De 24 ordene: hva en kryptografisk identitet er

En kryptografisk identitet er ikke et passord: ingen server lagrer den, og den kan ikke gjenopprettes. En didaktisk forklaring på BIP39-mekanismen, hvorfor akkurat tjuefire ord, og hvilken reell vekt som hviler på den som besitter dem.

For å forstå hverandre: Hvis du glemmer Gmail-passordet ditt, tilbakestiller Google det for deg. Hvis du mister de 24 ordene som utgjør en kryptografisk identitet, er det ingen å be om å få dem tilbake. Det er ikke det at prosedyren er streng – det er at det ikke finnes noen i den andre enden. Den forskjellen er hele forskjellen.

Forskjellen mellom et passord og en identitet

Et passord i den klassiske internettmodellen er ikke brukerens identitet. Det er et bevis. Brukeren har en identitet – et navn, en e-post, et kundenummer – og for å bevise overfor en server at man er den man utgir seg for å være, presenterer man et passord som serveren sammenligner med et lagret avtrykk. Hvis avtrykkene stemmer overens, gir serveren tilgang til sesjonen. Hvis passordet går tapt, forblir brukeren den samme brukeren; det man mister er beviset, og det finnes en gjenopprettingsprosedyre – en e-post til den registrerte adressen, et sikkerhetsspørsmål – for å gjenopprette det.

En kryptografisk identitet fungerer på en annen måte. Det er ikke en legitimasjon som noen sammenligner med et lagret avtrykk; det *er* en komplett matematisk hemmelighet i seg selv. Det spiller ingen rolle hvor den befinner seg – på et papir, i en enhet, eller til og med på en fremmed server: identiteten eksisterer i kraft av sin matematikk, ikke på grunn av hvem som validerer den. Her dukker det opp en egenskap som ligner den vi så i «Hva SHA-256 egentlig er»: besittelse bevises ikke ved å vise frem hemmeligheten, men ved å bruke den til å signere. Signaturen som produseres på denne måten kan kontrolleres av hvem som helst med en offentlig verdi som er matematisk utledet fra selve hemmeligheten, uten behov for å kjenne hemmeligheten selv, og uten at en tredjepart mekler i kontrollen. Den som har hemmeligheten, er identiteten; den som mister den, opphører å være det. Dommen er kategorisk: **det finnes ingen å be om å gi deg identiteten tilbake. Den personen eksisterer ikke, for vedkommende hadde den ikke i utgangspunktet.**

Hva tjuefire ord representerer

Den kryptografiske identiteten representeres vanligvis av en matematisk hemmelighet på trettito bytes – to hundre og femtiseks bits. Et tall som er vanskelig å huske og enda vanskeligere å skrive av uten feil. Kryptobransjen løste dette problemet i 2013 med en liten og elegant standard kalt BIP39: en måte å representere disse to hundre og femtiseks bitsene som en sekvens av tjuefire ord hentet fra en offisiell liste på to tusen og førtiåtte. Aritmetikken bak passer elegant; de som vil se den i detalj, finner den i margen.

Opptellingen starter bakfra. Vi ønsker å representere de to hundre og femtiseks bitene i hemmeligheten ved å legge til åtte bits sjekksum: totalt to hundre og sekstifire bits. Hvis vi fordeler dem på tjuefire ord – et håndterbart antall for å notere og diktere uten tap – må hvert ord bidra med nøyaktig elleve bits informasjon. Og elleve bits er to i ellevte potens muligheter, det vil si to tusen og førtiåtte. Derfor har det offisielle BIP39-vokabularet akkurat den størrelsen: listen eksisterer tilpasset problemet, ikke omvendt.

Opptellingen er ikke dekorativ. Hvis noen skriver av tjuetre ord riktig og gjør en feil i det tjuefjerde, vil sjekksummen oppdage det: programvaren vil si «denne sekvensen er ikke gyldig». Hvis noen skriver av alle tjuefire riktig, vil programvaren utlede den samme identiteten uten tvil. Valget av ordlisten er også bevisst: ordene i BIP39-vokabularet er korte, forskjellige fra hverandre, uten diakritiske tegn, valgt for å minimere fonetiske og ortografiske forvekslinger. Det er et vokabular designet for å bli husket, skrevet og diktert av mennesker uten tap.

Fra frase til nøkkel

De tjuefire ordene er ikke den kryptografiske nøkkelen som signerer meldinger. De er en gjenopprettbar representasjon av den opprinnelige entropien som, gjennom en deterministisk prosess kalt PBKDF2, transformeres til et seed på sekstifire byte. Fra dette seedet utledes, også deterministisk, de konkrete kryptografiske nøklene som brukeren benytter: en privat nøkkel for å signere og en tilsvarende offentlig nøkkel som publiseres for å verifisere signaturene. Samme mekanisme i ulike systemer: kryptovalutaer bruker secp256k1-kurven; Signal-protokollen og mange moderne systemer bruker Ed25519 på Curve25519-kurven. For en spesifikk kurve som Ed25519 tar standardene BIP32 og SLIP-0010 dette seedet på sekstifire byte og utleder deterministisk de trettito bytene som utgjør den effektive signeringsnøkkelen — de samme trettito bytene som kodeeksemplet i neste avsnitt starter med.

Dette er standardmåten hele bransjen presenterer mekanismen for brukeren på —kryptovaluta-lommebøker, desentraliserte identitetsbehandlere, Signal i sin persistente identitetsdel, Solo2 blant dem—: brukeren ser i praksis aldri seedet eller de utledede nøklene. Han ser de tjuefire ordene når han oppretter sin identitet, og noterer dem eventuelt på et papir. Ordene reiser deretter mellom enhetene hans når han ønsker å migrere identiteten: han skriver dem inn i den nye applikasjonen, applikasjonen utleder det samme seedet, de samme nøklene, den samme identiteten. Det er en bærbar, kryptografisk solid og, innen rimelighetens grenser, huskbar mekanisme.

Hvordan man signerer med nøkkelen (et penselstrøk i Zig)

I Zig, når man har seedet på trettito byte utledet fra de tjuefire ordene, kan signering av en melding med Ed25519 gjøres på få linjer:

```
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);
```

Signeringshandlingen produserer sekstifire byte —kalt en signatur— som kun kunne ha blitt generert ut fra den tilsvarende private nøkkelen. Verifiseringen er offentlig: alle med den offentlige nøkkelen kan kontrollere at signaturen tilsvarer meldingen. Uten den private nøkkelen kan ingen produsere en gyldig signatur for den meldingen; med den offentlige nøkkelen kan alle oppdage om en signatur er gyldig. Denne asymmetrien er det som gjør det mulig for underskriveren å bevise opphav uten å dele hemmeligheten.

Det forrige eksemplet er den minimale manualversjonen. I den virkelige Solo2-koden går kjeden gjennom to filer, en i JavaScript som lever i brukerens nettleser og rekonstruerer entropien fra de tjuefire ordene, en annen i

Zig i *zcatcrypto*-biblioteket som tar denne entropien og utleder de konkrete kryptografiske nøklene. Vi starter fra nettlesersiden:

```
// 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 };
}
```

De trettito bytene med entropi, sammen med ytterligere trettito utledet i samme trinn, går til Zigs WebAssembly-modul som genererer selve Ed25519-nøklene. Den komplette funksjonen, med sin endelige minneopprydding, får plass på én skjerm:

```
// 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;
}
```

To detaljer er verdt å merke seg. For det første: samme frø (seed) produserer alltid det samme nøkkelparet — det er nettopp dette som gjør det mulig å gjenopprette identiteten ved å skrive inn de tjuefire ordene på en ny enhet. For det andre: frøet slettes eksplisitt fra minnet i den siste linjen. Etter det punktet ville ikke engang selve funksjonen kunne rekonstruere nøklene; brukerens ord ville være den eneste kilden.

For de som vil sjekke det med små tall. Signaturskjemaet kan gjennomgås i sin helhet med tall som er små nok til å gjøre utregningene for hånd. De som foretrekker å ikke gå inn i aritmetikk kan hoppe over denne blokken uten å miste tråden i artikkelen; de som vil se mekanismen fungere trinn for trinn finner den her. **De offentlige reglene**, som alle kan lese: et primtall $p = 23$ (i ekte Ed25519 er det på rundt syttisju siffer; vi bruker tjuetre slik at utregningene får plass på én side), en base $g = 2$ hvis orden i denne gruppen er $q = 11$, og konvensjonen om at all aritmetikk med g gjøres *módulo p* og alle eksponenter reduseres *módulo q*. **Det private valget**, ett enkelt og aldri delt: hemmeligheten $x = 6$. Det er identiteten.

Trinn 1 — Den offentlige delen av identiteten. Den beregnes én gang og publiseres åpent.

$y = g^x \bmod p$

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

Den offentlige delen av identiteten er **18**. Alle kan ta den og bruke den til å verifisere signaturer laget med denne identiteten. Ingen kan, ved å bare observere 18, gjenopprette hemmeligheten 6: det er det diskrete logaritmeproblemet som vi vil komme tilbake til til slutt.

Trinn 2 — Signere en melding. Innehaveren av identiteten vil signere meldingen $m = 7$. Han starter med å velge en ny tilfeldig verdi $k = 4$, som vil bli brukt bare én gang og aldri delt (i ekte Ed25519 utledes k deterministisk fra meldingen og hemmeligheten for å unngå faren for gjenbruk, men rollen den spiller er akkurat denne). Deretter beregner han tre tall:

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

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

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

Signaturen er paret **(r, s) = (16, 10)**. Den reiser åpent sammen med meldingen. Alle kan lese den. Didaktisk note: i ekte Ed25519 er funksjonen H SHA-512, som er kryptografisk robust; her bruker vi forenklingen $e = (r + m) \bmod q$ slik at leseren kan gå gjennom trinnene uten å måtte beregne en hash. Strukturen i algoritmen er den samme.

Trinn 3 — Verifisere signaturen. Verifikatoren har den offentlige delen $y = 18$, meldingen $m = 7$, og signaturen $(r, s) = (16, 10)$. Han rekonstruerer e på samme måte — $e = (16 + 7) \bmod 11 = 1$ — og sjekker om denne likheten er oppfylt:

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

Beregner de to sidene hver for seg:

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

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

Begge sidene gir **12**. Signaturen er gyldig. Alle med den offentlige delen 18 kan komme til denne konklusjonen uten noen gang å ha visst at hemmeligheten var 6.

Hva med en tredjepart som prøver å forfalske? Eva har sett alt det offentlige passere gjennom kanalen: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. For å signere en *annen* melding i denne identitetens navn, ville hun trenge å kjenne x. Hennes eneste vei er å spørre seg selv: «for hvilken eksponent x er $2^x \bmod 23 = 18$ oppfylt?». Med $p = 23$ kan hun prøve 0, 1, 2, 3, … og finne det på sekunder. Men ved å erstatte 23 med et primtall av Ed25519s reelle dimensjoner, overstiger rommet av mulige eksponenter antall atomer i det observerbare universet. **Det finnes i dag ingen algoritme kjent for menneskeheten som kan gjennomløpe det rommet på mindre enn milliarder av år.** Det er det samme diskrete logaritmeproblemet som understøtter Diffie-Hellman fra den forrige artikkelen, anvendt her på signaturskjemaet.

Det vi nettopp har gått gjennom er *nøyaktig* Schnorr, signaturskjemaet som Ed25519 er en variant av, tilpasset en elliptisk kurve. I ekte Ed25519 gjøres alle operasjoner på punktene til en konkret kurve (Curve25519) i stedet for på heltall modulo et primtall, og funksjonen H er SHA-512 i stedet for den leketøyssummen vi brukte over. De to utskiftningene er implementeringsjusteringer — oppnå kryptografisk motstand mot brute force, oppnå ytterligere sikkerhetsegenskaper for k —. Den algoritmiske strukturen, de tre operasjonene, årsaken til asymmetrien, er de samme.

Det er på sin plass med et kort opphold her, fordi hele kjeden ved et raskt øyekast kan forveksles med en annen primitiv i trioen: hashen. Det er det ikke. En hash er en unik funksjon som komprimerer — mange byte går inn, et kort avtrykk kommer ut, der slutter veien. En kryptografisk identitet er et komplementært matematisk par: hemmeligheten blir igjen og signerer; dens offentlige motpart publiseres og verifiserer. Der hashen kollapser informasjon i én retning, etablerer identiteten en asymmetri mellom to halvdeler. Hashen vitner om hva som ble sagt; identiteten vitner om hvem som sa det.

Hva frasen ikke er

Tre hyppige misforståelser bør ryddes av veien. Frasen er ikke et passord i egentlig forstand: den sammenlignes ikke med et fingeravtrykk lagret på en server; den tastes inn på brukerens enhet for å rekonstruere identiteten matematisk. Frasen gjenopprettes ikke: hvis den mistes, er det ingen å be om den; hvis den dupliseres, dupliseres også identiteten. Frasen er ikke en legitimasjon som kan skilles fra identiteten: frasen *er* identiteten. Den som har den, kan agere som den identiteten, uten ytterligere tillatelse, uten autorisasjonsprosess, uten mulighet for gjenoppretting.

Nettopp denne tredje egenskapen er det som endrer sakens vekt. Et tapt passord er en administrativ plage. En tapt kryptografisk identitet er identiteten. Et papir med frasen funnet av tredjeparter er ikke en risiko for kontotyveri: det er overlevering av hele identiteten. Systemets løfte — at ingen kan tilbakekalle identiteten din eller blokkere deg vilkårlig — ledsages uadskillelig av ansvaret — at du er den eneste vokteren av noe som ingen kan gjenopprette for deg.

Løftet og vekten

Modellen for kryptografisk identitet får ofte betegnelsen *selvsuveren* —self-sovereign i den angelsaksiske litteraturen—. Valget av ord er bevisst og beskriver tilstanden ganske nøyaktig. Brukeren er suveren over sin identitet i en nesten middelaldersk forstand: den tildeles ikke av noen konge, noen utsteder eller noen sentral myndighet; ei heller kan den trekkes tilbake av noen av de foregående. Men i likhet med den middelalderske monarken bærer brukeren også den fulle konsekvensen av sine feil: det finnes ingen regent som tar avgjørelser i sitt sted hvis han mister seglet.

Valget mellom identitet administrert av en tredjepart og selvsuveren identitet har ikke ett universelt riktig svar. For en irrelevant forumkonto er administrert identitet sannsynligvis proporsjonal med risikoen. For en profesjonell identitet som signerer juridisk bindende dokumenter, for en økonomisk identitet som vokter egne

sparepenger, for en profesjonell kommunikasjonsidentitet med klienter som har betrodd sensitiv informasjon, endrer saken seg. Der slutter spørsmålet å være «er det praktisk?» og blir til «hvem, utenom meg selv, har makt til å agere som meg, og under hvilke omstendigheter?».

Hvor denne mekanismen opptrer i virkelige systemer

BIP39 ble født i Bitcoin-verdenen i 2013 og spredte sich raskt til hele kryptovaluta-økosystemet: enhver seriøs lommebok godtar i dag en BIP39-frase på tolv eller tjuefire ord som backup av innehaverens økonomiske identitet. Utenfor kryptovalutaer opptrer det samme underliggende konseptet — et kryptografisk par som beviser opphav uten en mellommann — i andre systemer med forskjellig syntaks. SSH-nøkler som en systemadministrator bruker for å få tilgang til sine servere er et klassisk eksempel: en privat nøkkel som administratoren oppbevarer på sin maskin og en offentlig som kopieres til hver server; ingen enhet som kan sammenlignes med en sentralisert tjeneste griper inn. Signal-protokollen bruker Ed25519 med persistent nøkkelmateriale på enheten; det europeiske eIDAS hviler, i sin del om kvalifisert signatur, på det samme kryptografiske prinsippet, med den forskjell at nøkkelen oppbevares av en kvalifisert tillitstjenesteyter i stedet for brukeren.

Solo2, utgiverplattformen for denne publikasjonen, bruker en BIP39-frase på tjuefire ord som hver brukers identitet. Brukeren ser ordene én gang når kontoen opprettes. De lagres ikke på noen Solo2-server eller hos noen andre: hvis brukeren noterer dem og passer på dem, beholder de sin identitet for alltid. Hvis de mister dem, mister de dem. Det er den logiske konsekvensen av en arkitektur uten en operatør i midten: hvis Solo2 kunne gi identiteten tilbake til brukeren som mistet den, kunne den også gi den til hvem som helst som presser Solo2 for å få den.

For den profesjonelle leseren

Fire overveielser for de som vurderer å adoptere en kryptografisk selvsuveren (autosoberana) identitet i en profesjonell sammenheng:

1. Frasen er identiteten. Fysisk oppbevaring — papir, flere kopier på forskjellige steder, eventuelt inngravert metall for langvarig bruk — gir flere garantier enn digital oppbevaring, som øker angrepsflaten uten å redusere risikoen for tap.
2. Det er ingen gjenoppretting. Å designe prosessen under forutsetning av at den primære kopien en dag går tapt, er mye klogere enn å oppdage det den dagen den går tapt. En annen geografisk adskilt kopi løser nesten alle scenarier.
3. Det er ikke det samme som et eIDAS-kvalifisert sertifikat. For kvalifisert signatur i Unionen — notarialforretninger, visse prosedyrer med forvaltningen — krever lovgivningen en kvalifisert tilbyder som oppbevarer nøkkelen. Kryptografisk selvsuveren identitet tjener til profesjonell kommunikasjon og dokumentunderskrift med bevisverdi, men erstatter ikke automatisk det kvalifiserte sertifikatet i de tilfellene hvor regelen krever det.
4. Hvis identiteten skal overføres — arv, profesjonell suksesjon, opphør av aktivitet — er det tilrådelig å forberede prosedyren før, ikke etter. Formelle prosedyrer med konvolutter forseglet med lakk (lacre), instruksjoner til en bobestyrer, deponering hos en notar, er klassiske ordninger som er fullt kompatible med aktivets kryptografiske natur.

Denne artikkelen avslutter den konseptuelle trioen som åpnet syklusen — hash, kryptering, identitet —. De tre idéene bygger på hverandre: hash gir det uforanderlige fingeravtrykket, kryptering gir konfidensialitet uten en betrodd tredjepart, identitet gir opphav uten en bevilgende tredjepart. De tre deler en egenskap som heller ikke er ideologisk: de overfører, fra den som administrerer en tjeneste til den som bruker den, tekniske egenskaper som tradisjonelt lå hos operatøren. De overfører også ansvar med dem. Å snakke ærlig om enhver av de tre krever også å snakke om de to andre.

Kilder og videre lesing

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, Bitcoin-forbedringsforslag fra 2013. De facto-standard for gjenopprettingsfraser i kryptoindustrien.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), inkludert Ed25519. IETF, januar 2017. Normativ spesifikasjon av signaturskjemaet som brukes i store deler av den moderne industrien.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, versjon 2.0. IETF, september 2000. Definerer PBKDF2-algoritmen som brukes i BIP39-avledning fra frase til seed.
- Forordning (EU) 910/2014 (eIDAS) og dens utvikling gjennom forordning (EU) 2024/1183 (eIDAS 2) — europeisk rammeverk for elektronisk identitet og kvalifisert signatur. Et annet regime enn det selvsuverene, men konseptuelt støttet av de samme kryptografiske primitivene.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Kanonisk tekst om prinsippene og forpliktelsene i den selvsuverene modellen, tidligere men relevant for forståelsen av familien av nåtidige løsninger.

← ForrigeForretningsmodellen som et signal om tillitNeste →Self-hosting som profesjonell praksis

Siste lesninger

- Refleksjon · 29. juni 2026 Du er ikke anonym
- Refleksjon · 27. mai 2026 Det en signatur ikke kan fikse
- Analyse · 26. mai 2026 Reelt vs. tilsynelatende personvern: Spørsmålene man bør stille seg selv

Ta med deg denne artikkelen dit du trenger den.

↓ Markdown ↓ Klartekst ↓ PDF

Filen lastes ned til enheten din. Derfra kan du lagre den, importere den til Solo2 eller dele den hvor du vil. Cuadernos bestemmer ikke destinasjonen for deg.

Lakksegl · SHA-256 91bb474f99e5cbf15d69f048dd0be73575505e5c2c1f0ff3cf872a0adbfa77ad

Cuadernos Lacre · En utgivelse fra Menzuri Gestión S.L. ·
skrevet av R.Eugenio · redigert av teamet bak Solo2.

Dette nettstedet bruker ikke informasjonskapsler. Alt som nettleseren din laster inn, er skrevet eller overvåket av oss og plassert på våre europeiske servere: den anonyme besøkstelleren (Umami, selvhostet) og det minimale JavaScript som kreves for språkvelgeren og innstillingen din for lyst eller mørkt tema, som lagres på din egen enhet. Ingen ressurser fra eksterne selskaper, ingen trackere, ingen profilering, ingen deling av data. Hvis du vil følge oss: RSS.