Konsepti · 23. toukokuuta 2026

← Edellinen·Seuraava →

24 sanaa: mikä on kryptografinen identiteetti

Kryptografinen identiteetti ei ole salasana: mikään palvelin ei tallenna sitä, eikä sitä voi palauttaa. Didaktinen selitys BIP39-mekanismista, miksi juuri kaksikymmentäneljä sanaa ja mikä todellinen painolasti lankeaa sille, joka ne omistaa.

Ymmärretään toisiamme: Jos unohdat Gmail-salasanasi, Google nollaa sen puolestasi. Jos kadotat kryptografisen identiteetin muodostavat 24 sanaa, ei ole ketään, jolta pyytää niitä. Kyse ei ole siitä, että menettely olisi tiukka – kyse on siitä, ettei toisessa päässä ole ketään. Tämä ero on ratkaiseva.

Salasanan ja identiteetin ero

Salasana perinteisessä internet-mallissa ei ole käyttäjän identiteetti. Se on tosite. Käyttäjällä on identiteetti – nimi, sähköposti, asiakasnumero – ja todistaakseen palvelimelle olevansa se, joka väittää olevansa, hän esittää salasanan, jota palvelin vertaa tallennettuun jälkeen. Jos jäljet täsmäävät, palvelin myöntää istunnon. Jos salasana katoaa, käyttäjä pysyy samana käyttäjänä; se, minkä hän menettää, on tosite, ja sen palauttamiseksi on olemassa menettely – sähköposti rekisteröityyn osoitteeseen, turvakysymys – sen korvaamiseksi.

Kryptografinen identiteetti toimii eri tavalla. Se ei ole valtuus, jota joku vertaa tallennettuun jälkeen; se *on* täydellinen matemaattinen salaisuus itsessään. On samantekevää, missä se sijaitsee – paperilla, laitteessa tai jopa vieraan palvelimella: identiteetti on olemassa matematiikkansa vuoksi, ei sen vahvistajan vuoksi. Tässä esiintyy ominaisuus, joka muistuttaa artikkelissa «Mitä SHA-256 todellisuudessa on» nähtyä: omistajuutta ei osoiteta näyttämällä salaisuutta, vaan käyttämällä sitä allekirjoittamiseen. Näin syntyneen allekirjoituksen kuka tahansa voi tarkistaa julkisella arvolla, joka on johdettu matemaattisesti itse salaisuudesta ilman tarvetta tuntea salaisuutta ja ilman kolmannen osapuolen välitystä. Se, jolla on salaisuus, on identiteetti; se, joka sen kadottaa, lakkaa olemasta se. Tuomio on ehdoton: **ei ole ketään, jolta pyytää identiteetin palauttamista. Tällaista henkilöä ei ole olemassa, koska hänellä ei ollut sitä alun perinkään.**

Mitä kaksikymmentäneljä sanaa edustavat

Kryptografinen identiteetti esitetään yleensä kolmenkymmenenkahden tavun matemaattisena salaisuutena – kaksisataaviisikymmentäkuusi bittiä. Luku, joka on vaikea muistaa ja vielä vaikeampi kirjoittaa muistiin virheettömästi. Kryptoteollisuus ratkaisi tämän ongelman vuonna 2013 pienellä ja tyylikkäällä standardilla nimeltä BIP39: tapa esittää nämä kaksisataaviisikymmentäkuusi bittiä 24 sanan sarjana, joka on otettu virallisesta 2048 sanan luettelosta. Taustalla oleva matematiikka täsmää tyylikkäästi; ne, jotka haluavat nähdä sen yksityiskohtaisesti, löytävät sen marginaalista.

Lasku alkaa lopusta. Haluamme esittää salaisuuden 256 bittiä lisäämällä kahdeksan bitin tarkistussumman: yhteensä 264 bittiä. Jos jaamme ne 24 sanaan – hallittava määrä muistiin merkitsemistä ja sanelemista varten ilman hävikkiä – jokaisen sanan on tuotava tasan yksitoista bittiä tietoa. Ja yksitoista bittiä on kaksi potenssiin yksitoista mahdollisuutta eli 2048. Tästä syystä virallinen BIP39-sanasto on juuri tämän kokoinen: luettelo on tehty ongelman mittojen mukaan, ei päinvastoin.

Laskenta ei ole koristelua. Jos joku kirjoittaa 23 sanaa oikein ja erehtyy 24. sanassa, tarkistussumma havaitsee sen: ohjelmisto sanoo hänelle "tämä sarja ei ole kelvollinen". Jos joku kirjoittaa kaikki 24 sanaa oikein, ohjelmisto johtaa saman identiteetin yksiselitteisesti. Sanaluettelon valinta on myös harkittu: BIP39-sanaston sanat ovat lyhyitä, toisistaan eroavia, ilman diakriittisiä merkkejä ja valittu minimoimaan foneettiset ja oikeinkirjoitukseen liittyvät sekaannukset. Se on sanasto, joka on suunniteltu ihmisten muistettavaksi, kirjoitettavaksi ja sanelemaksi ilman hävikkiä.

Lauseesta avaimeksi

Nuo kaksikymmentäneljä sanaa eivät ole se kryptografinen avain, joka allekirjoittaa viestit. Ne ovat palautettavissa oleva esitys alkuperäisestä entropiasta, joka PBKDF2-nimisen deterministisen prosessin kautta muunnetaan kuudenkymmenenneljän tavun siemeneksi (seed). Tästä siemenestä johdetaan, myös deterministisesti, ne konkreettiset kryptografiset avaimet, joita käyttäjä käyttää: yksityinen avain allekirjoittamiseen ja vastaava julkinen avain, joka julkaistaan allekirjoitusten todentamiseksi. Sama mekanismi eri järjestelmissä: kryptovaluutat käyttävät secp256k1-käyrää; Signal-protokolla ja monet nykyjärjestelmät käyttävät Ed25519-algoritmia Curve25519-käyrällä. Tietylle käyrälle, kuten Ed25519, BIP32- ja SLIP-0010-standardit ottavat tuon kuudenkymmenenneljän tavun siemenen ja johtavat deterministisesti ne kolmekymmentäkaksi tavua, jotka muodostavat varsinaisen allekirjoitusavaimen — ne samat kolmekymmentäkaksi tavua, joilla seuraavan osion koodiesimerkki alkaa.

Tämä on se standarditapa, jolla koko ala esittelee mekanismin käyttäjälle —kryptovaluuttalompakot, hajautetun identiteetin hallintatyökalut, Signal pysyvän identiteetin osaltaan, Solo2 näiden joukossa—: käyttäjä ei käytännössä koskaan näe siementä tai johdettuja avaimia. Hän näkee ne kaksikymmentäneljä sanaa identiteettiä luodessaan ja valinnaisesti kirjoittaa ne paperille. Sanat matkaavat sitten hänen laitteidensa välillä, kun hän haluaa siirtää identiteetin: hän syöttää ne uuteen sovellukseen, sovellus johtaa saman siemenen, samat avaimet, saman identiteetin. Se on siirrettävä, kryptografisesti vankka ja kohtuuden rajoissa muistettavissa oleva mekanismi.

Miten avaimella allekirjoitetaan (Zig-sivallus)

Zig-kielellä viestin allekirjoittaminen Ed25519-algoritmilla mahtuu muutamalle riville, kun käytössä on kaksikymmentäneljästä sanasta johdettu kolmenkymmenenkahden tavun siemen:

```
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);
```

Allekirjoitustoiminto tuottaa kuusikymmentäneljä tavua —allekirjoitukseksi kutsuttuna— jotka on voitu luoda vain vastaavasta yksityisestä avaimesta. Todentaminen on julkista: kuka tahansa, jolla on julkinen avain, voi tarkistaa, että allekirjoitus vastaa viestiä. Ilman yksityistä avainta kukaan ei voi tuottaa viestiin pätevää allekirjoitusta; julkisen avaimen avulla kaikki voivat havaita, onko allekirjoitus pätevä. Tämä epäsymmetria mahdollistaa sen, että allekirjoittaja osoittaa tekijyyden paljastamatta salaisuutta.

Edellinen esimerkki on ohjekirjan minimiversio. Solo2:n todellisessa koodissa ketju kulkee kahden tiedoston läpi: toinen JavaScriptillä, joka toimii käyttäjän selaimessa ja rekonstruoi entropian kahdestakymmenestäneljästä sanasta, ja toinen Zigillä *zcatcrypto*-kirjastossa, joka ottaa kyseisen entropian ja johtaa siitä tarkat kryptografiset avaimet. Aloittaen selaimen puolelta:

```
// 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 };
}
```

Nuo kolmekymmentäkaksi tavua entropiaa, yhdessä samassa vaiheessa johdetun toisen kolmenkymmenenkahden tavun kanssa, siirtyvät Zigin WebAssembly-moduuliin, joka generoi varsinaiset Ed25519-avaimet. Koko funktio lopullisine muistin puhdistuksineen mahtuu yhdelle näytölle:

```
// 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;
}
```

Kaksi yksityiskohtaa on syytä huomata. Ensimmäinen: sama siemen (seed) tuottaa aina saman avainparin — juuri tämä mahdollistaa identiteetin palauttamisen syöttämällä ne kaksikymmentäneljä sanaa uuteen laitteeseen. Toinen: siemen pyyhitään nimenomaisesti muistista viimeisellä rivillä. Tämän pisteen jälkeen edes funktio itse ei voisi rekonstruoida avaimia; käyttäjän sanat olisivat ainoa lähde.

Niille, jotka haluavat tarkistaa asian pienillä luvuilla. Allekirjoituskaavio voidaan käydä läpi kokonaisuudessaan luvuilla, jotka ovat riittävän pieniä laskutoimitusten tekemiseen käsin. Ne, jotka eivät halua syventyä aritmetiikkaan, voivat hypätä tämän lohkon yli menettämättä artikkelin juonta; ne, jotka haluavat nähdä mekanismin toimivan vaihe vaiheelta, löytävät sen täältä. **Julkiset säännöt**, jotka kuka tahansa voi lukea: alkuluku $p = 23$ (todellisessa Ed25519:ssä se on noin seitsemänkymmentäseitsemän numeroa pitkä; käytämme kahtakymmentäkolmea, jotta laskut mahtuvat yhdelle sivulle), kantaluku $g = 2$, jonka kertaluku tässä ryhmässä on $q = 11$, sekä käytäntö, että kaikki aritmetiikka g:llä tehdään *módulo p* ja kaikki eksponentit supistetaan *módulo q*. **Yksityinen valinta**, yksi ainoa eikä koskaan jaettu: salaisuus $x = 6$. Tämä on identiteetti.

Vaihe 1 — Identiteetin julkinen osa. Se lasketaan kerran ja julkaistaan avoimesti.

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

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

Identiteetin julkinen osa on **18**. Kuka tahansa voi ottaa sen ja käyttää sitä tällä identiteetillä tehtyjen allekirjoitusten todentamiseen. Kukaan ei voi pelkästään lukua 18 tarkkailemalla palauttaa salaisuutta 6: tämä on diskreetin logaritmin ongelma, johon palaamme lopussa.

Vaihe 2 — Viestin allekirjoittaminen. Identiteetin haltija haluaa allekirjoittaa viestin $m = 7$. Hän aloittaa valitsemalla uuden satunnaisen arvon $k = 4$, jota käytetään vain kerran eikä sitä koskaan jaeta (todellisessa Ed25519:ssä k johdetaan deterministisesti viestistä ja salaisuudesta uudelleenkäytön vaaran välttämiseksi, mutta sen rooli on juuri tämä). Tämän jälkeen hän laskee kolme lukua:

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

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

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

Allekirjoitus on pari **(r, s) = (16, 10)**. Se kulkee avoimesti viestin mukana. Kuka tahansa voi lukea sen. Didaktinen huomautus: todellisessa Ed25519:ssä funktio H on SHA-512, kryptografisesti vankka; tässä käytämme yksinkertaistusta $e = (r + m) \bmod q$, jotta lukija voi seurata vaiheita tarvitsematta laskea hashia. Algoritmin rakenne on sama.

Vaihe 3 — Allekirjoituksen todentaminen. Todentajalla on julkinen osa $y = 18$, viesti $m = 7$ ja allekirjoitus $(r, s) = (16, 10)$. Hän rekonstruoi e:n samalla tavalla — $e = (16 + 7) \bmod 11 = 1$ — ja tarkistaa, toteutuuko tämä yhtäsuuruus:

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

Laskee kumpikin puoli erikseen:

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

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

Molemmat puolet antavat tulokseksi **12**. Allekirjoitus on pätevä. Kuka tahansa, jolla on julkinen osa 18, voi päätyä tähän johtopäätökseen tietämättä koskaan, että salaisuus oli 6.

Entä kolmas osapuoli, joka yrittää väärentää? Eva on nähnyt kaiken julkisen kulkevan kanavan kautta: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Allekirjoittaakseen *toisenlaisen* viestin tämän identiteetin nimissä hänen täytyisi tietää x. Hänen ainoa keinonsa on kysyä itseltään: "millä eksponentilla x toteutuu $2^x \bmod 23 = 18$?". Jos $p = 23$, hän voi kokeilla lukuja 0, 1, 2, 3, … ja löytää sen sekunneissa. Mutta kun 23 korvataan Ed25519:n todellisten ulottuvuuksien alkuluvulla, mahdollisten eksponenttien avaruus ylittää havaittavan universumin atomien määrän. **Ihmiskunta ei tunne nykyään mitään algoritmia, joka pystyisi läpikäymään tuon avaruuden alle miljardeissa vuosissa.** Kyseessä on sama diskreetin logaritmin ongelma, johon edellisen artikkelin Diffie-Hellman perustuu, sovellettuna tässä allekirjoituskaavioon.

Tämä, jonka olemme juuri käyneet läpi, on *tarkalleen* Schnorr, allekirjoituskaavio, jonka elliptiselle käyrälle sovitettu muunnelma Ed25519 on. Todellisessa Ed25519:ssä kaikki operaatiot tehdään tietyn käyrän (Curve25519) pisteillä sen sijaan, että ne tehtäisiin kokonaisluvuilla modulo alkuluku, ja funktio H on SHA-512 yllä käyttämämme lelusumman sijasta. Nämä kaksi korvausta ovat toteutuksen hienosäätöjä — kryptografisen kestävyyden saavuttaminen raakaa voimaa vastaan sekä k:n turvaominaisuuksien parantaminen. Algoritminen rakenne, ne kolme operaatiota ja epäsymmetrian syy ovat samat.

Tässä on syytä pysähtyä hetkeksi, sillä koko ketju voidaan nopealla silmäyksellä sekoittaa kolmikon toiseen primitiiviin: hashiin. Se ei ole sitä. Hash on ainutlaatuinen funktio, joka tiivistää — sisään menee paljon tavuja, ulos tulee lyhyt sormenjälki, ja tie päättyy siihen. Kryptografinen identiteetti on matemaattisesti täydentävä pari: salaisuus jää ja allekirjoittaa; sen julkinen vastinpari julkaistaan ja todennetaan. Siinä missä hash romahduttaa tiedon yhteen suuntaan, identiteetti luo epäsymmetrian kahden puoliskon välille. Hash todistaa, mitä sanottiin; identiteetti todistaa, kuka sen sanoi.

Mitä lause ei ole

Kolme yleistä väärinkäsitystä on syytä korjata. Lause ei ole salasana varsinaisessa merkityksessä: sitä ei verrata palvelimelle tallennettuun sormenjälkeen; se syötetään käyttäjän laitteeseen identiteetin matemaattiseksi palauttamiseksi. Lausetta ei voi palauttaa: jos se katoaa, sitä ei voi pyytää keneltäkään; jos se kopioidaan, myös identiteetti kopioituu. Lause ei ole identiteetistä erillinen tunnistetieto: lause *on* identiteetti. Se, jolla se on hallussaan, voi toimia tuona identiteettinä ilman erillistä lupaa, ilman valtuutusprosessia, ilman palautusmahdollisuutta.

Tämä kolmas ominaisuus on se, mikä muuttaa asian painoarvon. Kadonnut salasana on hallinnollinen vaiva. Kadonnut kryptografinen identiteetti on identiteetti itse. Kolmansien osapuolten löytämä paperi, jossa lause on, ei ole vain tilin ryöstön riski: se on koko identiteetin luovuttaminen. Järjestelmän lupaukseen —että kukaan ei voi peruuttaa identiteettiäsi tai estää sinua mielivaltaisesti— liittyy erottamattomasti vastuu —että olet ainoa haltija jollekin, mitä kukaan ei voi palauttaa puolestasi.

Lupaus ja painoarvo

Kryptografista identiteettimallia kutsutaan usein *itsesuvereeniksi* —self-sovereign englanninkielisessä kirjallisuudessa—. Sanavalinta on harkittu ja kuvaa tilaa varsin tarkasti. Käyttäjä on identiteettinsä suvereeni lähes keskiaikaisessa mielessä: mikään kuningas, liikkeeseenlaskija tai keskushallinto ei sitä myönnä, eikä kukaan edellä mainituista voi sitä myöskään peruuttaa. Mutta keskiaikaisen monarkin tavoin käyttäjä kantaa myös virheidensä täydet seuraukset: ei ole sijaishallitsijaa, joka tekisi päätöksiä hänen puolestaan, jos hän kadottaa sinetin.

Valinnalla kolmannen osapuolen hallinnoiman identiteetin ja itsesuvereenin identiteetin välillä ei ole yhtä yleispätevää oikeaa vastausta. Merkityksettömän foorumitilin kohdalla hallinnoitu identiteetti on todennäköisesti oikeassa suhteessa riskiin. Mutta kun kyseessä on ammatillinen identiteetti, jolla allekirjoitetaan oikeudellisesti sitovia asiakirjoja, taloudellinen identiteetti, joka vartioi omia säästöjä, tai ammatillinen viestintäidentiteetti asiakkaiden kanssa, jotka ovat uskoneet haltuunsa arkaluonteista tietoa, asia muuttuu. Silloin kysymys ei ole enää «onko se mukavaa?» vaan «kenellä muulla kuin minulla on valta toimia minuna, ja missä olosuhteissa?».

Missä tämä mekanismi esiintyy todellisissa järjestelmissä

BIP39 syntyi Bitcoin-maailmassa vuonna 2013 ja levisi nopeasti koko kryptovaluuttaekosysteemiin: mikä tahansa vakavasti otettava lompakko hyväksyy nykyään kahdentoista tai kahdenkymmenen neljän sanan BIP39-lauseen haltijansa taloudellisen identiteetin varmuuskopioksi. Kryptovaluuttojen ulkopuolella sama peruskäsite — kryptografinen pari, joka todistaa tekijyyden ilman välikättä — esiintyy muissa järjestelmissä eri syntaksilla. SSH-avaimet, joita järjestelmänvalvoja käyttää palvelimiinsa pääsyyn, ovat klassinen tapaus: yksityinen avain, jota ylläpitäjä säilyttää koneellaan, ja julkinen avain, joka kopioidaan jokaiselle palvelimelle; mikään keskitettyyn palveluun verrattava taho ei puutu asiaan. Signal-protokolla käyttää Ed25519-algoritmia laitteessa olevalla pysyvällä avainmateriaalilla; eurooppalainen eIDAS perustuu pätevän allekirjoituksen osalta samaan kryptografiseen periaatteeseen sillä erolla, että avainta säilyttää pätevä luottamuspalvelun tarjoaja käyttäjän sijasta.

Solo2, tämän julkaisun kustantaja, käyttää kahdenkymmenen neljän sanan BIP39-lausetta kunkin käyttäjän identiteettinä. Käyttäjä näkee sanat kerran luodessaan tilinsä. Niitä ei tallenneta millekään Solo2:n tai kenenkään muun palvelimelle: jos käyttäjä merkitsee ne muistiin ja säilyttää niitä, hän säilyttää identiteettinsä ikuisesti. Jos hän kadottaa ne, hän kadottaa ne. Tämä on looginen seuraus arkkitehtuurista, jossa ei ole operaattoria välissä: jos Solo2 voisi palauttaa identiteetin sen kadottaneelle käyttäjälle, se voisi antaa sen myös kenelle tahansa, joka painostaa Solo2:ta luovuttamaan sen.

Ammattilukijalle

Neljä huomiota niille, jotka harkitsevat kryptografisen itsenäisen (autosoberana) identiteetin käyttöönottoa ammatillisessa yhteydessä:

1. Lause on identiteetti. Fyysinen säilytys — paperi, useat kopiot eri paikoissa, mahdollisesti kaiverrettu metalli pitkäaikaiseen käyttöön — tarjoaa enemmän takuita kuin digitaalinen säilytys, joka lisää hyökkäyspinta-alaa vähentämättä katoamisriskiä.
2. Palautusmahdollisuutta ei ole. Prosessin suunnittelu olettaen, että ensisijainen kopio joskus katoaa, on paljon suositeltavampaa kuin sen huomaaminen katoamispäivänä. Toinen maantieteellisesti erillään oleva kopio ratkaisee melkein kaikki skenaariot.
3. Se ei ole sama asia kuin eIDAS-pätevä varmenne. Unionin pätevää allekirjoitusta varten — notaarin asiakirjat, tietyt asioinnit hallinnon kanssa — lainsäädäntö edellyttää pätevää tarjoajaa, joka säilyttää avainta. Kryptografinen itsenäinen identiteetti palvelee ammatillista viestintää ja todistusvoimaista asiakirjojen allekirjoittamista, mutta se ei korvaa automaattisesti pätevää varmennetta tapauksissa, joissa säännös sitä edellyttää.
4. Jos identiteetti on tarkoitus siirtää — perintö, ammatillinen seuraanto, toiminnan lopettaminen — on suositeltavaa valmistella menettely etukäteen, ei jälkikäteen. Muodolliset menettelyt, joissa käytetään sinettivahalla (lacre) suljettuja kirjekuoria, ohjeita testamentin toimeenpanijalle ja talletusta notaarin luokse, ovat klassisia järjestelyjä, jotka ovat täysin yhteensopivia omaisuuserän kryptografisen luonteen kanssa.

Tämä artikkeli päättää käsitteellisen trion, joka aloitti syklin — hash, salaus, identiteetti —. Nämä kolme ideaa rakentuvat toistensa päälle: hash antaa muuttumattoman sormenjäljen, salaus antaa luottamuksellisuuden ilman luotettua kolmatta osapuolta, identiteetti antaa tekijyyden ilman lupaa myöntävää kolmatta osapuolta. Kaikilla

kolmella on ominaisuus, joka ei ole myöskään ideologinen: ne siirtävät palvelun hallinnoijalta sen käyttäjälle teknisiä kykyjä, jotka perinteisesti kuuluivat operaattorille. Niiden mukana siirtyy myös vastuita. Rehellinen puhuminen mistä tahansa näistä kolmesta edellyttää puhumista myös kahdesta muusta.

Lähteet ja lisälukemista

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, Bitcoin-parannusehdotus vuodelta 2013. Kryptoteollisuuden palautuslausekkeiden de facto -standardi.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), mukaan lukien Ed25519. IETF, tammikuu 2017. Normatiivinen spesifikaatio allekirjoitusmenetelmästä, jota käytetään suuressa osassa nykyteollisuutta.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, versio 2.0. IETF, syyskuu 2000. Määrittelee PBKDF2-algoritmin, jota käytetään BIP39-johtamisessa lauseesta siemeneksi (seed).
- Asetus (EU) 910/2014 (eIDAS) ja sen kehitys asetuksella (EU) 2024/1183 (eIDAS 2) — eurooppalainen sähköisen tunnistamisen ja pätevän allekirjoituksen kehys. Eri järjestelmä kuin itsenäinen, mutta rakentuu käsitteellisesti samojen kryptografisten primitiivien varaan.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Kanoninen teksti itsenäisen mallin periaatteista ja sitoumuksista, aiempi mutta merkityksellinen nykyisten ratkaisujen ymmärtämiseksi.

← EdellinenLiiketoimintamalli luottamussignaalinaSeuraava →Self-hosting ammatillisena käytäntönä

Viimeaikaiset lukemiset

- Pohdintaa · 29. kesäkuuta 2026 Et ole anonyymi
- Pohdintaa · 27. toukokuuta 2026 Mitä allekirjoitus ei voi korjata
- Analyysi · 26. toukokuuta 2026 Todellinen vs. näennäinen yksityisyys: kysymykset, jotka kannattaa kysyä

Ota tämä artikkeli mukaasi minne tarvitset.

↓ Markdown ↓ Pelkkä teksti ↓ PDF

Tiedosto ladataan laitteellesi. Voit tallentaa sen, tuoda sen Solo2-sovellukseen tai jakaa sen haluamallasi tavalla. Cuadernos ei päätä tiedoston kohtaloa puolestasi.

Cuadernos Lacre · Menzuri Gestión S.L. -julkaisu ·
kirjoittanut R.Eugenio · toimittanut Solo2-tiimi.