24 słowa: czym jest tożsamość kryptograficzna

Tożsamość kryptograficzna to nie hasło: żaden serwer jej nie przechowuje i nie można jej odzyskać. Dydaktyczne wyjaśnienie mechanizmu BIP39, dlaczego dokładnie dwadzieścia cztery słowa i jaki realny ciężar spoczywa na tym, kto je posiada.

Abyśmy się dobrze zrozumieli: Jeśli zapomnisz hasła do Gmaila, Google je zresetuje. Jeśli zgubisz 24 słowa tworzące tożsamość kryptograficzną, nie ma kogo o nie poprosić. To nie tak, że procedura jest surowa – po prostu po drugiej stronie nie ma nikogo. Ta różnica to cała różnica.

Różnica między hasłem a tożsamością

Hasło w klasycznym modelu internetu nie jest tożsamością użytkownika. Jest dowodem. Użytkownik ma tożsamość – imię, e-mail, numer klienta – i aby udowodnić serwerowi, że jest tym, za kogo się podaje, przedstawia hasło, które serwer porównuje z zapisanym odciskiem. Jeśli odciski są zgodne, serwer udziela sesji. Jeśli hasło zostanie zgubione, użytkownik pozostaje tym samym użytkownikiem; to, co traci, to dowód, a istnieje procedura odzyskiwania – e-mail na zarejestrowany adres, pytanie pomocnicze – aby go przywrócić.

Tożsamość kryptograficzna działa inaczej. To nie są dane uwierzytelniające, które ktoś porównuje z zapisanym odciskiem; to *jest* kompletny matematyczny sekret sam w sobie. Nie ma znaczenia, gdzie się znajduje – na papierze, w urządzeniu, czy nawet na obcym serwerze: tożsamość istnieje dzięki swojej matematyce, a nie dzięki temu, kto ją waliduje. Tutaj pojawia się właściwość podobna do tej, którą widzieliśmy w artykule «Czym naprawdę jest SHA-256»: posiadania nie udowadnia się przez okazanie sekretu, lecz przez użycie go do podpisania. Tak wytworzony podpis każdy może sprawdzić za pomocą wartości publicznej, która jest matematycznie wyprowadzona z samego sekretu, bez konieczności znajomości sekretu i bez pośrednictwa strony trzeciej w weryfikacji. Kto ma sekret, ten jest tożsamością; kto go zgubi, przestaje nią być. Wyrok jest kategoryczny: **nie ma nikogo, kogo można by poprosić o zwrócenie tożsamości. Taka osoba nie istnieje, ponieważ w ogóle jej nie posiadała.**

Co reprezentuje dwadzieścia cztery słowa

Tożsamość kryptograficzna jest zazwyczaj reprezentowana przez matematyczny sekret o długości trzydziestu dwóch bajtów – dwustu pięćdziesięciu sześciu bitów. Liczba, którą trudno zapamiętać i jeszcze trudniej bezbłędnie przepisać. Branża kryptograficzna rozwiązała ten problem w 2013 roku za pomocą małego i eleganckiego standardu o nazwie BIP39: sposobu reprezentacji tych dwustu pięćdziesięciu sześciu bitów jako sekwencji dwudziestu czterech słów zaczerpniętych z oficjalnej listy dwóch tysięcy czterdziestu ośmiu słów. Kryjąca się za tym arytmetyka pasuje idealnie; ci, którzy chcą ją zobaczyć szczegółowo, znajdą ją na marginesie.

Liczenie zaczyna się od końca. Chcemy reprezentować dwieście pięćdziesiąt sześć bitów sekretu, dodając osiem bitów sumy kontrolnej (checksum): łącznie dwieście sześćdziesiąt cztery bity. Jeśli rozdzielimy je na dwadzieścia cztery słowa – liczbę możliwą do zapisania i podyktowania bez strat – każde słowo musi dostarczać dokładnie jedenaście bitów informacji. A jedenaście bitów to dwa do potęgi jedenastej możliwości, czyli dwa tysiące czterdzieści osiem. Stąd oficjalne słownictwo BIP39 ma dokładnie taki rozmiar: lista istnieje na miarę problemu, a nie odwrotnie.

Liczenie nie jest dekoracyjne. Jeśli ktoś przepisze dwadzieścia trzy słowa poprawnie i pomyli się przy dwudziestym czwartym, suma kontrolna to wykryje: oprogramowanie powie mu „ta sekwencja jest nieprawidłowa”. Jeśli ktoś przepisze wszystkie dwadzieścia cztery słowa poprawnie, oprogramowanie jednoznacznie wyprowadzi tę samą tożsamość. Wybór listy słów jest również celowy: słowa ze słownika BIP39 są krótkie, wyraźnie różniące się od siebie, bez znaków diakrytycznych, wybrane tak, aby zminimalizować pomyłki fonetyczne i ortograficzne. Jest to słownictwo zaprojektowane tak, aby ludzie mogli je zapamiętać, zapisać i podyktować bez strat.

Od frazy do klucza

Te dwadzieścia cztery słowa nie są kluczem kryptograficznym, który podpisuje wiadomości. Są odtwarzalną reprezentacją oryginalnej entropii, która poprzez deterministyczny proces zwany PBKDF2 zostaje przekształcona w ziarno (seed) o długości sześćdziesięciu czterech bajtów. Z tego ziarna, również w sposób deterministyczny, wywodzone są konkretne klucze kryptograficzne, którymi posługuje się użytkownik: klucz prywatny do podpisywania oraz odpowiadający mu klucz publiczny, który jest publikowany w celu weryfikacji podpisów. Ten sam mechanizm w różnych systemach: kryptowaluty używają krzywej secp256k1; protokół Signal i wiele nowoczesnych systemów używa Ed25519 na krzywej Curve25519. Dla konkretnej krzywej, takiej jak Ed25519, standardy BIP32 i SLIP-0010 pobierają to sześćdziesięcioczterobajtowe ziarno i deterministycznie wywodzą trzydzieści dwa bajty, które stanowią efektywny klucz podpisujący — te same trzydzieści dwa bajty, od których zaczyna się przykład kodu w następnej sekcji.

To standardowy sposób, w jaki cała branża prezentuje mechanizm użytkownikowi —portfele kryptowalutowe, menedżerowie tożsamości zdecentralizowanej, Signal w części dotyczącej tożsamości trwałej, a wśród nich Solo2—: użytkownik w praktyce nigdy nie widzi ziarna ani wywodzonych kluczy. Widzi dwadzieścia cztery słowa podczas tworzenia swojej tożsamości i opcjonalnie zapisuje je na kartce papieru. Słowa te podróżują następnie między jego urządzeniami, gdy chce przenieść tożsamość: wpisuje je w nowej aplikacji, aplikacja wywodzi to samo ziarno, te same klucze, tę samą tożsamość. Jest to mechanizm przenośny, solidny pod względem kryptograficznym i, w granicach rozsądku, możliwy do zapamiętania.

Jak podpisywać kluczem (muśnięcie Zig)

W Zig, gdy posiada się już trzydziestodwubajtowe ziarno wywiedzione z dwudziestu czterech słów, podpisanie wiadomości za pomocą Ed25519 mieści się w kilku liniach:

```
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);
```

Operacja podpisywania generuje sześćdziesiąt cztery bajty —zwane podpisem— które mogły zostać utworzone wyłącznie na podstawie odpowiedniego klucza prywatnego. Weryfikacja jest publiczna: każdy, kto posiada klucz publiczny, może sprawdzić, czy podpis odpowiada wiadomości. Bez klucza prywatnego nikt nie może wygenerować poprawnego podpisu dla danej wiadomości; posiadając klucz publiczny, każdy może wykryć, czy podpis jest prawidłowy. Ta asymetria pozwala sygnatariuszowi udowodnić autorstwo bez dzielenia się sekretem.

Poprzedni przykład to minimalna wersja z podręcznika. W prawdziwym kodzie Solo2 łańcuch przechodzi przez dwa pliki: jeden w JavaScript, który żyje w przeglądarce użytkownika i rekonstruuje entropię z dwudziestu czterech słów, drugi w Zig wewnątrz biblioteki *zcatcrypto*, który przyjmuje tę entropię i wyprowadza konkretne klucze kryptograficzne. Zaczynając od strony przeglądarki:

```
// 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 };
}
```

Te trzydzieści dwa bajty entropii, wraz z kolejnymi trzydziestoma dwoma wyprowadzonymi w tym samym kroku, trafiają do modułu WebAssembly Zig, który generuje właściwe klucze Ed25519. Pełna funkcja, wraz z końcowym czyszczeniem pamięci, mieści się na jednym ekranie:

```
// 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;
}
```

Warto odnotować dwa szczegóły. Pierwszy: ten sam seed zawsze produkuje tę samą parę kluczy — to właśnie pozwala na odzyskanie tożsamości poprzez wpisanie dwudziestu czterech słów na nowym urządzeniu. Drugi: seed jest jawnie usuwany z pamięci w ostatniej linii. Po tym punkcie nawet sama funkcja nie mogłaby zrekonstruować kluczy; jedynym źródłem byłyby słowa użytkownika.

Dla tych, którzy chcą to sprawdzić na małych liczbach. Schemat podpisu można prześledzić w całości na liczbach na tyle małych, by obliczenia wykonać ręcznie. Kto woli nie wchodzić w arytmetykę, może pominąć ten blok bez utraty wątku artykułu; kto chce zobaczyć mechanizm działający krok po kroku, znajdzie go tutaj. **Zasady publiczne**, które każdy może przeczytać: liczba pierwsza $p = 23$ (w prawdziwym Ed25519 ma ona około siedemdziesięciu siedmiu cyfr; używamy dwudziestu trzech, aby obliczenia zmieściły się na jednej stronie), podstawa $g = 2$, której rząd w tej grupie wynosi $q = 11$, oraz konwencja, że cała arytmetyka z g odbywa się *módulo p*, a wszystkie wykładniki są redukowane *módulo q*. **Wybór prywatny**, jedyny i nigdy nieudostępniany: sekret $x = 6$. To jest tożsamość.

Krok 1 — Publiczna część tożsamości. Jest obliczana raz i publikowana otwarcie.

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

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

Publiczna część tożsamości to **18**. Każdy może ją wziąć i użyć do weryfikacji podpisów złożonych za pomocą tej tożsamości. Nikt, obserwując tylko 18, nie może odzyskać sekretu 6: to jest problem logarytmu dyskretnego, do którego wrócimy na końcu.

Krok 2 — Podpisywanie wiadomości. Posiadacz tożsamości chce podpisać wiadomość $m = 7$. Zaczyna od wyboru nowej losowej wartości $k = 4$, która zostanie użyta tylko raz i nigdy nie zostanie udostępniona (w prawdziwym Ed25519 k jest wyprowadzane deterministycznie z wiadomości i sekretu, aby uniknąć niebezpieczeństwa ponownego użycia, ale rolę odgrywa dokładnie taką). Następnie oblicza trzy liczby:

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

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

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

Podpis to para **(r, s) = (16, 10)**. Podróżuje jawnie wraz z wiadomością. Każdy może go przeczytać. Uwaga dydaktyczna: w prawdziwym Ed25519 funkcja H to SHA-512, kryptograficznie solidna; tutaj używamy uproszczenia $e = (r + m) \bmod q$, aby czytelnik mógł prześledzić kroki bez konieczności obliczania hasha. Struktura algorytmu jest taka sama.

Krok 3 — Weryfikacja podpisu. Weryfikujący posiada część publiczną $y = 18$, wiadomość $m = 7$ oraz podpis $(r, s) = (16, 10)$. Rekonstruuje e w ten sam sposób — $e = (16 + 7) \bmod 11 = 1$ — i sprawdza, czy ta równość jest spełniona:

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

Oblicza obie strony oddzielnie:

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

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

Obie strony dają **12**. Podpis jest ważny. Każdy, kto posiada część publiczną 18, może dojść do tego wniosku, nie wiedząc nigdy, że sekretem było 6.

A co z osobą trzecią, która próbowałaby sfałszować podpis? Ewa widziała wszystko, co publiczne, przechodzące przez kanał: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Aby podpisać *inną* wiadomość w imieniu tej tożsamości, musiałaby znać x. Jej jedyną drogą jest zadanie sobie pytania: „dla jakiego wykładnika x spełnione jest $2^x \bmod 23 = 18$?”. Przy $p = 23$ może spróbować 0, 1, 2, 3, … i znaleźć go w kilka sekund. Ale po zastąpieniu 23 liczbą pierwszą o wymiarach rzeczywistych Ed25519, przestrzeń możliwych wykładników przekracza liczbę atomów w obserwowalnym wszechświecie. **Nie istnieje dziś żaden algorytm znany ludzkości, który mógłby przemierzyć tę przestrzeń w czasie krótszym niż miliardy lat.** To ten sam problem logarytmu dyskretnego, który leży u podstaw Diffie-Hellman z poprzedniego artykułu, zastosowany tutaj do schematu podpisu.

To, co właśnie przeszliśmy, to *dokładnie* Schnorr, schemat podpisu, którego Ed25519 jest wariantem dostosowanym do krzywej eliptycznej. W prawdziwym Ed25519 wszystkie operacje są wykonywane na punktach konkretnej krzywej (Curve25519) zamiast na liczbach całkowitych modulo liczba pierwsza, a funkcja H to SHA-512 zamiast uproszczonej sumy, której użyliśmy powyżej. Obie zmiany to korekty implementacyjne — zyskanie odporności kryptograficznej na ataki typu brute force, zyskanie dodatkowych właściwości bezpieczeństwa dla k. Struktura algorytmiczna, trzy operacje i przyczyna asymetrii są takie same.

Warto tutaj zrobić krótką przerwę, ponieważ cały łańcuch może na pierwszy rzut oka zostać pomylony z inną prymitywą z tej trójki: hashem. Tak nie jest. Hash to unikalna funkcja, która kompresuje — wchodzi wiele bajtów, wychodzi krótki odcisk, tam kończy się droga. Tożsamość kryptograficzna to komplementarna para matematyczna: sekret zostaje i podpisuje; jego publiczny odpowiednik jest publikowany i weryfikuje. Tam, gdzie hash zapada informacje w jednym kierunku, tożsamość ustanawia asymetrię między dwiema połówkami. Hash poświadcza, co zostało powiedziane; tożsamość poświadcza, kto to powiedział.

Czym fraza nie jest

Warto wyjaśnić trzy częste nieporozumienia. Fraza nie jest hasłem w ścisłym tego słowa znaczeniu: nie jest porównywana z odciskiem przechowywanym na serwerze; wpisuje się ją do urządzenia użytkownika, aby matematycznie zrekonstruować tożsamość. Frazy nie da się odzyskać: jeśli zostanie zgubiona, nie ma kogo o nią poprosić; jeśli zostanie powielona, powielona zostanie również tożsamość. Fraza nie jest poświadczeniem oddzielnym od tożsamości: fraza *jest* tożsamością. Kto ją posiada, może działać jako ta tożsamość, bez dodatkowej zgody, bez procesu autoryzacji, bez możliwości odzyskania.

Właśnie ta trzecia właściwość zmienia wagę zagadnienia. Zagubione hasło to uciążliwość administracyjna. Zagubiona tożsamość kryptograficzna to sama tożsamość. Kartka z frazą znaleziona przez osoby trzecie to nie ryzyko kradzieży konta: to przekazanie całej tożsamości. Obietnicy systemu — że nikt nie może unieważnić Twojej tożsamości ani arbitralnie Cię zablokować — towarzyszy nierozerwalnie odpowiedzialność — że jesteś jedynym powiernikiem czegoś, czego nikt nie może za Ciebie przywrócić.

Obietnica i ciężar

Model tożsamości kryptograficznej często otrzymuje miano *samosuwerennej* —self-sovereign w literaturze anglosaskiej—. Wybór słowa jest celowy i dość dokładnie opisuje ten stan. Użytkownik jest suwerenem swojej tożsamości w sensie niemal średniowiecznym: nie nadaje jej żaden król, żaden wydawca, żadna władza centralna; nikt z powyższych nie może jej również odebrać. Ale też, niczym średniowieczny monarcha,

użytkownik ponosi pełne konsekwencje swoich błędów: nie ma regenta, który podjąłby decyzje za niego, jeśli zgubi pieczęć.

Wybór między tożsamością zarządzaną przez stronę trzecią a tożsamością samosuwerenną nie ma jednej uniwersalnej poprawnej odpowiedzi. W przypadku konta na nieistotnym forum tożsamość zarządzana jest prawdopodobnie proporcjonalna do ryzyka. W przypadku tożsamości zawodowej, która podpisuje dokumenty wiążące prawnie, tożsamości ekonomicznej, która strzeże własnych oszczędności, czy tożsamości do komunikacji zawodowej z klientami, którzy powierzyli poufne informacje, kwestia ta ulega zmianie. Tam pytanie przestaje brzmieć „czy to wygodne?”, a staje się pytaniem „kto, poza mną, ma moc działania jako ja i w jakich okolicznościach?”.

Gdzie ten mechanizm pojawia se w rzeczywistych systemach

Standard BIP39 narodził się w świecie Bitcoin w 2013 roku i szybko rozprzestrzenił się na cały ekosystem kryptowalut: każdy poważny portfel akceptuje dziś dwunasto- lub dwudziestoczterowyrazową frazę BIP39 jako zabezpieczenie tożsamości ekonomicznej jej posiadacza. Poza kryptowalutami ta sama podstawowa koncepcja — para kryptograficzna potwierdzająca autorstwo bez pośrednika — pojawia się w innych systemach o innej składni. Klucze SSH, których administrator systemów używa do uzyskiwania dostępu do swoich serwerów, są klasycznym przypadkiem: klucz prywatny, który administrator przechowuje na swojej maszynie, oraz klucz publiczny, który jest kopiowany na każdy serwer; nie interweniuje żaden podmiot porównywalny z usługą scentralizowaną. Protokół Signal wykorzystuje Ed25519 z trwałym materiałem klucza na urządzeniu; europejskie rozporządzenia eIDAS, w części dotyczącej podpisu kwalifikowanego, opierają się na tej samej zasadzie kryptograficznej, z tą różnicą, że klucz jest przechowywany przez kwalifikowanego dostawcę usług zaufania zamiast przez użytkownika.

Solo2, platforma wydawnicza niniejszej publikacji, używa dwudziestoczterowyrazowej frazy BIP39 jako tożsamości każdego użytkownika. Użytkownik, podczas zakładania konta, widzi słowa raz. Nie są one przechowywane na żadnym serwerze Solo2 ani nikogo innego: jeśli użytkownik je zapisze i będzie je chronił, zachowa swoją tożsamość na zawsze. Jeśli je zgubi, traci je bezpowrotnie. Jest to logiczna konsekwencja architektury bez operatora pośredniczącego: gdyby Solo2 mogło zwrócić tożsamość użytkownikowi, który ją zgubił, mogłoby ją również przekazać każdemu, kto wywarłby presję na Solo2, aby ją otrzymać.

Dla czytelnika zawodowego

Cztery kwestie dla osób rozważających przyjęcie kryptograficznej tożsamości samostanowiącej (autosoberana) w kontekście zawodowym:

1. Fraza jest tożsamością. Przechowywanie fizyczne — papier, kilka kopii w różnych miejscach, ostatecznie grawerowany metal do długotrwałego użytku — oferuje więcej gwarancji niż przechowywanie cyfrowe, które zwiększa powierzchnię ataku bez zmniejszania ryzyka utraty.
2. Brak możliwości odzyskania. Zaprojektowanie procesu przy założeniu, że pewnego dnia kopia pierwotna zostanie utracona, jest o wiele rozsądniejsze niż odkrycie tego w dniu jej utraty. Druga kopia oddzielona geograficznie rozwiązuje niemal wszystkie scenariusze.
3. To nie to samo co certyfikat kwalifikowany eIDAS. W przypadku podpisu kwalifikowanego w Unii — akty notarialne, niektóre formalności w urzędach — przepisy wymagają kwalifikowanego dostawcy, który przechowuje klucz. Kryptograficzna tożsamość samostanowiąca służy do komunikacji zawodowej i podpisywania dokumentów o wartości dowodowej, ale nie zastępuje automatycznie certyfikatu kwalifikowanego w przypadkach, gdy wymagają tego przepisy.
4. Jeśli tożsamość ma zostać przeniesiona — dziedziczenie, sukcesja zawodowa, zakończenie działalności — warto przygotować procedurę wcześniej, a ne po fakcie. Formalne procedury z kopertami zapieczętowanymi lakie (lacre), instrukcje dla wykonawcy testamentu, depozyt w kancelarii notarialnej to klasyczne rozwiązania w pełni kompatybilne z kryptograficzną naturą tego aktywa.

Ten artykuł zamyka koncepcyjne trio, które otworzyło cykl — hash, szyfrowanie, tożsamość —. Te trzy idee budują się na sobie: hash daje niezmienny odcisk, szyfrowanie daje poufność bez zaufanej strony trzeciej, tożsamość daje autorstwo bez strony trzeciej nadającej uprawnienia. Wszystkie trzy łączy cecha, która również nie jest ideologiczna: przenoszą one z podmiotu zarządzającego usługą na użytkownika możliwości techniczne, które tradycyjnie należały do operatora. Przenoszą wraz z nimi również odpowiedzialność. Uczciwa rozmowa o którejkolwiek z tych trzech idei wymaga rozmowy także o dwóch pozostałych.

Źródła i dodatkowa lektura

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, propozycja ulepszenia Bitcoin z 2013 roku. De facto standard dla fraz odzyskiwania w branży kryptowalut.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), w tym Ed25519. IETF, styczeń 2017. Normatywna specyfikacja schematu podpisu stosowanego w dużej części współczesnego przemysłu.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, wersja 2.0. IETF, wrzesień 2000. Definiuje algorytm PBKDF2 stosowany w wyprowadzaniu BIP39 z frazy do ziarna (seed).
- Rozporządzenie (UE) 910/2014 (eIDAS) oraz jego ewolucja poprzez Rozporządzenie (UE) 2024/1183 (eIDAS 2) — europejskie ramy tożsamości elektronicznej i podpisu kwalifikowanego. Reżim inny niż samostanowiący, ale koncepcyjny oparty na tych samych prymitywach kryptograficznych.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Kanoniczny tekst na temat zasad i zobowiązań modelu samostanowiącego, wcześniejszy, ale istotny dla zrozumienia rodziny współczesnych rozwiązań.

← PoprzedniModel biznesowy jako sygnał zaufaniaNastępny →Self-hosting jako praktyka zawodowa

Ostatnie lektury

- Refleksja · 29 czerwca 2026 Nie jesteś anonimowy
- Refleksja · 27 maja 2026 Czego podpis nie może naprawić
- Analiza · 26 maja 2026 Prywatność rzeczywista vs pozorna: pytania, które warto sobie zadać

Zabierz ten artykuł tam, gdzie go potrzebujesz.

↓ Markdown ↓ Zwykły tekst ↓ PDF

Plik zostanie pobrany na Twoje urządzenie. Stamtąd możesz go zapisać, zaimportować do Solo2 lub udostępnić w dowolnym miejscu. Cuadernos nie decyduje o miejscu docelowym za Ciebie.

Pieczęć lakowa · SHA-256 0355d6114f14ae5ff3f1ac554b78b8a55a7d549769d32c36a7ea7bce8683ff28

Cuadernos Lacre · Publikacja Menzuri Gestión S.L. ·
napisana przez R.Eugenio · redagowana przez zespół Solo2.