As 24 palabras: que é unha identidade criptográfica

Unha identidade criptográfica non é un contrasinal: non a garda ningún servidor e non se recupera. Unha explicación didáctica do mecanismo BIP39, por que exactamente vinte e catro palabras, e que peso real recae sobre quen as posúe.

Para entendernos: Se esqueces o teu contrasinal de Gmail, Google restabléceo por ti. Se perdes as 24 palabras que compoñen unha identidade criptográfica, non hai a quenllas pedir. Non é que o procedemento sexa estrito — é que non existe ninguén na outra punta. Esa diferenza é toda a diferenza.

A diferenza entre un contrasinal e unha identidade

Un contrasinal, no modelo clásico de internet, non é a identidade do usuario. É un comprobante. O usuario ten unha identidade —un nome, un correo electrónico, un número de cliente— e, para demostrar ante un servidor que é quen di ser, presenta un contrasinal que o servidor compara contra unha pegada que tiña almacenada. Se as pegadas coinciden, o servidor concede a sesión. Se o contrasinal se perde, o usuario segue sendo o mesmo usuario; o que perde é o comprobante, e existe un procedemento de recuperación —un correo ao enderezo rexistrado, unha pregunta de seguridade— para restituílo.

Unha identidade criptográfica funciona doutro xeito. Non é unha credencial que alguén compare contra unha pegada almacenada; *é* un segredo matemático completo en si mesmo. Dá igual onde resida —nun papel, nun dispositivo, incluso nun servidor alleo—: a identidade existe pola súa matemática, non por quen a valida. Aquí aparece unha propiedade parecida á que vimos en «Que é realmente SHA-256»: a posesión non se demostra exhibindo o segredo, senón usándoo para asinar. A firma así producida calquera pode comprobala cun valor público que se deriva matematicamente do propio segredo, sen necesidade de coñecer o segredo en si, e sen que un terceiro medie na comprobación. Quen ten o segredo, é a identidade; quen o perde, deixa de sela. A sentenza é categórica: **non hai ninguén a quen pedirlle que che devolva a identidade. Non existe ese alguén, porque non a tiña en primeiro lugar.**

O que representan vinte e catro palabras

A identidade criptográfica represéntase habitualmente cun segredo matemático de trinta e dous bytes — douscentos cincuenta e seis bits—. Un número difícil de reter e aínda máis difícil de transcribir sen erro. A industria criptográfica resolveu este problema en 2013 cun estándar pequeno e elegante chamado BIP39: unha forma de representar eses douscentos cincuenta e seis bits como unha secuencia de vinte e catro palabras tomadas dunha lista oficial de dúas mil corenta e oito. A aritmética detrás encaixa con elegancia; quen queira vela en detalle atópana na marxe.

A conta empeza polo final. Queremos representar os douscentos cincuenta e seis bits do segredo engadindo oito bits de suma de comprobación: douscentos sesenta e catro bits en total. Se os repartimos en vinte e catro palabras —un número manexable para anotar e ditar sen perda—, cada palabra ha de achegar exactamente once bits de información. E once bits son dous elevado a once posibilidades distintas, é dicir, dúas mil corenta e oito. Daí que o vocabulario oficial BIP39 teña precisamente ese tamaño: a lista existe a medida do problema, no revés.

A conta non é decorativa. Se alguén transcribe vinte e tres palabras correctamente e se equivoca na vinte e catro, a suma de comprobación detectarao: o software diralle «esta secuencia non é válida». Se alguén transcribe as vinte e catro correctas, o software derivará a mesma identidade sen ambigüidade. A elección da lista de palabras tamén é deliberada: as palabras do vocabulario BIP39 son curtas, distintas entre si, sen diacríticos, escollidas para minimizar confusións fonéticas e ortográficas. É un vocabulario deseñado para ser lembrado, escrito e ditado por seres humanos sen perda.

Da frase á clave

As vinte e catro palabras non son a clave criptográfica que asina mensaxes. Son unha representación recuperable da entropía orixinal que, mediante un proceso determinista chamado PBKDF2, transfórmase nunha semente de sesenta e catro bytes. Desa semente derivan, tamén de forma determinista, as claves criptográficas concretas que o usuario emprega: unha clave privada para asinar e unha clave pública correspondente que se publica para verificar as asinas. Mesmo mecanismo en sistemas distintos: as criptomoedas usan a curva secp256k1; o protocolo Signal e moitos sistemas modernos usan Ed25519 sobre a curva Curve25519. Para unha curva concreta como Ed25519, os estándares BIP32 e SLIP-0010 toman esa semente de sesenta e catro bytes e derivan, de forma determinista, os trinta e dous bytes que constitúen a clave de asina efectiva — os mesmos trinta e dous bytes cos que arranca o exemplo de código da próxima sección.

Esta é a forma estándar en que a industria enteira presenta o mecanismo ao usuario —moedeiros de criptomoedas, xestores de identidade descentralizada, Signal na súa parte de identidade persistente, Solo2 entre eles—: o usuario, na práctica, nunca ve a semente nin as claves derivadas. Ve as vinte e catro palabras ao crear a súa identidade e, opcionalmente, anótaas nun papel. As palabras viaxan logo entre os seus dispositivos cando quere migrar a identidade: introdúceas na aplicación nova, a aplicación deriva a mesma semente, as mesmas claves, a mesma identidade. É un mecanismo portable, criptograficamente sólido e, dentro dos límites do razoable, recordable.

Como se asina coa clave (pincelada Zig)

En Zig, unha vez se ten a semente de trinta e dous bytes derivada das vinte e catro palabras, asinar un mensaxe con Ed25519 cabe en poucas liñas:

```
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);
```

A operación de asinar produce sesenta e catro bytes —chamados asina— que só puideron xerarse a partir da clave privada correspondente. A verificación é pública: calquera coa clave pública pode comprobar que a asina corresponde ao mensaxe. Sen a clave privada, ninguén pode producir unha asina válida para ese mensaxe; coa clave pública, todos poden detectar se unha asina é válida. Esa asimetría é o que permite que o asinante demostre autoría sen compartir o segredo.

O exemplo anterior é a versión mínima de manual. No código real de Solo2 a cadea atravesa dous ficheiros, un en JavaScript que vive no navegador do usuario e reconstrúe a entropía a partir das vinte e catro palabras, outro

en Zig dentro da biblioteca *zcatcrypto* que toma esa entropia e deriva as claves criptográficas concretas. Comezando polo lado do navegador:

```
// 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 };
}
```

Eses trinta e dous bytes de entropía, xunto con outros trinta e dous derivados no mesmo paso, viaxan ao módulo WebAssembly de Zig que xera as claves Ed25519 propiamente ditas. A función completa, coa súa limpeza de memoria final, cabe nunha pantalla:

```
// 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;
}
```

Dous detalles vale a pena sinalar. O primeiro: unha mesma semente produce sempre o mesmo par de claves —é exactamente iso o que permite recuperar a identidade introducindo as vinte e catro palabras nun dispositivo novo. O segundo: a semente bórrase explicitamente da memoria na última liña. Pasado ese punto, nin sequera a propia función podería reconstruír as claves; as palabras do usuario serían a única orixe.

Para quen queira comprobalo con números pequenos. O esquema de sinatura pódese percorrer enteiro con cifras o bastante reducidas como para facer as contas a man. Quen prefira no entrar en aritmética pode saltarse este bloque sen perder o fío do artigo; quen queira ver o mecanismo funcionando paso a paso atoparao aquí. **As regras públicas**, que calquera pode ler: un primo $p = 23$ (en Ed25519 real é dunhas setenta e sete cifras; usamos vinte e tres para que as contas quepan nunha páxina), unha base $g = 2$ cuxa orde neste grupo é $q = 11$, e a convención de que toda a aritmética con g se fai *módulo p* e todos os expoñentes se reducen *módulo q*. **A elección privada**, unha soa e xamais compartida: o segredo $x = 6$. Esa é a identidade.

Paso 1 — A parte pública da identidade. Calcúlase unha vez e publícase abertamente.

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

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

A parte pública da identidade é **18**. Calquera pode tomala e usala para verificar sinaturas feitas con esta identidade. Ninguén, observando só o 18, pode recuperar o segredo 6: ese é o problema do logaritmo discreto ao que volveremos ao final.

Paso 2 — Signar unha mensaxe. O posuidor da identidade quere signar a mensaxe $m = 7$. Comeza elixindo un valor aleatorio novo $k = 4$, que se usará unha soa vez e non se compartirá xamais (en Ed25519 real, k derívase deterministicamente da mensaxe e do segredo para evitar o perigo de reutilizalo, pero o papel que xoga é exactamente este). Despois calcula tres números:

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

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

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

A sinatura é o par **(r, s) = (16, 10)**. Viaxa en aberto xunto á mensaxe. Calquera pode lela. Nota didáctica: en Ed25519 real a función H é SHA-512, criptograficamente robusta; aquí usamos a simplificación $e = (r + m) \bmod q$ para que o lector poida percorrer os pasos sen necesidade de calcular un hash. A estrutura do algoritmo é a mesma.

Paso 3 — Verificar a sinatura. O verificador ten a parte pública $y = 18$, a mensaxe $m = 7$, e a sinatura $(r, s) = (16, 10)$. Reconstrúe e da mesma forma — $e = (16 + 7) \bmod 11 = 1$ — e comproba se esta igualdade se cumpre:

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

Calcula os dous lados por separado:

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

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

Os dous lados dan **12**. A sinatura é válida. Calquera coa parte pública 18 pode chegar a esta conclusión sen ter sabido nunca que o segredo era 6.

E un terceiro que intentase falsificar? Eva viu pasar pola canle todo o público: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Para signar unha mensaxe *distinta* en nome desta identidade, necesitaría coñecer x. A súa única vía é preguntarse: «para que expoñente x se cumpre $2^x \bmod 23 = 18$?». Con $p = 23$ pode probar 0, 1, 2, 3, … e atopalo en segundos. Pero ao substituír 23 por un primo das dimensións reais de Ed25519, o espazo de expoñentes posibles supera o número de átomos do universo observable. **Non existe a día de hoxe ningún algoritmo coñecido pola humanidade que poida percorrer ese espazo en menos de miles de millóns de anos.** É o mesmo problema do logaritmo discreto que sustenta o Diffie-Hellman do artigo anterior, aplicado aquí ao esquema de sinatura.

Isto que acabamos de percorrer é *exactamente* Schnorr, o esquema de sinatura do que Ed25519 é unha variante adaptada a unha curva elíptica. En Ed25519 real, todas as operacións se fan sobre os puntos dunha curva concreta (Curve25519) en vez de sobre números enteiros módulo un primo, e a función H é SHA-512 en vez da suma de xoguete que usamos arriba. As dúas substitucións son axustes de implementación — gañar resistencia criptográfica á forza bruta, gañar propiedades adicionais de seguridade para k —. A estrutura algorítmica, as tres operacións, o porqué da asimetría, son as mesmas.

Convén aquí un alto breve, porque a cadea enteira pode confundirse dunha ollada rápida con outra primitiva do trío: o hash. Non o é. Un hash é unha función única que comprime — entran moitos bytes, sae unha pegada curta, aí acaba o camiño —. Unha identidade criptográfica é unha parella matemática complementaria: o segredo queda e signa; a súa contraparte pública publícase e verifícase. Onde o hash colapsa información nun sentido único, a identidade establece unha asimetría entre dúas metades. O hash testemuña que se dixo; a identidade testemuña quen o dixo.

O que a frase non é

Tres equivocacións frecuentes convén despexar. A frase non é un contrasinal en sentido propio: no se compara contra unha pegada almacenada nun servidor; introdúcese no dispositivo do usuario para reconstruír matematicamente a identidade. A frase non se recupera: se se perde, non hai ninguén a quen pedila; se se duplica, duplícase tamén a identidade. A frase non é unha credencial separable da identidade: a frase *é* a identidade. Quen a ten pode actuar como ela, sen permiso adicional, sen proceso de autorización, sen recuperación posible.

Esta terceira propiedade é a que cambia o peso do asunto. Un contrasinal perdido é unha molestia administrativa. Unha identidade criptográfica perdida é a identidade. Un papel coa frase atopado por terceiros non é un risco de roubo de conta: é a entrega da identidade enteira. A promesa do sistema —que ninguén poida revocarche a túa identidade nin bloquearte arbitrariamente— vén acompañada inseparablemente da responsabilidade — que ti es o único custodio de algo que ninguén pode restituír por ti.

A promesa e o peso

O modelo de identidade criptográfica adoita recibir o cualificativo de *autosoberana* —self-sovereign na literatura anglosaxoa—. A elección de palabra é deliberada e describe con bastante exactitude a condición. O usuario é soberano sobre a súa identidade nun sentido case medieval: no a concede ningún rei, ningún emisor, ningunha autoridade central; tampouco a poden retirar ningún dos anteriores. Pero tamén, como o monarca medieval, o usuario carga coa consecuencia enteira dos seus erros: no hai rexente que tome decisións no seu lugar se perde o selo.

A elección entre identidade xestionada por un terceiro e identidade autosoberana non ten unha resposta universal correcta. Para a conta dun foro irrelevante, a identidade xestionada é probablemente proporcional ao risco. Para unha identidade profesional que asina documentos xuridicamente vinculantes, para unha identidade económica que custodia aforros propios, para unha identidade de comunicación profesional con clientes que confiaron

información sensible, a cuestión cambia. Alí a pregunta deixa de ser «é cómodo?» e vólvese «quen, ademais de eu, ten o poder de actuar como eu, e baixo que circunstancias?».

Onde aparece este mecanismo en sistemas reais

O BIP39 naceu no mundo de Bitcoin en 2013 e estendeuse rapidamente a todo o ecosistema de criptomonedas: calquera moedeiro serio acepta hoxe unha frase BIP39 de doce ou vinte e catro palabras como respaldo da identidade económica do seu posuidor. Fóra das criptomonedas, o mesmo concepto subxacente —par criptográfico que proba a autoría sen intermediario— aparece noutros sistemas con sintaxe distinta. As claves SSH que un administrador de sistemas usa para acceder aos seus servidores son un caso clásico: unha clave privada que o administrador garda na súa máquina e unha pública que se copia en cada servidor; ninguén comparable a un servizo centralizado intervén. O protocolo Signal usa Ed25519 con material de clave persistente no dispositivo; as eIDAS europeas, na súa parte de sinatura cualificada, descansan sobre o mesmo principio criptográfico, coa diferenza de que a clave a custodia un provedor de servizos de confianza cualificado en lugar do usuario.

Solo2, plataforma editora desta publicación, usa unha frase BIP39 de vinte e catro palabras como identidade de cada usuario. O usuario, ao crear a súa conta, ve as palabras unha vez. Non se almacenan en ningún servidor de Solo2 nin de ninguén: si o usuario as anota e as custodia, mantén a súa identidade para sempre. Si as perde, pérdeas. É a consecuencia coherente coa arquitectura de non haber operador no medio: si Solo2 puidera devolverlle a identidade ao usuario que a perdeu, podería tamén dalla a quen presione a Solo2 para que lla dea.

Para o lector profesional

Catro consideracións para quen avalía adoptar identidade criptográfica autosoberana (autosoberana) nun contexto profesional:

1. A frase é a identidade. Custodia física —papel, varias copias en lugares distintos, eventualmente metal gravado para uso de longo prazo— ofrece máis garantías que a custodia dixital, que engade superficie de ataque sen reducir o risco de perda.
2. Non hai recuperación. Deseñar o proceso asumindo que un día se perde a copia primaria convén moito máis que descubrilo o día que se perde. Unha segunda copia xeograficamente separada resolve case todos os escenarios.
3. Non é o mesmo que un certificado cualificado eIDAS. Para sinatura cualificada na Unión —escrituras notariais, certos trámites coa Administración— a lexislación esixe un provedor cualificado que custodia a clave. A identidade criptográfica autosoberana serve para a comunicación profesional e a sinatura documental con valor probatorio, pero non substitúe automaticamente ao certificado cualificado nos casos onde a norma o esixe.
4. Si a identidade vai transferirse —herdanza, sucesión profesional, peche de actividade— convén preparar o procedemento antes, non despois. Procedementos formais con sobres selados con lacre (lacre), instrucións a un albacea, depósito en notaría, son arranxos clásicos perfectamente compatibles coa natureza criptográfica do activo.

Este artigo pecha o trío conceptual que abriu o ciclo —hash, cifrado, identidade—. As tres ideas constrúense unhas sobre as outras: o hash dá a pegada inalterable, o cifrado dá a confidencialidade sen terceiro de confianza, a identidade dá a autoría sen terceiro de concesión. As tres comparten unha propiedade que tampoc é ideolóxica: trasladan, de quen xestiona un servizo a quen o usa, capacidades técnicas que tradicionalmente residían no operador. Trasladan con elas tamén responsabilidades. Falar con honestidade de calquera das tres esixe falar tamén das outras duás.

Fontes e lectura adicional

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, proposta de mellora de Bitcoin de 2013. Estándar de facto para frases de recuperación na industria criptográfica.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), incluíndo Ed25519. IETF, xaneiro de 2017. Especificación normativa do esquema de sinatura usado en boa parte da industria contemporánea.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, versión 2.0. IETF, setembro de 2000. Define o algoritmo PBKDF2 usado na derivación BIP39 de frase a semente (seed).
- Regulamento (UE) 910/2014 (eIDAS) e a súa evolución polo Regulamento (UE) 2024/1183 (eIDAS 2) — marco europeo de identidade electrónica e sinatura cualificada. Réxime distinto do autosoberano, pero conceptualmente apoiado nos mesmos primitivos criptográficos.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Texto canónico sobre os principios e compromisos do modelo autosoberano, anterior pero relevante para a comprensión da familia de solucións contemporáneas.

← AnteriorO modelo de negocio como sinal de confianzaSeguinte →Self-hosting como práctica profesional

Lecturas recentes

- Reflexión · 29 de xuño de 2026 Non es anónimo
- Reflexión · 27 de maio de 2026 O que unha firma non pode arranxar
- Análise · 26 de maio de 2026 Privacidade real vs aparente: as preguntas que convén facerse

Leva este artigo onde o necesites.

↓ Markdown ↓ Texto plano ↓ PDF

O arquivo descárgase no teu dispositivo. Desde aí podes gardalo, importalo a Solo2, o compartilo onde queiras. Cuadernos no decide o destino por ti.

Selo de lacre · SHA-256 4f72fdb0fb9f1ddb2cfc1508c33fe8bc35531e24f022da48c355bbdb1de3ec91

Cuadernos Lacre · Unha publicación de Menzuri Gestión S.L. ·
escrita por R.Eugenio · editada polo equipo de Solo2.