Концепция · 23 май 2026 г.

← Предишна·Следваща →

24-те думи: какво е криптографска идентичност

Криптографската идентичност не е парола: никой сървър не я съхранява и тя не може да бъде възстановена. Дидактично обяснение на механизма BIP39, защо точно двадесет и четири думи и каква реална отговорност пада върху този, който ги притежава.

За да се разберем: Ако забравите паролата си за Gmail, Google я нулира. Ако загубите 24-те думи, които съставляват криптографска идентичност, няма от кого да ги поискате. Не че процедурата е строга – просто на другия край няма никого. Тази разлика е фундаментална.

Разликата между парола и идентичност

Паролата в класическия модел на интернет не е идентичността на потребителя. Тя е доказателство. Потребителят има идентичност – име, имейл, клиентски номер – и за да докаже пред сървър, че е този, за когото се представя, представя парола, която сървърът сравнява със съхранен отпечатък. Ако отпечатъците съвпадат, сървърът разрешава сесията. Ако паролата бъде загубена, потребителят остава същият потребител; това, което губи, е доказателството, и съществува процедура за възстановяване – имейл до регистрирания адрес, таен въпрос – за възстановяването му.

Криптографската идентичност работи по друг начин. Тя не е удостоверение, което някой сравнява със съхранен отпечатък; тя *е* пълна математическа тайна сама по себе си. Без значение къде се намира – на хартия, в устройство или дори в чужд сървър – идентичността съществува благодарение на своята математика, а не заради този, който я валидира. Тук се появява свойство, подобно на това, което видяхме в „Какво всъщност е SHA-256“: притежанието не се доказва чрез показване на тайната, а чрез използването ѝ за подписване. Така произведеният подпис може да бъде проверен от всеки с публична стойност, която се извежда математически от самата тайна, без да е необходимо тя да бъде известна и без намесата на трета страна. Който има тайната, е идентичността; който я загуби, престава да бъде такава. Присъдата е категорична: **няма от кого да поискате да ви върне идентичността. Такъв човек не съществува, защото той изначално не я е притежавал.**

Какво представляват двадесет и четири думи

Криптографската идентичност обикновено се представя чрез математическа тайна от тридесет и два байта – двеста петдесет и шест бита. Число, което е трудно за запомняне и още по-трудно за преписване без грешка. Криптографската индустрия реши този проблем през 2013 г. с малък и елегантен стандарт, наречен BIP39: начин за представяне на тези двеста петдесет и шест бита като последователност от двадесет и четири думи, взети от официален списък от две хиляди четиридесет и осем. Аритметиката зад него пасва елегантно; желаещите да я видят в детайли могат да я намерят в страничната бележка.

Броенето започва отзад напред. Искаме да представим двеста петдесет и шест бита на тайната, добавяйки осем бита контролна сума: общо двеста шестдесет и четири бита. Ако ги разделим на двадесет и четири думи – управляемо число за записване и диктуване без загуба – всяка дума трябва да носи точно единадесет бита информация. Единадесет бита са две на единадесета степен възможности, тоест две

хиляди четиридесет и осем. Ето защо официалният речник на BIP39 има точно този размер: списъкът съществува според проблема, а не обратното.

Броенето не е декоративно. Ако някой препише двадесет и три думи правилно и сгреши двадесет и четвъртата, контролната сума ще го засече: софтуерът ще му каже „тази последователност е невалидна“. Ако някой препише и двадесет и четирите думи правилно, софтуерът ще изведе същата идентичност без неясноти. Изборът на списъка с думи също е преднамерен: думите в речника на BIP39 са кратки, различни една от друга, без диакритични знаци, избрани да минимизират фонетични и правописни грешки. Това е речник, проектиран да бъде запомнян, записван и диктуван от хора без загуби.

От фразата до ключа

Двадесет и четирите думи не са криптографският ключ, който подписва съобщения. Те са възстановимо представяне на оригиналната ентропия, която чрез детерминиран процес, наречен PBKDF2, се трансформира в семе (seed) от шестдесет и четири байта. От това семе се извеждат, също по детерминиран начин, конкретните криптографски ключове, които потребителят използва: частен ключ за подписване и съответен публичен ключ, който се публикува за проверка на подписите. Същият механизъм в различни системи: криптовалутите използват кривата secp256k1; протоколът Signal и много модерни системи използват Ed25519 върху кривата Curve25519. За конкретна крива като Ed25519 стандартите BIP32 и SLIP-0010 вземат това семе от шестдесет и четири байта и извеждат детерминирано тридесет и двата байта, които съставляват ефективния ключ за подписване — същите тридесет и два байта, с които започва примерният код в следващия раздел.

Това е стандартният начин, по който цялата индустрия представя механизма на потребителя —портфейли за криптовалути, мениджъри на децентрализирана идентичност, Signal в неговата част за постоянна идентичност, Solo2 сред тях—: потребителят на практика никога не вижда семето или извлечените ключове. Той вижда двадесет и четирите думи при създаването на идентичността си и по желание ги записва на хартия. След това думите пътуват между неговите устройства, когато иска да мигрира идентичността: той ги въвежда в новото приложение, приложението извлича същото семе, същите ключове, същата идентичност. Това е преносим, криптографски надежден и в рамките на разумното, запомнящ се механизъм.

Как се подписва с ключа (щрих от Zig)

В Zig, след като имате тридесет и двата байта семе, извлечени от двадесет и четирите думи, подписването на съобщение с Ed25519 се вписва в няколко реда:

```
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);
```

Операцията по подписване произвежда шестдесет и четири байта —наречени подпис— които могат да бъдат генерирани само от съответния частен ключ. Проверката е публична: всеки с публичния ключ може да провери дали подписът съответства на съобщението. Без частния ключ никой не може да произведе

валиден подпис за това съобщение; с публичния ключ всеки може да открие дали един подпис е валиден. Тази асиметрия е това, което позволява на подписващия да докаже авторство, без да споделя тайната.

Предишният пример е минималната версия на ръководството. В реалния код на Solo2 веригата преминава през два файла, единият в JavaScript, който живее в браузъра на потребителя и реконструира ентропията от двадесет и четирите думи, а другият в Zig в рамките на библиотеката *zcatcrypto*, която взема тази ентропия и извежда конкретните криптографски ключове. Започвайки от страната на браузъра:

```
// 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 };
}
```

Тези тридесет и два байта ентропия, заедно с други тридесет и два, извлечени в същата стъпка, пътуват до модула WebAssembly на Zig, който генерира самите Ed25519 ключове. Пълната функция, с нейното окончателно почистване на паметта, се побира на един екран:

```
// 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;
}
```

Два детайла си струва да се отбележат. Първият: една и съща семейна стойност винаги произвежда една и съща двойка ключове — именно това позволява възстановяването на самоличността чрез въвеждане на двадесет и четирите думи в ново устройство. Вторият: семейната стойност се изтрива изрично от паметта на последния ред. След тази точка дори самата функция не би могла да реконструира ключовете; думите на потребителя биха били единственият източник.

За тези, които искат да го проверят с малки числа. Схемата за подписване може да бъде премината изцяло с достатъчно малки цифри, за да се направят сметките на ръка. Тези, които предпочитат да не навлизат в аритметика, могат да прескочат този блок, без да губят нишката на статията; тези, които искат да видят механизма работещ стъпка по стъпка, ще го намерят тук. **Публичните правила**, които всеки може да прочете: просто число $p = 23$ (в реалния Ed25519 то е от около седемдесет и седем цифри; използваме двадесет и три, за да се поберат сметките на една страница), основа $g = 2$, чийто ред в тази група е $q = 11$, и конвенцията, че цялата аритметика с g се прави *módulo p* и всички експоненти се редуцират *módulo q*. **Частният избор**, единствен и никога несподелян: тайната $x = 6$. Това е самоличността.

Стъпка 1 — Публичната част на самоличността. Изчислява се веднъж и се публикува открито.

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

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

Публичната част на самоличността е **18**. Всеки може да я вземе и да я използва за проверка на подписи, направени с тази самоличност. Никой, наблюдавайки само 18, не може да възстанови тайната 6: това е проблемът с дискретния логаритъм, към който ще се върнем накрая.

Стъпка 2 — Подписване на съобщение. Притежателят на самоличността иска да подпише съобщението $m = 7$. Той започва, като избира нова случайна стойност $k = 4$, която ще се използва само веднъж и никога няма да бъде споделяна (в реалния Ed25519 k се извежда детерминистично от съобщението и тайната, за да се избегне опасността от повторно използване, но ролята, която играе, е точно тази). След това изчислява три числа:

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

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

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

Подписът е двойката **(r, s) = (16, 10)**. Пътува открито заедно със съобщението. Всеки може да го прочете. Дидактическа бележка: в реалния Ed25519 функцията H е SHA-512, криптографски надеждна; тук използваме опростяването $e = (r + m) \bmod q$, така че читателят да може да премине през стъпките, без да е необходимо да изчислява хеш. Структурата на алгоритъма е същата.

Стъпка 3 — Проверка на подписа. Проверяващият има публичната част $y = 18$, съобщението $m = 7$ и подписа $(r, s) = (16, 10)$. Реконструира e по същия начин —$e = (16 + 7) \bmod 11 = 1$— и проверява дали това равенство е изпълнено:

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

Изчислява двете страни поотделно:

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

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

Двете страни дават **12**. Подписът е валиден. Всеки с публичната част 18 може да стигне до това заключение, без изобщо да е знаел, че тайната е 6.

Ами трета страна, която се опитва да фалшифицира? Ева е видяла всичко публично, преминаващо през канала: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. За да подпише *различно* съобщение от името на тази самоличност, тя би трябвало да знае x. Единственият ѝ начин е да се запита: „за коя експонента x е изпълнено $2^x \bmod 23 = 18$?“. С $p = 23$ тя може да опита 0, 1, 2, 3, … и да го намери за секунди. Но при замяната на 23 с просто число от реалните измерения на Ed25519, пространството от възможни експоненти надвишава броя на атомите в наблюдаемата вселена. **Днес не съществува алгоритъм, известен на човечеството, който да може да премине през това пространство за по-малко от милиарди години.** Това е същият проблем с дискретния логаритъм, който е в основата на Diffie-Hellman от предишната статия, приложен тук към схемата за подписване.

Това, през което току-що преминахме, е *точно* Schnorr, схемата за подписване, на която Ed25519 е вариант, адаптиран към елиптична крива. В реалния Ed25519 всички операции се извършват върху точките на конкретна крива (Curve25519), вместо върху цели числа по модул на просто число, а функцията H е SHA-512 вместо опростената сума, която използвахме по-горе. Двете замени са корекции на изпълнението — за спечелване на криптографска устойчивост срещу груба сила и спечелване на допълнителни свойства за сигурност на k. Алгоритмичната структура, трите операции и причината за асиметрията са едни и същи.

Тук е уместно да направим кратка пауза, тъй като цялата верига може да бъде объркана на пръв поглед с друга примитива от триото: хеша. Тя не е такава. Хешът е уникална функция, която компресира — влизат много байтове, излиза кратък отпечатък и там пътят свършва. Криптографската самоличност е математически допълваща се двойка: тайната остава и подписва; нейната публична част се публикува и проверява. Докато хешът срива информацията в една посока, самоличността установява асиметрия между две половини. Хешът свидетелства за това какво е казано; самоличността свидетелства за това кой го е казал.

Какво фразата не е

Три чести погрешни схващания трябва да бъдат изяснени. Фразата не е парола в истинския смисъл на думата: тя не се сравнява с отпечатък, съхранен на сървър; тя се въвежда в устройството на потребителя, за да се възстанови математически идентичността. Фразата не се възстановява: ако бъде загубена, няма от кого да я поискате; ако бъде дублирана, се дублира и идентичността. Фразата не е удостоверение, което може да се отдели от идентичността: фразата *е* идентичността. Който я притежава, може да действа като нея, без допълнително разрешение, без процес на оторизация, без възможност за възстановяване.

Това трето свойство е това, което променя тежестта на въпроса. Загубената парола е административно неудобство. Загубената криптографска идентичност е идентичността. Хартия с фразата, намерена от трети лица, не е риск от кражба на акаунт: това е предаване на цялата идентичност. Обещанието на системата — че никой не може да отнеме идентичността ви или да ви блокира произволно — е придружено неразривно от отговорността — че вие сте единственият пазител на нещо, което никой не може да възстанови вместо вас.

Обещанието и тежестта

Моделът на криптографска идентичност често получава определението *самосуверенна* —self-sovereign в англосаксонската литература—. Изборът на дума е съзнателен и описва състоянието доста точно. Потребителят е суверен на своята идентичност в почти средновековен смисъл: тя не се предоставя от никой крал, никой издател, никоя централна власт; нито някой от изброените може да я отнеме. Но също така, като средновековния монарх, потребителят носи цялата последица от своите грешки: няма регент, който да взема решения вместо него, ако загуби печата.

Изборът между идентичност, управлявана от трета страна, и самосуверенна идентичност няма универсален правилен отговор. За акаунт в маловажен форум управляваната идентичност вероятно е пропорционална на риска. За професионална идентичност, която подписва правно обвързващи документи, за икономическа идентичност, която съхранява собствени спестявания, за професионална комуникационна идентичност с клиенти, които са поверили чувствителна информация, въпросът се променя. Там въпросът престава да бъде „удобно ли е?“ и става „кой, освен мен, има силата да действа като мен и при какви обстоятелства?“.

Къде се появява този механизъм в реалните системи

BIP39 се роди в света на Bitcoin през 2013 г. и бързо се разпространи в цялата екосистема на криптовалутите: всеки сериозен портфейл днес приема фраза BIP39 от дванадесет или двадесет и четири думи като резервно копие на икономическата идентичност на своя притежател. Извън криптовалутите, същата основна концепция — криптографска двойка, която доказва авторство без посредник — се появява в други системи с различен синтаксис. SSH ключовете, които системният администратор използва за достъп до своите сървъри, са класически случай: частен ключ, който администраторът съхранява на своята машина, и публичен, който се копира на всеки сървър; не се намесва субект, сравним с централизирана услуга. Протоколът Signal използва Ed25519 с постоянен ключов материал на устройството; европейският eIDAS, в частта си за квалифициран подпис, се основава на същия криптографски принцип, с разликата, че ключът се съхранява от квалифициран доставчик на доверителни услуги вместо от потребителя.

Solo2, издателската платформа на тази публикация, използва фраза BIP39 от двадесет и четири думи като идентичност на всеки потребител. Потребителят вижда думите веднъж при създаването на своя акаунт. Те не се съхраняват на нито един сървър на Solo2 или на когото и да било друг: ако потребителят ги запише и съхранява, той запазва идентичността си завинаги. Ако ги загуби, ги губи. Това е логичното следствие от архитектурата без оператор в средата: ако Solo2 можеше да върне идентичността на потребителя, който я е загубил, тя щеше да може да я даде и на всеки, който окаже натиск върху Solo2 да я предостави.

За професионалния читател

Четири съображения за тези, които обмислят приемането на криптографска самосуверенна (autosoberana) идентичност в професионален контекст:

1. Фразата е идентичността. Физическото съхранение — хартия, няколко копия на различни места, евентуално гравиран метал за дългосрочна употреба — предлага повече гаранции от дигиталното съхранение, което увеличава повърхността за атака, без да намалява риска от загуба.
2. Няма възстановяване. Проектирането на процеса с предположението, че един ден основното копие ще бъде загубено, е много по-разумно, отколкото да го откриете в деня на загубата. Второ географски отделено копие решава почти всички сценарии.
3. Това не е същото като квалифициран сертификат eIDAS. За квалифициран подпис в Съюза — нотариални актове, определени процедури с Администрацията — законодателството изисква

квалифициран доставчик, който съхранява ключа. Криптографската самосуверенна идентичност служи за професионална комуникация и документално подписване с доказателствена стойност, но не заменя автоматично квалифицирания сертификат в случаите, когато нормата го изисква.

4. Ако идентичността предстои да бъде прехвърлена — наследство, професионално правоприемство, прекратяване на дейност — е препоръчително процедурата да се подготви предварително, а не след това. Формални процедури със запечатани с червен восък (lacre) пликове, инструкции към изпълнител на завещание, депозит в нотариална кантора, са класически договорености, напълно съвместими с криптографския характер на актива.

Тази статия затваря концептуалното трио, което откри цикъла — хеш, криптиране, идентичност. Трите идеи се изграждат една върху друга: хешът дава неизменния отпечатък, криптирането дава поверителността без доверена трета страна, идентичността дава авторството без предоставяща трета страна. Трите споделят свойство, което също не е идеологическо: те прехвърлят от този, който управлява услугата, към този, който я използва, технически способности, които традиционно са принадлежали на оператора. Заедно с тях се прехвърлят и отговорности. Честният разговор за всяка от трите изисква разговор и за другите две.

Източници и допълнително четиво

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, предложение за подобрение на Bitcoin от 2013 г. Де факто стандарт за фрази за възстановяване в крипто индустрията.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), включително Ed25519. IETF, януари 2017 г. Нормативна спецификация на схемата за подпис, използвана в голяма част от съвременната индустрия.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, версия 2.0. IETF, септември 2000 г. Определя алгоритъма PBKDF2, използван при извличането на BIP39 от фраза към seed.
- Регламент (ЕС) 910/2014 (eIDAS) и неговата еволюция чрез Регламент (ЕС) 2024/1183 (eIDAS 2) — европейска рамка за електронна идентичност и квалифициран подпис. Режим, различен от самосуверенния, но концептуално подкрепен от същите криптографски примитиви.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Каноничен текст за принципите и ангажиментите на самосуверенния модел, предходящ, но актуален за разбирането на фамилията от съвременни решения.

← ПредишнаБизнес моделът като сигнал за доверие Следваща →Self-hosting като професионална практика

Скорошни четива

- Размисъл · 29 юни 2026 г. Не сте анонимни
- Размисъл · 27 май 2026 г. Какво един подпис не може да поправи
- Анализ · 26 май 2026 г. Реална срещу привидна поверителност: въпросите, които е добре да си зададете

Вземете тази статия със себе си навсякъде, където ви е необходима.

↓ Markdown ↓ Обикновен текст ↓ PDF

Файлът ще бъде изтеглен на вашето устройство. Оттам можете да го запазите, импортирате в Solo2 или споделите където пожелаете. Cuadernos не решава дестинацията вместо вас.

Восъчен печат · SHA-256 977d3476c88f13a95d50bdac0b1bec6caaa12d6ddef7ce077c945ff7b2abeda3

Cuadernos Lacre · Публикация на Menzuri Gestión S.L. ·
написана от R.Eugenio · редактирана от екипа на Solo2.

Този уебсайт не използва бисквитки. Всичко, което зарежда вашият браузър, е написано или контролирано от нас и се съхранява на нашите европейски сървъри: анонимният брояч на посещения (Umami, самостоятелно хостван) и минималният необходим JavaScript за избора на език и вашата настройка за светла/тъмна тема, която се запазва на собственото ви устройство. Без ресурси от трети страни, без тракери, без профилиране, без споделяне на данни. Ако искате да ни последвате: RSS.