# Les 24 mots : qu'est-ce qu'une identité cryptographique

Une identité cryptographique n'est pas un mot de passe : aucun serveur ne la conserve et elle ne se récupère pas. Une explication didactique du mécanisme BIP39, pourquoi exactement vingt-quatre mots, et quel poids réel repose sur celui qui les possède.

**Pour nous comprendre :** Si vous oubliez votre mot de passe Gmail, Google vous le réinitialise. Si vous perdez les 24 mots qui composent une identité cryptographique, il n'y a personne à qui les demander. Ce n'est pas que la procédure soit stricte — c'est qu'il n'existe personne à l'autre bout. Cette différence est toute la différence.

## La différence entre un mot de passe et une identité

Un mot de passe, dans le modèle classique d'internet, n'est pas l'identité de l'utilisateur. C'est un justificatif. L'utilisateur a une identité —un nom, un e-mail, un numéro de client— et, pour prouver devant un serveur qu'il est bien celui qu'il prétend être, présente un mot de passe que le serveur compare à une empreinte qu'il avait mémorisée. Si les empreintes coïncident, le serveur accorde la session. Si le mot de passe est perdu, l'utilisateur reste le même utilisateur ; ce qu'il perd, c'est le justificatif, et il existe une procédure de récupération —un courriel à l'adresse enregistrée, une question de sécurité— pour le restituer.

Une identité cryptographique fonctionne autrement. Ce n'est pas un justificatif que quelqu'un compare à une empreinte mémorisée ; *c'est* un secret mathématique complet en soi. Peu importe où il réside —sur un papier, dans un dispositif, même sur un serveur tiers— : l'identité existe par sa mathématique, non par celui qui la valide. Ici apparaît une propriété semblable à celle que nous avons vue dans « Qu'est-ce que réellement SHA-256 » : la possession ne se démontre pas en exhibant le secret, mais en l'utilisant pour signer. La signature ainsi produite peut être vérifiée par n'importe qui à l'aide d'une valeur publique qui dérive mathématiquement du secret lui-même, sans avoir besoin de connaître le secret en soi, et sans qu'un tiers n'intervienne dans la vérification. Celui qui détient le secret, est l'identité ; celui qui le perd, cesse de l'être. La sentence est catégorique : **il n'y a personne à qui demander de vous rendre l'identité. Cet quelqu'un n'existe pas, car il ne la possédait pas en premier lieu.**

## Ce que représentent vingt-quatre mots

L'identité cryptographique est habituellement représentée par un secret mathématique de trente-deux octets — deux cent cinquante-six bits. Un nombre difficile à retenir et encore plus difficile à transcrire sans erreur. L'industrie cryptographique a résolu ce problème en 2013 avec un standard petit et élégant appelé BIP39 : une façon de représenter ces deux cent cinquante-six bits comme une séquence de vingt-quatre mots tirés d'une liste officielle de deux mille quarante-huit. L'arithmétique sous-jacente s'emboîte avec élégance ; celui qui veut la voir en détail la trouve en marge.

Le compte commence par la fin. Nous voulons représenter les deux cent cinquante-six bits du secret en ajoutant huit bits de somme de contrôle (checksum) : deux cent soixante-quatre bits au total. Si nous les répartissons en vingt-quatre mots —un nombre gérable pour noter et dicter sans perte—, chaque mot doit apporter exactement

onze bits d'information. Et onze bits font deux puissance onze possibilités, c'est-à-dire deux mille quarante-huit. D'où le fait que le vocabulaire officiel BIP39 ait précisément cette taille : la liste existe sur mesure pour le problème, pas l'inverse.

Le compte n'est pas décoratif. Si quelqu'un transcrit vingt-trois mots correctement et se trompe au vingt-quatrième, la somme de contrôle le détectera : le logiciel lui dira « cette séquence n'est pas valide ». Si quelqu'un transcrit les vingt-quatre correctement, le logiciel dérivera la même identité sans ambiguïté. Le choix de la liste de mots est également délibéré : les mots du vocabulaire BIP39 sont courts, distincts les uns des autres, sans diacritiques, choisis pour minimiser les confusions phonétiques et orthographiques. C'est un vocabulaire conçu pour être mémorisé, écrit et dicté par des êtres humains sans perte.

# De la phrase à la clé

Les vingt-quatre mots ne sont pas la clé cryptographique qui signe les messages. Ils sont une représentation récupérable de l'entropie originale qui, par un processus déterministe appelé PBKDF2, est transformée en une graine (seed) de soixante-quatre octets. De cette graine dérivent, également de manière déterministe, les clés cryptographiques concrètes que l'utilisateur emploie : une clé privée pour signer et une clé publique correspondante qui est publiée pour vérifier les signatures. Même mécanisme dans différents systèmes : les crypto-monnaies utilisent la courbe secp256k1 ; le protocole Signal et de nombreux systèmes modernes utilisent Ed25519 sur la courbe Curve25519. Pour une courbe concrète comme Ed25519, les standards BIP32 et SLIP-0010 prennent cette graine de soixante-quatre octets et dérivent, de manière déterministe, les trente-deux octets qui constituent la clé de signature effective — les mêmes trente-deux octets avec lesquels commence l'exemple de code de la section suivante.

C'est la manière standard dont toute l'industrie présente le mécanisme à l'utilisateur —portefeuilles de crypto-monnaies, gestionnaires d'identité décentralisée, Signal dans sa partie d'identité persistante, Solo2 parmi eux— : l'utilisateur, en pratique, ne voit jamais la graine ni les clés dérivées. Il voit les vingt-quatre mots lors de la création de son identité et, facultativement, les note sur un papier. Les mots voyagent ensuite entre ses appareils lorsqu'il souhaite migrer l'identité : il les saisit dans la nouvelle application, l'application dérive la même graine, les mêmes clés, la même identité. C'est un mécanisme portable, cryptographiquement solide et, dans les limites du raisonnable, mémorisable.

# Comment signer avec la clé (un coup de pinceau Zig)

En Zig, une fois que l'on a la graine de trente-deux octets dérivée des vingt-quatre mots, signer un message avec Ed25519 tient en quelques lignes :

```
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'opération de signature produit soixante-quatre octets —appelés signature— qui n'ont pu être générés qu'à partir de la clé privée correspondante. La vérification est publique : n'importe qui possédant la clé publique peut vérifier que la signature correspond au message. Sans la clé privée, personne ne peut produire une signature

valide pour ce message ; avec la clé publique, tout le monde peut détecter si une signature est valide. Cette asymétrie est ce qui permet au signataire de prouver son auteur sans partager le secret.

L'exemple précédent est la version minimale du manuel. Dans le code réel de Solo2, la chaîne traverse deux fichiers : l'un en JavaScript qui vit dans le navigateur de l'utilisateur et reconstruit l'entropie à partir des vingt-quatre mots, l'autre en Zig au sein de la bibliothèque *zcatcrypto* qui prend cette entropie et en dérive les clés cryptographiques concrètes. En commençant par le côté navigateur :

```
// 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 };
}
```

Ces trente-deux octets d'entropie, ainsi que trente-deux autres dérivés au cours de la même étape, voyagent vers le module WebAssembly de Zig qui génère les clés Ed25519 proprement dites. La fonction complète, avec son nettoyage final de la mémoire, tient sur un seul écran :

```
// 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;
}
```

Deux détails méritent d'être signalés. Le premier : une même graine (seed) produit toujours la même paire de clés — c'est précisément ce qui permet de récupérer l'identité en saisissant les vingt-quatre mots sur un nouvel appareil. Le second : la graine est explicitement effacée de la mémoire à la dernière ligne. Passé ce point, même la fonction elle-même ne pourrait pas reconstruire les clés ; les mots de l'utilisateur seraient la seule source.

**Pour ceux qui veulent vérifier avec de petits nombres.** Le schéma de signature peut être parcouru en entier avec des chiffres suffisamment réduits pour faire les calculs à la main. Ceux qui préfèrent ne pas entrer dans l'arithmétique peuvent sauter ce bloc sans perdre le fil de l'article ; ceux qui veulent voir le mécanisme fonctionner étape par étape le trouveront ici. **Les règles publiques**, que tout le monde peut lire : un nombre premier $p = 23$ (dans le Ed25519 réel, il comporte environ soixante-dix-sept chiffres ; nous utilisons vingt-trois pour que les calculs tiennent sur une page), une base $g = 2$ dont l'ordre dans ce groupe est $q = 11$, et la convention selon laquelle toute l'arithmétique avec $g$ se fait *módulo p* et tous les exposants sont réduits *módulo q*. **Le choix privé**, unique et jamais partagé : le secret $x = 6$. C'est cela, l'identité.

**Étape 1 — La partie publique de l'identité.** Elle est calculée une fois et publiée ouvertement.

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

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

La partie publique de l'identité est **18**. N'importe qui peut la prendre et l'utiliser pour vérifier des signatures faites avec cette identité. Personne, en observant seulement le 18, ne peut récupérer le secret 6 : c'est le problème du logarithme discret sur lequel nous reviendrons à la fin.

**Étape 2 — Signer un message.** Le détenteur de l'identité veut signer le message $m = 7$. Il commence par choisir une nouvelle valeur aléatoire $k = 4$, qui ne sera utilisée qu'une seule fois et ne sera jamais partagée (dans le Ed25519 réel, $k$ est dérivé de manière déterministe du message et du secret pour éviter le danger de réutilisation, mais le rôle qu'il joue est exactement celui-ci). Il calcule ensuite trois nombres :

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

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

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

La signature est la paire **(r, s) = (16, 10)**. Elle voyage en clair avec le message. Tout le monde peut la lire. Note didactique : dans le Ed25519 réel, la fonction $H$ est SHA-512, cryptographiquement robuste ; ici, nous utilisons la simplification $e = (r + m) \bmod q$ pour que le lecteur puisse suivre les étapes sans avoir à calculer de hash. La structure de l'algorithme est la même.

**Étape 3 — Vérifier la signature.** Le vérificateur dispose de la partie publique $y = 18$, du message $m = 7$ et de la signature $(r, s) = (16, 10)$. Il reconstruit $e$ de la même manière — $e = (16 + 7) \bmod 11 = 1$ — et vérifie si cette égalité est respectée :

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

Calcule les deux côtés séparément :

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

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

Les deux côtés donnent **12**. La signature est valide. Quiconque possède la partie publique 18 peut arriver à cette conclusion sans avoir jamais su que le secret était 6.

**Et un tiers qui tenterait de falsifier ?** Eva a vu passer par le canal tout ce qui est public : $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Pour signer un message *différent* au nom de cette identité, elle aurait besoin de connaître $x$. Sa seule voie est de se demander : « pour quel exposant $x$ a-t-on $2^x \bmod 23 = 18$ ? ». Avec $p = 23$, elle peut essayer 0, 1, 2, 3, … et le trouver en quelques secondes. Mais en remplaçant 23 par un nombre premier aux dimensions réelles de Ed25519, l'espace des exposants possibles dépasse le nombre d'atomes de l'univers observable. **Il n'existe à ce jour aucun algorithme connu de l'humanité capable de parcourir cet espace en moins de milliards d'années.** C'est le même problème du logarithme discret qui sous-tend le Diffie-Hellman de l'article précédent, appliqué ici au schéma de signature.

Ce que nous venons de parcourir est *exactement* Schnorr, le schéma de signature dont Ed25519 est une variante adaptée à une courbe elliptique. Dans le Ed25519 réel, toutes les opérations se font sur les points d'une courbe concrète (Curve25519) au lieu de nombres entiers modulo un nombre premier, et la fonction $H$ est SHA-512 au lieu de la somme de pacotille que nous avons utilisée plus haut. Les deux substitutions sont des ajustements d'implémentation — gagner en résistance cryptographique à la force brute, gagner des propriétés de sécurité supplémentaires pour $k$. La structure algorithmique, les trois opérations, la raison de l'asymétrie, sont les mêmes.

Une brève pause s'impose ici, car la chaîne entière peut être confondue d'un coup d'œil rapide avec une autre primitive du trio : le hash. Ce n'en est pas un. Un hash est une fonction unique qui compresse — beaucoup d'octets entrent, une empreinte courte sort, le chemin s'arrête là. Une identité cryptographique est un couple mathématique complémentaire : le secret reste et signe ; sa contrepartie publique est publiée et vérifie. Là où le hash condense l'information dans un sens unique, l'identité établit une asymétrie entre deux moitiés. Le hash atteste de ce qui a été dit ; l'identité atteste de qui l'a dit.

# Ce que la phrase n'est pas

Il convient de dissiper trois idées reçues fréquentes. La phrase n'est pas un mot de passe au sens propre : elle n'est pas comparée à une empreinte stockée sur un serveur ; elle est saisie sur l'appareil de l'utilisateur pour reconstruire mathématiquement l'identité. La phrase ne se récupère pas : si elle est perdue, il n'y a personne à qui la demander ; si elle est dupliquée, l'identité est également dupliquée. La phrase n'est pas un identifiant séparable de l'identité : la phrase *est* l'identité. Quiconque la possède peut agir en son nom, sans permission supplémentaire, sans processus d'autorisation, sans récupération possible.

C'est cette troisième propriété qui change le poids de l'affaire. Un mot de passe perdu est un désagrément administratif. Une identité cryptographique perdue est l'identité elle-même. Un papier avec la phrase trouvé par des tiers n'est pas un risque de vol de compte : c'est la remise de l'identité entière. La promesse du système — que personne ne puisse révoquer votre identité ni vous bloquer arbitrairement — s'accompagne inséparablement de la responsabilité — que vous êtes le seul dépositaire de quelque chose que personne ne peut restituer pour vous.

# La promesse et le poids

Le modèle d'identité cryptographique reçoit souvent le qualificatif d'*auto-souveraine* —self-sovereign dans la littérature anglo-saxonne—. Le choix du mot est délibéré et décrit assez précisément la condition. L'utilisateur est souverain sur son identité dans un sens presque médiéval : elle n'est concédée par aucun roi, aucun émetteur, aucune autorité centrale ; elle ne peut pas non plus être retirée par l'un des précédents. Mais aussi, comme le

monarque médiéval, l'utilisateur assume l'entière conséquence de ses erreurs : il n'y a pas de régent pour prendre des décisions à sa place s'il perd le sceau.

Le choix entre une identité gérée par un tiers et une identité auto-souveraine n'a pas de réponse universelle correcte. Pour le compte d'un forum sans importance, l'identité gérée est probablement proportionnelle au risque. Pour une identité professionnelle qui signe des documents juridiquement contraignants, pour une identité économique qui garde ses propres économies, pour une identité de communication professionnelle avec des clients qui ont confié des informations sensibles, la question change. Là, la question n'est plus « est-ce pratique ? » mais devient « qui, à part moi, a le pouvoir d'agir en mon nom, et dans quelles circonstances ? ».

## Où ce mécanisme apparaît dans les systèmes réels

Le BIP39 est né dans le monde de Bitcoin en 2013 et s’est rapidement étendu à tout l’écosystème des cryptomonnaies : tout portefeuille sérieux accepte aujourd’hui une phrase BIP39 de douze ou vingt-quatre mots comme sauvegarde de l’identité économique de son détenteur. En dehors des cryptomonnaies, le même concept sous-jacent — paire cryptographique prouvant l’auteur sans intermédiaire — apparaît dans d’autres systèmes avec une syntaxe différente. Les clés SSH qu’un administrateur système utilise pour accéder à ses serveurs sont un cas classique : une clé privée que l’administrateur garde sur sa machine et une publique qui est copiée sur chaque serveur ; aucun service centralisé n’intervient. Le protocole Signal utilise Ed25519 avec un matériel de clé persistant sur l’appareil ; l’eIDAS européen, dans sa partie signature qualifiée, repose sur le même principe cryptographique, avec la différence que la clé est gardée par un prestataire de services de confiance qualifié au lieu de l’utilisateur.

Solo2, plateforme éditrice de cette publication, utilise une phrase BIP39 de vingt-quatre mots comme identité pour chaque utilisateur. L’utilisateur, lors de la création de son compte, voit les mots une seule fois. Ils ne sont stockés sur aucun serveur de Solo2 ni de quiconque : si l’utilisateur les note et les garde, il conserve son identité pour toujours. S’il les perd, il les perd. C’est la conséquence cohérente d’une architecture sans opérateur intermédiaire : si Solo2 pouvait rendre son identité à l’utilisateur qui l’a perdue, elle pourrait aussi la donner à quiconque ferait pression sur Solo2 pour l’obtenir.

## Pour le lecteur professionnel

Quatre considérations pour celui qui envisage d’adopter une identité cryptographique autosouveraine (autosoberana) dans un contexte professionnel :

1. La phrase est l’identité. La garde physique — papier, plusieurs copies dans des lieux différents, éventuellement métal gravé pour un usage à long terme — offre plus de garanties que la garde numérique, qui augmente la surface d’attaque sans réduire le risque de perte.
2. Il n’y a pas de récupération. Concevoir le processus en supposant qu’un jour la copie primaire sera perdue vaut bien mieux que de le découvrir le jour où on la perd. Une seconde copie géographiquement séparée résout presque tous les scénarios.
3. Ce n’est pas la même chose qu’un certificat qualifié eIDAS. Pour la signature qualifiée dans l’Union — actes notariés, certaines démarches administratives — la législation impose un prestataire qualifié qui garde la clé. L’identité cryptographique autosouveraine sert à la communication professionnelle et à la signature documentaire avec valeur probante, mais ne remplace pas automatiquement le certificat qualifié dans les cas où la norme l’exige.
4. Si l’identité doit être transférée — héritage, succession professionnelle, cessation d’activité — il convient de préparer la procédure avant, pas après. Des procédures formelles avec enveloppes scellées à la cire (lacre), des instructions à un exécuteur testamentaire, un dépôt chez un notaire, sont des arrangements classiques parfaitement compatibles avec la nature cryptographique de l’actif.

---

*Cet article clôt le trio conceptuel qui a ouvert le cycle — hash, chiffrement, identité —. Les trois idées se construisent les unes sur les autres : le hash donne l'empreinte inaltérable, le chiffrement donne la confidentialité sans tiers de confiance, l'identité donne l'auteur sans tiers de concession. Les trois partagent une propriété qui n'est pas non plus idéologique : elles transfèrent, du gestionnaire d'un service à celui qui l'utilise, des capacités techniques qui résidaient traditionnellement chez l'opérateur. Elles transfèrent avec elles également des responsabilités. Parler avec honnêteté de l'une des trois exige de parler également des deux autres.*

## Sources et lectures complémentaires


- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, Bitcoin improvement proposal de 2013. Standard de fait pour les phrases de récupération dans l'industrie cryptographique.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), incluant Ed25519. IETF, janvier 2017. Spécification normative du schéma de signature utilisé dans une grande partie de l'industrie contemporaine.
- RFC 2898 — PKCS #5 : Password-Based Cryptography Specification, version 2.0. IETF, septembre 2000. Définit l'algorithme PBKDF2 utilisé dans la dérivation BIP39 de phrase à graine (seed).
- Règlement (UE) n° 910/2014 (eIDAS) et son évolution par le Règlement (UE) 2024/1183 (eIDAS 2) — cadre européen d'identité électronique et de signature qualifiée. Régime différent de l'autosouverain, mais conceptuellement appuyé sur les mêmes primitives cryptographiques.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Texte canonique sur les principes et les engagements du modèle autosouverain, antérieur mais pertinent pour la compréhension de la famille de solutions contemporaines.



← PrécédentLe modèle économique comme signal de confianceSuivant →Self-hosting comme pratique professionnelle

## Lectures recentes

- Réflexion · 29 juin 2026 Vous n'êtes pas anonyme
- Réflexion · 27 mai 2026 Ce qu'une signature ne peut pas réparer
- Analyse · 26 mai 2026 Confidentialité réelle vs apparente : les questions à se poser

Emportez cet article où vous en avez besoin.

↓ Markdown ↓ Texte brut ↓ PDF

Le fichier est téléchargé sur votre appareil. À partir de là, vous pouvez l'enregistrer, l'importer dans Solo2 ou le partager comme vous le souhaitez. Cuadernos ne décide pas de la destination pour vous.



Cachet de cire · SHA-256 081ecbd14d51270c38f6a307d3dff147e813d768ff8316faae5a56492b588ae3