24 слова: що таке криптографічна ідентичність

Криптографічна ідентичність — це не пароль: жоден сервер її не зберігає, і вона не відновлюється. Дидактичне пояснення механізму BIP39, чому саме двадцять чотири слова і яка реальна відповідальність лягає на того, хто ними володіє.

Щоб ми розуміли один одного: якщо ви забудете пароль від Gmail, Google скине його для вас. Якщо ви втратите 24 слова, з яких складається криптографічна ідентичність, їх нема в кого просити. Справа не в тому, що процедура сувора, а в тому, що на іншому кінці нікого немає. У цьому й полягає вся різниця.

Різниця між паролем та ідентичністю

Пароль у класичній моделі інтернету не є ідентичністю користувача. Це квитанція. У користувача є ідентичність — ім'я, електронна пошта, номер клієнта — і, щоб довести серверу, що він той, за кого себе видає, він пред'являє пароль, який сервер порівнює з відбитком, що зберігається. Якщо відбитки збігаються, сервер дозволяє сесію. Якщо пароль втрачено, користувач залишається тим самим користувачем; він втрачає лише квитанцію, і існує процедура відновлення — лист на зареєстровану адресу, секретне питання — для її заміни.

Криптографічна ідентичність працює інакше. Це не облікові дані, які хтось порівнює з відбитком, що зберігається; це *сам по собі* повний математичний секрет. Неважливо, де він знаходиться — на папері, у пристрої чи навіть на чужому сервері: ідентичність існує завдяки своїй математиці, а не тому, хто її підтверджує. Тут проявляється властивість, схожа на ту, що ми бачили в статті «Що таке насправді SHA-256»: володіння доводиться не пред'явленням секрету, а використанням його для підпису. Створений таким чином підпис може перевірити будь-хто за допомогою публічного значення, яке математично виводиться із самого секрету, без необхідності знати сам секрет і без посередництва третьої сторони. Той, у кого є секрет, і є ідентичність; той, хто його втрачає, перестає нею бути. Вердикт категоричний: **нема в кого просити повернути вам ідентичність. Цієї «когось» не існує, бо у нього її спочатку й не було.**

Що являють собою двадцять чотири слова

Криптографічна ідентичність зазвичай представляється математичним секретом довжиною тридцять два байти — двісті п'ятдесят шість біт. Число, яке важко запам'ятати і ще важче безпомилково переписати. Криптографічна індустрія вирішила цю проблему у 2013 році за допомогою невеликого та елегантного стандарту під назвою BIP39: способу представлення цих двохсот п'ятдесяти шести біт у вигляді послідовності з двадцяти чотирьох слів, взятих з офіційного списку з двох тисяч сорока восьми. Математика, що лежить в його основі, ідеально підходить; ті, хто хоче побачити її в деталях, знайдуть її на полях.

Відлік починається з кінця. Ми хочемо представити двісті п'ятдесят шість біт секрету, додавши вісім біт контрольної суми: всього двісті шістдесят чотири біти. Якщо ми розділимо їх на двадцять чотири слова — число, зручне для запису та диктування без втрат, — кожне слово має містити рівно одинадцять біт інформації. А одинадцять біт — це два в одинадцятому ступені можливостей, тобто дві тисячі сорок вісім. Ось чому офіційний словник BIP39 має саме такий розмір: список існує під завдання, а не навпаки.

Цей розрахунок — не просто декорація. Якщо хтось правильно перепише двадцять три слова і помилиться у двадцять четвертому, контрольна сума виявить це: програмне забезпечення скаже йому «ця послідовність недійсна». Якщо хтось перепише всі двадцять чотири правильно, програмне забезпечення однозначно виведе ту саму ідентичність. Вибір списку слів також не є випадковим: слова в словнику BIP39 короткі, відмінні одне від одного, без діакритичних знаків, обрані так, щоб звести до мінімуму фонетичну та орфографічну плутанину. Це словник, створений для того, щоб люди могли його запам'ятовувати, записувати та диктувати без втрат.

Від фрази до ключа

Ці двадцять чотири слова не є криптографічним ключем, яким підписуються повідомлення. Вони є відновлюваним представленням вихідної ентропії, яка за допомогою детермінованого процесу під назвою PBKDF2 перетворюється на 64-байтне «зерно» (seed). З цього зерна також детерміновано виводяться конкретні криптографічні ключі, які використовує користувач: закритий ключ для підпису та відповідний відкритий ключ, який публікується для перевірки підписів. Той самий механізм у різних системах: криптовалюти використовують криву secp256k1; протокол Signal та багато сучасних систем використовують Ed25519 на кривій Curve25519. Для конкретної кривої, такої як Ed25519, стандарти BIP32 та SLIP-0010 беруть це 64-байтне зерно і детерміновано виводять 32 байти, що складають ефективний ключ підпису — ті самі 32 байти, з яких починається приклад коду в наступному розділі.

Це стандартний спосіб, яким уся індустрія представляє механізм користувачеві — криптовалютні гаманці, менеджери децентралізованої ідентифікації, Signal у своїй частині постійної ідентифікації, Solo2 серед них: користувач на практиці ніколи не бачить ні зерна, ні похідних ключів. Він бачить двадцять чотири слова при створенні своєї особистості і, за бажанням, записує їх на папері. Потім слова переміщуються між його пристроями, коли він хоче мігрувати особистість: він вводить їх у новий додаток, додаток виводить те саме зерно, ті самі ключі, ту саму особистість. Це портативний, криптографічно надійний і, в розумних межах, запам'ятовуваний механізм.

Як підписувати ключем (штрих Zig)

У Zig, як тільки ви маєте 32-байтне зерно, отримане з двадцяти чотирьох слів, підпис повідомлення за допомогою Ed25519 вкладається в кілька рядків:

```
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);
```

Операція підпису створює шістдесят чотири байти — званих підписом — які могли бути створені лише на основі відповідного закритого ключа. Перевірка є публічною: будь-хто, у кого є відкритий ключ, може перевірити, чи відповідає підпис повідомленню. Без закритого ключа ніхто не може створити дійсний підпис для цього повідомлення; з відкритим ключем кожен може виявити, чи є підпис дійсним. Ця асиметрія дозволяє підписувачу довести авторство, не ділячись секретом.

Попередній приклад — це мінімальна версія з посібника. У реальному коді Solo2 ланцюжок проходить через два файли: один на JavaScript, який живе в браузері користувача і відновлює ентропію з двадцяти чотирьох слів, інший на Zig у складі бібліотеки *zcatcrypto*, який бере цю ентропію і виводить конкретні криптографічні ключі. Починаючи зі сторони браузера:

```
// 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 };
}
```

Ці тридцять два байти ентропії разом з іншими тридцятьма двома, отриманими на тому ж етапі, відправляються в модуль WebAssembly на Zig, який генерує власне ключі Ed25519. Повна функція з фінальним очищенням пам'яті вміщується на одному екрані:

```
// 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;
}
```

Варто відзначити дві деталі. Перша: одне і те ж початкове число (seed) завжди створює одну і ту ж пару ключів — саме це дозволяє відновити особистість, ввівши двадцять чотири слова на новому пристрої. Друга: початкове число явно стирається з пам'яті в останньому рядку. Після цього моменту навіть сама функція не змогла б відновити ключі; слова користувача були б єдиним джерелом.

Для тих, хто хоче перевірити це на маленьких числах. Схему підпису можна повністю простежити на цифрах, достатньо малих для ручного розрахунку. Ті, хто воліє не заглиблюватися в арифметику, можуть пропустити цей блок, не втрачаючи нитки статті; ті, хто хоче побачити механізм у дії крок за кроком, знайдуть його тут. **Публічні правила**, які може прочитати кожен: просте число $p = 23$ (у реальному Ed25519 воно складається приблизно з сімдесяти семи цифр; ми використовуємо двадцять три, щоб розрахунки вмістилися на одній сторінці), основа $g = 2$, порядок якої в цій групі дорівнює $q = 11$, і угода про те, що вся арифметика з g виконується *módulo p*, а всі показники степеня скорочуються *módulo q*. **Приватний вибір**, єдиний і ніколи не розголошуваний: секрет $x = 6$. Це і є особистість.

Крок 1 — Публічна частина особистості. Обчислюється один раз і публікується відкрито.

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

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

Публічна частина особистості — **18**. Будь-хто може взяти її і використовувати для перевірки підписів, зроблених від імені цієї особистості. Ніхто, знаючи лише 18, не може відновити секрет 6: у цьому і полягає проблема дискретного логарифма, до якої ми повернемося в кінці.

Крок 2 — Підписання повідомлення. Власник особистості хоче підписати повідомлення $m = 7$. Він починає з вибору нового випадкового значення $k = 4$, яке буде використано лише один раз і ніколи не буде передано (у реальному Ed25519 k виводиться детерміновано з повідомлення та секрету, щоб уникнути небезпеки повторного використання, але роль воно відіграє саме таку). Потім він обчислює три числа:

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

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

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

Підпис — це пара **(r, s) = (16, 10)**. Він передається відкрито разом із повідомленням. Будь-хто може його прочитати. Методична примітка: у реальному Ed25519 функція H — це SHA-512, криптографічно стійка; тут ми використовуємо спрощення $e = (r + m) \bmod q$, щоб читач міг простежити кроки без необхідності обчислювати хеш. Структура алгоритму та ж сама.

Крок 3 — Перевірка підпису. Той, хто перевіряє, має публічну частину $y = 18$, повідомлення $m = 7$ та підпис $(r, s) = (16, 10)$. Він відновлює e тим самим способом — $e = (16 + 7) \bmod 11 = 1$ — і перевіряє, чи виконується ця рівність:

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

Обчислює обидві сторони окремо:

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

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

Обидві сторони дають **12**. Підпис дійсний. Будь-хто, маючи публічну частину 18, може прийти до цього висновку, ніколи не знаючи, що секретом було число 6.

А що, якщо хтось третій спробує підробити? Єва бачила все публічне, що проходить по каналу: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. Щоб підписати *інше* повідомлення від імені цієї особистості, їй потрібно було б знати x. Її єдиний шлях — задатися питанням: «при якому показнику степеня x виконується $2^x \bmod 23 = 18$?». При $p = 23$ вона може перебрати 0, 1, 2, 3, … і знайти його за секунди. Але якщо замінити 23 на просте число реальних масштабів Ed25519, простір можливих експонент перевищить кількість атомів в осяжному Всесвіті. **На сьогодні людству не відомо жодного алгоритму, здатного перебрати цей простір менш ніж за мільярди років.** Це та сама проблема дискретного логарифма, яка лежить в основі протоколу Diffie-Hellman з попередньої статті, застосована тут до схеми підпису.

Те, що ми щойно розібрали, — це *саме* Schnorr, схема підпису, варіантом якої (адаптованим до еліптичної кривої) є Ed25519. У реальному Ed25519 всі операції виконуються над точками конкретної кривої (Curve25519), а не над цілими числами за модулем простого числа, і функція H — це SHA-512 замість спрощеної суми, яку ми використовували вище. Обидві заміни є коригуваннями реалізації — для підвищення криптостійкості до повного перебору та отримання додаткових властивостей безпеки для k. Алгоритмічна структура, три операції та причина асиметрії залишаються тими ж самими.

Тут доречно зробити коротку зупинку, оскільки весь ланцюжок можна при побіжному погляді сплутати з іншим примітивом з цієї трійки — хешем. Це не він. Хеш — це унікальна функція, яка стискає: на вході багато байтів, на виході короткий відбиток, і на цьому шлях закінчується. Криптографічна особистість — це математично доповнююча одна одну пара: секрет залишається у власника і підписує, а його публічна частина публікується і перевіряє. Там, де хеш згортає інформацію в одному напрямку, особистість встановлює асиметрію між двома половинами. Хеш свідчить про те, що було сказано; особистість свідчить про те, хто це сказав.

Чим фраза не є

Варто розвіяти три поширені помилки. Фраза не є паролем у власному розумінні слова: вона не порівнюється з відбитком, що зберігається на сервері; вона вводиться в пристрій користувача для математичного відновлення особистості. Фраза не відновлюється: якщо вона втрачена, немає в кого її просити; якщо вона дублюється, дублюється і особистість. Фраза не є обліковими даними, що відокремлюються від особистості: фраза *є* особистість. Той, у кого вона є, може діяти від її імені без додаткового дозволу, без процесу авторизації, без можливості відновлення.

Саме ця третя властивість змінює вагу справи. Втрачений пароль — це адміністративна незручність. Втрачена криптографічна особистість — це сама особистість. Папір із фразою, знайдений третіми особами, — це не ризик крадіжки аккаунта: це передача всієї особистості. Обіцянка системи — що ніхто не може відкликати вашу особистість або довільно заблокувати вас — нерозривно супроводжується відповідальністю — що ви є єдиним охоронцем того, чого ніхто не може відновити за вас.

Обіцянка та вага

Модель криптографічної ідентифікації часто називають *самосуверенною* — self-sovereign в англосаксонській літературі. Вибір слова є навмисним і досить точно описує стан. Користувач є сувереном над своєю особистістю майже в середньовічному сенсі: її не жалує жоден король, жоден емітент, жодна центральна влада; і ніхто з вищеперелічених не може її відкликати. Але також, подібно до

середньовічного монарха, користувач несе повну відповідальність за свої помилки: немає регента, який би приймав рішення за нього, якщо він втратить печатку.

Вибір між особистістю, керованою третьою стороною, та самосуверенною особистістю не має єдиної універсальної правильної відповіді. Для облікового запису на неважливому форумі керована особистість, ймовірно, пропорційна ризику. Для професійної особистості, що підписує юридично зобов'язуючі документи, для економічної особистості, що охороняє власні заощадження, для особистості професійного спілкування з клієнтами, які довірили конфіденційну інформацію, питання змінюється. Там питання перестає бути «чи це зручно?» і стає «хто, крім мене, має владу діяти від мого імені і за яких обставин?»

Де цей механізм з'являється в реальних системах

BIP39 народився у світі Bitcoin у 2013 році та швидко поширився на всю екосистему криптовалют: будь-який серйозний гаманець сьогодні приймає фразу BIP39 із дванадцяти або двадцяти чотирьох слів як резервну копію економічної ідентичності її власника. За межами криптовалют та сама базова концепція — криптографічна пара, що доводить авторство без посередника — з'являється в інших системах з іншим синтаксисом. Ключі SSH, які системний адміністратор використовує для доступу до своїх серверів, є класичним випадком: приватний ключ, який адміністратор зберігає на своїй машині, і відкритий, який копіюється на кожен сервер; жодна структура, порівнянна з централізованою службою, не втручається. Протокол Signal використовує Ed25519 з постійним матеріалом ключа на пристрої; європейські регламенти eIDAS у частині кваліфікованого підпису ґрунтуються на тому самому криптографічному принципі, з тією різницею, що ключ зберігається у кваліфікованого постачальника довірчих послуг, а не у користувача.

Solo2, видавнича платформа цього видання, використовує фразу BIP39 із двадцяти чотирьох слів як ідентичність кожного користувача. Користувач під час створення облікового запису бачить слова один раз. Вони не зберігаються на жодному сервері Solo2 або будь-кого іншого: якщо користувач занотує їх і збереже, він збереже свою ідентичність назавжди. Якщо він їх втратить, він їх втратить. Це логічний наслідок архітектури без посередника: якби Solo2 могла повернути ідентичність користувачеві, який її втратив, вона також могла б віддати її будь-кому, хто натисне на Solo2, щоб її отримати.

Для професійного читача

Чотири міркування для тих, хто розглядає можливість впровадження криптографічної самосуверенної (autosoberana) ідентичності у професійному контексті:

1. Фраза і є ідентичність. Фізичне зберігання — папір, кілька копій у різних місцях, зрештою гравірування на металі для довгострокового використання — пропонує більше гарантій, ніж цифрове зберігання, яке збільшує поверхню атаки, не знижуючи ризику втрати.
2. Відновлення неможливе. Проєктування процесу виходячи з припущення, що одного разу основна копія буде втрачена, набагато доцільніше, ніж виявити це в день втрати. Друга географічно віддалена копія вирішує майже всі сценарії.
3. Це не те саме, що кваліфікований сертифікат eIDAS. Для кваліфікованого підпису в Союзі — нотаріальні акти, певні процедури в адміністрації — законодавство вимагає наявності кваліфікованого постачальника, який зберігає ключ. Криптографічна самосуверенна ідентичність служить для професійного спілкування та документального підписання з доказовою силою, але не замінює автоматично кваліфікований сертифікат у випадках, коли цього вимагає норма.
4. Якщо ідентичність підлягає передачі — спадщина, професійне наступництво, припинення діяльності — доцільно підготувати процедуру заздалегідь, а не після. Формальні процедури з конвертами, запечатаними сургучем (lacre), інструкції виконавцю заповіту, зберігання в нотаріальній конторі — це класичні механізми, повністю сумісні з криптографічною природою активу.

Ця стаття завершує концептуальне тріо, що відкрило цикл — hash, шифрування, ідентичність —. Ці три ідеї будуються одна на одній: hash дає незмінний відбиток, шифрування дає конфіденційність без довіреної третьої сторони, ідентичність дає авторство без сторони, що надає повноваження. Усі три мають властивість, яка також не є ідеологічною: вони передають від того, хто керує сервісом, тому, хто його використовує, технічні можливості, які традиційно належали оператору. Разом з ними передається і відповідальність. Чесна розмова про будь-яку з цих трьох ідей вимагає розмови і про дві інші.

Джерела та додаткова література

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, пропозиція щодо вдосконалення Bitcoin 2013 року. Стандарт де-факто для фраз відновлення в криптоіндустрії.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), включаючи Ed25519. IETF, січень 2017 р. Нормативна специфікація схеми підпису, що використовується в значній частині сучасної індустрії.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, версія 2.0. IETF, вересень 2000 р. Визначає алгоритм PBKDF2, що використовується в похідній BIP39 від фрази до початкового числа (seed).
- Регламент (ЄС) 910/2014 (eIDAS) та його розвиток Регламентом (ЄС) 2024/1183 (eIDAS 2) — європейська база електронної ідентифікації та кваліфікованого підпису. Режим, відмінний від самосуверенного, але концептуально заснований на тих самих криптографічних примітивах.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Канонічний текст про принципи та зобов'язання самосуверенної моделі, раніший, але актуальний для розуміння сімейства сучасних рішень.

← ПопереднійБізнес-модель як сигнал довіриНаступний →Self-hosting як професійна практика

Останні матеріали

- Роздуми · 29 червня 2026 р. Ви не анонімні
- Рефлексія · 27 травня 2026 р. Те, що підпис не може виправити
- Аналіз · 26 травня 2026 р. Реальна vs уявна конфіденційність: питання, які варто собі поставити

Візьміть цю статтю з собою куди завгодно.

↓ Markdown ↓ Звичайний текст ↓ PDF

Файл буде завантажено на ваш пристрій. Звідти ви можете зберегти його, імпортувати в Solo2 або поділитися ним де завгодно. Cuadernos не вирішує місце призначення за вас.

Сургучна печатка · SHA-256 020b714f9674e5fe48cfca57ece3b6e534580f1705bd432c04b637dac950c676

Cuadernos Lacre · Видання Menzuri Gestión S.L. ·
автор R.Eugenio · під редакцією команди Solo2.