۲۴ کلمه: هویت رمزنگاری شده چیست

یک هویت رمزنگاری شده رمز عبور نیست: هیچ سروری آن را ذخیره نمی‌کند و قابل بازیابی نیست. توضیحی آموزشی از مکانیسم BIP39، اینکه چرا دقیقاً بیست و چهار کلمه، و چه بار واقعی بر دوش کسی است که آن‌ها را در اختیار دارد.

برای درک بهتر: اگر رمز عبور جیمیل خود را فراموش کنید، گوگل آن را برای شما بازنشانی می‌کند. اگر ۲۴ کلمه‌ای را که یک هویت رمزنگاری شده را تشکیل می‌دهند گم کنید، کسی نیست که آن‌ها را از او بخواهید. نه اینکه روند سخت‌گیرانه باشد - بلکه کسی در آن طرف وجود ندارد. این تفاوت، تمام تفاوت است.

تفاوت بین رمز عبور و هویت

رمز عبور، در مدل کلاسیک اینترنت، هویت کاربر نیست. بلکه یک رسید است. کاربر دارای یک هویت است - یک نام، یک ایمیل، یک شماره مشتری - و برای اثبات اینکه همان کسی است که ادعا می‌کند به سرور، رمز عبوری را ارائه می‌دهد که سرور آن را با اثر ذخیره شده مقایسه می‌کند. اگر اثرها مطابقت داشته باشند، سرور اجازه دسترسی به نشست را می‌دهد. اگر رمز عبور گم شود، کاربر همان کاربر باقی می‌ماند؛ آنچه گم می‌شود رسید است و یک روند بازیابی وجود دارد - ایمیلی به آدرس ثبت شده، یک سوال امنیتی - برای بازگرداندن آن.

یک هویت رمزنگاری شده به گونه دیگری عمل می‌کند. این یک اعتبارنامه نیست که کسی آن را با یک اثر ذخیره شده مقایسه کند؛ بلکه خود یک راز ریاضی کامل است. فرقی نمی‌کند کجا باشد - روی کاغذ، در یک دستگاه، یا حتی در سرور دیگران - هویت به دلیل ریاضیاتش وجود دارد، نه به خاطر کسی که آن را تایید می‌کند. در اینجا ویژگی مشابهی با آنچه در «SHA-256 واقعاً چیست» دیدیم ظاهر می‌شود: مالکیت با نمایش راز ثابت نمی‌شود، بلکه با استفاده از آن برای امضا ثابت می‌شود. امضای تولید شده به این صورت را هر کسی می‌تواند با یک مقدار عمومی که به صورت ریاضی از خود راز مشتق شده است، بدون نیاز به دانستن خود راز و بدون میانجیگری شخص ثالث در بررسی، تایید کند. کسی که راز را دارد، هویت است؛ کسی که آن را گم کند، دیگر هویت نیست. حکم قاطع است: **کسی وجود ندارد که از او بخواهید هویت را به شما برگرداند. چنین کسی وجود ندارد، زیرا او از ابتدا آن را نداشته است.**

آنچه بیست و چهار کلمه نشان می‌دهند

هویت رمزنگاری شده معمولاً با یک راز ریاضی سی و دو بایتی - دویست و پنجاه و شش بیت - نشان داده می‌شود. عددی که حفظ کردنش دشوار و بازنویسی‌اش بدون خطا دشوارتر است. صنعت رمزنگاری این مشکل را در سال ۲۰۱۳ با یک استاندارد کوچک و زیبا به نام BIP39 حل کرد: راهی برای نمایش آن دویست و پنجاه و شش بیت به عنوان دنباله‌ای از بیست و چهار کلمه گرفته شده از یک لیست رسمی دو هزار و چهل و هشت کلمه‌ای. محاسبات پشت آن به زیبایی با هم جور در می‌آیند؛ هر کسی که بخواهد آن را با جزئیات ببیند، در حاشیه پیدا می‌کند.

شمارش از آخر شروع می‌شود. ما می‌خواهیم دویست و پنجاه و شش بیت راز را با اضافه کردن هشت بیت چک‌سام نمایش دهیم: در مجموع دویست و شصت و چهار بیت. اگر آن‌ها را به بیست و چهار کلمه تقسیم کنیم - عددی قابل مدیریت برای یادداشت کردن و دیکته کردن بدون خطا - هر کلمه باید دقیقاً یازده بیت اطلاعات ارائه دهد. و یازده بیت یعنی دو به توان یازده احتمال مختلف، یعنی دو هزار و چهل و هشت. به همین دلیل است که واژگان رسمی BIP39 دقیقاً همین اندازه را دارند: لیست به اندازه مشکل وجود دارد، نه برعکس.

شمارش تزئینی نیست. اگر کسی بیست و سه کلمه را به درستی بازنویسی کند و در کلمه بیست و چهارم اشتباه کند، چک‌سام آن را تشخیص خواهد داد: نرم‌افزار به او خواهد گفت «این دنباله معتبر نیست». اگر کسی تمام بیست و چهار کلمه را به درستی بازنویسی کند، نرم‌افزار بدون ابهام همان هویت را مشتق خواهد کرد. انتخاب لیست کلمات نیز عمدی است: کلمات واژگان BIP39 کوتاه، متفاوت از یکدیگر، بدون علائم تلفظی هستند که برای به حداقل رساندن اشتباهات آوایی و املایی انتخاب شده‌اند. این واژگانی است که برای به خاطر سپردن، نوشتن و دیکته کردن توسط انسان‌ها بدون خطا طراحی شده است.

از عبارت تا کلید

بیست و چهار کلمه، کلید رمزنگاری نیستند که پیام‌ها را امضا می‌کنند. آن‌ها یک نمایش قابل بازیابی از انتروپی اصلی هستند که از طریق فرآیندی معین به نام PBKDF2، به یک دانه (seed) شصت و چهار بایتی تبدیل می‌شوند. از آن دانه، باز هم به روشی معین، کلیدهای رمزنگاری خاصی که کاربر استفاده می‌کند مشتق می‌شوند: یک کلید خصوصی برای امضا و یک کلید عمومی متناظر که برای تأیید امضاها منتشر می‌شود. سازوکار یکسان در سیستم‌های مختلف: ارزهای دیجیتال از منحنی secp256k1 استفاده می‌کنند؛ پروتکل Signal و بسیاری از سیستم‌های مدرن از Ed25519 روی منحنی Curve25519 استفاده می‌کنند. برای یک منحنی خاص مانند Ed25519، استانداردهای BIP32 و SLIP-0010 آن دانه شصت و چهار بایتی را می‌گیرند و به طور معین، سی و دو بایتی را که کلید امضای مؤثر را تشکیل می‌دهند مشتق می‌کنند — همان سی و دو بایتی که مثال کد در بخش بعدی با آن شروع می‌شود.

این روش استانداردی است که کل صنعت، سازوکار را به کاربر ارائه می‌دهد —کیف پول‌های ارز دیجیتال، مدیریت هویت غیرمتمرکز، Signal در بخش هویت پایدار خود، و Solo2 در میان آن‌ها—: کاربر در عمل هرگز دانه یا کلیدهای مشتق شده را نمی‌بیند. او هنگام ایجاد هویت خود بیست و چهار کلمه را می‌بیند و در صورت تمایل، آن‌ها را روی کاغذ یادداشت می‌کند. کلمات سپس بین دستگاه‌های او هنگامی که می‌خواهد هویت را منتقل کند جابجا می‌شوند: او آن‌ها را در برنامه جدید وارد می‌کند، برنامه همان دانه، همان کلیدها و همان هویت را مشتق می‌کند. این یک سازوکار قابل حمل، از نظر رمزنگاری مستحکم و در حد معقول، قابل به خاطر سپردن است.

چگونه با کلید امضا کنیم (یک نگاه کوتاه در Zig)

در Zig، هنگامی که دانه سی و دو بایتی مشتق شده از بیست و چهار کلمه را داشته باشید، امضای یک پیام با Ed25519 در چند خط جا می‌گیرد:

```
const std = @import("std");
const Ed25519 = std.crypto.sign.Ed25519;

// 'La 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);
```

عملیات امضا شصت و چهار بایت تولید می‌کند —که امضا نامیده می‌شود— که فقط می‌توانسته از کلید خصوصی متناظر تولید شده باشد. تأیید عمومی است: هر کسی با کلید عمومی می‌تواند بررسی کند که امضا با پیام مطابقت دارد. بدون کلید خصوصی، هیچ‌کس نمی‌تواند امضای معتبری برای آن پیام تولید کند؛ با کلید عمومی، همه می‌توانند تشخیص دهند که آیا یک امضا معتبر است یا خیر. این عدم تقارن همان چیزی است که به امضاکننده اجازه می‌دهد بدون به اشتراک گذاشتن راز، اصالت خود را ثابت کند.

مثال قبلی نسخه حداقلی دفترچه راهنما است. در کد واقعی Solo2، زنجیره از دو فایل عبور می‌کند: یکی در JavaScript که در مرورگر کاربر قرار دارد و آنتروپی را از بیست و چهار کلمه بازسازی می‌کند، و دیگری در Zig در کتابخانه *zcatcrypto* که آن آنتروپی را می‌گیرد و کلیدهای رمزنگاری خاص را مشتق می‌کند. با شروع از سمت مرورگر:

```
                                                    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')
                                                                                {

                                       .palabras = 264 bits. Los primeros 256 son la entropía 24 //
                                                ;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 }
                                                                            {
```

آن سی و دو بایت آنتروپی، همراه با سی و دو بایت دیگر که در همان مرحله مشتق شده‌اند، به ماژول WebAssembly در Zig می‌روند که کلیدهای Ed25519 واقعی را تولید می‌کند. تابع کامل، با پاکسازی نهایی حافظه، در یک صفحه جا می‌شود:

```
                                                    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
                                                                      {
```

دو جزئیات ارزش اشاره دارند. اول: یک دانه (seed) واحد همیشه همان جفت کلید را تولید می‌کند — دقیقاً همین است که امکان بازیابی هویت را با وارد کردن بیست و چهار کلمه در یک دستگاه جدید فراهم می‌کند. دوم: دانه به صراحت در آخرین خط از حافظه پاک می‌شود. پس از آن نقطه، حتی خود تابع هم نمی‌تواند کلیدها را بازسازی کند؛ کلمات کاربر تنها منبع خواهند بود.

برای کسانی که می‌خواهند آن را با اعداد کوچک بررسی کنند. طرح امضا را می‌توان به طور کامل با اعدادی پیمود که به اندازه کافی کوچک هستند تا محاسبات به صورت دستی انجام شود. کسانی که ترجیح می‌دهند وارد ریاضیات نشوند، می‌توانند این بلوک را بدون از دست دادن رشته مطلب مقاله نادیده بگیرند؛ کسانی که می‌خواهند مکانیسم را گام به گام در حال کار ببینند، آن را در اینجا خواهند یافت. **قوانین عمومی**، که هر کسی می‌تواند بخواند: یک عدد اول $p = 23$ (در Ed25519 واقعی حدود هفتاد و هفت رقم است؛ ما از بیست و سه استفاده می‌کنیم تا محاسبات در یک صفحه جا شود)، یک پایه $g = 2$ که مرتبه آن در این گروه $q = 11$ است، و این قرارداد که تمام محاسبات با g به صورت *módulo p* انجام می‌شود و تمام توان‌ها *módulo q* کاهش می‌یابند. **انتخاب خصوصی**، تنها یکی و هرگز به اشتراک گذاشته نشده: راز $x = 6$. این همان هویت است.

گام ۱ — بخش عمومی هویت. یک بار محاسبه می‌شود و به طور آشکار منتشر می‌گردد.

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

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

بخش عمومی هویت **18** است. هر کسی می‌تواند آن را بردارد و برای تأیید امضاهای ساخته شده با این هویت استفاده کند. هیچ‌کس، تنها با مشاهده عدد 18، نمی‌تواند راز 6 را بازیابی کند: این همان مسئله لگاریتم گسسته است که در پایان به آن بازخواهیم گشت.

گام ۲ — امضای یک پیام. دارنده هویت می‌خواهد پیام $m = 7$ را امضا کند. او با انتخاب یک مقدار تصادفی جدید $k = 4$ شروع می‌کند که فقط یک بار استفاده می‌شود و هرگز به اشتراک گذاشته نخواهد شد (در Ed25519 واقعی، k به طور قطعی از پیام و راز مشتق می‌شود تا از خطر استفاده مجدد جلوگیری شود، اما نقشی که ایفا می‌کند دقیقاً همین است). سپس سه عدد را محاسبه می‌کند:

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

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

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

امضا جفت **(r, s) = (16, 10)** است. همراه با پیام به صورت آشکار منتقل می‌شود. هر کسی می‌تواند آن را بخواند. نکته آموزشی: در Ed25519 واقعی تابع H همان SHA-512 است که از نظر رمزنگاری قدرتمند است؛ در اینجا ما از ساده‌سازی $e = (r + m) \bmod q$ استفاده می‌کنیم تا خواننده بتواند مراحل را بدون نیاز به محاسبه هش طی کند. ساختار الگوریتم یکسان است.

گام ۳ — تأیید امضا. تأییدکننده بخش عمومی $y = 18$، پیام $m = 7$ و امضای $(r, s) = (16, 10)$ را در اختیار دارد. او e را به همان روش بازسازی می‌کند — $e = (16 + 7) \bmod 11 = 1$ — و بررسی می‌کند که آیا این برابری برقرار است یا خیر:

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

هر دو طرف را به طور جداگانه محاسبه می‌کند:

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

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

هر دو طرف عدد **12** را می‌دهند. امضا معتبر است. هر کسی با داشتن بخش عمومی 18 می‌تواند به این نتیجه برسد بدون اینکه هرگز بداند راز عدد 6 بوده است.

و نفر سومی که سعی در جعل داشته باشد؟ اوا تمام موارد عمومی را که از کانال عبور می‌کنند دیده است: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. برای امضای یک پیام *متفاوت* به نام این هویت، او نیاز به دانستن x دارد. تنها راه او این است که از خود بپرسد: «برای کدام توان x رابطه $2^x \bmod 23 = 18$ برقرار است؟». با $p = 23$ او می‌تواند 0، 1، 2، 3، ... را امتحان کند و در عرض چند ثانیه آن را بیابد. اما با جایگزینی 23 با یک عدد اول در ابعاد واقعی Ed25519، فضای توان‌های ممکن از تعداد اتم‌های جهان قابل مشاهده فراتر می‌رود. **امروزه هیچ الگوریتمی شناخته شده برای بشر وجود ندارد که بتواند آن فضا را در کمتر از میلیاردها سال پیمایش کند.** این همان مسئله لگاریتم گسسته است که زیربنای Diffie-Hellman در مقاله قبلی است و در اینجا برای طرح امضا به کار گرفته شده است.

آنچه اکنون پیمودیم *دقیقاً* Schnorr است، طرح امضایی که Ed25519 نوعی از آن است که برای یک منحنی بیضوی تطبیق یافته است. در Ed25519 واقعی، تمام عملیات روی نقاط یک منحنی خاص (Curve25519) به جای اعداد صحیح به پیمانه یک عدد اول انجام می‌شود و تابع H به جای مجموع ساده‌ای که در بالا استفاده کردیم، SHA-512 است. این دو جایگزینی تنظیمات پیاده‌سازی هستند — برای به دست آوردن مقاومت رمزنگاری در برابر حمله جستجوی فراگیر (brute force) و به دست آوردن ویژگی‌های امنیتی اضافی برای k —. ساختار الگوریتمی، سه عملیات و دلیل عدم تقارن، یکسان هستند.

در اینجا توقفی کوتاه لازم است، زیرا کل زنجیره ممکن است با یک نگاه سریع با یکی دیگر از اعضای این سه گانه اشتباه گرفته شود: هش. این‌طور نیست. هش تابعی منحصر به فرد است که فشرده‌سازی می‌کند — بایت‌های زیادی وارد می‌شوند، یک ردپای کوتاه خارج می‌شود و مسیر در همان‌جا به پایان می‌رسد. هویت رمزنگاری یک جفت ریاضی مکمل است: راز می‌ماند و امضا می‌کند؛ همتای عمومی آن منتشر می‌شود و تأیید می‌کند. در جایی که هش اطلاعات را در یک جهت فرو می‌ریزد، هویت عدم تقارنی را بین دو نیمه برقرار می‌کند. هش گواهی می‌دهد که چه چیزی گفته شده است؛ هویت گواهی می‌دهد که چه کسی آن را گفته است.

آنچه عبارت نیست

سه اشتباه رایج را باید برطرف کرد. عبارت به معنای واقعی کلمه یک رمز عبور نیست: با اثر انگشت ذخیره شده در سرور مقایسه نمی‌شود؛ بلکه برای بازسازی ریاضی هویت، در دستگاه کاربر وارد می‌شود. عبارت قابل بازیابی نیست: اگر گم شود، کسی نیست که آن را از او بخواهید؛ اگر کپی شود، هویت نیز کپی می‌شود. عبارت یک اعتبارنامه جدا شدنی از هویت نیست: عبارت خودِ هویت *است*. هر کسی آن را داشته باشد می‌تواند به عنوان آن هویت عمل کند، بدون اجازه اضافی، بدون فرآیند مجوز و بدون امکان بازیابی.

این ویژگی سوم است که وزن موضوع را تغییر می‌دهد. رمز عبور گم شده یک دردسر اداری است. هویت رمزنگاری گم شده، خودِ هویت است. کاغذی با عبارت که توسط اشخاص ثالث پیدا شود، خطر سرقت حساب نیست: بلکه تحویل کل هویت است. وعده سیستم —اینکه هیچ‌کس نتواند هویت شما را باطل کند یا شما را به طور خودسرانه مسدود کند— به طور جدایی‌ناپذیری با مسئولیت همراه است —اینکه شما تنها نگهبان چیزی هستید که هیچ‌کس نمی‌تواند آن را برای شما بازگرداند.

وعده و وزن

مدل هویت رمزنگاری معمولاً با صفت *خود-حاکم* —self-sovereign در ادبیات انگلیسی— توصیف می‌شود. انتخاب کلمه آگاهانه است و وضعیت را با دقت زیادی توصیف می‌کند. کاربر به معنای تقریباً قرون وسطایی بر هویت خود حاکم است: هیچ پادشاهی، هیچ صادرکننده‌ای، هیچ مرجع مرکزی آن را اعطا نمی‌کند؛ و هیچ یک از موارد فوق نیز نمی‌توانند آن را پس بگیرند. اما کاربر نیز مانند پادشاه قرون وسطایی، تمام عواقب اشتباهات خود را بر عهده دارد: اگر مهر را گم کند، هیچ نایب‌السلطنه‌ای وجود ندارد که به جای او تصمیم بگیرد.

انتخاب بین هویت مدیریت شده توسط شخص ثالث و هویت خود-حاکم پاسخ صحیح جهانی واحدی ندارد. برای یک حساب کاربری در انجمنی بی‌اهمیت، هویت مدیریت شده احتمالاً با خطر متناسب است. اما برای یک هویت حرفه‌ای که اسناد الزام‌آور قانونی را امضا می‌کند، برای یک هویت اقتصادی که از پس‌اندازهای شخصی محافظت می‌کند، برای یک هویت ارتباط حرفه‌ای با مشتریانی که اطلاعات حساسی را به امانت گذاشته‌اند، موضوع تغییر می‌کند. در آنجا دیگر سوال این نیست که «آیا راحت است؟» بلکه این است که «چه کسی به جز من، قدرت عمل به عنوان من را دارد و تحت چه شرایطی؟».

این سازوکار در کجا در سیستم‌های واقعی ظاهر می‌شود

BIP39 در سال ۲۰۱۳ در دنیای Bitcoin متولد شد و به سرعت در کل اکوسیستم ارزهای دیجیتال گسترش یافت: امروزه هر کیف پول جدی یک عبارت BIP39 دوازده یا بیست و چهار کلمه‌ای را به عنوان پشتیبان هویت اقتصادی دارنده آن می‌پذیرد. خارج از ارزهای دیجیتال، همان مفهوم اساسی — جفت رمزنگاری که مالکیت را بدون واسطه اثبات می‌کند — در سیستم‌های دیگر با دستور زبان متفاوت ظاهر می‌شود. کلیدهای SSH که یک مدیر سیستم برای دسترسی به سرورهای خود استفاده می‌کند، یک مورد کلاسیک است: یک کلید خصوصی که مدیر در دستگاه خود نگه می‌دارد و یک کلید عمومی که در هر سرور کپی می‌شود؛ هیچ نهاد قابل مقایسه با یک سرویس متمرکز مداخله نمی‌کند. پروتکل Signal از Ed25519 با متریال کلید پایدار در دستگاه استفاده می‌کند؛ eIDAS اروپایی، در بخش امضای واجد شرایط خود، بر همان اصل رمزنگاری استوار است، با این تفاوت که کلید به جای کاربر توسط یک ارائه دهنده خدمات اعتماد واجد شرایط نگهداری می‌شود.

Solo2، پلتفرم ناشر این نشریه، از یک عبارت BIP39 بیست و چهار کلمه‌ای به عنوان هویت هر کاربر استفاده می‌کند. کاربر هنگام ایجاد حساب خود، کلمات را یک بار می‌بیند. آنها در هیچ سرور Solo2 یا هیچ کس دیگری ذخیره نمی‌شوند: اگر کاربر آنها را یادداشت کرده و نگه دارد، هویت خود را برای همیشه حفظ می‌کند. اگر آنها را گم کند، آنها را از دست داده است. این نتیجه منطقی معماری بدون اپراتور در میان است: اگر Solo2 می‌توانست هویت را به کاربری که آن را گم کرده بازگرداند، می‌توانست آن را به هر کسی که به Solo2 فشار می‌آورد تا آن را دریافت کند، بدهد.

برای خواننده حرفه‌ای

چهار نکته برای کسانی که پذیرش هویت رمزنگاری شده خود-حاکم (autosoberana) را در یک زمینه حرفه‌ای ارزیابی می‌کنند:

1. عبارت همان هویت است. نگهداری فیزیکی — کاغذ، چندین نسخه در مکان‌های مختلف، در نهایت فلز حکاکی شده برای استفاده طولانی مدت — تضمین‌های بیشتری نسبت به نگهداری دیجیتال ارائه می‌دهد که بدون کاهش خطر گم شدن، سطح حمله را افزایش می‌دهد.
2. هیچ بازیابی وجود ندارد. طراحی فرآیند با این فرض که روزی نسخه اصلی گم می‌شود، بسیار عاقلانه‌تر از کشف آن در روز گم شدن است. نسخه دوم که از نظر جغرافیایی جدا شده است، تقریباً همه سناریوها را حل می‌کند.
3. این همان گواهی واجد شرایط eIDAS نیست. برای امضای واجد شرایط در اتحادیه — اسناد رسمی، برخی مراحل با مدیریت — قانون یک ارائه دهنده واجد شرایط را می‌طلبد که کلید را نگه دارد. هویت رمزنگاری شده خود-حاکم (autosoberana) برای ارتباطات حرفه‌ای و امضای اسناد با ارزش اثباتی مفید است، اما به طور خودکار جایگزین گواهی واجد شرایط در مواردی که استاندارد آن را می‌طلبد، نمی‌شود.
4. اگر قرار است هویت منتقل شود — ارث، جانشینی حرفه‌ای، پایان فعالیت — بهتر است رویه را قبل از آن آماده کنید، نه بعد از آن. رویه‌های رسمی با پاکت‌های مهر و موم شده با موم (lacre)، دستورالعمل به وصی، سپرده‌گذاری در دفتر اسناد رسمی، ترتیبات کلاسیکی هستند که کاملاً با ماهیت رمزنگاری شده دارایی سازگار هستند.

این مقاله سه‌گانه مفهومی را که چرخه را آغاز کرد — hash، رمزگذاری، هویت — می‌بندد. این سه ایده بر روی یکدیگر ساخته شده‌اند: hash اثر انگشت تغییرناپذیر را می‌دهد، رمزگذاری محرمانگی بدون شخص ثالث قابل اعتماد را می‌دهد، هویت مالکیت بدون شخص ثالث اعطاکننده را می‌دهد. هر سه دارای ویژگی هستند که ایدئولوژیک هم نیست: آنها توانایی‌های فنی را که به طور سنتی در اختیار اپراتور بود، از مدیر سرویس به کاربر آن

منتقل می‌کنند. آنها همچنین مسئولیت‌هایی را با خود منتقل می‌کنند. صحبت صادقانه در مورد هر یک از این سه مستلزم صحبت در مورد دو مورد دیگر نیز هست.

منابع و مطالعه بیشتر

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*، پیشنهاد بهبود Bitcoin در سال ۲۰۱۳. استاندارد دو فاکتو برای عبارت‌های بازیابی در صنعت رمزنگاری.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA)، شامل Ed25519. IETF، ژانویه ۲۰۱۷. مشخصات هنجاری طرح امضای مورد استفاده در بخش بزرگی از صنعت معاصر.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification، نسخه ۲.۰. IETF، سپتامبر ۲۰۰۰. الگوریتم PBKDF2 مورد استفاده در مشتق BIP39 از عبارت به دانه (seed) را تعریف می‌کند.
- مقررات (اتحادیه اروپا) ۹۱۰/۲۰۱۴ (eIDAS) و تکامل آن توسط مقررات (اتحادیه اروپا) ۲۰۲۴/۱۱۸۳ (eIDAS 2) — چارچوب اروپایی برای هویت الکترونیکی و امضای واجد شرایط. رژیمی متفاوت از خود-حاکم، اما از نظر مفهومی بر پایه همان اصول رمزنگاری.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). متنی کلاسیک درباره اصول و تعهدات مدل خود-حاکم، قدیمی‌تر اما مرتبط برای درک خانواده راه‌حل‌های معاصر.

← السابقمدل کسب‌وکار به عنوان سیگنال اعتمادالتالي → Self-hosting به عنوان یک فعالیت حرفه‌ای

قراءات حديثة

- تأمل ۰ ۲۹ ژوئن ۲۰۲۶ شما ناشناس نیستید
- تامل ۰ ۲۷ مۀ ۲۰۲۶ آنچه یک امضا نمی‌تواند اصلاح کند
- تحلیل ۰ ۲۶ مۀ ۲۰۲۶ حریم خصوصی واقعی در مقابل ظاهری: سؤالاتی که باید از خود بپرسید

این مقاله را هر کجا که نیاز دارید همراه خود ببرید.

↓ مارک‌داون ↓ متن ساده ↓ PDF

فایل در دستگاه شما دانلود خواهد شد. از آنجا می‌توانید آن را ذخیره کنید، به Solo2 وارد کنید یا در هر کجا که می‌خواهید به اشتراک بگذارید. Cuadernos مقصد را برای شما تعیین نمی‌کند.

ختم شمعي · SHA-256 04909b4128d196c446b81ab238b5a9c5dc831b70f7f9579bc9315c3643526450

Cuadernos Lacre · نشريّة من Menzuri Gestión S.L. ·
كتبها R.Eugenio · حرّرها فريق Solo2.