De 24 ord: hvad er en kryptografisk identitet

En kryptografisk identitet er ikke en adgangskode: ingen server gemmer den, og den kan ikke gendannes. En didaktisk forklaring på BIP39-mekanismen, hvorfor præcis fireogtyve ord, og hvilken reel vægt der hviler på den, der besidder dem.

For at forstå hinanden: Hvis du glemmer din adgangskode til Gmail, nulstiller Google den for dig. Hvis du mister de 24 ord, der udgør en kryptografisk identitet, er der ingen at bede om dem. Det er ikke fordi proceduren er streng – det er fordi der ikke findes nogen i den anden ende. Den forskel er hele forskellen.

Forskellen mellem en adgangskode og en identitet

En adgangskode i den klassiske internetmodel er ikke brugerens identitet. Det er et bevis. Brugeren har en identitet – et navn, en e-mail, et kundenummer – og for at bevise over for en server, at man er den, man udgiver sig for at være, præsenterer man en adgangskode, som serveren sammenligner med et gemt aftryk. Hvis aftrykkene stemmer overens, giver serveren adgang til sessionen. Hvis adgangskoden mistes, forbliver brugeren den samme bruger; det, man mister, er beviset, og der findes en gendannelsesprocedure – en e-mail til den registrerede adresse, et sikkerhedsspørgsmål – til at genoprette det.

En kryptografisk identitet fungerer på en anden måde. Det er ikke en legitimationsoplysning, som nogen sammenligner med et gemt aftryk; det *er* en komplet matematisk hemmelighed i sig selv. Det er ligegyldigt, hvor den befinder sig – på et papir, i en enhed eller endda på en fremmed server: identiteten eksisterer i kraft af sin matematik, ikke på grund af hvem der validerer den. Her dukker en egenskab op, der ligner den, vi så i «Hvad SHA-256 egentlig er»: besiddelse bevises ikke ved at fremvise hemmeligheden, men ved at bruge den til at signere. Den således frembragte signatur kan enhver kontrollere med en offentlig værdi, der er matematisk afledt af selve hemmeligheden, uden behov for at kende hemmeligheden selv og uden at en tredjepart mægler i kontrollen. Den, der har hemmeligheden, er identiteten; den, der mister den, ophører med at være det. Dommen is kategorisk: **der er ingen at bede om at give dig identiteten tilbage. Den person findes ikke, for vedkommende havde den ikke i første omgang.**

Hvad fireogtyve ord repræsenterer

Den kryptografiske identitet repræsenteres normalt af en matematisk hemmelighed på toogtredive bytes – to hundrede og seksoghalvtreds bits. Et tal, der er svært at huske og endnu sværere at skrive af uden fejl. Den kryptografiske industri løste dette problem i 2013 med en lille og elegant standard kaldet BIP39: en måde at repræsentere disse to hundrede og seksoghalvtreds bits som en sekvens af fireogtyve ord taget fra en officiel liste på to tusind og otteogfyrre. Aritmetikken bag passer elegant; de, der vil se den i detaljer, finder den i margenen.

Optællingen starter bagfra. Vi ønsker at repræsentere de to hundrede og seksoghalvtreds bits af hemmeligheden ved at tilføje otte bits tjeksum: i alt to hundrede og fireogtres bits. Hvis vi fordeler dem på fireogtyve ord – et overkommeligt antal at notere og diktere uden tab – skal hvert ord bidrage med præcis elleve bits information. Og elleve bits er to i ellevte potens muligheder, det vil sige to tusind og otteogfyrre. Derfor har det officielle BIP39-ordforråd præcis den størrelse: listen findes tilpasset problemet, ikke omvendt.

Optællingen er ikke dekorativ. Hvis nogen skriver treogtyve ord korrekt og laver en fejl i det fireogtyvende, vil tjeksummen opdage det: softwaren vil sige «denne sekvens er ikke gyldig». Hvis nogen skriver alle fireogtyve korrekt, vil softwaren aflede den samme identitet uden tvetydighed. Valget af ordlisten er også bevidst: ordene i BIP39-ordforrådet er korte, forskellige fra hinanden, uden diakritiske tegn, valgt for at minimere fonetiske og ortografiske forvekslinger. Det er et ordforråd designet til at blive husket, skrevet og dikteret af mennesker uden tab.

Fra sætning til nøgle

De fireogtyve ord er ikke den kryptografiske nøgle, der signerer beskeder. De er en genoprettelig repræsentation af den oprindelige entropi, som gennem en deterministisk proces kaldet PBKDF2 transformeres til et seed på fireogtres byte. Fra dette seed udledes, også deterministisk, de specifikke kryptografiske nøgler, som brugeren anvender: en privat nøgle til at signere og en tilsvarende offentlig nøgle, der offentliggøres for at verificere signaturerne. Samme mekanisme i forskellige systemer: kryptovalutaer bruger secp256k1-kurven; Signal-protokollen og mange moderne systemer bruger Ed25519 på Curve25519-kurven. For en specifik kurve som Ed25519 tager standarderne BIP32 og SLIP-0010 dette seed på fireogtres byte og udleder deterministisk de toogtredive byte, der udgør den effektive signaturnøgle — de samme toogtredive byte, som kodeeksemplet i næste afsnit starter med.

Dette er standardmåden, hvorpå hele branchen præsenterer mekanismen for brugeren —kryptovaluta-wallets, decentraliserede identitetsadministratorer, Signal i dens persistente identitetsdel, Solo2 blandt dem—: brugeren ser i praksis aldrig seedet eller de udledte nøgler. Han ser de fireogtyve ord, når han opretter sin identitet, og noterer dem eventuelt på et stykke papir. Ordene rejser derefter mellem hans enheder, når han ønsker at migrere identiteten: han indtaster dem i den nye applikation, applikationen udleder det samme seed, de samme nøgler, den samme identitet. Det er en bærbar, kryptografisk solid og, inden for rimelighedens grænser, huskbar mekanisme.

Hvordan man signerer med nøglen (et strejf af Zig)

I Zig, når man har seedet på toogtredive byte udledt fra de fireogtyve ord, kan signering af en besked med Ed25519 gøres 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 producerer fireogtres byte —kaldet en signatur— som kun kunne være genereret ud fra den tilsvarende private nøgle. Verificeringen er offentlig: enhver med den offentlige nøgle kan kontrollere, at signaturen svarer til beskeden. Uden den private nøgle kan ingen producere en gyldig signatur til den besked; med den offentlige nøgle kan alle opdage, om en signatur er gyldig. Denne asymmetri er det, der gør det muligt for underskriveren at bevise forfatterskab uden at dele hemmeligheden.

Det foregående eksempel er den minimale manualversion. I Solo2's rigtige kode løber kæden gennem to filer, en i JavaScript, der lever i brugerens browser og rekonstruerer entropien ud fra de fireogtyve ord, en anden i Zig i

zcatcrypto-biblioteket, der tager denne entropi og afleder de konkrete kryptografiske nøgler. Startende fra browsersiden:

```
// 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 toogtredive bytes entropi, sammen med yderligere toogtredive afledt i samme trin, rejser til Zigs WebAssembly-modul, som genererer de faktiske Ed25519-nøgler. Den fulde funktion, med dens endelige hukommelsesrensning, fylder én skærm:

```
// 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 værd at bemærke. Den første: det samme seed producerer altid det samme nøglepar — det er præcis det, der gør det muligt at gendanne identiteten ved at indtaste de fireogtyve ord på en ny enhed. Den anden: seedet slettes eksplicit fra hukommelsen på den sidste linje. Efter dette punkt ville selv funktionen selv ikke kunne rekonstruere nøglerne; brugerens ord ville være den eneste kilde.

For dem, der vil tjekke det med små tal. Signaturskemaet kan gennemgås helt med tal, der er små nok til at lave beregningerne manuelt. De, der foretrækker ikke at gå ind i aritmetik, kan springe denne blok over uden at miste tråden i artiklen; de, der vil se mekanismen fungere trin for trin, finder den her. **De offentlige regler**, som alle kan læse: et primtal $p = 23$ (i rigtig Ed25519 er det på omkring syvoghalvfjerds cifre; vi bruger treogtyve, så beregningerne kan være på én side), en base $g = 2$, hvis orden i denne gruppe er $q = 11$, og konventionen om, at al aritmetik med g udføres *módulo p*, og alle eksponenter reduceres *módulo q*. **Det private valg**, ét enkelt og aldrig delt: hemmeligheden $x = 6$. Det er identiteten.

Trin 1 — Den offentlige del af identiteten. Den beregnes én gang og offentliggøres åbent.

$y = g^x \bmod p$

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

Den offentlige del af identiteten er **18**. Enhver kan tage den og bruge den til at verificere signaturer lavet med denne identitet. Ingen, der kun observerer 18, kan gendanne hemmeligheden 6: det er det diskrete logaritmeproblem, som vi vender tilbage til til sidst.

Trin 2 — At signere en besked. Indehaveren af identiteten vil signere beskeden $m = 7$. Han starter med at vælge en ny tilfældig værdi $k = 4$, som kun bruges én gang og aldrig deles (i rigtig Ed25519 afledes k deterministisk fra beskeden og hemmeligheden for at undgå faren ved genbrug, men den rolle, den spiller, er præcis denne). Derefter beregner han tre tal:

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

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

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

Signaturen er parret **(r, s) = (16, 10)**. Den rejser åbent sammen med beskeden. Enhver kan læse den. Didaktisk note: I rigtig Ed25519 er funktionen H SHA-512, som er kryptografisk robust; her bruger vi forenklingen $e = (r + m) \bmod q$, så læseren kan gennemgå trinene uden at skulle beregne en hash. Algoritmens struktur er den samme.

Trin 3 — Verificering af signaturen. Verifikatoren har den offentlige del $y = 18$, beskeden $m = 7$ og signaturen $(r, s) = (16, 10)$. Han rekonstruerer e på samme måde — $e = (16 + 7) \bmod 11 = 1$ — og tjekker, om denne lighed er opfyldt:

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

Beregn de to sider hver for sig:

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

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

De to sider giver **12**. Signaturen er gyldig. Enhver med den offentlige del 18 kan nå frem til denne konklusion uden nogensinde at have vidst, at hemmeligheden var 6.

Hvad med en tredjepart, der forsøger at forfalske? Eva har set alt det offentlige passere gennem kanalen: p = 23, g = 2, q = 11, y = 18, m = 7, r = 16, s = 10. For at signere en *anden* besked i denne identitets navn, skal hun kende x. Hendes eneste vej er at spørge sig selv: "for hvilken eksponent x er $2^x \bmod 23 = 18$?". Med p = 23 kan hun prøve 0, 1, 2, 3, ... og finde det på få sekunder. Men ved at erstatte 23 med et primtal af Ed25519's reelle dimensioner overstiger rummet af mulige eksponenter antallet af atomer i det observerbare univers. **Der findes i dag ingen algoritme kendt af menneskeheden, der kan gennemløbe det rum på mindre end milliarder af år.** Det er det samme diskrete logaritmeproblem, som understøtter Diffie-Hellman fra den forrige artikel, anvendt her på signaturskemaet.

Det, vi lige har gennemgået, er *præcis* Schnorr, det signaturskema, som Ed25519 er en variant af, tilpasset til en elliptisk kurve. I rigtig Ed25519 udføres alle operationer på punkter på en specifik kurve (Curve25519) i stedet for på heltal modulo et primtal, og funktionen H er SHA-512 i stedet for den legetøjssum, vi brugte ovenfor. De to udskiftninger er implementeringsjusteringer — for at opnå kryptografisk modstandsdygtighed over for brute force og opnå yderligere sikkerhedsegenskaber for k. Den algoritmiske struktur, de tre operationer og årsagen til asymmetrien er de samme.

Et kort ophold er på sin plads her, fordi hele kæden ved et hurtigt blik kan forveksles med en anden primitiv i trioen: hashen. Det er det ikke. En hash er en unik funktion, der komprimerer — mange bytes kommer ind, et kort aftryk kommer ud, og der slutter vejen. En kryptografisk identitat er et komplementært matematisk par: hemmeligheden bliver og signerer; dens offentlige modpart offentliggøres og verificerer. Hvor hashen kollapser information i én retning, etablerer identiteten en asymmetri mellem to halvdele. Hashen vidner om, hvad der blev sagt; identiteten vidner om, hvem der sagde det.

Hvad sætningen ikke er

Tre hyppige misforståelser bør ryddes af vejen. Sætningen er ikke en adgangskode i egentlig forstand: den sammenlignes ikke med et fingeraftryk gemt på en server; den indtastes på brugerens enhed for matematisk at rekonstruere identiteten. Sætningen genoprettes ikke: hvis den mistes, er der ingen at bede om den; hvis den duplikeres, duplikeres identiteten også. Sætningen er ikke en legitimation, der kan adskilles fra identiteten: sætningen *er* identiteten. Den, der har den, kan agere som den, uden yderligere tilladelse, uden autorisationsproces, uden mulighed for genoprettelse.

Netop denne tredje egenskab er det, der ændrer sagens vægt. En mistet adgangskode er en administrativ gene. En mistet kryptografisk identitet er identiteten. Et papir med sætningen fundet af tredjeparter er ikke en risiko for kontotyveri: det er udlevering af hele identiteten. Systemets løfte — at ingen kan tilbagekalde din identitet eller blokere dig vilkårligt — ledsages uadskilleligt af ansvaret — at du er den eneste vogter af noget, som ingen kan genoprette for dig.

Løftet og vægten

Modellen for kryptografisk identitet får ofte betegnelsen *selvsuveræn* —self-sovereign i den angelsaksiske litteratur—. Valget af ord er bevidst og beskriver tilstanden ret præcist. Brugeren er suveræn over sin identitet i en næsten middelalderlig forstand: den tildeles ikke af nogen konge, nogen udsteder, nogen central myndighed; ej heller kan den trækkes tilbage af nogen af de foregående. Men ligesom den middelalderlige monark bærer brugeren også den fulde konsekvens af sine fejl: der er ingen regent til at træffe beslutninger i sit sted, hvis han mister seglet.

Valget mellem identitet administreret af en tredjepart og selvsuveræn identitet har ikke ét universelt rigtigt svar. For en irrelevant forumkonto er administreret identitet sandsynligvis proportional med risikoen. For en professionel identitet, der signerer juridisk bindende dokumenter, for en økonomisk identitet, der bevogter egne

opsparinger, for en professionel kommunikationsidentitet med klienter, der har betroet følsomme oplysninger, ændrer spørgsmålet sig. Der ophører spørgsmålet med at være «er det bekvemt?» og bliver til «hvem, udover mig selv, har magten til at agere som mig, og under hvilke omstændigheder?».

Hvor denne mekanisme optræder i virkelige systemer

BIP39 blev født i Bitcoin-verdenen i 2013 og spredte sig hurtigt til hele kryptovaluta-økosystemet: enhver seriøs wallet accepterer i dag en BIP39-frase på tolv eller fireogtyve ord som backup af sin indehavers økonomiske identitet. Uden for kryptovalutaer optræder det samme underliggende koncept — et kryptografisk par, der beviser ophav uden en mellemand — i andre systemer med forskellig syntaks. SSH-nøgler, som en systemadministrator bruger til at få adgang til sine servere, er et klassisk eksempel: en privat nøgle, som administratoren gemmer på sin maskine, og en offentlig, der kopieres til hver server; ingen enhed, der kan sammenlignes med en centraliseret tjeneste, griber ind. Signal-protokollen bruger Ed25519 med persistent nøglemateriale på enheden; det europæiske eIDAS hviler, i sin del om kvalificeret signatur, på det samme kryptografiske princip, med den forskel at nøglen opbevares af en kvalificeret tillidstjenesteudbyder i stedet for brugeren.

Solo2, udgiverplatformen for denne publikation, bruger en BIP39-frase på fireogtyve ord som hver brugers identitet. Brugeren ser ordene én gang, når kontoen oprettes. De gemmes ikke på nogen Solo2-server eller hos nogen andre: hvis brugeren noterer dem og passer på dem, bevarer de deres identitet for evigt. Hvis de mister dem, mister de dem. Det er den logiske konsekvens af en arkitektur uden en operatør i midten: hvis Solo2 kunne give identiteten tilbage til den bruger, der mistede den, kunne den også give den til hvem end der presser Solo2 for at få den.

Til den professionelle læser

Fire overvejelser for dem, der overvejer at adoptere en kryptografisk selvsuveræn (autosoberana) identitet i en professionel sammenhæng:

1. Frasen er identiteten. Fysisk opbevaring — papir, flere kopier på forskellige steder, eventuelt indgraveret metal til langvarig brug — giver flere garantier end digital opbevaring, som øger angrebsfladen uden at mindske risikoen for tab.
2. Der er ingen genoprettelse. At designe processen under antagelse af, at den primære kopi en dag går tabt, er meget klogere end at opdage det den dag, den går tabt. En anden geografisk adskilt kopi løser næsten alle scenarier.
3. Det er ikke det samme som et eIDAS-kvalificeret certifikat. For kvalificeret signatur i Unionen — notarialakter, visse procedurer med forvaltningen — kræver lovgivningen en kvalificeret udbyder, der opbevarer nøglen. Kryptografisk selvsuveræn identitet tjener til professionel kommunikation og dokumentunderskrift med bevisværdi, men erstatter ikke automatisk det kvalificerede certifikat i de tilfælde, hvor reglen kræver det.
4. Hvis identiteten skal overføres — arv, professionel succession, ophør af aktivitet — er det tilrådeligt at forberede proceduren før, ikke efter. Formelle procedurer med kuverter forseglet med lak (lacre), instruktioner til en bobestyrer, deponering hos en notar, er klassiske arrangementer, der er fuldt kompatible med aktivets kryptografiske natur.

Denne artikel afslutter den konceptuelle trio, der åbnede cyklussen — hash, kryptering, identitet —. De tre idéer bygger på hinanden: hash giver det uforanderlige fingeraftryk, kryptering giver fortrolighed uden en betroet tredjepart, identitet giver ophav uden en bevilgende tredjepart. De tre deler en egenskab, som heller ikke er ideologisk: de overfører, fra den der administrerer en tjeneste til den der bruger den, tekniske egenskaber, der traditionelt lå hos operatøren. De overfører også ansvar med dem. At tale ærligt om enhver af de tre kræver også at tale om de to andre.

Kilder og yderligere læsning

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, Bitcoin-forbedringsforslag fra 2013. De facto-standard for genoprettelsesfraser i kryptoindustrien.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), herunder Ed25519. IETF, januar 2017. Normativ specifikation af den signaturskema, der anvendes i store dele af den moderne industri.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, version 2.0. IETF, september 2000. Definerer PBKDF2-algoritmen, der anvendes i BIP39-afledning fra frase til seed.
- Forordning (EU) 910/2014 (eIDAS) og dens udvikling gennem forordning (EU) 2024/1183 (eIDAS 2) — europæisk ramme for elektronisk identitet og kvalificeret signatur. Et andet regime end det selvsuveræne, men konceptuelt støttet af de samme kryptografiske primitiver.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Kanonisk tekst om principperne og forpligtelserne i den selvsuveræne model, tidligere men relevant for forståelsen af familien af nutidige løsninger.

← ForrigeForretningsmodellen som et signal om tillidNæste →Self-hosting som professionel praksis

Seneste læsning

- Refleksion · 29. juni 2026 Du er ikke anonym
- Refleksion · 27. maj 2026 Hvad en underskrift ikke kan løse
- Analyse · 26. maj 2026 Reelt vs. tilsyneladende privatliv: De spørgsmål, man bør stille sig selv

Tag denne artikel med dig, hvor du har brug for den.

↓ Markdown ↓ Almindelig tekst ↓ PDF

Filen downloades til din enhed. Derfra kan du gemme den, importere den til Solo2 eller dele den, hvor du vil. Cuadernos beslutter ikke destinationen for dig.

Laksegl · SHA-256 3bdfcd1fc28168f2c71296c991a60b3bb30889d32488120b6b924e0257bc0897

Cuadernos Lacre · En udgivelse fra Menzuri Gestión S.L. ·
skrevet af R.Eugenio · redigeret af holdet bag Solo2.