Konzept · 23. Mai 2026

← Zurück·Weiter →

Die 24 Wörter: Was eine kryptografische Identität ist

Eine kryptografische Identität ist kein Passwort: Kein Server speichert sie und sie kann nicht wiederhergestellt werden. Eine didaktische Erklärung des BIP39-Mechanismus, warum es genau vierundzwanzig Wörter sind und welche tatsächliche Last auf demjenigen liegt, der sie besitzt.

Um uns zu verstehen: Wenn Sie Ihr Gmail-Passwort vergessen, setzt Google es für Sie zurück. Wenn Sie die 24 Wörter verlieren, aus denen eine kryptografische Identität besteht, gibt es niemanden, den Sie darum bitten könnten. Es ist nicht so, dass das Verfahren streng wäre – es gibt einfach niemanden am anderen Ende. Dieser Unterschied ist der entscheidende Punkt.

Der Unterschied zwischen einem Passwort und einer Identität

Ein Passwort im klassischen Internetmodell ist nicht die Identität des Nutzers. Es ist ein Beleg. Der Nutzer hat eine Identität – einen Namen, eine E-Mail-Adresse, eine Kundennummer – und um gegenüber einem Server zu beweisen, dass er derjenige ist, für den er sich ausgibt, legt er ein Passwort vor, das der Server mit einem gespeicherten Fingerabdruck vergleicht. Wenn die Fingerabdrücke übereinstimmen, gewährt der Server die Sitzung. Wenn das Passwort verloren geht, bleibt der Nutzer derselbe Nutzer; was er verliert, ist der Beleg, und es gibt ein Wiederherstellungsverfahren – eine E-Mail an die registrierte Adresse, eine Sicherheitsfrage –, um ihn zu ersetzen.

Eine kryptografische Identität funktioniert anders. Sie ist kein Berechtigungsnachweis, den jemand mit einem gespeicherten Fingerabdruck vergleicht; sie *ist* ein vollständiges mathematisches Geheimnis für sich. Es ist egal, wo sie sich befindet – auf einem Papier, in einem Gerät, sogar auf einem fremden Server: Die Identität existiert durch ihre Mathematik, nicht durch denjenigen, der sie validiert. Hier taucht eine Eigenschaft auf, die derjenigen ähnelt, die wir in «Was SHA-256 wirklich ist» gesehen haben: Der Besitz wird nicht durch Vorzeigen des Geheimnisses bewiesen, sondern indem man es zum Signieren verwendet. Die so erzeugte Signatur kann jeder mit einem öffentlichen Wert überprüfen, der mathematisch aus dem Geheimnis selbst abgeleitet wird, ohne das Geheimnis kennen zu müssen und ohne dass ein Dritter bei der Überprüfung vermittelt. Wer das Geheimnis hat, ist die Identität; wer es verliert, hört auf, sie zu sein. Das Urteil ist kategorisch: **Es gibt niemanden, den man bitten könnte, einem die Identität zurückzugeben. Diese Person existiert nicht, weil sie die Identität gar nicht erst besaß.**

Was vierundzwanzig Wörter repräsentieren

Die kryptografische Identität wird üblicherweise durch ein mathematisches Geheimnis von zweiunddreißig Byte – zweihundertsechsundfünfzig Bit – dargestellt. Eine Zahl, die schwer zu behalten und noch schwerer fehlerfrei zu transkribieren ist. Die Kryptobranche löste dieses Problem 2013 mit einem kleinen und eleganten Standard namens BIP39: eine Möglichkeit, diese zweihundertsechsundfünfzig Bit als eine Folge von vierundzwanzig Wörtern darzustellen, die aus einer offiziellen Liste von zweitausendachtundvierzig Wörtern stammen. Die dahinterstehende Arithmetik fügt sich elegant zusammen; wer sie im Detail sehen möchte, findet sie am Rand.

Die Rechnung beginnt am Ende. Wir wollen die zweihundertsechsundfünfzig Bit des Geheimnisses darstellen und fügen acht Bit Prüfsumme hinzu: insgesamt zweihundertvierundsechzig Bit. Wenn wir diese auf vierundzwanzig Wörter verteilen – eine handhabbare Anzahl, um sie ohne Verlust zu notieren und zu diktieren –, muss jedes Wort genau elf Bit Information beisteuern. Und elf Bit sind zwei hoch elf Möglichkeiten, also zweitausendachtundvierzig. Daher hat der offizielle BIP39-Wortschatz genau diese Größe: Die Liste existiert maßgeschneidert für das Problem, nicht umgekehrt.

Die Rechnung ist nicht dekorativ. Wenn jemand dreiundzwanzig Wörter korrekt transkribiert und sich beim vierundzwanzigsten irrt, wird die Prüfsumme dies erkennen: Die Software wird ihm sagen: „Diese Sequenz ist ungültig“. Wenn jemand alle vierundzwanzig korrekt transkribiert, wird die Software eindeutig dieselbe Identität ableiten. Auch die Wahl der Wortliste ist wohlüberlegt: Die Wörter des BIP39-Vokabulars sind kurz, voneinander verschieden, ohne Diakritika und so gewählt, dass phonetische und orthografische Verwechslungen minimiert werden. Es ist ein Vokabular, das darauf ausgelegt ist, von Menschen verlustfrei erinnert, geschrieben und diktiert zu werden.

Vom Satz zum Schlüssel

Die vierundzwanzig Wörter sind nicht der kryptografische Schlüssel, der Nachrichten signiert. Sie sind eine wiederherstellbare Darstellung der ursprünglichen Entropie, die durch einen deterministischen Prozess namens PBKDF2 in einen vierundsechzig Byte langen Seed umgewandelt wird. Von diesem Seed leiten sich, ebenfalls deterministisch, die konkreten kryptografischen Schlüssel ab, die der Benutzer verwendet: ein privater Schlüssel zum Signieren und ein entsprechender öffentlicher Schlüssel, der zur Überprüfung der Signaturen veröffentlicht wird. Derselbe Mechanismus in verschiedenen Systemen: Kryptowährungen verwenden die secp256k1-Kurve; das Signal-Protokoll und viele moderne Systeme verwenden Ed25519 auf der Curve25519-Kurve. Für eine konkrete Kurve wie Ed25519 nehmen die Standards BIP32 und SLIP-0010 diesen vierundsechzig Byte langen Seed und leiten deterministisch die zweiunddreißig Byte ab, die den effektiven Signierschlüssel bilden — dieselben zweiunddreißig Byte, mit denen das Codebeispiel im nächsten Abschnitt beginnt.

Dies ist die Standardart und -weise, wie die gesamte Branche dem Benutzer den Mechanismus präsentiert — Kryptowährungs-Wallets, dezentrale Identitätsmanager, Signal in seinem persistenten Identitätsteil, Solo2 darunter—: Der Benutzer sieht in der Praxis niemals den Seed oder die abgeleiteten Schlüssel. Er sieht die vierundzwanzig Wörter bei der Erstellung seiner Identität und notiert sie sich optional auf einem Blatt Papier. Die Wörter reisen dann zwischen seinen Geräten, wenn er die Identität migrieren möchte: Er gibt sie in die neue Anwendung ein, die Anwendung leitet denselben Seed, dieselben Schlüssel, dieselbe Identität ab. Es ist ein portabler, kryptografisch solider und, in vernünftigem Rahmen, merkbarer Mechanismus.

Wie man mit dem Schlüssel signiert (ein Zig-Pinselstrich)

In Zig passt das Signieren einer Nachricht mit Ed25519 in wenige Zeilen, sobald man den von den vierundzwanzig Wörtern abgeleiteten zweiunddreißig Byte langen Seed hat:

```
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);
```

Der Signiervorgang erzeugt vierundsechzig Byte —Signatur genannt—, die nur aus dem entsprechenden privaten Schlüssel generiert werden konnten. Die Verifizierung ist öffentlich: Jeder mit dem öffentlichen Schlüssel kann überprüfen, ob die Signatur der Nachricht entspricht. Ohne den privaten Schlüssel kann niemand eine gültige Signatur für diese Nachricht erstellen; mit dem öffentlichen Schlüssel kann jeder erkennen, ob eine Signatur gültig ist. Diese Asymmetrie ist es, die es dem Unterzeichner ermöglicht, die Urheberschaft nachzuweisen, ohne das Geheimnis preiszugeben.

Das vorangegangene Beispiel ist die Minimalversion aus dem Handbuch. Im echten Solo2-Code durchläuft die Kette zwei Dateien: eine in JavaScript, die im Browser des Benutzers ausgeführt wird und die Entropie aus den vierundzwanzig Wörtern rekonstruiert, und eine andere in Zig innerhalb der *zcatcrypto*-Bibliothek, die diese Entropie übernimmt und die konkreten kryptografischen Schlüssel ableitet. Beginnend auf der Browserseite:

```
// 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 };
}
```

Diese zweiunddreißig Bytes Entropie reisen zusammen mit weiteren zweiunddreißig im selben Schritt abgeleiteten Bytes zum WebAssembly-Modul von Zig, das die eigentlichen Ed25519-Schlüssel generiert. Die vollständige Funktion mit ihrer abschließenden Speicherbereinigung passt auf einen Bildschirm:

```
// 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;
}
```

Zwei Details sind bemerkenswert. Erstens: Derselbe Seed erzeugt immer dasselbe Schlüsselpaar – genau das ermöglicht die Wiederherstellung der Identität durch Eingabe der vierundzwanzig Wörter auf einem neuen Gerät. Zweitens: Der Seed wird in der letzten Zeile explizit aus dem Speicher gelöscht. Ab diesem Punkt könnte nicht einmal die Funktion selbst die Schlüssel rekonstruieren; die Wörter des Benutzers wären die einzige Quelle.

Für diejenigen, die es mit kleinen Zahlen überprüfen wollen. Das Signaturschema kann vollständig mit Zahlen durchlaufen werden, die klein genug sind, um die Rechnungen von Hand durchzuführen. Wer lieber nicht in die Arithmetik einsteigen möchte, kann diesen Block überspringen, ohne den Faden des Artikels zu verlieren; wer den Mechanismus Schritt für Schritt in Aktion sehen möchte, findet ihn hier. **Die öffentlichen Regeln**, die jeder lesen kann: eine Primzahl $p = 23$ (im echten Ed25519 hat sie etwa siebenundsiebzig Stellen; wir verwenden dreiundzwanzig, damit die Rechnungen auf eine Seite passen), eine Basis $g = 2$, deren Ordnung in dieser Gruppe $q = 11$ ist, und die Konvention, dass die gesamte Arithmetik mit *g módulo p* erfolgt und alle Exponenten *módulo q* reduziert werden. **Die private Wahl**, eine einzige und niemals geteilte: das Geheimnis $x = 6$. Das ist die Identität.

Schritt 1 – Der öffentliche Teil der Identität. Er wird einmal berechnet und offen veröffentlicht.

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

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

Der öffentliche Teil der Identität ist **18**. Jeder kann ihn nehmen und verwenden, um Signaturen zu überprüfen, die mit dieser Identität erstellt wurden. Niemand kann allein durch Beobachtung der 18 das Geheimnis 6 wiederherstellen: Das ist das Problem des diskreten Logarithmus, auf das wir am Ende zurückkommen werden.

Schritt 2 – Eine Nachricht signieren. Der Inhaber der Identität möchte die Nachricht $m = 7$ signieren. Er beginnt mit der Wahl eines neuen Zufallswerts $k = 4$, der nur einmal verwendet und niemals geteilt wird (im echten Ed25519 wird *k* deterministisch aus der Nachricht und dem Geheimnis abgeleitet, um die Gefahr der Wiederverwendung zu vermeiden, aber die Rolle, die er spielt, ist genau diese). Dann berechnet er drei Zahlen:

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

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

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

Die Signatur ist das Paar **(r, s) = (16, 10)**. Sie reist offen zusammen mit der Nachricht. Jeder kann sie lesen. Didaktischer Hinweis: Im echten Ed25519 ist die Funktion *H* SHA-512, kryptografisch robust; hier verwenden wir die Vereinfachung $e = (r + m) \bmod q$, damit der Leser die Schritte nachvollziehen kann, ohne einen Hash berechnen zu müssen. Die Struktur des Algorithmus ist die gleiche.

Schritt 3 – Die Signatur überprüfen. Der Verifizierer verfügt über den öffentlichen Teil $y = 18$, die Nachricht $m = 7$ und die Signatur $(r, s) = (16, 10)$. Er rekonstruiert e auf die gleiche Weise – $e = (16 + 7) \bmod 11 = 1$ – und prüft, ob diese Gleichheit erfüllt ist:

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

Berechnet die beiden Seiten getrennt:

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

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

Beide Seiten ergeben **12**. Die Signatur ist gültig. Jeder mit dem öffentlichen Teil 18 kann zu diesem Schluss kommen, ohne jemals gewusst zu haben, dass das Geheimnis 6 war.

Und ein Dritter, der versuchen würde zu fälschen? Eva hat alles Öffentliche durch den Kanal gehen sehen: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Um eine *andere* Nachricht im Namen dieser Identität zu signieren, müsste sie x kennen. Ihr einziger Weg ist die Frage: „Für welchen Exponenten x gilt $2^x \bmod 23 = 18$?“. Bei $p = 23$ kann sie 0, 1, 2, 3, … ausprobieren und es in Sekunden finden. Aber wenn man 23 durch eine Primzahl in den realen Dimensionen von Ed25519 ersetzt, übersteigt der Raum der möglichen Exponenten die Anzahl der Atome im beobachtbaren Universum. **Es gibt bis heute keinen der Menschheit bekannten Algorithmus, der diesen Raum in weniger als Milliarden von Jahren durchlaufen könnte.** Es ist das gleiche Problem des diskreten Logarithmus, das dem Diffie-Hellman des vorherigen Artikels zugrunde liegt, hier angewendet auf das Signaturschema.

Was wir gerade durchlaufen haben, ist *exakt* Schnorr, das Signaturschema, von dem Ed25519 eine an eine elliptische Kurve angepasste Variante ist. Im echten Ed25519 werden alle Operationen auf Punkten einer konkreten Kurve (Curve25519) statt auf ganzen Zahlen modulo einer Primzahl durchgeführt, und die Funktion H ist SHA-512 anstelle der oben verwendeten Spielzeugsumme. Die beiden Ersetzungen sind Anpassungen der Implementierung – Gewinnung kryptografischer Widerstandsfähigkeit gegen Brute-Force, Gewinnung zusätzlicher Sicherheitseigenschaften für k. Die algorithmische Struktur, die drei Operationen und der Grund für die Asymmetrie sind dieselben.

An dieser Stelle ist ein kurzer Halt angebracht, da die gesamte Kette auf den ersten Blick mit einer anderen Primitve des Trios verwechselt werden kann: dem Hash. Das ist sie nicht. Ein Hash ist eine einzigartige Funktion, die komprimiert – viele Bytes gehen hinein, ein kurzer Fingerabdruck kommt heraus, dort endet der Weg. Eine kryptografische Identität ist ein komplementäres mathematisches Paar: Das Geheimnis bleibt und signiert; sein öffentliches Gegenstück wird veröffentlicht und verifiziert. Wo der Hash Informationen in eine einzige Richtung kollabiert, stellt die Identität eine Asymmetrie zwischen zwei Hälften her. Der Hash bezeugt, was gesagt wurde; die Identität bezeugt, wer es gesagt hat.

Was der Satz nicht ist

Drei häufige Missverständnisse sollten ausgeräumt werden. Der Satz ist kein Passwort im eigentlichen Sinne: Er wird nicht mit einem auf einem Server gespeicherten Fingerabdruck verglichen; er wird in das Gerät des Benutzers eingegeben, um die Identität mathematisch zu rekonstruieren. Der Satz kann nicht wiederhergestellt werden: Wenn er verloren geht, gibt es niemanden, den man danach fragen kann; wenn er dupliziert wird, wird auch die Identität dupliziert. Der Satz ist kein von der Identität trennbarer Berechtigungsnachweis: Der Satz *ist* die Identität. Wer ihn hat, kann als diese handeln, ohne zusätzliche Erlaubnis, ohne Autorisierungsprozess, ohne mögliche Wiederherstellung.

Diese dritte Eigenschaft ist es, die das Gewicht der Angelegenheit verändert. Ein verlorenes Passwort ist ein administratives Ärgernis. Eine verlorene kryptografische Identität ist die Identität. Ein von Dritten gefundenes Papier mit dem Satz ist kein Risiko für einen Kontodiebstahl: Es ist die Übergabe der gesamten Identität. Das

Versprechen des Systems — dass niemand einem die Identität entziehen oder einen willkürlich blockieren kann — geht untrennbar mit der Verantwortung einher — dass man der einzige Hüter von etwas ist, das niemand für einen wiederherstellen kann.

Das Versprechen und das Gewicht

Das Modell der kryptografischen Identität wird oft als *selbstsouverän* bezeichnet —self-sovereign in der englischsprachigen Literatur—. Die Wortwahl ist bewusst und beschreibt den Zustand recht genau. Der Benutzer ist souverän über seine Identität in einem fast mittelalterlichen Sinne: Sie wird von keinem König, keinem Emittenten, keiner zentralen Behörde verliehen; sie kann auch von keinem der Genannten entzogen werden. Aber auch der Benutzer trägt, wie der mittelalterliche Monarch, die gesamte Konsequenz seiner Fehler: Es gibt keinen Regenten, der an seiner Stelle Entscheidungen trifft, wenn er das Siegel verliert.

Die Wahl zwischen einer von einem Dritten verwalteten Identität und einer selbstsouveränen Identität hat keine universell richtige Antwort. Für ein irrelevantes Forenkonto ist die verwaltete Identität wahrscheinlich proportional zum Risiko. Für eine berufliche Identität, die rechtlich bindende Dokumente unterzeichnet, für eine wirtschaftliche Identität, die eigene Ersparnisse verwaltet, für eine berufliche Kommunikationsidentität mit Kunden, die sensible Informationen anvertraut haben, ändert sich die Frage. Dort hört die Frage auf, «ist es bequem?» zu sein, und wird zu «wer außer mir hat die Macht, als ich zu handeln, und unter welchen Umständen?».

Wo dieser Mechanismus in realen Systemen auftaucht

BIP39 entstand 2013 in der Welt von Bitcoin und verbreitete sich schnell im gesamten Kryptowährungs-Ökosystem: Jedes seriöse Wallet akzeptiert heute eine BIP39-Phrase aus zwölf oder vierundzwanzig Wörtern als Backup für die wirtschaftliche Identität seines Inhabers. Außerhalb von Kryptowährungen taucht das gleiche zugrunde liegende Konzept — ein kryptografisches Paar, das die Urheberschaft ohne Zwischenhändler beweist — in anderen Systemen mit unterschiedlicher Syntax auf. SSH-Schlüssel, die ein Systemadministrator für den Zugriff auf seine Server verwendet, sind ein klassischer Fall: ein privater Schlüssel, den der Administrator auf seinem Rechner aufbewahrt, und ein öffentlicher Schlüssel, der auf jeden Server kopiert wird; keine Instanz, die mit einem zentralisierten Dienst vergleichbar wäre, greift ein. Das Signal-Protokoll verwendet Ed25519 mit persistentem Schlüsselmaterial auf dem Gerät; das europäische eIDAS stützt sich im Teil der qualifizierten Signatur auf das gleiche kryptografische Prinzip, mit dem Unterschied, dass der Schlüssel von einem qualifizierten Vertrauensdiensteanbieter anstelle des Benutzers verwaltet wird.

Solo2, die Verlagsplattform dieser Publikation, verwendet eine BIP39-Phrase aus vierundzwanzig Wörtern als Identität für jeden Benutzer. Der Benutzer sieht die Wörter bei der Erstellung seines Kontos einmal. Sie werden auf keinem Server von Solo2 oder anderen gespeichert: Wenn der Benutzer sie notiert und verwahrt, behält er seine Identität für immer. Wenn er sie verliert, verliert er sie. Dies ist die konsequente Folge einer Architektur ohne zwischengeschalteten Betreiber: Wenn Solo2 die Identität an den Benutzer zurückgeben könnte, der sie verloren hat, könnte sie sie auch jedem geben, der Solo2 unter Druck setzt, sie herauszugeben.

Für den professionellen Leser

Vier Überlegungen für diejenigen, die die Einführung einer kryptografischen autosouveränen Identität im beruflichen Kontext prüfen:

1. Die Phrase ist die Identität. Physische Verwahrung — Papier, mehrere Kopien an verschiedenen Orten, eventuell graviertes Metall für den Langzeitgebrauch — bietet mehr Garantien als die digitale Verwahrung, die die Angriffsfläche vergrößert, ohne das Verlustrisiko zu verringern.
2. Es gibt keine Wiederherstellung. Den Prozess unter der Annahme zu gestalten, dass die Primärkopie eines Tages verloren geht, ist viel ratsamer, als es am Tag des Verlusts festzustellen. Eine zweite, geografisch

getrennte Kopie löst fast alle Szenarien.

3. Es ist nicht dasselbe wie ein qualifiziertes eIDAS-Zertifikat. Für qualifizierte Signaturen in der Union — notarielle Urkunden, bestimmte Behördengänge — schreibt die Gesetzgebung einen qualifizierten Anbieter vor, der den Schlüssel verwahrt. Die kryptografische autosouveräne Identität dient der beruflichen Kommunikation und der dokumentarischen Unterzeichnung mit Beweiswert, ersetzt aber nicht automatisch das qualifizierte Zertifikat in Fällen, in denen die Norm es erfordert.
4. Wenn die Identität übertragen werden soll — Erbschaft, berufliche Nachfolge, Geschäftsaufgabe —, sollte das Verfahren vorher und nicht nachher vorbereitet werden. Formelle Verfahren mit Siegellack (lacre) versiegelten Umschlägen, Anweisungen an einen Testamentsvollstrecker, Hinterlegung beim Notar sind klassische Vereinbarungen, die perfekt mit der kryptografischen Natur des Vermögenswertes vereinbar sind.

Dieser Artikel schließt das konzeptionelle Trio ab, das den Zyklus eröffnet hat — Hash, Verschlüsselung, Identität —. Die drei Ideen bauen aufeinander auf: Der Hash liefert den unveränderlichen Fingerabdruck, die Verschlüsselung liefert die Vertraulichkeit ohne vertrauenswürdigen Dritten, die Identität liefert die Urheberschaft ohne konzessionierenden Dritten. Alle drei teilen eine Eigenschaft, die ebenfalls nicht ideológico ist: Sie übertragen technische Fähigkeiten, die traditionell beim Betreiber lagen, vom Dienstverwalter auf den Benutzer. Mit ihnen werden auch Verantwortlichkeiten übertragen. Um ehrlich über eines der drei zu sprechen, muss man auch über die anderen beiden sprechen.

Quellen und weiterführende Literatur

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, Bitcoin-Verbesserungsvorschlag von 2013. De-facto-Standard für Wiederherstellungsphrasen in der Kryptoindustrie.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), einschließlich Ed25519. IETF, Januar 2017. Normative Spezifikation des Signaturschemas, das in weiten Teilen der zeitgenössischen Industrie verwendet wird.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, Version 2.0. IETF, September 2000. Definiert den PBKDF2-Algorithmus, der bei der BIP39-Ableitung von Phrase zu Seed verwendet wird.
- Verordnung (EU) 910/2014 (eIDAS) und ihre Weiterentwicklung durch die Verordnung (EU) 2024/1183 (eIDAS 2) — europäischer Rahmen für elektronische Identität und qualifizierte Signatur. Anderes System als das autosouveräne, aber konzeptionell auf denselben kryptografischen Primitiven basierend.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Kanonischer Text über die Prinzipien und Verpflichtungen des autosouveränen Modells, älter, aber relevant für das Verständnis der Familie zeitgenössischer Lösungen.

← ZurückDas Geschäftsmodell als VertrauenssignalWeiter →Self-Hosting als berufliche Praxis

Aktuelle Lektüre

- Reflexion · 29. Juni 2026 Du bist nicht anonym
- Reflexion · 27. Mai 2026 Was eine Unterschrift nicht in Ordnung bringen kann
- Analyse · 26. Mai 2026 Echte vs. scheinbare Privatsphäre: Die Fragen, die man sich stellen sollte

Nehmen Sie diesen Artikel mit, wohin Sie ihn brauchen.

↓ Markdown ↓ Klartext ↓ PDF

Die Datei wird auf Ihr Gerät heruntergeladen. Von dort aus können Sie sie speichern, in Solo2 importieren oder teilen, wo immer Sie möchten. Cuadernos entscheidet nicht über den Zielort für Sie.

Siegellack-Siegel · SHA-256 8b33b2970a795d0ded20b460f9a14d16b5a60557330e045b7b6d79b660b2428d

Cuadernos Lacre · Eine Publikation von Menzuri Gestión S.L. ·
geschrieben von R.Eugenio · herausgegeben vom Team von Solo2.

Diese Website verwendet keine Cookies. Alles, was Ihr Browser lädt, ist von uns geschrieben oder überwacht und auf unseren europäischen Servern gehostet: das anonyme Besuchsanalysetool (Umami, selbst gehostet) und das Minimum an JavaScript, das für die Sprachauswahl und Ihre Einstellung für helles/dunkles Design erforderlich ist, die auf Ihrem eigenen Gerät gespeichert wird. Keine Ressourcen von Drittanbietern, keine Tracker, kein Profiling, keine Datenweitergabe. Wenn Sie uns folgen möchten: RSS.