แนวคิด · 23 พฤษภาคม 2026

← ก่อนหน้า·ถัดไป →

24 คำ: อัตลักษณ์การเข้ารหัสคืออะไร

อัตลักษณ์การเข้ารหัสไม่ใช่รหัสผ่าน: ไม่มีเซิร์ฟเวอร์ใดเก็บไว้และไม่สามารถกู้คืนได้ คำอธิบายเชิงวิชาการเกี่ยวกับกลไก BIP39 ทำไมต้องเป็นยี่สิบสี่คำ และภาระที่แท้จริงที่ตกอยู่กับผู้ที่ครอบครองมันคืออะไร

เพื่อความเข้าใจที่ตรงกัน: หากคุณลืมรหัสผ่าน Gmail ของคุณ Google จะรีเซ็ตให้คุณ หากคุณทำ 24 คำที่ประกอบเป็นอัตลักษณ์การเข้ารหัสหาย จะไม่มีใครให้ขอได้ ไม่ใช่ว่าขั้นตอนนั้นเข้มงวด แต่เป็นเพราะไม่มีใครอยู่อีกฝั่งหนึ่ง ความแตกต่างนั้นคือความแตกต่างทั้งหมด

ความแตกต่างระหว่างรหัสผ่านและอัตลักษณ์

รหัสผ่านในรูปแบบอินเทอร์เน็ตคลาสสิกไม่ใช่อัตลักษณ์ของผู้ใช้ แต่มันคือใบรับรอง ผู้ใช้มีอัตลักษณ์ — ชื่อ อีเมล หมายเลขลูกค้า — และเพื่อพิสูจน์ต่อเซิร์ฟเวอร์ว่าตนเองเป็นใครตามที่กล่าวอ้าง ผู้ใช้จะนำเสนอรหัสผ่านที่เซิร์ฟเวอร์จะนำไปเปรียบเทียบกับรอยเท้าที่เก็บไว้ หากรอยเท้าตรงกัน เซิร์ฟเวอร์จะอนุญาตเซสชัน หากรหัสผ่านสูญหาย ผู้ใช้ยังคงเป็นผู้ใช้คนเดิม สิ่งที่สูญหายไปคือใบรับรอง และมีขั้นตอนการกู้คืน — อีเมลไปยังที่อยู่ที่ลงทะเบียนไว้ คำถามเพื่อความปลอดภัย — เพื่อกู้คืนมันกลับมา

อัตลักษณ์การเข้ารหัสทำงานในลักษณะที่แตกต่างออกไป มันไม่ใช่ข้อมูลประจำตัวที่ใครจะนำไปเปรียบเทียบกับรอยเท้าที่เก็บไว้ แต่มัน *คือ* ความลับทางคณิตศาสตร์ที่สมบูรณ์ในตัวเอง ไม่สำคัญว่ามันจะอยู่ที่ใด — บนกระดาษ ในอุปกรณ์ หรือแม้แต่ในเซิร์ฟเวอร์ของผู้อื่น: อัตลักษณ์นั้นมีอยู่ด้วยคณิตศาสตร์ของมัน ไม่ใช่ด้วยผู้ที่ตรวจสอบมัน ตรงนี้มีคุณสมบัติที่คล้ายกับที่เราเห็นใน «SHA-256 คืออะไรกันแน่»: การครอบครองไม่ได้พิสูจน์ด้วยการแสดงความลับ แต่พิสูจน์ด้วยการใช้มันเพื่อลงลายมือชื่อ ลายมือชื่อที่ผลิตขึ้นในลักษณะนี้ใครๆ ก็สามารถตรวจสอบได้ด้วยค่าสาธารณะที่สืบทอดมาจากความลับนั้นเองตามหลักคณิตศาสตร์ โดยไม่จำเป็นต้องรู้ความลับนั้น และโดยไม่มีบุคคลที่สามมาเกี่ยวข้องในการตรวจสอบ ใครที่มีความลับคืออัตลักษณ์ ใครที่ทำหายก็สิ้นสุดการเป็นอัตลักษณ์นั้น คำตัดสินนั้นชัดเจน: **ไม่มีใครให้ขอให้อัตลักษณ์กลับคืนมาให้คุณ บุคคลนั้นไม่มีอยู่จริง เพราะเขาไม่ได้ครอบครองมันตั้งแต่แรก**

ยี่สิบสี่คำเป็นตัวแทนของอะไร

โดยปกติแล้วอัตลักษณ์การเข้ารหัสจะแสดงด้วยความลับทางคณิตศาสตร์ขนาด 32 ไบต์ — 256 บิต ตัวเลขที่จำยากและยิ่งยากขึ้นไปอีกที่จะคัดลอกโดยไม่มีข้อผิดพลาด อุตสาหกรรมการเข้ารหัสลับได้แก้ไขปัญหานี้ในปี 2013 ด้วยมาตรฐานขนาดเล็กและสง่างามที่เรียกว่า BIP39: วิธีการแสดง 256 บิตเหล่านั้นเป็นลำดับของยี่สิบสี่คำที่นำมาจากรายการที่เป็นทางการจำนวน 2,048 คำ คณิตศาสตร์ที่อยู่เบื้องหลังสอดรับกันอย่างลงตัว ใครที่ต้องการดูรายละเอียดสามารถดูได้ที่ส่วนเสริม

การนับเริ่มต้นจากตอนท้าย เราต้องการแสดงความลับ 256 บิตโดยเพิ่ม 8 บิตของ Checksum: รวมเป็น 264 บิต หากเราแบ่งมันออกเป็นยี่สิบสี่คำ — ซึ่งเป็นจำนวนที่จัดการได้เพื่อจดบันทึกและบอกตามคำบอกโดยไม่มีการสูญหาย — แต่ละคำจะต้องให้ข้อมูลที่แม่นยำ 11 บิต และ 11 บิตคือ 2 ยกกำลัง 11 ของความเป็นไปได้ นั่นคือ 2,048 ด้วยเหตุนี้ คำศัพท์อย่างเป็นทางการของ BIP39 จึงมีขนาดเท่านี้พอดี: รายการนี้มีอยู่ตามขนาดของปัญหา ไม่ใช่ในทางกลับกัน

การนับนั้นไม่ได้มีไว้เพื่อความสวยงาม หากใครคัดลอกยี่สิบสามคำได้อย่างถูกต้องและผิดพลาดในคำที่ยี่สิบสี่ Checksum จะตรวจพบ: ซอฟต์แวร์จะบอกเขาว่า "ลำดับนี้ไม่ถูกต้อง" หากใครคัดลอกทั้งยี่สิบสี่คำได้อย่างถูกต้อง ซอฟต์แวร์จะสืบทอดอัตลักษณ์เดียวกันออกมาอย่างชัดเจน การเลือกรายการคำก็เป็นไปอย่างจงใจ: คำศัพท์ใน BIP39 นั้นสั้น แตกต่างกันอย่างชัดเจน ไม่มีเครื่องหมายกำกับการออกเสียง และถูกเลือกมาเพื่อลดความสับสนด้านเสียงและการสะกดคำให้เหลือน้อยที่สุด มันเป็นคำศัพท์ที่ออกแบบมาเพื่อให้มนุษย์จดจำ เขียน และบอกตามคำบอกได้โดยไม่มีการสูญหาย

จากวลีสู่กุญแจ

คำทั้งยี่สิบสี่คำไม่ใช่กุญแจเข้ารหัสลับที่ใช้ลงนามข้อความ แต่เป็นตัวแทนที่กู้คืนได้ของเอนโทรปีดั้งเดิม ซึ่งผ่านกระบวนการเชิงกำหนดที่เรียกว่า PBKDF2 แล้วจะถูกเปลี่ยนเป็นเมล็ดพันธุ์ (seed) ขนาดหกสิบสี่ไบต์ จากเมล็ดพันธุ์นั้น กุญแจเข้ารหัสลับที่เฉพาะเจาะจงซึ่งผู้ใช้ใช้งานจะถูกสร้างขึ้นมาอย่างเป็นระบบเช่นกัน ได้แก่ กุญแจส่วนตัวสำหรับการลงนาม และกุญแจสาธารณะที่สอดคล้องกันซึ่งจะถูกเผยแพร่เพื่อตรวจสอบลายเซ็น กลไกเดียวกันนี้ถูกใช้ในระบบที่แตกต่างกัน เช่น สกุลเงินดิจิทัลใช้เส้นโค้ง secp256k1 ส่วนโปรโตคอล Signal และระบบสมัยใหม่หลายระบบใช้ Ed25519 บนเส้นโค้ง Curve25519 สำหรับเส้นโค้งเฉพาะอย่าง Ed25519 มาตรฐาน BIP32 และ SLIP-0010 จะนำเมล็ดพันธุ์หกสิบสี่ไบต์นั้นมาสร้างเป็นสามสิบสองไบต์ ซึ่งเป็นกุญแจลงนามที่มีผลบังคับใช้ — ซึ่งเป็นสามสิบสองไบต์เดียวกันกับที่ตัวอย่างโค้ดในส่วนถัดไปเริ่มต้นขึ้น

นี่คือวิธีมาตรฐานที่อุตสาหกรรมทั้งหมดนำเสนอกลไกนี้ต่อผู้ใช้ ไม่ว่าจะเป็นกระเป๋าเงินสกุลเงินดิจิทัล, ตัวจัดการอัตลักษณ์แบบกระจายศูนย์, Signal ในส่วนของอัตลักษณ์ถาวร รวมถึง Solo2 ด้วย ในทางปฏิบัติ ผู้ใช้จะไม่เคยเห็นเมล็ดพันธุ์หรือกุญแจที่สร้างขึ้นเลย เขาจะเห็นคำยี่สิบสี่คำเมื่อสร้างอัตลักษณ์ของตน และอาจจดบันทึกไว้ในกระดาษ จากนั้นคำเหล่านี้จะเดินทางไปพร้อมกับอุปกรณ์ของเขาเมื่อเขาต้องการย้ายอัตลักษณ์ โดยเขาจะกรอกคำเหล่านั้นในแอปพลิเคชันใหม่ แอปพลิเคชันก็จะสร้างเมล็ดพันธุ์เดิม กุญแจเดิม

และอัตลักษณ์เดิมขึ้นมา เป็นกลไกที่พกพาได้ มีความปลอดภัยทางเข้ารหัสลับ และจดจำได้ภายในขอบเขตที่สมเหตุสมผล

วิธีการลงนามด้วยกุญแจ (แต้มพู่กันด้วย Zig)

ในภาษา Zig เมื่อคุณมีเมล็ดพันธุ์สามสิบสองไบต์ที่สร้างมาจากคำยี่สิบสี่คำแล้ว การลงนามข้อความด้วย Ed25519 จะใช้โค้ดเพียงไม่กี่บรรทัด:

```
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);
```

การลงนามจะสร้างข้อมูลขนาดหกสิบสี่ไบต์ที่เรียกว่า ลายเซ็น ซึ่งสามารถสร้างขึ้นได้จากกุญแจส่วนตัวที่สอดคล้องกันเท่านั้น การตรวจสอบเป็นเรื่องสาธารณะ ใครก็ตามที่มีกุญแจสาธารณะสามารถตรวจสอบได้ว่าลายเซ็นนั้นตรงกับข้อความหรือไม่ หากไม่มีกุญแจส่วนตัว จะไม่มีใครสามารถสร้างลายเซ็นที่ถูกต้องสำหรับข้อความนั้นได้ แต่ด้วยกุญแจสาธารณะ ทุกคนสามารถตรวจพบได้ว่าลายเซ็นนั้นถูกต้องหรือไม่ ความไม่สมมาตรนี้เองที่ช่วยให้ผู้ลงนามสามารถพิสูจน์ความเป็นเจ้าของได้โดยไม่ต้องแบ่งปันความลับ

ตัวอย่างข้างต้นคือเวอร์ชันคู่มือขั้นต่ำ ในโค้ดจริงของ Solo2 ห่วงโซ่จะส่งผ่านไฟล์สองไฟล์ ไฟล์หนึ่งใน JavaScript ที่อยู่ในเบราว์เซอร์ของผู้ใช้และสร้างเอนโทรปีใหม่จากคำยี่สิบสี่คำ อีกไฟล์หนึ่งใน Zig ภายในไลบรารี *zcatcrypto* ที่นำเอนโทรปีนั้นมาและรับคีย์การเข้ารหัสเฉพาะ เริ่มจากฝั่งเบราว์เซอร์:

```
// 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 };
}
```

เอนโทรปีสามสิบสองไบต์เหล่านั้น พร้อมกับอีกสามสิบสองไบต์ที่ได้ในขั้นตอนเดียวกัน จะเดินทางไปยังโมดูล WebAssembly ของ Zig ที่สร้างคีย์ Ed25519 ที่แท้จริง ฟังก์ชันทั้งหมดพร้อมการล้างหน่วยความจำขั้นสุดท้ายจะพอดีกับหน้าจอเดียว:

```
// 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;
}
```

มีรายละเอียดสองประการที่ควรค่าแก่การสังเกต ประการแรก: ซีด (seed) เดียวกันจะสร้างคู่คีย์เดิมเสมอ — นี่คือสิ่งที่ช่วยให้กู้คืนตัวตนได้โดยการป้อนคำยี่สิบสี่คำในอุปกรณ์ใหม่ ประการที่สอง: ซีดจะถูกลบออกจากหน่วยความจำอย่างชัดเจนในบรรทัดสุดท้าย หลังจากจุดนั้น แม้แต่ตัวฟังก์ชันเองก็ไม่สามารถสร้างคีย์ขึ้นมาใหม่ได้ คำพูดของผู้ใช้จะเป็นที่มาเพียงอย่างเดียว

สำหรับผู้ที่ต้องการตรวจสอบด้วยตัวเลขขนาดเล็ก รูปแบบการลงนามสามารถไล่ตามได้ทั้งหมดด้วยตัวเลขที่น้อยพอที่จะคำนวณด้วยมือ ใครที่ไม่อยากเข้าสู่การคำนวณสามารถข้ามบล็อกนี้ไปได้โดยไม่เสียเนื้อหาของบทความ ใครที่ต้องการเห็นกลไกทำงานทีละขั้นตอนจะพบได้ที่นี่ **กฎสาธารณะ** ที่ใครๆ ก็อ่านได้: เลขฐาน $p = 23$ (ใน Ed25519 จริงมีประมาณเจ็ดสิบเจ็ดหลัก เราใช้ยี่สิบสามเพื่อให้การคำนวณพอดีในหน้าเดียว),

ฐาน $g = 2$ ซึ่งอันดับในกลุ่มนี้คือ $q = 11$ และข้อกำหนดที่ว่าการคำนวณทั้งหมดด้วย g จะทำแบบ *módulo p* และเลขชี้กำลังทั้งหมดจะลดลงแบบ *módulo q* **ตัวเลือกส่วนตัว** เพียงหนึ่งเดียวและไม่เคยแชร์: ความลับ $x = 6$ นั่นคือตัวตน

ขั้นตอนที่ 1 — ส่วนสาธารณะของตัวตน คำนวณเพียงครั้งเดียวและเผยแพร่อย่างเปิดเผย

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

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

ส่วนสาธารณะของตัวตนคือ **18** ใครๆ ก็สามารถนำไปใช้เพื่อตรวจสอบลายเซ็นที่สร้างขึ้นด้วยตัวตนนี้ ไม่มีใครที่สังเกตเห็นเพียงเลข 18 จะสามารถกู้คืนความลับ 6 ได้: นั่นคือปัญหาลอการิทึมแบบไม่ต่อเนื่อง (discrete logarithm) ที่เราจะกลับมาพูดถึงในตอนท้าย

ขั้นตอนที่ 2 — การลงนามข้อความ ผู้ครอบครองตัวตนต้องการลงนามในข้อความ $m = 7$ เขาเริ่มจากการเลือกค่าสุ่มใหม่ $k = 4$ ซึ่งจะใช้เพียงครั้งเดียวและไม่แชร์เด็ดขาด (ใน Ed25519 จริง k จะได้มาจากข้อความและความลับอย่างเป็นระบบเพื่อหลีกเลี่ยงอันตรายจากการใช้ซ้ำ แต่บทบาทที่เล่นคือสิ่งนี้พอดี) จากนั้นเขาคำนวณตัวเลขสามตัว:

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

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

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

ลายเซ็นคือคู่ **(r, s) = (16, 10)** เดินทางไปอย่างเปิดเผยพร้อมกับข้อความ ใครๆ ก็อ่านได้ หมายเหตุเพื่อการเรียนรู้: ใน Ed25519 จริง ฟังก์ชัน H คือ SHA-512 ซึ่งแข็งแกร่งในการเข้ารหัส ที่นี่เราใช้การทำให้ง่ายขึ้น $e = (r + m) \bmod q$ เพื่อให้ผู้อ่านสามารถไล่ตามขั้นตอนได้โดยไม่ต้องคำนวณแฮช โครงสร้างของอัลกอริทึมจะเหมือนกัน

ขั้นตอนที่ 3 — การตรวจสอบลายเซ็น ผู้ตรวจสอบมีส่วนสาธารณะ $y = 18$, ข้อความ $m = 7$ และลายเซ็น $(r, s) = (16, 10)$ เขาสร้าง e ขึ้นมาใหม่ด้วยวิธีเดียวกัน — $e = (16 + 7) \bmod 11 = 1$ — และตรวจสอบว่าความเท่ากันนี้เป็นจริงหรือไม่:

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

คำนวณทั้งสองด้านแยกกัน:

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

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

ทั้งสองด้านได้ผลลัพธ์ **12** ลายเซ็นนั้นถูกต้อง ใครก็ตามที่มีส่วนสาธารณะ 18 สามารถมาถึงข้อสรุปนี้ได้โดยไม่ต้องรู้เลยว่าความลับคือ 6

แล้วบุคคลที่สามที่พยายามปลอมแปลงล่ะ? เอวาเห็นทุกสิ่งที่เป็นสาธารณะผ่านช่องทาง: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$ ในการลงนามข้อความที่ *แตกต่าง* ในนามของตัวตนนี้ เธอจำเป็นต้องรู้ x ทางเดียวของเธอคือการถามตัวเองว่า: "เลขชี้กำลัง x ใดที่ทำให้ $2^x \bmod 23 = 18$ เป็นจริง?" ด้วย $p = 23$ เธอสามารถลอง 0, 1, 2, 3, … และพบได้ในไม่กี่วินาที แต่เมื่อแทนที่ 23 ด้วยเลขฐานในมิติจริงของ Ed25519 พื้นที่ของเลขชี้กำลังที่เป็นไปได้จะเกินจำนวนอะตอมในจักรวาลที่สังเกตได้ **ในปัจจุบันยังไม่มีอัลกอริทึมใดที่มนุษยชาติรู้จักที่สามารถสำรวจพื้นที่นั้นได้ในเวลาน้อยกว่าพันล้านปี** มันคือปัญหาลอการิทึมแบบไม่ต่อเนื่องแบบเดียวกับที่เป็นฐานของ Diffie-Hellman ในบทความก่อนหน้านี้ โดยนำมาใช้ที่นี่กับรูปแบบการลงนาม

สิ่งที่เราเพิ่งไล่ตามไปคือ Schnorr *พอดิบพอดี* ซึ่งเป็นรูปแบบการลงนามที่ Ed25519 เป็นรุ่นที่ปรับให้เข้ากับเส้นโค้งวงรี ใน Ed25519 จริง การดำเนินการทั้งหมดทำบนจุดของเส้นโค้งเฉพาะ (Curve25519) แทนที่จะเป็นจำนวนเต็มมอดูโลเลขฐาน และฟังก์ชัน H คือ SHA-512 แทนที่จะเป็นการบวกแบบของเล่นที่เราใช้ข้างต้น การแทนที่ทั้งสองคือการปรับเปลี่ยนการใช้งาน — เพื่อให้ได้ความต้านทานในการเข้ารหัสต่อการโจมตีแบบสุ่ม (brute force) เพื่อให้ได้คุณสมบัติความปลอดภัยเพิ่มเติมสำหรับ k — โครงสร้างอัลกอริทึม การดำเนินการสามอย่าง และเหตุผลของความไม่สมมาตรนั้นเหมือนกัน

ควรหยุดพักสั้นๆ ที่นี่ เพราะห่วงโซ่ทั้งหมดอาจถูกสับสนได้ในแวบเดียวกับฟังก์ชันดั้งเดิมอีกตัวในกลุ่มสามตัวนั่นคือ แฮช (hash) แต่มันไม่ใช่ แฮชคือฟังก์ชันเฉพาะที่ทำการบีบอัด — ไบต์จำนวนมากเข้ามา ลายนิ้วมือสั้นๆ ออกไป เส้นทางสิ้นสุดลงตรงนั้น ตัวตนทางการเข้ารหัสคือคู่ทางคณิตศาสตร์ที่ส่งเสริมกัน: ความลับยังคงอยู่และลงนาม; ส่วนสาธารณะจะถูกเผยแพร่และตรวจสอบ ในขณะที่แฮชจะยุบข้อมูลไปในทิศทางเดียว ตัวตนจะสร้างความไม่สมมาตรระหว่างสองส่วน แฮชเป็นพยานว่ามีการพูดอะไร; ตัวตนเป็นพยานว่าใครเป็นคนพูด

สิ่งที่วลีนั้นไม่ใช่

ควรทำความเข้าใจความเข้าใจผิดที่พบบ่อยสามประการ วลีไม่ใช่รหัสผ่านในความหมายที่แท้จริง เพราะมันไม่ได้ถูกเปรียบเทียบกับลายนิ้วมือที่เก็บไว้ในเซิร์ฟเวอร์ แต่มันถูกกรอกลงในอุปกรณ์ของผู้ใช้เพื่อสร้างอัตลักษณ์ขึ้นมาใหม่ด้วยวิธีทางคณิตศาสตร์ วลีไม่สามารถกู้คืนได้ หากทำหาย จะไม่มีใครให้คุณไปขอคืนได้ หากมีการคัดลอก อัตลักษณ์ก็จะถูกคัดลอกไปด้วย วลีไม่ใช่ข้อมูลรับรองที่แยกออกจากอัตลักษณ์ แต่วลีนั้น *คือ* อัตลักษณ์ ใครก็ตามที่มีวลีนี้สามารถกระทำการในฐานะอัตลักษณ์นั้นได้ โดยไม่ต้องขออนุญาตเพิ่มเติม ไม่ต้องผ่านกระบวนการอนุมัติ และไม่สามารถกู้คืนได้

คุณสมบัติประการที่สามนี้เองที่เปลี่ยนน้ำหนักของเรื่อง รหัสผ่านที่หายไปเป็นเพียงความยุ่งยากในการบริหารจัดการ แต่อัตลักษณ์ทางเข้ารหัสลับที่หายไปคือการสูญเสียตัวตน กระดาษที่จดวลีซึ่งถูกบุคคลที่สามพบไม่ใช่แค่ความเสี่ยงในการถูกขโมยบัญชี แต่มันคือการส่งมอบอัตลักษณ์ทั้งหมดให้ผู้อื่น คำมั่นสัญญาของระบบที่ว่าไม่มีใครสามารถเพิกถอนอัตลักษณ์ของคุณหรือบล็อกคุณโดยพลการได้ มาพร้อมกับความรับผิดชอบที่แยกไม่ออกว่าคุณคือผู้ดูแลเพียงคนเดียวของสิ่งที่ไม่มีใครสามารถกู้คืนให้คุณได้

คำมั่นสัญญาและภาระ

รูปแบบของอัตลักษณ์ทางเข้ารหัสลับมักจะถูกเรียกว่า *อัตตธิปไตย* (self-sovereign ในภาษาอังกฤษ) การเลือกใช้คำนี้มีความตั้งใจและอธิบายสถานะได้อย่างแม่นยำ ผู้ใช้เป็นอธิปไตยเหนืออัตลักษณ์ของตนในความหมายที่เกือบจะเหมือนในยุคกลาง คือไม่มีกษัตริย์ ผู้พิจารณา หรืออำนาจส่วนกลางใดเป็นผู้มอบให้ และไม่มีใครสามารถถอนคืนไปได้เช่นกัน แต่เช่นเดียวกับกษัตริย์ในยุคกลาง ผู้ใช้ต้องรับผิดชอบต่อผลลัพธ์ของความผิดพลาดของตนเองทั้งหมด ไม่มีผู้สำเร็จราชการแทนพระองค์ที่จะมาตัดสินใจแทนคุณหากคุณทำตราประทับหาย

การเลือกระหว่างอัตลักษณ์ที่จัดการโดยบุคคลที่สามกับอัตลักษณ์แบบอัตตธิปไตยไม่มีคำตอบที่ถูกต้องตายตัวสำหรับทุกคน สำหรับบัญชีฟอรัมที่ไม่สำคัญ อัตลักษณ์ที่ได้รับการจัดการอาจจะเหมาะสมกับความเสี่ยงแล้ว แต่สำหรับอัตลักษณ์ทางวิชาชีพที่ต้องลงนามในเอกสารที่มีผลผูกพันทางกฎหมาย สำหรับอัตลักษณ์ทางเศรษฐกิจที่ดูแลเงินออมส่วนตัว หรือสำหรับอัตลักษณ์ในการสื่อสารทางวิชาชีพกับลูกค้าที่ไว้วางใจมอบข้อมูลที่ละเอียดอ่อนให้ เรื่องนี้จะเปลี่ยนไป เมื่อนั้นคำถามจะไม่ใช่แค่ «สะดวกไหม?» แต่จะกลายเป็น «ใครนอกจากฉันที่มีอำนาจกระทำการในฐานะตัวฉัน และภายใต้สถานการณ์ใด?»

กลไกนี้ปรากฏให้เห็นในระบบที่ใช้งานจริงที่ไหนบ้าง

BIP39 ถือกำเนิดขึ้นในโลกของ Bitcoin ในปี 2013 และแพร่กระจายไปยังระบบนิเวศคริปโตเคอร์เรนซีทั้งหมดอย่างรวดเร็ว: วอลเล็ตที่ได้มาตรฐานทุกวันนี้ล้วนยอมรับวลี BIP39 ขนาด 12 หรือ 24 คำ เพื่อเป็นข้อมูลสำรองสำหรับอัตลักษณ์ทางเศรษฐกิจของผู้ครอบครอง นอกเหนือจากคริปโตเคอร์เรนซี แนวคิดพื้นฐานเดียวกันนี้ — คู่รหัสลับที่พิสูจน์ความเป็นเจ้าของโดยไม่ต้องมีคนกลาง — ยังปรากฏในระบบอื่นๆ ด้วยไวยากรณ์ที่ต่างกัน กุญแจ SSH ที่ผู้ดูแลระบบใช้ในการเข้าถึงเซิร์ฟเวอร์คือกรณีตัวอย่างคลาสสิก: กุญแจส่วนตัวที่ผู้ดูแลระบบเก็บไว้ในเครื่องของตนและกุญแจสาธารณะที่คัดลอกไปยังเซิร์ฟเวอร์แต่ละเครื่อง โดยไม่มีองค์กรใดที่เทียบได้กับบริการรวมศูนย์เข้ามาเกี่ยวข้อง โปรโตคอล Signal ใช้ Ed25519 พร้อมวัสดุกุญแจที่คงอยู่ในอุปกรณ์ ส่วน eIDAS ของยุโรปในส่วนของลายเซ็นที่มีคุณสมบัติเหมาะสมนั้น ตั้งอยู่บนหลักการรหัสลับเดียวกัน โดยมีความแตกต่างคือผู้ให้บริการความเชื่อถือที่มีคุณสมบัติเหมาะสมเป็นผู้ดูแลกุญแจแทนผู้ใช้

Solo2 ซึ่งเป็นแพลตฟอร์มผู้จัดพิมพ์สิ่งพิมพ์ฉบับนี้ ใช้วลี BIP39 ขนาด 24 คำเป็นอัตลักษณ์ของผู้ใช้แต่ละราย เมื่อผู้ใช้สร้างบัญชี จะเห็นคำเหล่านั้นเพียงครั้งเดียว คำเหล่านี้จะไม่ถูกเก็บไว้ในเซิร์ฟเวอร์ใดๆ ของ Solo2 หรือของใครเลย: หากผู้ใช้จดบันทึกและดูแลรักษาคำเหล่านี้ไว้ ก็จะคงอัตลักษณ์ของตนไว้ได้ตลอดไป หากทำหาย ก็คือหายไปเลย นี่คือผลลัพธ์ที่สอดคล้องกับสถาปัตยกรรมที่ไม่มีผู้ให้บริการอยู่ตรงกลาง: หาก Solo2 สามารถคืนอัตลักษณ์ให้กับผู้ใช้ที่ทำหายได้ Solo2 ก็จะสามารถมอบอัตลักษณ์นั้นให้กับใครก็ตามที่กดดันให้ Solo2 มอบให้ได้เช่นกัน

สำหรับผู้อ่านมืออาชีพ

ข้อควรพิจารณาสี่ประการสำหรับผู้ที่กำลังประเมินการนำอัตลักษณ์แบบรหัสลับที่ผู้ถือครองมีอำนาจปกครองตนเอง (autosoberana) มาใช้ในบริบททางวิชาชีพ:

1. วลีคืออัตลักษณ์ การดูแลรักษาทางกายภาพ — กระดาษ, สำเนาหลายฉบับในที่ต่างกัน, หรือแม้แต่แผ่นโลหะสลักเพื่อการใช้งานในระยะยาว — ให้การรับประกันได้มากกว่าการดูแลรักษาแบบดิจิทัล ซึ่งเป็นการเพิ่มช่องทางในการโจมตีโดยไม่ได้ลดความเสี่ยงจากการสูญหาย
2. ไม่มีการกู้คืน การออกแบบกระบวนการโดยสมมติว่าวันหนึ่งสำเนาหลักจะสูญหายนั้นเหมาะสมกว่าการไปพบความจริงในวันที่สายเกินไป สำเนาฉบับที่สองที่แยกเก็บไว้ในที่ต่างๆ ทางภูมิศาสตร์สามารถแก้ปัญหาได้เกือบทุกสถานการณ์
3. ไม่ใช่สิ่งเดียวกับใบรับรองที่มีคุณสมบัติเหมาะสมของ eIDAS สำหรับลายเซ็นที่มีคุณสมบัติเหมาะสมในสหภาพยุโรป — นิติกรรมสัญญา, ขั้นตอนบางอย่างกับหน่วยงานราชการ — กฎหมายกำหนดให้มีผู้ให้บริการที่มีคุณสมบัติเหมาะสมเป็นผู้ดูแลกุญแจ อัตลักษณ์แบบรหัสลับที่ผู้ถือครองมีอำนาจปกครองตนเอง (autosoberana) ใช้เพื่อการสื่อสารทางวิชาชีพและการลงนามในเอกสารที่มีคุณค่าทางการพิสูจน์ แต่ไม่สามารถทดแทนใบรับรองที่มีคุณสมบัติเหมาะสมได้โดยอัตโนมัติในกรณีที่กฎระเบียบกำหนดไว้
4. หากต้องมีการส่งต่ออัตลักษณ์ — มรดก, การสืบทอดทางวิชาชีพ, การปิดกิจการ — ควรเตรียมขั้นตอนไว้ล่วงหน้า ไม่ใช่ทำในภายหลัง ขั้นตอนอย่างเป็นทางการด้วยซองจดหมายที่ผนึกด้วยครั่ง (lacre), คำสั่งถึงผู้จัดการมรดก, การฝากไว้ที่สำนักงานโนตารี คือการจัดการแบบคลาสสิกที่สอดคล้องกับธรรมชาติทางรหัสลับของสินทรัพย์นี้อย่างสมบูรณ์

บทความนี้เป็นการปิดท้ายชุดแนวคิดสามประการที่เปิดวงจร — hash, การเข้ารหัส, อัตลักษณ์ — ทั้งสามไอเดียสร้างขึ้นต่อยอดกันและกัน: hash ให้รอยพิมพ์ที่ไม่สามารถเปลี่ยนแปลงได้, การเข้ารหัสให้การรักษาความลับโดยไม่มีบุคคลที่สามที่เชื่อถือได้, อัตลักษณ์ให้ความเป็นเจ้าของโดยไม่มีบุคคลที่สามผู้ประทานให้ ทั้งสามอย่างมีคุณสมบัติร่วมกันที่ไม่ใช่เรื่องของอุดมการณ์: คือการถ่ายโอนขีดความสามารถทางเทคนิคที่เดิมเคยอยู่ที่ผู้ให้บริการ จากผู้จัดการบริการไปยังผู้ใช้ และยังเป็นการถ่ายโอนความรับผิดชอบไปพร้อมกันด้วย การพูดคุยเรื่องใดเรื่องหนึ่งในสามเรื่องนี้อย่างซื่อตรง จำเป็นต้องพูดถึงอีกสองเรื่องที่เหลือด้วย

แหล่งข้อมูลและการอ่านเพิ่มเติม

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, ข้อเสนอการปรับปรุง Bitcoin ปี 2013 มาตรฐานโดยพฤตินัยสำหรับวลีการกู้คืนในอุตสาหกรรมคริปโต
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), รวมถึง Ed25519 IETF, มกราคม 2017 ข้อกำหนดเชิงบรรทัดฐานของรูปแบบลายเซ็นที่ใช้ในอุตสาหกรรมร่วมสมัยส่วนใหญ่
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, เวอร์ชัน 2.0 IETF, กันยายน 2000 กำหนดอัลกอริทึม PBKDF2 ที่ใช้ในการสกัด BIP39 จากวลีไปเป็น seed

- กฎระเบียบ (EU) 910/2014 (eIDAS) และวิวัฒนาการโดยกฎระเบียบ (EU) 2024/1183 (eIDAS 2) — กรอบการทำงานของยุโรปด้านอัตลักษณ์อิเล็กทรอนิกส์และลายเซ็นที่มีคุณสมบัติเหมาะสม เป็นระบบที่ต่างจากระบบปกครองตนเอง แต่ในเชิงแนวคิดได้รับการสนับสนุนโดยพื้นฐานรหัสลับเดียวกัน
- Allen, C. — *The Path to Self-Sovereign Identity* (2016) ข้อความหลักเกี่ยวกับหลักการและข้อผูกพันของโมเดลการปกครองตนเอง แม้จะเขียนไว้ก่อนหน้าแต่ยังคงเกี่ยวข้องกับการทำความเข้าใจตระกูลของโซลูชันร่วมสมัย

← ก่อนหน้าโมเดลธุรกิจในฐานะสัญญาณแห่งความไว้วางใจถัดไป →Self-hosting ในฐานะการปฏิบัติทางวิชาชีพ

บทความล่าสุด

- บทสะท้อนความคิด · 29 มิถุนายน 2026 คุณไม่ได้เป็นนิรนาม
- บทสะท้อน · 27 พฤษภาคม 2026 สิ่งที่ลายเซ็นไม่สามารถแก้ไขได้
- การวิเคราะห์ · 26 พฤษภาคม 2026 ความเป็นส่วนตัวที่แท้จริง vs ความเป็นส่วนตัวที่ฉาบฉวย: คำถามที่คุณควรตั้งกับตัวเอง

ดาวน์โหลดบทความนี้เก็บไว้เพื่อใช้งานได้ทุกที่ที่คุณต้องการ

↓ Markdown ↓ ข้อความธรรมดา ↓ PDF

ไฟล์จะถูกดาวน์โหลดลงในอุปกรณ์ของคุณ คุณสามารถบันทึก นำเข้าสู่ Solo2 หรือแชร์ได้ทุกที่ตามต้องการ Cuadernos จะไม่กำหนดปลายทางแทนคุณ

ตราประทับครั่ง · SHA-256 b59f868640aff8d4929cebdb980cf8857ad5395ea4cedb69106264a142a575b3

Cuadernos Lacre · สิ่งพิมพ์ของ Menzuri Gestión S.L. ·
เขียนโดย R.Eugenio · เรียบเรียงโดยทีมงาน Solo2