Концепция · 23 мая 2026 г.

← Предыдущий·Следующий →

24 слова: что такое криптографическая идентичность

Криптографическая идентичность — это не пароль: ни один сервер её не хранит, и она не восстанавливается. Дидактическое объяснение механизма BIP39, почему именно двадцать четыре слова и какая реальная ответственность ложится на того, кто ими владеет.

Чтобы мы понимали друг друга: если вы забудете пароль от Gmail, Google сбросит его для вас. Если вы потеряете 24 слова, из которых состоит криптографическая идентичность, их не у кого просить. Дело не в том, что процедура строгая, а в том, что на другом конце никого нет. В этом и заключается вся разница.

Разница между паролем и идентичностью

Пароль в классической модели интернета не является идентичностью пользователя. Это квитанция. У пользователя есть идентичность — имя, электронная почта, номер клиента — и, чтобы доказать серверу, что он тот, за кого себя выдает, он предъявляет пароль, который сервер сравнивает с хранящимся отпечатком. Если отпечатки совпадают, сервер разрешает сессию. Если пароль утерян, пользователь остается тем же пользователем; он теряет лишь квитанцию, и существует процедура восстановления — письмо на зарегистрированный адрес, секретный вопрос — для её замены.

Криптографическая идентичность работает иначе. Это не учетные данные, которые кто-то сравнивает с хранящимся отпечатком; это *сам по себе* полный математический секрет. Неважно, где он находится — на бумаге, в устройстве или даже на чужом сервере: идентичность существует благодаря своей математике, а не тому, кто её подтверждает. Здесь проявляется свойство, похожее на то, что мы видели в статье «Что такое SHA-256 на самом деле»: владение доказывается не предъявлением секрета, а использованием его для подписи. Созданную таким образом подпись может проверить любой желающий с помощью публичного значения, которое математически выводится из самого секрета, без необходимости знать сам секрет и без посредничества третьей стороны. Тот, у кого есть секрет, и есть идентичность; тот, кто его теряет, перестает ею быть. Вердикт категоричен: **некого просить вернуть вам идентичность. Этого «кого-то» не существует, потому что у него её изначально и не было.**

Что представляют собой двадцать четыре слова

Криптографическая идентичность обычно представляется математическим секретом длиной тридцать два байта — двести пятьдесят шесть бит. Число, которое трудно запомнить и еще труднее безошибочно переписать. Криптографическая индустрия решила эту проблему в 2013 году с помощью небольшого и элегантного стандарта под названием BIP39: способа представления этих двухсот пятидесяти шести бит в виде последовательности из двадцати четырех слов, взятых из официального списка из двух тысяч сорока восьми. Лежащая в его основе арифметика идеально подходит; желающие увидеть её в деталях найдут её на полях.

Отсчет начинается с конца. Мы хотим представить двести пятьдесят шесть бит секрета, добавив восемь бит контрольной суммы: всего двести шестьдесят четыре бита. Если мы разделим их на двадцать четыре

слова — число, удобное для записи и диктовки без потерь, — каждое слово должно содержать ровно одиннадцать бит информации. А одиннадцать бит — это два в одиннадцатой степени возможностей, то есть две тысячи сорок восемь. Вот почему официальный словарь BIP39 имеет именно такой размер: список существует под задачу, а не наоборот.

Этот расчет — не просто декорация. Если кто-то правильно перепишет двадцать три слова и ошибется в двадцать четвертом, контрольная сумма обнаружит это: программное обеспечение скажет ему «эта последовательность недействительна». Если кто-то перепишет все двадцать четыре правильно, программное обеспечение однозначно выведет ту же идентичность. Выбор списка слов также не случаен: слова в словаре BIP39 короткие, отличные друг от друга, без диакритических знаков, выбраны так, чтобы свести к минимуму фонетическую и орфографическую путаницу. Это словарь, созданный для того, чтобы люди могли его запоминать, записывать и диктовать без потерь.

От фразы к ключу

Двадцать четыре слова — это не криптографический ключ, которым подписываются сообщения. Они представляют собой восстанавливаемое представление исходной энтропии, которая с помощью детерминированного процесса под названием PBKDF2 превращается в 64-байтную «seed-фразу» (зерно). Из этого зерна также детерминированным образом выводятся конкретные криптографические ключи, используемые пользователем: закрытый ключ для подписи и соответствующий открытый ключ, который публикуется для проверки подписей. Тот же механизм в различных системах: криптовалюты используют кривую secp256k1; протокол Signal и многие современные системы используют Ed25519 на кривой Curve25519. Для конкретной кривой, такой как Ed25519, стандарты BIP32 и SLIP-0010 берут это 64-байтное зерно и детерминированно выводят 32 байта, составляющие эффективный ключ подписи — те самые 32 байта, с которых начинается пример кода в следующем разделе.

Это стандартный способ, которым вся индустрия представляет механизм пользователю — криптовалютные кошельки, менеджеры децентрализованной идентификации, Signal в своей части постоянной идентификации, Solo2 среди них: пользователь на практике никогда не видит ни зерна, ни производных ключей. Он видит двадцать четыре слова при создании своей личности и, при желании, записывает их на бумаге. Затем слова перемещаются между его устройствами, когда он хочет мигрировать личность: он вводит их в новое приложение, приложение выводит то же зерно, те же ключи, ту же личность. Это портативный, криптографически надежный и, в разумных пределах, запоминающийся механизм.

Как подписывать ключом (штрих Zig)

В Zig, как только у вас есть 32-байтное зерно, полученное из двадцати четырех слов, подпись сообщения с помощью Ed25519 укладывается в несколько строк:

```
const std = @import("std");
const Ed25519 = std.crypto.sign.Ed25519;

// 'semilla' son los 32 bytes derivados de las 24 palabras.
const par = Ed25519.KeyPair.create(semilla);

// Firmar un mensaje con la clave privada:
const mensaje = "Este mensaje lo escribí yo.";
const firma = try par.sign(mensaje, null);

// Cualquiera con la clave pública del par puede verificar:
try Ed25519.Signature.verify(firma, mensaje, par.public_key);
```

Операция подписи производит шестьдесят четыре байта — называемых подписью — которые могли быть созданы только на основе соответствующего закрытого ключа. Проверка является публичной: любой, у кого есть открытый ключ, может проверить, соответствует ли подпись сообщению. Без закрытого ключа никто не может создать действительную подпись для этого сообщения; с открытым ключом каждый может обнаружить, является ли подпись действительной. Эта асимметрия позволяет подписывающему лицу доказать авторство, не делясь секретом.

Предыдущий пример — это минимальная версия из учебника. В реальном коде Solo2 цепочка проходит через два файла: один на JavaScript, который живет в браузере пользователя и восстанавливает энтропию из двадцати четырех слов, другой на Zig в составе библиотеки *zcatcrypto*, который берет эту энтропию и выводит конкретные криптографические ключи. Начиная со стороны браузера:

```
// solo2/web-app/js/lib/bip39.js
async function mnemonicToEntropy(mnemonic, lang) {
    const validation = await validateMnemonic(mnemonic, lang);
    if (!validation.valid) {
        return { entropy: null, valid: false, error: validation.error };
    }
    const wordlist = WORDLISTS[lang || 'en'];
    const words = mnemonic.trim().split(/\s+/);

    // Cada palabra aporta 11 bits (su índice en la lista de 2048).
    let bits = '';
    for (let i = 0; i < words.length; i++) {
        bits += wordlist.indexOf(words[i]).toString(2).padStart(11, '0');
    }

    // 24 palabras = 264 bits. Los primeros 256 son la entropía.
    const entropyBytes = new Uint8Array(32);
    for (let j = 0; j < 32; j++) {
        entropyBytes[j] = parseInt(bits.slice(j * 8, (j + 1) * 8), 2);
    }
    return { entropy: entropyBytes, valid: true };
}
```

Эти тридцать два байта энтропии вместе с другими тридцатью двумя, полученными на том же этапе, отправляются в модуль WebAssembly на Zig, который генерирует сами ключи Ed25519. Полная функция с финальной очисткой памяти умещается на одном экране:

```
// zcatcrypto/wasm/bindings/identity.zig
const Ed25519 = std.crypto.sign.Ed25519;
const X25519  = std.crypto.dh.X25519;

export fn identity_generate() ?*IdentityHandle {
    var seed: [64]u8 = undefined;
    if (!common.getRandomBytes(&seed)) return null;

    const handle = common.wasm_allocator.create(IdentityHandle) catch return null;

    // Bytes 0..31: semilla determinista del par Ed25519 (firma).
    const sign_kp = Ed25519.KeyPair.generateDeterministic(seed[0..32].*) catch {
        common.wasm_allocator.destroy(handle);
        return null;
    };
    handle.sign_secret = sign_kp.secret_key.toBytes();
```

```
    handle.sign_public = sign_kp.public_key.toBytes();

    // Bytes 32..63: secreto X25519 (para acordar claves de cifrado con el otro).
    handle.exchange_secret = seed[32..64].*;
    handle.exchange_public = X25519.recoverPublicKey(handle.exchange_secret) catch {
        common.wasm_allocator.destroy(handle);
        return null;
    };

    @memset(&seed, 0);  // Borra la semilla de la memoria.
    return handle;
}
```

Стоит отметить две детали. Первая: одно и то же начальное число (seed) всегда создает одну и ту же пару ключей — именно это позволяет восстановить личность, введя двадцать четыре слова на новом устройстве. Вторая: начальное число явно стирается из памяти в последней строке. После этого момента даже сама функция не сможет восстановить ключи; единственным источником будут слова пользователя.

Для тех, кто хочет проверить это на маленьких числах. Схему подписи можно полностью проследить на цифрах, достаточно малых для ручного расчета. Те, кто предпочитает не углубляться в арифметику, могут пропустить этот блок, не теряя нити статьи; те, кто хочет увидеть механизм в действии шаг за шагом, найдут его здесь. **Публичные правила**, которые может прочитать каждый: простое число $p = 23$ (в реальном Ed25519 оно состоит примерно из семидесяти семи цифр; мы используем двадцать три, чтобы расчеты уместились на одной странице), основание $g = 2$, порядок которого в этой группе равен $q = 11$, и соглашение о том, что вся арифметика с g выполняется *módulo p*, а все показатели степени сокращаются *módulo q*. **Приватный выбор**, единственный и никогда не разглашаемый: секрет $x = 6$. Это и есть личность.

Шаг 1 — Публичная часть личности. Вычисляется один раз и публикуется открыто.

$$y = g^x \bmod p$$

$$y = 2^6 \bmod 23 = 64 \bmod 23 = 18$$

Публичная часть личности — **18**. Любой может взять ее и использовать для проверки подписей, сделанных от имени этой личности. Никто, зная только 18, не может восстановить секрет 6: в этом и заключается проблема дискретного логарифма, к которой мы вернемся в конце.

Шаг 2 — Подписание сообщения. Владелец личности хочет подписать сообщение $m = 7$. Он начинает с выбора нового случайного значения $k = 4$, которое будет использовано только один раз и никогда не будет передано (в реальном Ed25519 k выводится детерминировано из сообщения и секрета во избежание опасности повторного использования, но роль оно играет именно такую). Затем он вычисляет три числа:

$$r = g^k \bmod p = 2^4 \bmod 23 = 16$$

$$e = H(r, m) \bmod q = (16 + 7) \bmod 11 = 1$$

$$s = (k + x \cdot e) \bmod q = (4 + 6 \cdot 1) \bmod 11 = 10$$

Подпись — это пара **(r, s) = (16, 10)**. Она передается открыто вместе с сообщением. Любой может ее прочитать. Методическое примечание: в реальном Ed25519 функция H — это SHA-512, криптографически стойкая; здесь мы используем упрощение $e = (r + m) \bmod q$, чтобы читатель мог проследить шаги без необходимости вычислять хеш. Структура алгоритма та же.

Шаг 3 — Проверка подписи. Проверяющий имеет публичную часть $y = 18$, сообщение $m = 7$ и подпись $(r, s) = (16, 10)$. Он восстанавливает e тем же способом — $e = (16 + 7) \bmod 11 = 1$ — и проверяет, выполняется ли это равенство:

$$g^s \bmod p \stackrel{?}{=} r \cdot y^e \bmod p$$

Вычисляет обе стороны отдельно:

Izquierda: $2^{10} \bmod 23 = 1024 \bmod 23 = 12$

Derecha: $16 \cdot 18^1 \bmod 23 = 288 \bmod 23 = 12$

Обе стороны дают **12**. Подпись действительна. Любой, имея публичную часть 18, может прийти к этому выводу, никогда не зная, что секретом было число 6.

А что, если кто-то третий попытается подделать? Ева видела все публичные данные, проходящие по каналу: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Чтобы подписать *другое* сообщение от имени этой личности, ей нужно было бы знать x. Ее единственный путь — задаться вопросом: «при каком показателе степени x выполняется $2^x \bmod 23 = 18$?». При $p = 23$ она может перебрать 0, 1, 2, 3, … и найти его за секунды. Но если заменить 23 на простое число реальных масштабов Ed25519, пространство возможных экспонент превысит число атомов в обозримой Вселенной. **На сегодняшний день человечеству не известно ни одного алгоритма, способного перебрать это пространство менее чем за миллиарды лет.** Это та же проблема дискретного логарифма, которая лежит в основе протокола Diffie-Hellman из предыдущей статьи, примененная здесь к схеме подписи.

То, что мы только что разобрали, — это *именно* Schnorr, схема подписи, вариантом которой (адаптированным к эллиптической кривой) является Ed25519. В реальном Ed25519 все операции выполняются над точками конкретной кривой (Curve25519), а не над целыми числами по модулю простого числа, и функция H — это SHA-512 вместо упрощенной суммы, которую мы использовали выше. Обе замены являются корректировками реализации — для повышения криптостойкости к полному перебору и получения дополнительных свойств безопасности для k. Алгоритмическая структура, три операции и причина асимметрии остаются теми же.

Здесь уместно сделать короткую остановку, так как всю цепочку можно при беглом взгляде спутать с другим примитивом из этой тройки — хешем. Это не он. Хеш — это уникальная функция, которая сжимает: на входе много байтов, на выходе короткий отпечаток, и на этом путь заканчивается. Криптографическая личность — это математически дополняющая друг друга пара: секрет остается у владельца и подписывает, а его публичная часть публикуется и проверяет. Там, где хеш сворачивает информацию в одном направлении, личность устанавливает асимметрию между двумя половинами. Хеш свидетельствует о том, что было сказано; личность свидетельствует о том, кто это сказал.

Чем фраза не является

Следует развеять три распространенных заблуждения. Фраза не является паролем в собственном смысле слова: она не сравнивается с отпечатком, хранящимся на сервере; она вводится в устройство пользователя для математического восстановления личности. Фраза не восстанавливается: если она потеряна, не у кого ее просить; если она дублируется, дублируется и личность. Фраза не является учетными данными, отделяемыми от личности: фраза *есть* личность. Тот, у кого она есть, может действовать от ее имени без дополнительного разрешения, без процесса авторизации, без возможности восстановления.

Именно это третье свойство меняет вес дела. Потерянный пароль — это административное неудобство. Потерянная криптографическая личность — это сама личность. Бумага с фразой, найденная третьими лицами, — это не риск кражи аккаунта: это передача всей личности. Обещание системы — что никто не может отозвать вашу личность или произвольно заблокировать вас — неразрывно сопровождается

ответственностью — что вы являетесь единственным хранителем того, что никто не может восстановить за вас.

Обещание и вес

Модель криптографической идентификации часто называют *самосуверенной* — self-sovereign в англосаксонской литературе. Выбор слова является преднамеренным и довольно точно описывает состояние. Пользователь суверенен над своей личностью почти в средневековом смысле: ее не жалует ни король, ни эмитент, ни центральная власть; и никто из вышеперечисленных не может ее отозвать. Но также, подобно средневековому монарху, пользователь несет полную ответственность за свои ошибки: нет регента, который принимал бы решения за него, если он потеряет печать.

Выбор между личностью, управляемой третьей стороной, и самосуверенной личностью не имеет единственного универсального правильного ответа. Для учетной записи на несущественном форуме управляемая личность, вероятно, пропорциональна риску. Для профессиональной личности, подписывающей юридически обязывающие документы, для экономической личности, охраняющей собственные сбережения, для личности профессионального общения с клиентами, которые доверили конфиденциальную информацию, вопрос меняется. Там вопрос перестает быть «это удобно?» и становится «кто, кроме меня, имеет власть действовать от моего имени и при каких обстоятельствах?»

Где этот механизм появляется в реальных системах

BIP39 появился в мире Bitcoin в 2013 году и быстро распространился на всю экосистему криптовалют: любой серьезный кошелек сегодня принимает фразу BIP39 из двенадцати или двадцати четырех слов в качестве резервной копии экономической личности ее владельца. За пределами криптовалют та же базовая концепция — криптографическая пара, доказывающая авторство без посредника — появляется в других системах с иным синтаксисом. SSH-ключи, которые системный администратор использует для доступа к своим серверам, являются классическим случаем: закрытый ключ администратор хранит на своей машине, а открытый копируется на каждый сервер; никакая структура, сопоставимая с централизованной службой, не вмешивается. Протокол Signal использует Ed25519 с постоянным материалом ключа на устройстве; европейский стандарт eIDAS в части квалифицированной подписи основывается на том же криптографическом принципе, с той разницей, что ключ хранится у квалифицированного поставщика доверенных услуг, а не у пользователя.

Solo2, издательская платформа этой публикации, использует фразу BIP39 из двадцати четырех слов в качестве личности каждого пользователя. Пользователь видит эти слова один раз при создании своей учетной записи. Они не хранятся ни на одном сервере Solo2 или кого-либо еще: если пользователь запишет их и сохранит, он сохранит свою личность навсегда. Если он их потеряет, он их потеряет. Это логическое следствие архитектуры без оператора-посредника: если бы Solo2 могла вернуть личность пользователю, который ее потерял, она также могла бы отдать ее любому, кто нажмет на Solo2, чтобы ее получить.

Для профессионального читателя

Четыре соображения для тех, кто рассматривает возможность внедрения криптографической самосуверенной (autosoberana) личности в профессиональном контексте:

1. Фраза и есть личность. Физическое хранение — бумага, несколько копий в разных местах, в конечном итоге гравировка на металле для долгосрочного использования — предлагает больше гарантий, чем цифровое хранение, которое увеличивает поверхность атаки, не снижая риска потери.

2. Восстановления нет. Разработка процесса исходя из предположения, что однажды основная копия будет потеряна, гораздо целесообразнее, чем обнаружить это в день потери. Вторая географически удаленная копия решает почти все сценарии.
3. Это не то же самое, что квалифицированный сертификат eIDAS. Для квалифицированной подписи в Союзе — нотариальные акты, определенные процедуры в администрации — законодательство требует наличия квалифицированного поставщика, который хранит ключ. Криптографическая самосуверенная личность служит для профессионального общения и документального подписания с доказательной силой, но не заменяет автоматически квалифицированный сертификат в случаях, когда этого требует норма.
4. Если личность подлежит передаче — наследство, профессиональное преемство, прекращение деятельности — целесообразно подготовить процедуру заранее, а не после. Формальные процедуры с запечатанными сургучом (lacre) конвертами, инструкции исполнителю завещания, хранение в нотариальной конторе — это классические механизмы, полностью совместимые с криптографической природой актива.

Эта статья завершает концептуальное трио, открывшее цикл — хэш, шифрование, личность. Эти три идеи строятся одна на другой: хэш дает неизменяемый отпечасок, шифрование дает конфиденциальность без доверенной третьей стороны, личность дает авторство без предоставляющей третьей стороны. Все три обладают свойством, которое также не является идеологическим: они передают от того, кто управляет сервисом, тому, кто его использует, технические возможности, которые традиционно принадлежали оператору. Вместе с ними передается и ответственность. Честный разговор о любой из этих трех идей требует разговора и о двух других.

Источники и дополнительная литература

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, предложение по улучшению Bitcoin 2013 года. Стандарт де-факто для фраз восстановления в криптоиндустрии.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), включая Ed25519. IETF, январь 2017 г. Нормативная спецификация схемы подписи, используемой в значительной части современной индустрии.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, версия 2.0. IETF, сентябрь 2000 г. Определяет алгоритм PBKDF2, используемый в производной BIP39 от фразы к начальному числу (seed).
- Регламент (ЕС) 910/2014 (eIDAS) и его развитие Регламентом (ЕС) 2024/1183 (eIDAS 2) — европейская база электронной идентификации и квалифицированной подписи. Режим, отличный от самосуверенного, но концептуально опирающийся на те же криптографические примитивы.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Канонический текст о принципах и обязательствах самосуверенной модели, более ранний, но актуальный для понимания семейства современных решений.

← ПредыдущийБизнес-модель как сигнал доверияСледующий →Self-hosting как профессиональная практика

Недавние материалы

- Размышление · 29 июня 2026 г. Вы не анонимны
- Размышления · 27 мая 2026 г. То, что подпись не может исправить
- Анализ · 26 мая 2026 г. Реальная vs мнимая конфиденциальность: вопросы, которые стоит себе задать

Возьмите эту статью с собой туда, где она вам понадобится.

↓ Markdown ↓ Простой текст ↓ PDF

Файл будет загружен на ваше устройство. Оттуда вы можете сохранить его, импортировать в Solo2 или поделиться им где угодно. Cuadernos не решает место назначения за вас.

Сургучная печать · SHA-256 8aae4f5046e4aa699a94a4949b14bd3fad741e1ede1295469d25b6fd041d5f3b

Cuadernos Lacre · Издание Menzuri Gestión S.L. ·
текст R.Eugenio · под редакцией команды Solo2.

Этот сайт не использует куки. Всё, что загружает ваш браузер, написано или контролируется нами и размещено на наших европейских серверах: анонимный счетчик посещений (Umami, самостоятельно размещенный) и минимум JavaScript, необходимый для выбора языка и вашей настройки светлой/темной темы, которая сохраняется на вашем собственном устройстве. Без сторонних ресурсов, без трекеров, без профилирования, без передачи данных. Если вы хотите следить за нами: RSS.