Les 24 paraules: què és una identitat criptogràfica

Una identitat criptogràfica no és una contrasenya: no la guarda cap servidor i no es recupera. Una explicació didàctica del mecanisme BIP39, per què exactament vint-i-quatre paraules, i quin pes real recau sobre qui les posseeix.

Per entendre'ns: Si oblides la teva contrasenya de Gmail, Google te la restableix. Si perds les 24 paraules que componen una identitat criptogràfica, no hi ha a qui demanar-les. No és que el procediment sigui estricte — és que no existeix ningú a l'altra punta. Aquesta diferència és tota la diferència.

La diferència entre una contrasenya i una identitat

Una contrasenya, en el model clàssic d'internet, no és la identitat de l'usuari. És un comprovant. L'usuari té una identitat —un nom, un correu electrònic, un número de client— i, per demostrar davant d'un servidor que és qui diu ser, presenta una contrasenya que el servidor compara contra una empremta que tenia emmagatzemada. Si les empremtes coincideixen, el servidor concedeix la sessió. Si la contrasenya es perd, l'usuari continua sent el mismo usuari; el que perd és el comprovant, i existeix un procediment de recuperació —un correu a l'adreça registrada, una pregunta de seguretat— per restituir-lo.

Una identitat criptogràfica funciona d'una altra manera. No és una credencial que algú compari contra una empremta emmagatzemada; *és* un secret matemàtic complet en si mateix. Tant és on resideixi —en un paper, en un dispositiu, fins i tot en un servidor aliè—: la identitat existeix per la seva matemàtica, no per qui la valida. Aquí apareix una propietat semblant a la que vam veure a «Què és realment SHA-256»: la possessió no es demostra exhibint el secret, sinó usant-lo per signar. La signatura així produïda qualsevol pot comprovar-la amb un valor públic que es deriva matemàticament del propi secret, sense necessitat de conèixer el secret en si, i sense que un tercer mediï en la comprovació. Qui té el secret, és la identitat; qui el perd, deixa de ser-la. La sentència és categòrica: **no hi ha ningú a qui demanar-li que et torni la identitat. No existeix aquest algú, perquè no la tenia en primer lloc.**

El que representen vint-i-quatre paraules

La identitat criptogràfica es representa habitualment amb un secret matemàtic de trenta-dos bytes —dos-cents cinquanta-sis bits—. Un número difícil de retenir i encara més difícil de transcriure sense error. La indústria criptogràfica va resoldre aquest problema el 2013 amb un estàndard petit i elegant anomenat BIP39: una forma de representar aquests dos-cents cinquanta-sis bits com una seqüència de vint-i-quatre paraules agafades d'una llista oficial de dues mil quaranta-vuit. L'aritmètica de darrere encaixa amb elegància; qui vulgui veure-la en detall la troba al marge.

El compte comença pel final. Volem representar els dos-cents cinquanta-sis bits del secret afegint vuit bits de suma de comprovació: dos-cents seixanta-quatre bits en total. Si els repartim en vint-i-quatre paraules —un número manejable per anotar i dictar sense pèrdua—, cada paraula ha d'aportar exactament onze bits d'informació. I onze bits són dos elevat a onze possibilitats distintes, és a dir, dues mil quaranta-vuit. D'aquí que el vocabulari oficial BIP39 tingui precisament aquesta mida: la llista existeix a mida del problema, no al revés.

El compte no és decoratiu. Si algú transcriu vint-i-tres paraules correctament i s'equivoca en la vint-i-quatrena, la suma de comprovació ho detectarà: el programari li dirà «aquesta seqüència no és vàlida». Si algú transcriu les vint-i-quatre correctes, el programari derivarà la mateixa identitat sense ambigüitat. L'elecció de la llista de paraules també és deliberada: les paraules del vocabulari BIP39 són curtes, distintes entre si, sense diacrítics, escollides per minimitzar confusions fonètiques i ortogràfiques. És un vocabulari dissenyat per ser recordat, escrit i dictat per éssers humans sense pèrdua.

De la frase a la clau

Les vint-i-quatre paraules no són la clau criptogràfica que signa missatges. Són una representació recuperable de l'entropia original que, mitjançant un procés determinista anomenat PBKDF2, es transforma en una llavor de seixanta-quatre bytes. D'aquesta llavor deriven, també de forma determinista, les claus criptogràfiques concretes que l'usuari empra: una clau privada per signar i una clau pública corresponent que es publica per verificar les signatures. Mateix mecanisme en sistemes diferents: les criptomonedes usen la corba secp256k1; el protocol Signal i molts sistemes moderns usen Ed25519 sobre la corba Curve25519. Per a una corba concreta com Ed25519, els estàndards BIP32 i SLIP-0010 prenen aquesta llavor de seixanta-quatre bytes i deriven, de forma determinista, els trenta-dos bytes que constitueixen la clau de signatura efectiva — els mateixos trenta-dos bytes amb què arrenca l'exemple de codi de la secció següent.

Aquesta és la forma estàndard en què la indústria sencera presenta el mecanisme a l'usuari —moneders de criptomonedes, gestors d'identitat descentralitzada, Signal en la seva part d'identitat persistent, Solo2 entre ells —: l'usuari, en la pràctica, mai veu la llavor ni les claus derivades. Veu les vint-i-quatre paraules en crear la seva identitat i, opcionalment, les anota en un paper. Les paraules viatgen després entre els seus dispositius quan vol migrar la identitat: les introdueix en l'aplicació nova, l'aplicació deriva la mateixa llavor, les mateixes claus, la mateixa identitat. És un mecanisme portable, criptogràficament sòlid i, dins dels límits del que és raonable, recordable.

Com es signa amb la clau (pinzellada Zig)

En Zig, un cop es té la llavor de trenta-dos bytes derivada de les vint-i-quatre paraules, signar un missatge amb Ed25519 cap en poques línies:

```
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);
```

L'operació de signar produeix seixanta-quatre bytes —anomenats signatura— que només es van poder generar a partir de la clau privada corresponent. La verificació és pública: qualsevol amb la clau pública pot comprovar que la signatura correspon al missatge. Sense la clau privada, ningú pot produir una signatura vàlida per a aquest missatge; amb la clau pública, tothom pot detectar si una signatura és vàlida. Aquesta asimetria és la que permet que el signant demostri autoria sense compartir el secret.

L'exemple anterior és la versió mínima de manual. En el codi real de Solo2 la cadena travessa dos fitxers, un en JavaScript que viu al navegador de l'usuari i reconstrueix l'entropia a partir de les vint-i-quatre paraules, un altre

en Zig dins la biblioteca *zcatcrypto* que pren aquesta entropia i deriva les claus criptogràfiques concretes. Començant pel costat del 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 };
}
```

Aquests trenta-dos bytes d'entropia, juntament amb uns altres trenta-dos derivats en el mateix pas, viatgen al mòdul WebAssembly de Zig que genera les claus Ed25519 pròpiament dites. La funció completa, amb la seva neteja de memòria final, cap en una 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;
}
```

Dos detalls val la pena assenyalar. El primer: una mateixa llavor produeix sempre el mateix parell de claus —és exactament això el que permet recuperar la identitat introduint les vint-i-quatre paraules en un dispositiu nou. El segon: la llavor s'esborra explícitament de la memòria en l'última línia. Passat aquest punt, ni tan sols la pròpia funció podria reconstruir les claus; les paraules de l'usuari serien l'únic origen.

Per a qui vulgui comprovar-ho amb números petits. L'esquema de signatura es pot recórrer sencer amb xifres prou reduïdes com per fer els comptes a mà. Qui prefereixi no entrar en aritmètica pot saltar-se aquest bloc sense perdre el fil de l'article; qui vulgui veure el mecanisme funcionant pas a pas ho trobarà aquí. **Les regles públiques**, que qualsevol pot llegir: un primer $p = 23$ (en Ed25519 real és d'unes setanta-set xifres; en fem servir vint-i-tres perquè els comptes caben en una pàgina), una base $g = 2$ l'ordre de la qual en aquest grup és $q = 11$, i la convenció que tota l'aritmètica amb g es fa *módulo p* i tots els exponents es redueixen *módulo q*. **L'elecció privada**, una de sola i mai compartida: el secret $x = 6$. Aquesta és la identitat.

Pas 1 — La part pública de la identitat. Es calcula una vegada i es publica obertament.

$y = g^x \bmod p$

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

La part pública de la identitat és **18**. Qualsevol pot prendre-la i usar-la per verificar signatures fetes amb aquesta identitat. Ningú, observant només el 18, pot recuperar el secret 6: aquest és el problema del logaritme discret al qual tornarem al final.

Pas 2 — Signar un missatge. El posseïdor de la identitat vol signar el missatge $m = 7$. Comença triant un valor aleatori nou $k = 4$, que es farà servir una sola vegada i no es compartirà mai (en Ed25519 real, k es deriva determinísticament del missatge i del secret per evitar el perill de reutilitzar-lo, però el paper que juga és exactament aquest). Després calcula tres números:

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

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

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

La signatura és el parell **(r, s) = (16, 10)**. Viatja en obert juntament amb el missatge. Qualsevol pot llegir-la. Nota didàctica: en Ed25519 real la funció H és SHA-512, criptogràficament robusta; aquí fem servir la simplificació $e = (r + m) \bmod q$ perquè el lector pugui recórrer els passos sense necessitat de calcular un hash. L'estructura de l'algoritme és la mateixa.

Pas 3 — Verificar la signatura. El verificador té la part pública $y = 18$, el missatge $m = 7$, i la signatura $(r, s) = (16, 10)$. Reconstrueix e de la mateixa forma —$e = (16 + 7) \bmod 11 = 1$— i comprova si aquesta igualtat es compleix:

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

Calcula els dos costats per separat:

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

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

Els dos costats donen **12**. La signatura és vàlida. Qualsevol amb la part pública 18 pot arribar a aquesta conclusió sense haver sabut mai que el secret era 6.

I un tercer que intentés falsificar? L'Eva ha vist passar pel canal tot el que és públic: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Per signar un missatge *diferent* en nom d'aquesta identitat, necessitaria conèixer x. La seva única via és preguntar-se: «per a quin exponent x es compleix $2^x \bmod 23 = 18$?». Amb $p = 23$ pot provar 0, 1, 2, 3, … i trobar-lo en segons. Però en substituir 23 per un primer de les dimensions reals d'Ed25519, l'espai d'exponents possibles supera el nombre d'àtoms de l'univers observable. **No existeix avui dia cap algoritme conegut per la humanitat que pugui recórrer aquest espai en menys de milers de milions d'anys.** És el mateix problema del logaritme discret que sustenta el Diffie-Hellman de l'article anterior, aplicat aquí a l'esquema de signatura.

Això que acabem de recórrer és *exactament* Schnorr, l'esquema de signatura del qual Ed25519 és una variant adaptada a una corba el·líptica. En Ed25519 real, totes les operacions es fan sobre els punts d'una corba concreta (Curve25519) en comptes de sobre números enters mòdul un primer, i la funció H és SHA-512 en comptes de la suma de joguina que hem fet servir a dalt. Les dues substitucions són ajustos d'implementació —guanyar resistència criptogràfica a la força bruta, guanyar propietats addicionals de seguretat per a k—. L'estructura algorítmica, les tres operacions, el perquè de l'asimetria, són les mateixes.

Convé aquí un alt breu, perquè la cadena sencera pot confondre's d'una ullada ràpida amb una altra primitiva del trio: el hash. No ho és. Un hash és una funció única que comprimeix —hi entren molts bytes, en surt una empremta curta, aquí acaba el camí—. Una identitat criptogràfica és una parella matemàtica complementària: el secret es queda i signa; la seva contrapart pública es publica i verifica. On el hash col·lapsa informació en un sentit únic, la identitat estableix una asimetria entre dues meitats. El hash testifica què s'ha dit; la identitat testifica qui ho ha dit.

El que la frase no és

Convé aclarir tres equivocacions freqüents. La frase no és una contrasenya en sentit propi: no es compara contra una empremta emmagatzemada en un servidor; s'introdueix en el dispositiu de l'usuari per reconstruir matemàticament la identitat. La frase no es recupera: si es perd, no hi ha ningú a qui demanar-la; si es duplica, es duplica també la identitat. La frase no és una credencial separable de la identitat: la frase *és* la identitat. Qui la té pot actuar com ella, sense permís addicional, sense procés d'autorització, sense recuperació possible.

Aquesta tercera propietat és la que canvia el pes de l'assumpte. Una contrasenya perduda és una molèstia administrativa. Una identitat criptogràfica perduda és la identitat. Un paper amb la frase trobat per tercers no és un risc de robatori de compte: és el lliurament de la identitat sencera. La promesa del sistema —que ningú pugui revocar-te la teva identitat ni bloquejar-te arbitràriament— ve acompanyada inseparablement de la responsabilitat — que tu ets l'únic custodi d'una cosa que ningú pot restituir per tu.

La promesa i el pes

El model d'identitat criptogràfica sol rebre el qualificatiu d'*autosoberana* —self-sovereign en la literatura anglosaxona—. L'elecció de paraula és deliberada i descriu amb força exactitud la condició. L'usuari és sobirà sobre la seva identitat en un sentit gairebé medieval: no la concedeix cap rei, cap emissor, cap autoritat central; tampoc la pot retirar cap dels anteriors. Però també, com el monarca medieval, l'usuari carrega amb la conseqüència sencera dels seus errors: no hi ha regent que prengui decisions en el seu lloc si perd el segell.

L'elecció entre identitat gestionada per un tercer i identitat autosoberana no té una resposta universal correcta. Per al compte d'un fòrum irrellevant, la identitat gestionada és probablement proporcional al risc. Per a una identitat professional que signa documents jurídicament vinculants, per a una identitat econòmica que custodia estalvis propis, per a una identitat de comunicació professional amb clients que han confiat informació sensible,

la qüestió canvia. Allà la pregunta deixa de ser «és còmode?» i es torna «qui, a més de jo, té el poder d'actuar com jo, i sota quines circumstàncies?».

On apareix aquest mecanisme en sistemes reals

El BIP39 va néixer al món de Bitcoin el 2013 i es va estendre ràpidament a tot l'ecosistema de criptomonedes: qualsevol moneder seriós accepta avui una frase BIP39 de dotze o vint-i-quatre paraules com a suport de la identitat econòmica del seu posseïdor. Fora de les criptomonedes, el mateix concepte subjacent —parell criptogràfic que prova l'autoria sense intermediari— apareix en altres sistemes amb sintaxi diferent. Les claus SSH que un administrador de sistemes usa per accedir als seus servidors són un cas clàssic: una clau privada que l'administrador guarda a la seva màquina i una pública que es copia a cada servidor; ningú comparable a un servei centralitzat hi intervé. El protocol Signal usa Ed25519 amb material de clau persistent al dispositiu; les eIDAS europees, en la seva part de signatura qualificada, descansen sobre el mateix principi criptogràfic, amb la diferència que la clau la custodia un proveïdor de serveis de confiança qualificat en lloc de l'usuari.

Solo2, plataforma editora d'aquesta publicació, usa una frase BIP39 de vint-i-quatre paraules com a identitat de cada usuari. L'usuari, en crear el seu compte, veu les paraules una vegada. No s'emmagatzemen en cap servidor de Solo2 ni de ningú: si l'usuari les anota i les custodia, manté la seva identitat per sempre. Si les perd, les perd. És la conseqüència coherent amb l'arquitectura de no haver-hi operador pel mig: si Solo2 pogués retornar-li la identitat a l'usuari que la va perdre, podria també donar-li a qui pressioni Solo2 perquè li doni.

Per al lector professional

Quatre consideracions per a qui avalua adoptar identitat criptogràfica autosoberana (autosoberana) en un context professional:

1. La frase és la identitat. Custòdia física —paper, diverses còpies en llocs diferents, eventualment metall gravat per a ús de llarg termini— ofereix més garanties que la custòdia digital, que afegeix superfície d'atac sense reduir el risc de pèrdua.
2. No hi ha recuperació. Dissenyar el procés assumint que un dia es perd la còpia primària convé molt més que descobrir-ho el dia que es perd. Una segona còpia geogràficament separada resol gairebé tots els escenaris.
3. No és el mateix que un certificat qualificat eIDAS. Per a signatura qualificada a la Unió —escriptures notarials, certs tràmits amb l'Administració— la legislació exigeix un proveïdor qualificat que custodia la clau. La identitat criptogràfica autosoberana serveix per a la comunicació professional i la signatura documental amb valor probatori, però no substitueix automàticament el certificat qualificat en els casos on la norma l'exigeix.
4. Si la identitat s'ha de transferir —herència, successió professional, tancament d'activitat— convé preparar el procediment abans, no després. Procediments formals amb sobres lacrats amb lacre (lacre), instruccions a un marmessor, dipòsit en notaria, són arranjaments clàssics perfectament compatibles amb la natura criptogràfica de l'actiu.

Aquest article tanca el trio conceptual que va obrir el cicle —hash, xifrat, identitat—. Les tres idees es construeixen unes sobre les altres: el hash dóna l'empremta inalterable, el xifrat dóna la confidencialitat sense tercer de confiança, la identitat dóna l'autoria sense tercer de concessió. Les tres comparteixen una propietat que tampoc és ideològica: traslladen, de qui gestiona un servei a qui l'usa, capacitats tècniques que tradicionalment residien en l'operador. Traslladen amb elles també responsabilitats. Parlar amb honestedat de qualsevol de les tres exigeix parlar també de les altres dues.

Fonts i lectura addicional

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, proposta de millora de Bitcoin de 2013. Estàndard de facto per a frases de recuperació en la indústria criptogràfica.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), incloent Ed25519. IETF, gener de 2017. Especificació normativa de l'esquema de signatura usat en bona part de la indústria contemporània.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, versió 2.0. IETF, setembre de 2000. Defineix l'algoritme PBKDF2 usat en la derivació BIP39 de frase a llavor.
- Reglament (UE) 910/2014 (eIDAS) i la seva evolució pel Reglament (UE) 2024/1183 (eIDAS 2) — marc europeu d'identitat electrònica i signatura qualificada. Règim diferent de l'autosoberà, però conceptualment recolzat en els mateixos primitius criptogràfics.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Text canònic sobre els principis i compromisos del model autosoberà, anterior però rellevant per a la comprensió de la família de solucions contemporànies.

← AnteriorEl model de negoci com a senyal de confiançaSegüent →Self-hosting com a pràctica professional

Lectures recents

- Reflexió · 29 de juny del 2026 No ets anònim
- Reflexió · 27 de maig del 2026 El que una signatura no pot arreglar
- Anàlisi · 26 de maig del 2026 Privadesa real vs aparent: les preguntes que convé fer-se

Emporta't aquest article on el necessitis.

↓ Markdown ↓ Text pla ↓ PDF

L'arxiu es descarrega al teu dispositiu. Des d'allà pots guardar-lo, importar-lo a Solo2, o compartir-lo on vulguis. Cuadernos no decideix el destí per tu.

Segell de lacre · SHA-256 b2f58e871dfeb3a0654b8c4bcb48f34d5c7d615cd14bde8cd633893072ea7d37

Cuadernos Lacre · Una publicació de Menzuri Gestión S.L. ·
escrita per R.Eugenio · editada per l'equip de Solo2.