного одноразового числа возникает некоторая дополнительная вычислительная сложность – это связано с тем, что каждый подписант, предоставляющий 2 или 4 одноразовых числа в MuSig2, сталкивается с дополнительной нагрузкой при объединении этих одноразовых чисел для формирования агрегированного одноразового числа. Данный процесс включает в себя дополнительные вычисления для получения скаляра (b) с помощью хеш-функции. Кроме того, сложность для подписанта несколько увеличивается из-за необходимости обрабатывать несколько одноразовых чисел и выполнять необходимые вычисления для их агрегации

3. Предлагаемое решение

В этом разделе разъясняется предлагаемое решение проблемы генерации одноразовых чисел в схемах цифровой подписи и мультиподписи Шнорра, рассмотренных выше. Предложенное решение включает следующие положения.

Использование подхода генерации закрытого ключа, аналогичного тому, что используется в алгоритме цифровой подписи (EdDSA), для генерации детерминированных закрытых одноразовых чисел. В этом исследовании предлагается генерировать закрытое одноразовое число путём комбинирования сообщения с хешированным значением закрытого ключа (x).

Исключение открытого одноразового числа (R) из расчёта вызова $c = H_{sig}(X, R, m)$. Для обеспечения уникальности каждой подписи следует внедрить дополнительные контекстные и протокольные меры безопасности, аналогичные тем, которые используются в рамках EdDSA (что выходит за рамки данного исследования). Это включает интеграцию последовательных номеров, временных меток и других данных, специфичных для сессии, в сообщения. Эти компоненты, работающие вне механизма

цифровой подписи, предназначены для предотвращения определенных атак, особенно повторных атак.

В предлагаемой схеме мультиподписи агрегированный открытый ключ $\tilde{X}$ используется вместо доказательства с нулевым разглашением (NIZK), чтобы продемонстрировать, что владелец открытых ключей знает соответствующие секретные ключи. Следовательно, если агрегированный открытый ключ $\tilde{X}$ совпадает с любым из открытых ключей одного или нескольких совподписантов, процесс подписания отклоняется.

Процесс реализации этих решений описан в двух подразделах.

Предлагаемый алгоритм цифровой подписи Шнорра

Процесс генерации и проверки подписи в Шнорре следующий:

Чтобы подписать сообщение (m), Алиса выполняет следующие шаги:

1. Выбрать закрытый ключ (x), такой что $0 < x < p$;
2. Вычислить открытый ключ $X = g^x$;
3. Вычислить $h = H(x)$;
4. Вычислить закрытое одноразовое число $r = H(h \parallel m)$;
5. Вычислить открытое одноразовое число $R = g^r$;
6. Вычислить $c = H(X, m)$;
7. Вычислить $s \equiv (r + cx) \bmod p$;
8. Подписью является пара (R, s) . Алиса отправляет сообщение (m), открытый ключ (X), (R) и (s) Бобу.

Боб считает подпись Алисы действительной только в том случае, если $g^s = RX^c$.

Доказательство корректности алгоритма: поскольку $R = g^r$ и $X = g^x$, тогда:

$$RX^c = g^r (g^x)^c = g^{r+xc} = g^s, \text{ где } s = r + xc.$$

В текущем сценарии Алиса не может подписать два разных сообщения, используя одно и то же одноразовое число: закрытое одноразовое число меняется при изменении сообщения. Однако, если Алиса подписывает одно и то же сообщение дважды, она использует одно и то же одноразовое число. Рассмотрим шаги, которые Оскар может выполнить для реализации атаки с повторным использованием одноразового числа.

Алиса подписывает сообщение m_1	Алиса снова подписывает сообщение m_1
1. Выбрать x_1 и вычислить $X_1 = g^{x_1}$	1. Ключи x_1 и X_1 остаются теми же
2. Вычислить $h = H(x_1)$	2. $h = H(x_1)$ остаётся тем же
3. Вычислить $r_1 = H(h \parallel m_1)$	3. $r_2 = r_1 = H(h \parallel m_1)$
4. Вычислить $R_1 = g^{r_1}$	4. $R_2 = R_1$
5. Вычислить $c_1 = H(X_1, m_1)$	5. $c_2 = c_1 = H(X_1, m_1)$
6. Вычислить $s_1 = r_1 + c_1 x_1$	6. $s_2 = s_1 = r_2 + c_2 x_1$
7. Подпись – пара (R_1, s_1)	7. Подпись – та же пара (R_1, s_1)

Это показывает, что две подписи, (s_1) и (s_2) , идентичны, и вычитание одной из другой даёт ноль ($s_1 - s_2 = 0$). Следовательно, от дублирующих подписей не извлекается никакой пользы или информации. Таким образом безопасность системы не подвергается угрозе, когда одно и то же сообщение подписывается дважды с одним и тем же

одноразовым числом. Предложенный метод эффективно предотвращает атаки с повторным использованием одноразового числа, детерминированно вычисляя закрытое одноразовое число, обеспечивая его изменение с каждым новым сообщением – если одно и то же сообщение подписывается несколько раз, процесс остаётся безопасным.

Кроме того, предложенный метод подписания сообщений не требует хранения какой-либо информации во время процесса подписания. Необходимые вычисления можно выполнить на основе входных данных и полученных сообщений, что устраняет риск манипуляций через атаки с откатом виртуальной машины.

Для предотвращения возможной эксплуатации вероятностей коллизий рекомендуется использовать соответствующие криптографические хеш-функции.

Предлагаемая схема мультиподписи.

Для различения схем мультиподписей классификация следующая:

- Схемы мультиподписей, в которых требуется n из n подписантов для совместного создания действительной подписи.
- Схемы пороговых подписей, в которых любая подгруппа размером t из n подписантов из общего числа n подписантов может совместно создать действительную подпись.

Следует отметить, что данный раздел касается исключительно схем мультиподписей. Предлагаемая схема генерации и проверки мультиподписей следующая.

– Раунд 1

Группа из n подписантов хочет совместно подписать сообщение m .

Пусть X_1 и x_1 – открытый и закрытый ключи конкретного подписанта, где $X_1 = g^{x_1}$.

Пусть $X_2, \dots, X_n$ – открытые ключи других подписантов, а L – хеш-значение мультимножества всех открытых ключей: $L = H_{agg}(X_1 \parallel \dots \parallel X_n)$.

Для $i \in \{1, \dots, n\}$ каждый подписант вычисляет $a_i = H_{agg}(L, X_i)$ и агрегированный открытый ключ: $\tilde{X} = \prod_{i=1}^n X_i^{a_i}$. Подписант должен проверить, выполняется ли $\tilde{X} = X_i^{a_i}$. Если это так, операция подписания прекращается.

Подписант вычисляет $h = H_{agg}(x_1)$, закрытое одноразовое число $r_1 = H_{agg}(h \parallel m)$ и открытое одноразовое число $R_1 = g^{r_1}$, после чего отправляет R_1 остальным подписантам и получает от них $R_2, \dots, R_n$.

– Раунд 2

Вычислить $c = H_{sig}(\tilde{X}, m)$ и $s_1 \equiv (r_1 + ca_1x_1) \bmod p$.

Подпись s_1 отправляется всем другим подписантам. После получения $s_2, \dots, s_n$ они вычисляют $R = \prod_{i=1}^n R_i$, $s \equiv \sum_{i=1}^n s_i \pmod{p}$.

Подписью будет пара (R, s) . Сообщение m и пара (R, s) отправляются получателю. Получатель выполняет следующие шаги для проверки агрегированной подписи:

- 1) Вычислить $a_i = H_{agg}(L, X_i)$ для $i \in \{1, \dots, n\}$;
- 2) Вычислить $\tilde{X} = \prod_{i=1}^n X_i^{a_i}$;
- 3) Вычислить $c = H_{sig}(\tilde{X}, m)$;
- 4) Принять подпись, если $g^s = R\tilde{X}^c$.

Доказательство корректности алгоритма:

$g^s = R\tilde{X}^c$, поскольку $R = \prod_{i=1}^n R_i = \prod_{i=1}^n g^{r_i}$ и $\tilde{X} = \prod_{i=1}^n X_i^{a_i} = \prod_{i=1}^n (g^{x_i})^{a_i}$, то

$g^s = (\prod_{i=1}^n g^{r_i})(\prod_{i=1}^n g^{x_i a_i})^c = \prod_{i=1}^n g^{r_i + x_i a_i c} = g^{\sum_{i=1}^n s_i} = g^s$, поскольку $s = \sum_{i=1}^n s_i$.

Чтобы улучшить понимание рабочего механизма предложенного метода и его устойчивости к атаке с подменой ключа, используется простой пример для его разъяснения:

Предположим, что Алиса и Боб хотят совместно подписать сообщение (m), а затем отправить эту мультиподпись Роберту (получателю). Шаги для предложенной схемы мультиподписи следующие

– Раунд 1

Пусть X_a и x_a – открытый и закрытый ключи Алисы, где $X_a = g^{x_a}$, а X_b и x_b – открытый и закрытый ключи Боба, где $X_b = g^{x_b}$. Тогда $L = H_{agg}(X_a \parallel X_b)$.

Алиса вычисляет $a_a = H_{agg}(L, X_a)$ и, зная открытый ключ Боба X_b , вычисляет $a_b = H(L \parallel X_b)$. После этого она вычисляет агрегированный открытый ключ $\tilde{X} = X_a^{a_a} \cdot X_b^{a_b}$.

Если Алиса обнаруживает, что $\tilde{X} = X_b^{a_b}$, протокол прекращается, поскольку открытый ключ, который использует Боб, поддельный $X_b = (g^{x_b})^{a_b} \cdot (X_a^{a_a})^{-1}$. Данная манипуляция указывает на попытку Боба выполнить атаку с подменой ключа, эксплуатируя знание открытого ключа Алисы для подрыва целостности агрегированного открытого ключа, вычисляемого как $\tilde{X} = X_a^{a_a} \cdot X_b^{a_b} = X_a^{a_a} \cdot g^{x_b} \cdot (X_a^{a_a})^{-1} = g^{x_b} = X_b^{a_b}$. Прекращая процесс подписания на этом этапе, протокол предотвращает подделку подписи Бобом путём аннулирования вклада Алисы в открытый ключ.

Аналогично Боб вычисляет $a_b = H_{agg}(L, X_b)$ и, зная открытый ключ Алисы X_a , вычисляет $a_a = H(L \parallel X_a)$. Затем Боб вычисляет $\tilde{X} = X_a^{a_a} \cdot X_b^{a_b}$.

Если Боб обнаруживает, что $\tilde{X} = X_b^{a_b}$, протокол прекращается, указывая на возможную попытку атаки с подменой ключа со стороны Алисы.

Если ни Алиса, ни Боб не выполняют атаку с подменой ключа, подписание продолжается. Алиса вычисляет $h = H_{agg}(x_a)$, закрытое одноразовое число $r_a = H_{agg}(h \parallel m)$ и открытое одноразовое число $R_a = g^{r_a}$. Она отправляет R_a Бобу и получает от него R_b , вычисленное аналогичным образом

– Раунд 2

Алиса вычисляет $R = \prod_{i=1}^n R_i$, $c = H_{sig}(\tilde{X}, m)$ и $s_a \equiv (r_a + c a_a x_a) \bmod p$, затем отправляет s_a Бобу и получает от него s_b , вычисленное аналогично. Агрегированная подпись: $s = s_a + s_b \pmod{p}$. Сообщение (m) и пара (R, s) отправляются Роберту.

Роберт считает мультиподпись действительной только при выполнении $g^s = R \tilde{X}^c$.

При оценке эффективности предложенного метода против атак с повторным использованием одноразовых чисел следует отметить, что Алиса не может подписать два разных сообщения, используя одно и то же одноразовое число, так как закрытое одноразовое число меняется при изменении сообщения. Однако, если Алиса подписывает одно и то же сообщение дважды, то она использует одно и то же одноразовое число, что вызывает обеспокоенность по поводу возможной компрометации безопасности системы.

Рассмотрим, будет ли безопасность системы нарушена, если Алиса дважды совместно с Бобом подпишет сообщение m , используя одно и то же одноразовое число.

Первое подписание сообщения m

Выбрать x_a и вычислить $X_a = g^{x_a}$.

Раунд 1.

Вычислить $L = H_{agg}(X_a \parallel X_b)$,

$a_a = H_{agg}(L, X_a)$ и $\tilde{X} = X_a^{a_a} \cdot X_b^{a_b}$.

Проверить, выполняется ли $\tilde{X} = X_b^{a_b}$.

Вычислить $h = H(x_a)$, $r_a = H(h \parallel m)$ и $R_a = g^{r_a}$.

Отправить R_a Бобу и получить R_b .

Раунд 2.

Вычислить $R = R_a R_b$, $c = H_{sig}(\tilde{X}, m)$ и

$s_a = r_a + ca_a x_a$.

Отправить s_a Бобу, получить s_b и вычислить $s \equiv \sum_{i=1}^n s_i \pmod{p} = s_a + s_b$.

Повторное подписание сообщения m

Ключи x_a и X_a остаются теми же.

Раунд 1.

L , a_a и $\tilde{X}$ остаются теми же.

Повторить проверку агрегированного ключа.

Вычислить $h' = h$, $r'_a = r_a$ и $R'_a = R_a$.

Отправить R'_a Бобу и получить R_b .

Раунд 2.

$R = R'_a R_b = R_a R_b$, $c' = c$, $s'_a = s_a$.

Отправить s'_a Бобу, получить s_b и вычислить ту же агрегированную подпись.

Поскольку две подписи s'_a и s_a идентичны, вычитание одной из другой даёт ноль: $s_a - s'_a = 0$. Следовательно, повторные подписи не предоставляют новой информации при многократном подписании одного сообщения с тем же одноразовым числом.

Оценим предложенный метод применительно ко второму сценарию, рассмотренному в разделе об агрегации подписей: прерыванию и возобновлению подписания.

Алиса подписывает сообщение m

Выбрать x_a , вычислить $X_a = g^{x_a}$.

Раунд 1.

Вычислить $L = H_{agg}(X_a \parallel X_b)$ и

$a_a = H_{agg}(L, X_a)$.

Вычислить $\tilde{X} = X_a^{a_a} \cdot X_b^{a_b}$; проверить $\tilde{X} = X_b^{a_b}$.

Вычислить $h_a = H_{agg}(x_a)$, $r_a = H_{agg}(h_a \parallel m)$, $R_a = g^{r_a}$.

Раунд 2.

Отправить R_a Бобу и получить R_b .

Вычислить $R = R_a \cdot R_b$,

$c = H_{sig}(\tilde{X}, m)$ и $s_a = r_a + ca_a x_a$.

Отправить s_a Бобу и получить s_b .

По какой-то причине подписание не завершилось.

Боб подписывает сообщение m

Выбрать x_b , вычислить $X_b = g^{x_b}$.

Раунд 1.

Вычислить $L = H_{agg}(X_a \parallel X_b)$ и

$a_b = H_{agg}(L, X_b)$.

Вычислить $\tilde{X} = X_a^{a_a} \cdot X_b^{a_b}$; проверить $\tilde{X} = X_b^{a_b}$.

Вычислить $h_b = H_{agg}(x_b)$, $r_b = H_{agg}(h_b \parallel m)$, $R_b = g^{r_b}$.

Раунд 2.

Отправить R_b Алисе и получить R_a .

Вычислить $R = R_a \cdot R_b$, $c = H_{sig}(\tilde{X}, m)$ и

$s_b = r_b + ca_b x_b$.

По какой-то причине подписание не завершилось.

При возобновлении снова отправить R_a Бобу и получить R'_b .

Вычислить $R' = R_a R'_b$, $c' = H_{sig}(\tilde{X}, m) = c$ и $s'_a = r_a + c' a_a x_a = s_a$. Отправить s'_a Бобу и продолжить протокол.

При возобновлении вычислить $a_b, \tilde{X}, h_b$, $r'_b = H_{agg}(h_b \parallel m) = r_b$ и $R'_b = g^{r'_b} = R_b$. Отправить R'_b Алисе и получить R_a .

Продолжить протокол с восстановленными значениями.

В предложенном подходе Боб не может выбрать разные одноразовые числа для одного и того же сообщения. Даже если Боб попытается сделать это намеренно, Алиса подпишет то же сообщение с тем же уникальным одноразовым числом, что приведёт к идентичным подписям. Подписи (s'_a и s_a) идентичны, потому что открытое одноразовое число исключено из расчётов вызова. Вычитание одной подписи из другой даёт ноль: ($s_a - s'_a = 0$), поэтому Оскар не получает новой информации от дублирующих подписей.

Таким образом, предложенный метод эффективно предотвращает атаки с повторным использованием одноразовых чисел, детерминированно вычисляя и изменяя закрытое одноразовое число для каждого нового сообщения. Кроме того, метод не требует хранения какой-либо информации во время процесса подписания, так как все необходимые вычисления могут быть получены из входных данных и полученных сообщений. Это устраняет риск манипуляций через атаки с откатом виртуальной машины. Для предотвращения возможной эксплуатации вероятностей коллизий рекомендуется использовать подходящие криптографические хеш-функции.

Заключение

Предложенный метод улучшения цифровой подписи Шнорра и двухраундовой MuSig показывает перспективы в достижении нескольких важных целей. Во-первых, он улучшает цифровую подпись Шнорра, комбинируя закрытый ключ и сообщение через хеширование, после чего снова хеширует результат для генерации закрытого одноразового числа; способ вдохновлён методом, используемым в EdDSA.

Предложенный метод исключает открытое одноразовое число (R) из расчётов вызова ($c = H_{sig}(X, R, m)$), обеспечивая уникальность каждой подписи. Кроме того, исключение (R) из этих расчетов снижает вычислительную сложность, что является важным фактором для устройств с ограниченными ресурсами, где каждый вычислительный шаг имеет значение. Значительное преимущество этого метода заключается в том, что он гарантирует, что закрытое одноразовое число будет разным для каждого нового сообщения. Даже если одно и то же сообщение подписывается несколько раз с тем же одноразовым числом, это не нарушит безопасность системы, таким образом эффективно решая проблему сбоя генерации случайных чисел.

В схемах мультиподписей предложенный метод сравнивается со схемой подписей MuSig. В то время как MuSig устойчива к атакам с аннулированием ключа и атакам с подменой ключа, она включает три раунда и уязвима к атакам с повторным использованием одноразовых чисел. Напротив, предложенный метод является двухраундовой схемой, которая устойчива к атакам с аннулированием ключа, атакам с подменой ключа и атакам с повторным использованием одноразовых чисел, предлагая повышенную безопасность с меньшей вычислительной сложностью.

По сравнению с MuSig-DN, очевидно, что MuSig-DN вводит внутреннюю сложность из-за необходимости развёртывания доказательств с нулевым разглашением (NIZK) для проверки детерминированной генерации одноразовых чисел. Это требование не только увеличивает вычислительную нагрузку, но и добавляет сложности в реализации, делая протокол менее практичным для устройств с ограниченными ресурсами. Напротив, предложенный метод использует агрегированный открытый ключ $\tilde{X}$, чтобы продемонстрировать владение соответствующими секретными ключами. Этот подход устраняет необходимость вычислять и отправлять доказательства NIZK вместе с подписями, что приводит к снижению вычислительной сложности и использования пропускной способности. Таким образом, предложенный метод более эффективен с точки зрения вычислительной сложности и использования пропускной способности, особенно для устройств с ограниченными ресурсами. Сравнивая предложенный метод с MuSig2, ключевое различие заключается в том, как они обрабатывают одноразовые числа. MuSig2 позволяет использовать несколько одноразовых чисел при заданных параметрах схемы, где каждый подписант должен предоставить как минимум два одноразовых числа. Данные одноразовые числа затем объединяются для создания агрегированного одноразового числа для процесса подписания. Таким образом, генерация и использование нескольких одноразовых чисел в MuSig2 добавляет вычислительную сложность, особенно при выводе скаляра « b » с использованием хеш-функции. Кроме того, отправка нескольких одноразовых чисел в MuSig2 приводит к увеличению использования пропускной способности. Напротив, предложенный метод использует только одно одноразовое число. Данный упрощённый подход устраняет дополнительные вычислительные задачи и минимизирует требования к пропускной способности, делая процесс более эффективным и простым по сравнению с MuSig2.

Хотя исследование не утверждает однозначного превосходства над устоявшимися и проверенными методами MuSig-DN и MuSig2, оно представляет собой ценный вклад в продолжающиеся усилия по улучшению технологий цифровых подписей. В предложенном методе открытое одноразовое число (R) служит для процесса проверки получателем, чтобы гарантировать действительность подписи. Кроме того, предложенный метод устраняет необходимость хранения любой информации между раундами подписания, что означает, что подписантам не нужно запоминать такие детали, как одноразовое число или частичная подпись – данные значения можно повторно вычислить по открытым входным данным и сообщению. Такое преимущество снижает требования к хранению и вычислительным ресурсам для подписантов и повышает устойчивость к атакам путём отката состояния виртуальной машины. В результате такой подход предотвращает мошенничество с открытыми ключами, снижает вычислительную сложность и минимизирует объём передаваемой информации, объединённой с доказательством NIZK, что приводит к лучшему использованию пропускной способности.

Благодарности

Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации № 075-00003-24-02 (проект FSEE-2024-0003).

Список литературы

1. Diffie, W. and Hellman, M. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6), 644–654. doi: 10.1109/TIT.1976.1055638;
2. Rivest, R. L., Shamir, A., and Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2), 120–126. doi: 10.1145/359340.359342;
3. National Institute of Standards and Technology. (1994). FIPS PUB 186: Digital Signature Standard (DSS). Available from: <https://csrc.nist.gov/pubs/fips/186/upd1/final>;
4. Schnorr, C. P. (1991). Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3), 161–174. doi: 10.1007/BF00196725;
5. Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. (2012). High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2, 77–89. doi: 10.1007/s13389-012-0027-1;
6. Johnson, D. B., Menezes, A., and Vanstone, S. A. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1, 36–63;
7. Josefsson, S. and Liusvaara, I. (2017). Edwards-Curve Digital Signature Algorithm (EdDSA). IRTF. RFC 8032. doi: 10.17487/RFC8032. ISSN 2070-1721;
8. Beunardeau, M., Connolly, A., Ferradi, H., Géraud-Stewart, R., Naccache, D., & Vergnaud, D. (2017). Reusing Nonces in Schnorr Signatures. In: *Progress in Cryptology – AFRICACRYPT 2017*. AFRICACRYPT 2017. *Lecture Notes in Computer Science*, vol 10239. Springer, Cham. doi:10.1007/978-3-319-66402-6_14;
9. Romailier, Yolan and Pelissier, Sylvain. (2017). Practical Fault Attack against the Ed25519 and EdDSA Signature Schemes. *Proceedings of the 2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 17–24. doi: 10.1109/FDTC.2017.12;
10. Chalkias, K., Garillot, F., Kondi, Y., and Nikolaenko, V. (2021). Non-interactive Half-Aggregation of EdDSA and Variants of Schnorr Signatures. In: K.G. Paterson (ed.), *Topics in Cryptology – CT-RSA 2021*, *Lecture Notes in Computer Science*, 12704, Springer, Cham, doi: 10.1007/978-3-030-75539-3_24;
11. Kondi, Y., Orlandi, C., and Roy, L. (2023). Two-Round Stateless Deterministic Two-Party Schnorr Signatures from Pseudorandom Correlation Functions. In: H. Handschuh and A. Lysyanskaya (eds), *Advances in Cryptology – CRYPTO 2023*, *Lecture Notes in Computer Science*, 14081, Springer, Cham, doi: 10.1007/978-3-031-38557-5_21;
12. Nick, J., Ruffing, T., Seurin, Y., and Wuille, P. (2020). MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 1717–1731. doi: 10.1145/3372297.3417236;
13. Nick, J., Ruffing, T., and Seurin, Y. (2021). MuSig2: Simple Two-Round Schnorr Multi-signatures. In: T. Malkin and C. Peikert (eds), *Advances in Cryptology – CRYPTO 2021*, *Lecture Notes in Computer Science*, 12825, Springer, Cham, doi: 10.1007/978-3-030-84242-0_8;
14. Nguyen, P. Q. and Shparlinski, I. E. (2003). The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. *Designs, Codes and Cryptography*, 30(2), 201–217. doi: 10.1023/A:1025436905711;
15. Bitcoin.org. (2013, August 11). Android Security Vulnerability. Available from: <https://bitcoin.org/en/alert/2013-08-11-android>;

16. Boneh, D. (2005). Schnorr Digital Signature Scheme. DOI: 10.1007/0-387-23483-7_369;
17. Michels, M., & Horster, P. (1996). On the Risk of Disruption in Several Multiparty Signature Schemes. International Conference on the Theory and Application of Cryptology and Information Security;
18. Maxwell, G., Poelstra, A., Seurin, Y., & Wuille, P. (2019). Simple Schnorr multi-signatures with applications to Bitcoin. Designs, Codes and Cryptography. doi:10.1007/s10623-019-00608-x.