Conceito · 23 de maio de 2026

← Anterior·Seguinte →

As 24 palavras: o que é uma identidade criptográfica

Uma identidade criptográfica não é uma senha: nenhum servidor a guarda e não se recupera. Uma explicação didática do mecanismo BIP39, por que exatamente vinte e quatro palavras, e que peso real recai sobre quem as possui.

Para nos entendermos: Se esquecer a sua senha do Gmail, a Google redefine-a para si. Se perder as 24 palavras que compõem uma identidade criptográfica, não há a quem as pedir. Não é que o procedimento seja rigoroso — é que não existe ninguém na outra ponta. Essa diferença é toda a diferença.

A diferença entre uma senha e uma identidade

Uma senha, no modelo clássico da internet, não é a identidade do utilizador. É um comprovativo. O utilizador tem uma identidade — um nome, um e-mail, um número de cliente — e, para demonstrar perante um servidor que é quem diz ser, apresenta uma senha que o servidor compara com uma pegada que tinha armazenada. Se as pegadas coincidirem, o servidor concede a sessão. Se a senha for perdida, o utilizador continua a ser o mesmo utilizador; o que perde é o comprovativo, e existe um procedimento de recuperação — um e-mail para o endereço registado, uma pergunta de segurança — para o restituir.

Uma identidade criptográfica funciona de outra maneira. Não é uma credencial que alguém compare com uma pegada armazenada; *é* um segredo matemático completo em si mesmo. Não importa onde resida — num papel, num dispositivo, até num servidor alheio —: a identidade existe pela sua matemática, não por quem a valida. Aqui aparece uma propriedade parecida com a que vimos em «O que é realmente o SHA-256»: a posse não se demonstra exibindo o segredo, mas usando-o para assinar. A assinatura assim produzida qualquer pessoa pode verificar com um valor público que deriva matematicamente do próprio segredo, sem necessidade de conhecer o segredo em si, e sem que um terceiro medie a verificação. Quem tem o segredo, é a identidade; quem o perde, deixa de o ser. A sentença é categórica: **não há ninguém a quem pedir que lhe devolva a identidade. Esse alguém não existe, porque não a possuía em primeiro lugar.**

O que representam vinte e quatro palavras

A identidade criptográfica é habitualmente representada por um segredo matemático de trinta e dois bytes — duzentos e cinquenta e seis bits. Um número difícil de reter e ainda mais difícil de transcrever sem erro. A indústria criptográfica resolveu este problema em 2013 com um padrão pequeno e elegante chamado BIP39: uma forma de representar esses duzentos e cinquenta e seis bits como uma sequência de vinte e quatro palavras retiradas de uma lista oficial de duas mil quarenta e oito. A aritmética por trás encaixa com elegância; quem quiser vê-la em detalhe encontra-a na margem.

A conta começa pelo fim. Queremos representar os duzentos e cinquenta e seis bits do segredo adicionando oito bits de soma de verificação (checksum): duzentos e sessenta e quatro bits no total. Se os repartirmos em vinte e quatro palavras — um número manuseável para anotar e ditar sem perda —, cada palavra deve fornecer exatamente onze bits de informação. E onze bits são dois elevado a onze possibilidades, ou seja, duas mil quarenta e oito. Daí que o vocabulário oficial BIP39 tenha precisamente esse tamanho: a lista existe à medida do problema, não ao contrário.

A conta não é decorativa. Se alguém transcrever vinte e três palavras corretamente e se equivocar na vigésima quarta, a soma de verificação detetará o erro: o software dirá "esta sequência não é válida". Se alguém transcrever as vinte e quatro corretas, o software derivará a mesma identidade sem ambiguidade. A escolha da lista de palavras também é deliberada: as palavras do vocabulário BIP39 são curtas, distintas entre si, sem diacríticos, escolhidas para minimizar confusões fonéticas e ortográficas. É um vocabulário desenhado para ser lembrado, escrito e ditado por seres humanos sem perda.

Da frase à chave

As vinte e quatro palavras não são a chave criptográfica que assina mensagens. São uma representação recuperável da entropia original que, através de um processo determinístico chamado PBKDF2, se transforma numa semente (seed) de sessenta e quatro bytes. Dessa semente derivam, também de forma determinística, as chaves criptográficas concretas que o usuário utiliza: uma chave privada para assinar e uma chave pública correspondente que é publicada para verificar as assinaturas. O mesmo mecanismo em sistemas diferentes: as criptomoedas usam a curva secp256k1; o protocolo Signal e muitos sistemas modernos usam Ed25519 sobre a curva Curve25519. Para uma curva concreta como Ed25519, os padrões BIP32 e SLIP-0010 pegam nessa semente de sessenta e quatro bytes e derivam, de forma determinística, os trinta e dois bytes que constituem a chave de assinatura efetiva — os mesmos trinta e dois bytes com os quais começa o exemplo de código da próxima seção.

Esta é a forma padrão como toda a indústria apresenta o mecanismo ao usuário —carteiras de criptomoedas, gestores de identidade descentralizada, o Signal na sua parte de identidade persistente, o Solo2 entre eles—: o usuário, na prática, nunca vê a semente nem as chaves derivadas. Vê as vinte e quatro palavras ao criar a sua identidade e, opcionalmente, anota-as num papel. As palavras viajam depois entre os seus dispositivos quando deseja migrar a identidade: insere-as na nova aplicação, a aplicação deriva a mesma semente, as mesmas chaves, a mesma identidade. É um mecanismo portátil, criptograficamente sólido e, dentro dos limites do razoável, memorizável.

Como assinar com a chave (uma pincelada de Zig)

Em Zig, uma vez que se tem a semente de trinta e dois bytes derivada das vinte e quatro palavras, assinar uma mensagem com Ed25519 cabe em poucas linhas:

```
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);
```

A operação de assinar produz sessenta e quatro bytes —chamados assinatura— que só poderiam ter sido gerados a partir da chave privada correspondente. A verificação é pública: qualquer pessoa com a chave pública pode verificar se a assinatura corresponde à mensagem. Sem a chave privada, ninguém pode produzir uma assinatura válida para essa mensagem; com la chave pública, todos podem detetar se uma assinatura é válida. Essa assimetria é o que permite ao signatário demonstrar autoria sem compartilhar o segredo.

O exemplo anterior é a versão mínima de manual. No código real do Solo2, a cadeia atravessa dois ficheiros, um em JavaScript que vive no navegador do utilizador e reconstrói a entropia a partir das vinte e quatro palavras, outro em Zig dentro da biblioteca *zcatcrypto* que toma essa entropia e deriva as chaves criptográficas concretas. Começando pelo lado do navegador:

```
// 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 };
}
```

Esses trinta e dois bytes de entropia, juntamente com outros trinta e dois derivados no mesmo passo, viajam para o módulo WebAssembly do Zig que gera as chaves Ed25519 propriamente ditas. A função completa, com a sua limpeza de memória final, cabe num ecrã:

```
// 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;
}
```

Vale a pena assinalar dois detalhes. O primeiro: uma mesma semente produz sempre o mesmo par de chaves — é exatamente isso que permite recuperar a identidade introduzindo as vinte e quatro palavras num novo dispositivo. O segundo: a semente é apagada explicitamente da memória na última linha. Passado esse ponto, nem a própria função poderia reconstruir as chaves; as palavras do utilizador seriam a única origem.

Para quem quiser comprovar com números pequenos. O esquema de assinatura pode ser percorrido na íntegra com algarismos suficientemente reduzidos para fazer as contas à mão. Quem preferir não entrar em aritmética pode saltar este bloco sem perder o fio do artigo; quem quiser ver o mecanismo a funcionar passo a passo encontrá-lo-á aqui. **As regras públicas**, que qualquer pessoa pode ler: um primo $p = 23$ (no Ed25519 real é de cerca de setenta e sete algarismos; usamos vinte e três para que as contas caibam numa página), uma base $g = 2$ cuja ordem neste grupo é $q = 11$, e a convenção de que toda a aritmética com g é feita *módulo p* e todos os expoentes são reduzidos *módulo q*. **A escolha privada**, uma só e jamais partilhada: o segredo $x = 6$. Essa é a identidade.

Passo 1 — A parte pública da identidade. É calculada uma vez e publicada abertamente.

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

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

A parte pública da identidade é **18**. Qualquer pessoa pode tomá-la e usá-la para verificar assinaturas feitas com esta identidade. Ninguém, observando apenas o 18, pode recuperar o segredo 6: esse é o problema do logaritmo discreto ao qual voltaremos no final.

Passo 2 — Assinar uma mensagem. O detentor da identidade quer assinar a mensagem $m = 7$. Começa por escolher um valor aleatório novo $k = 4$, que será usado uma única vez e não será jamais partilhado (no Ed25519 real, k é derivado deterministicamente da mensagem e do segredo para evitar o perigo de reutilização, mas o papel que desempenha é exatamente este). Depois calcula três números:

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

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

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

A assinatura é o par **(r, s) = (16, 10)**. Viaja em aberto junto da mensagem. Qualquer pessoa pode lê-la. Nota didática: no Ed25519 real a função H é SHA-512, criptograficamente robusta; aqui usamos a simplificação $e = (r + m) \bmod q$ para que o leitor possa percorrer os passos sem necessidade de calcular um hash. A estrutura do algoritmo é a mesma.

Passo 3 — Verificar a assinatura. O verificador tem a parte pública $y = 18$, a mensagem $m = 7$, e a assinatura $(r, s) = (16, 10)$. Reconstrói e da mesma forma — $e = (16 + 7) \bmod 11 = 1$ — e verifica se esta igualdade se cumpre:

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

Calcula os dois lados separadamente:

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

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

Os dois lados dão **12**. A assinatura é válida. Qualquer pessoa com a parte pública 18 pode chegar a esta conclusão sem ter sabido nunca que o segredo era 6.

E um terceiro que tentasse falsificar? Eva viu passar pelo canal tudo o que é público: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Para assinar uma mensagem *diferente* em nome desta identidade, ela precisaria de conhecer x. A sua única via é perguntar-se: «para que expoente x se cumpre $2^x \bmod 23 = 18$?». Com $p = 23$ ela pode testar 0, 1, 2, 3, … e encontrá-lo em segundos. Mas ao substituir 23 por um primo das dimensões reais do Ed25519, o espaço de expoentes possíveis supera o número de átomos do universo observável. **Não existe hoje em dia nenhum algoritmo conhecido pela humanidade que possa percorrer esse espaço em menos de mil milhões de anos.** É o mesmo problema do logaritmo discreto que sustenta o Diffie-Hellman do artigo anterior, aplicado aqui ao esquema de assinatura.

Isto que acabámos de percorrer é *exatamente* Schnorr, o esquema de assinatura do qual o Ed25519 é uma variante adaptada a uma curva elíptica. No Ed25519 real, todas as operações são feitas sobre os pontos de uma curva concreta (Curve25519) em vez de sobre números inteiros módulo um primo, e a função H é SHA-512 em vez da soma simplificada que usamos acima. As duas substituições são ajustes de implementação — ganhar resistência criptográfica à força bruta, ganhar propriedades adicionais de segurança para k —. A estrutura algorítmica, as três operações, o porquê da assimetria, são as mesmas.

Convém aqui uma breve paragem, porque a cadeia inteira pode ser confundida, num olhar rápido, com outra primitiva do trio: o hash. Não o é. Um hash é uma função única que comprime — entram muitos bytes, sai uma pegada curta, aí acaba o caminho. Uma identidade criptográfica é um par matemático complementar: o segredo fica e assina; a sua contraparte pública é publicada e verifica. Onde o hash colapsa informação num sentido único, a identidade estabelece uma assimetria entre duas metades. O hash atesta o que foi dito; a identidade atesta quem o disse.

O que a frase não é

Convém esclarecer três equívocos frequentes. A frase não é uma senha no sentido próprio: não é comparada com uma impressão digital armazenada num servidor; é inserida no dispositivo do usuário para reconstruir matematicamente a identidade. A frase não se recupera: se for perdida, não há ninguém a quem a pedir; se for duplicada, a identidade também é duplicada. A frase não é uma credencial separável da identidade: a frase *é* a identidade. Quem a possui pode agir como tal, sem permissão adicional, sem processo de autorização, sem recuperação possível.

Esta terceira propriedade é a que muda o peso do assunto. Uma senha perdida é um transtorno administrativo. Uma identidade criptográfica perdida é a identidade. Um papel com a frase encontrado por terceiros não é um risco de roubo de conta: é a entrega da identidade inteira. A promessa do sistema — que ninguém possa revogar a sua identidade ou bloqueá-lo arbitrariamente — vem acompanhada inseparavelmente da responsabilidade — que você é o único guardião de algo que ninguém pode restituir por você.

A promessa e o peso

O modelo de identidade criptográfica costuma receber o qualificativo de *autossoberana* —self-sovereign na literatura anglo-saxónica—. A escolha da palavra é deliberada e descreve a condição com bastante exatidão. O usuário é soberano sobre a sua identidade num sentido quase medieval: não é concedida por nenhum rei, nenhum emissor, nenhuma autoridade central; tampouco pode ser retirada por nenhum dos anteriores. Mas também, como o monarca medieval, o usuário carrega com a consequência inteira dos seus erros: não há regente que tome decisões no seu lugar se perder o selo.

A escolha entre uma identidade gerida por um terceiro e uma identidade autossoberana não tem uma resposta universal correta. Para a conta de um fórum irrelevante, a identidade gerida é provavelmente proporcional ao risco. Para uma identidade profissional que assina documentos juridicamente vinculativos, para uma identidade económica que guarda as próprias poupanças, para uma identidade de comunicação profissional com clientes que confiaram informações sensíveis, a questão muda. Aí a pergunta deixa de ser «é cómodo?» e torna-se «quem, além de mim, tem o poder de agir como eu, e em que circunstâncias?».

Onde aparece este mecanismo em sistemas reais

O BIP39 nasceu no mundo do Bitcoin em 2013 e espalhou-se rapidamente para todo o ecossistema das criptomoedas: qualquer carteira séria aceita hoje uma frase BIP39 de doze ou vinte e quatro palavras como backup da identidade económica do seu detentor. Fora das criptomoedas, o mesmo conceito subjacente — par criptográfico que prova a autoria sem intermediário — aparece noutros sistemas com sintaxe diferente. As chaves SSH que um administrador de sistemas usa para aceder aos seus servidores são um caso clássico: uma chave privada que o administrador guarda na sua máquina e uma pública que é copiada para cada servidor; ninguém comparável a um serviço centralizado intervém. O protocolo Signal usa Ed25519 com material de chave persistente no dispositivo; as eIDAS europeias, na sua parte de assinatura qualificada, descansam sobre o mesmo princípio criptográfico, com a diferença de que a chave é custodiada por um prestador de serviços de confiança qualificado em vez do utilizador.

Solo2, plataforma editora desta publicação, usa uma frase BIP39 de vinte e quatro palavras como identidade de cada utilizador. O utilizador, ao criar a sua conta, vê as palavras uma vez. Não são armazenadas em nenhum servidor da Solo2 nem de ninguém: se o utilizador as anota e as custodia, mantém a sua identidade para sempre. Se as perder, perdeu-as. É a consequência coerente com a arquitetura de não haver operador no meio: se a Solo2 pudesse devolver a identidade ao utilizador que a perdeu, poderia também dá-la a quem pressionar a Solo2 para que lha dê.

Para o leitor profissional

Quatro considerações para quem avalia adotar a identidade criptográfica autossoberana (autosoberana) num contexto profissional :

1. A frase é a identidade. Custódia física — papel, várias cópias em lugares diferentes, eventualmente metal gravado para uso a longo prazo — oferece mais garantias do que a custódia digital, que acrescenta superfície de ataque sem reduzir o risco de perda.
2. Não há recuperação. Desenhar o processo assumindo que um dia se perde a cópia primária convém muito mais do que descobri-lo no dia em que se perde. Uma segunda cópia geograficamente separada resolve quase todos os cenários.
3. Não é o mesmo que um certificado qualificado eIDAS. Para assinatura qualificada na União — escrituras notariais, certos trâmites com a Administração — a legislação exige um fornecedor qualificado que custodie a chave. A identidade criptográfica autossoberana serve para a comunicação profissional e a assinatura documental com valor probatório, mas não substitui automaticamente o certificado qualificado nos casos onde a norma o exige.
4. Se a identidade vai ser transferida — herança, sucessão profissional, encerramento de atividade — convém preparar o procedimento antes, não depois. Procedimentos formais com envelopes selados com lacre (lacre), instruções a um executor, depósito em cartório, são arranjos clássicos perfeitamente compatíveis com a natureza criptográfica do ativo.

Este artigo fecha o trio conceptual que abriu o ciclo — hash, cifragem, identidade —. As três ideias constroem-se umas sobre as outras: o hash dá a pegada inalterável, a cifragem dá a confidencialidade sem terceiro de confiança, a identidade dá a autoria sem terceiro de concessão. As tres partilham uma propriedade que também não é ideológica: trasladam, de quem gere um serviço para quem o usa, capacidades técnicas que

tradicionalmente residiam no operador. Trasladam com elas também responsabilidades. Falar com honestidade de qualquer uma das três exige falar também das outras duas.

Fontes e leitura adicional

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, proposta de melhoria do Bitcoin de 2013. Padrão de facto para frases de recuperação na indústria criptográfica.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), incluindo Ed25519. IETF, janeiro de 2017. Especificação normativa do esquema de assinatura usado em boa parte da indústria contemporânea.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, versão 2.0. IETF, setembro de 2000. Define o algoritmo PBKDF2 usado na derivação BIP39 de frase a semente (seed).
- Regulamento (UE) 910/2014 (eIDAS) e a sua evolução pelo Regulamento (UE) 2024/1183 (eIDAS 2) — quadro europeu de identidade eletrónica e assinatura qualificada. Regime distinto do autossoberano, mas conceptualmente apoiado nos mesmos primitivos criptográficos.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Texto canónico sobre os princípios e compromissos do modelo autossoberano, anterior mas relevante para a compreensão da família de soluções contemporâneas.

← AnteriorO modelo de negócio como sinal de confiançaSeguinte →Self-hosting como prática profissional

Leituras recentes

- Reflexão · 29 de junho de 2026 Não és anónimo
- Reflexão · 27 de maio de 2026 Lo que una firma no puede arreglar
- Análise · 26 de maio de 2026 Privacidade real vs aparente: as perguntas que convém fazer-se

Leve este artigo para onde precisar.

↓ Markdown ↓ Texto simples ↓ PDF

O arquivo é descarregado no seu dispositivo. A partir daí, pode guardá-lo, importá-lo no Solo2 ou partilhá-lo onde quiser. Cuadernos não decide o destino por si.

Selo de lacre · SHA-256 09a015ef34429a0db23dff4501f24fdc034f5ff8530f247a4482f3f9c18e0acb

Cuadernos Lacre · Uma publicação da Menzuri Gestión S.L. ·
escrita por R.Eugenio · editada pela equipa do Solo2.