24 המילים: מהי זהות קריפטוגרפית

זהות קריפטוגרפית אינה סיסמה: אף שרת לא שומר אותה ולא ניתן לשחזר אותה. הסבר דידקטי על מנגנון BIP39, מדוע בדיוק עשרים וארבע מילים, ומהו המשקל האמיתי המוטל על מי שמחזיק בהן.

כדי שנבין זה את זה: אם שכחת את סיסמת ה-Gmail שלך, גוגל מאפסת אותה עבורך. אם איבדת את 24 המילים המרכיבות זהות קריפטוגרפית, אין ממי לבקש אותן. זה לא שההליך נוקשה — זה שפשוט אין אף אחד בצד השני. ההבדל הזה הוא כל ההבדל.

ההבדל בין סיסמה לזהות

סיסמה, במודל האינטרנט הקלאסי, אינה הזהות של המשתמש. היא שובר אימות. למשתמש יש זהות — שם, אימייל, מספר לקוח — וכדי להוכיח בפני שרת שהוא מי שהוא טוען להיות, הוא מציג סיסמה שהשרת משווה לטביעת אצבע ששמר. אם טביעות האצבע תואמות, השרת מאשר את ההתחברות. אם הסיסמה אובדת, המשתמש נשאר אותו משתמש; מה שהוא מאבד זה את השובר, וקיים הליך שחזור — אימייל לכתובת הרשומה, שאלת אבטחה — כדי להחזירו.

זהות קריפטוגרפית פועלת אחרת. היא אינה אישור שמישהו משווה לטביעת אצבע שמורה; היא *עצמה* סוד מתמטי שלם. זה לא משנה איפה היא נמצאת — על נייר, במכשיר, או אפילו בשרת זר: הזהות קיימת בזכות המתמטיקה שלה, לא בזכות מי שמאמת אותה. כאן מופיעה תכונה דומה לזו שראינו ב«מה זה באמת SHA-256»: הבעלות אינה מוכחת על ידי הצגת הסוד, אלא על ידי שימוש בו לחתימה. כל אחד יכול לבדוק את החתימה שנוצרת כך באמצעות ערך ציבורי שנגזר מתמטית מהסוד עצמו, ללא צורך להכיר את הסוד עצמו, וללא תיווך של צד שלישי בבדיקה. מי שמחזיק בסוד, הוא הזהות; מי שמאבד אותו, מפסיק להיות הזהות. הפסיקה היא חד-משמעית: **אין אף אחד שאפשר לבקש ממנו להחזיר לך את הזהות. המישהו הזה לא קיים, כי היא לא הייתה אצלו מלכתחילה.**

מה מייצגות עשרים וארבע מילים

זהות קריפטוגרפית מיוצגת בדרך כלל על ידי סוד מתמטי של שלושים ושניים בייטים — מאתיים חמישים ושישה ביטים. מספר שקשה לזכור ועוד יותר קשה להעתיק ללא שגיאה. תעשיית הקריפטו פתרה את הבעיה הזו בשנת 2013 עם תקן קטן ואלגנטי בשם BIP39: דרך לייצג את אותם מאתיים חמישים ושישה ביטים כרצף של עשרים וארבע מילים שנלקחו מרשימה רשמית של אלפיים ארבעים ושמונה. האריתמטיקה שמאחורי זה משתלבת באלגנטיות; מי שרוצה לראות זאת בפירוט ימצא זאת בשוליים.

הספירה מתחילה מהסוף. אנחנו רוצים לייצג את מאתיים חמישים ושישה הביטים של הסוד בתוספת שמונה ביטים של סכום ביקורת (checksum): מאתיים שישים וארבעה ביטים בסך הכל. אם נחלק אותם לעשרים וארבע מילים — מספר נוח לרישום והכתבה ללא אובדן — כל מילה חייבת לתרום בדיוק אחת עשרה ביטים של מידע. ואחת עשרה ביטים הם שתיים בחזקת אחת עשרה אפשרויות שונות, כלומר, אלפיים ארבעים ושמונה. מכאן שאוצר המילים הרשמי של BIP39 הוא בדיוק בגודל הזה: הרשימה קיימת לפי מידת הבעיה, לא להפך.

הספירה אינה קישוטית. אם מישהו יעתיק עשרים ושלוש מילים נכון ויטעה במילה העשרים וארבע, סכום הביקורת יזהה זאת: התוכנה תגיד לו "הרצף הזה אינו חוקי". אם מישהו יעתיק את כל העשרים וארבע נכון, התוכנה תגזור את אותה הזהות ללא עמימות. גם בחירת רשימת המילים היא מכוונת: המילים באוצר המילים של BIP39 הן קצרות, שונות זו מזו, ללא סימני הלחם, ונבחרו כדי למזער בלבול פונטי ואורתוגרפי. זהו אוצר מילים שנועד להיזכר, להיכתב ולהיות מוכתב על ידי בני אדם ללא אובדן.

מהמשפט אל המפתח

עשרים וארבע המילים אינן המפתח הקריפטוגרפי שחותם על הודעות. הן ייצוג שניתן לשחזור של האנטרופיה המקורית אשר באמצעות תהליך דטרמיניסטי הנקרא PBKDF2, הופכת לזרע (seed) בן שישים וארבעה בתים. מאותו זרע נגזרים, גם באופן דטרמיניסטי, המפתחות הקריפטוגרפיים הספציפיים שבהם המשתמש משתמש: מפתח פרטי לחתימה ומפתח ציבורי תואם שמפורסם כדי לאמת את החתימות. אותו מנגנון במערכות שונות: מטבעות קריפטוגרפיים משתמשים בעקומת secp256k1; פרוטוקול Signal ומערכות מודרניות רבות משתמשים ב-Ed25519 מעל עקומת Curve25519. עבור עקומה ספציפית כמו Ed25519, תקני BIP32 ו-SLIP-0010 לוקחים את אותו זרע בן שישים וארבעה בתים וגוזרים, באופן דטרמיניסטי, את שלושים ושניים הבתים המהווים את מפתח החתימה האפקטיבי — אותם שלושים ושניים בתים שבהם מתחילה דוגמת הקוד בסעיף הבא.

זוהי הדרך הסטנדרטית שבה התעשייה כולה מציגה את המנגנון למשתמש —ארנקי מטבעות קריפטוגרפיים, מנהלי זהות מבוזרת, Signal בחלק של הזהות הקבועה שלו, Solo2 ביניהם—: המשתמש, בפועל, לעולם אינו רואה את הזרע או את המפתחות הנגזרים. הוא רואה את עשרים וארבע המילים בעת יצירת זהותו, ובאופן אופציונלי, רושם אותן על נייר. המילים עוברות לאחר מכן בין מכשיריו כאשר הוא רוצה להעביר את הזהות: הוא מזין אותן באפליקציה החדשה, האפליקציה גוזרת את אותו זרע, אותם מפתחות, אותה זהות. זהו מנגנון נייד, מוצק מבחינה קריפטוגרפית, ובמסגרת הסביר, ניתן לזכירה.

איך חותמים עם המפתח (נגיעת Zig)

ב-Zig, ברגע שיש לך את הזרע בן שלושים ושניים הבתים הנגזר מעשרים וארבע המילים, חתימה על הודעה באמצעות Ed25519 נכנסת למספר שורות בודדות:

```
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);
```

פעולת החתימה מייצרת שישים וארבעה בתים —הנקראים חתימה— שיכלו להיווצר רק מהמפתח הפרטי התואם. האימות הוא ציבורי: כל מי שיש לו את המפתח הציבורי יכול לבדוק שהחתימה תואמת להודעה. ללא המפתח הפרטי, אף אחד לא יכול לייצר חתימה תקפה לאותה הודעה; עם המפתח הציבורי, כולם יכולים לזהות אם חתימה היא תקפה. אסימטריה זו היא המאפשרת לחתום להוכיח בעלות מבלי לשתף את הסוד.

הדוגמה הקודמת היא גרסת המדריך המינימלית. בקוד האמיתי של Solo2 השרשרת עוברת דרך שני קבצים, אחד ב-JavaScript שחי בדפדפן המשתמש ומשחזר את האנטרופיה מעשרים וארבע המילים, אחר ב-Zig בתוך ספריית *zcatcrypto* שלוקחת את האנטרופיה הזו ומפיקה את המפתחות הקריפטוגרפיים הספציפיים. נתחיל מצד הדפדפן:

```
// 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 words = mnemonic.trim().split(/\s+/);
    const wordlist = WORDLISTS[lang || 'en'];
```

```
                .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 }
                                                                    {
```

שלושים ושניים הבתים של האנטרופיה, יחד עם שלושים ושניים נוספים שהופקו באותו שלב, עוברים למודול ה-WebAssembly של Zig המייצר את מפתחות ה-Ed25519 עצמם. הפונקציה המלאה, עם ניקוי הזיכרון הסופי שלה, נכנסת למסך אחד:

```
                                                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
                                                                        {
```

שני פרטים ראויים לציון. הראשון: אותו זרע (seed) תמיד מייצר את אותו זוג מפתחות – זה בדיוק מה שמאפשר שחזור זהות על ידי הזנת עשרים וארבע המילים במכשיר חדש. השני: הזרע נמחק במפורש מהזיכרון בשורה האחרונה. מעבר לנקודה זו, אפילו הפונקציה עצמה לא תוכל לשחזר את המפתחות; מילות המשתמש יהיו המקור היחיד.

למי שרוצה לבדוק זאת עם מספרים קטנים. ניתן לעבור על כל סכימת החתימה עם מספרים קטנים מספיק כדי לבצע את החישובים ידנית. מי שמעדיף לא להיכנס לאריתמטיקה יכול לדלג על הבלוק הזה מבלי לאבד את הקשר המאמר; מי שרוצה לראות את המנגנון פועל צעד אחר צעד ימצא אותו כאן. **הכללים הציבוריים**, שכל אחד יכול לקרוא: מספר ראשוני $p = 23$ (ב-Ed25519 אמיתי הוא בן כ-77 ספרות; אנו משתמשים ב-23 כדי שהחישובים ייכנסו לדף אחד), בסיס $g = 2$ שהסדר שלו

בקבוצה זו הוא $q = 11$, והמוסכמה שכל האריתמטיקה עם g נעשית *módulo p* וכל המעריכים מצומצמים *módulo q*.
הבחירה הפרטית, אחת ויחידה ולעולם לא משותפת: הסוד $x = 6$. זו הזהות.

שלב 1 – החלק הציבורי של הזהות. הוא מחושב פעם אחת ומפורסם בגלוי.

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

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

החלק הציבורי של הזהות הוא **18**. כל אחד יכול לקחת אותו ולהשתמש בו כדי לאמת חתימות שנעשו עם הזהות הזו. אף אחד, על ידי התבוננות ב-18 בלבד, לא יכול לשחזר את הסוד 6: זוהי בעיית הלוגריתם הבדיד שאליה נחזור בסוף.

שלב 2 – חתימה על הודעה. בעל הזהות רוצה לחתום על ההודעה $m = 7$. הוא מתחיל בבחירת ערך אקראי חדש $k = 4$, שישמש פעם אחת בלבד ולעולם לא ישותף (ב-Ed25519 אמיתי, k מופק באופן דטרמיניסטי מההודעה ומהסוד כדי למנוע סכנה של שימוש חוזר, אך התפקיד שהוא ממלא הוא בדיוק זה). לאחר מכן הוא מחשב שלושה מספרים:

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

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

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

החתימה היא הזוג **(r, s) = (16, 10)**. היא עוברת בגלוי יחד עם ההודעה. כל אחד יכול לקרוא אותה. הערה דידקטית: ב-Ed25519 אמיתי הפונקציה H היא SHA-512, חסונה מבחינה קריפטוגרפית; כאן אנו משתמשים בפישוט $e = (r + m) \bmod q$ כדי שהקורא יוכל לעבור על השלבים ללא צורך בחישוב האש (hash). מבנה האלגוריתם זהה.

שלב 3 – אימות החתימה. למאמת יש את החלק הציבורי $y = 18$, את ההודעה $m = 7$, ואת החתימה $(r, s) = (16, 10)$. הוא משחזר את e באותה דרך – $e = (16 + 7) \bmod 11 = 1$ – ובודק אם השוויון הזה מתקיים:

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

מחשב את שני הצדדים בנפרד:

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

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

שני הצדדים נותנים **12**. החתימה תקפה. כל אחד עם החלק הציבורי 18 יכול להגיע למסקנה זו מבלי שידע מעולם שהסוד היה 6.

ומה לגבי צד שלישי שינסה לזייף? אווה ראתה את כל המידע הציבורי שעבר בערוץ: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. כדי לחתום על הודעה *אחרת* בשם זהות זו, היא תצטרך לדעת את x. הדרך היחידה שלה היא לשאול את עצמה: "עבור איזה מעריך x מתקיים $2^x \bmod 23 = 18$?". עם $p = 23$ היא יכולה לנסות 0, 1, 2, 3, … ולמצוא אותו בשניות. אך בהחלפת 23 במספר ראשוני מהממדים האמיתיים של Ed25519, מרחב המעריכים האפשריים עולה על מספר האטומים ביקום הנראה. **אין כיום אף אלגוריתם הידוע לאנושות שיכול לעבור על המרחב הזה בפחות ממיליארדי שנים.** זוהי אותה בעיית לוגריתם בדיד שעומדת בבסיס ה-Diffie-Hellman מהמאמר הקודם, המיושמת כאן על סכימת החתימה.

מה שעברנו עכשיו הוא בדיוק Schnorr, סכימת החתימה ש-Ed25519 היא גרסה שלה המותאמת לעקומה אליפטית. ב-Ed25519 אמיתי, כל הפעולות מבוצעות על נקודות של עקומה ספציפית (Curve25519) במקום על מספרים שלמים מודולו ראשוני, והפונקציה H היא SHA-512 במקום סכום הצעצוע שהשתמשנו בו למעלה. שתי ההחלפות הן התאמות יישום – השגת עמידות קריפטוגרפית נגד כוח גס (brute force), והשגת תכונות אבטחה נוספות עבור k. המבנה האלגוריתמי, שלוש הפעולות והסיבה לאסימטריה, זהים.

כדאי לעצור כאן לרגע, כי ניתן להתבלבל במבט חטוף בין השרשרת כולה לבין פרימיטיב אחר מהשלישייה: ההאש (hash). זה לא זה. האש היא פונקציה ייחודית שדוחסת – הרבה בתים נכנסים, טביעת אצבע קצרה יוצאת, שם מסתיים המסלול. זהות קריפטוגרפית היא זוג מתמטי משלים: הסוד נשאר וחותם; החלק הציבורי שלו מפורסם ומאמת. במקום שבו ההאש

מקריסה מידע בכיוון אחד, הזהות מבססת אסימטריה בין *שני* חצאים. ההאש מעיד על מה *שנאמר*; הזהות מעידה על מי *שאמר* זאת.

מה המשפט אינו

כדאי להבהיר *שלוש* טעויות נפוצות. המשפט אינו סיסמה במובן המקובל: הוא אינו מושווה לטביעת אצבע המאוחסנת בשרת; הוא מוזן במכשיר *של* המשתמש כדי *לשחזר* מתמטית את הזהות. את המשפט לא ניתן *לשחזר*: אם הוא אובד, אין למי לפנות כדי לבקשו; אם הוא משוכפל, הזהות משוכפלת גם כן. המשפט אינו אישור הניתן להפרדה מהזהות: המשפט *הוא* הזהות. מי *שמחזיק* בו יכול לפעול בשמה, ללא אישור נוסף, ללא תהליך הרשאה, ללא אפשרות *שחזור*.

תכונה *שלישית* זו היא *שמשנה* את משקל העניין. סיסמה אבודה היא טרחה מנהלתית. זהות קריפטוגרפית אבודה היא הזהות עצמה. נייר עם המשפט *שנמצא* על ידי צדדים *שלישיים* אינו סכנה לגניבת חשבון: זוהי מסירה *של* הזהות כולה. הבטחת המערכת —*שאף* אחד לא יוכל לבטל את זהותך או לחסום אותך באופן *שרירותי*— מלווה באופן בלתי נפרד באחריות —*שאתה* השומר היחיד *של* משהו *שאף* אחד לא יכול להחזיר עבורך.

ההבטחה והמשקל

מודל הזהות הקריפטוגרפית זוכה לעתים קרובות לכינוי *ריבונית עצמית* —self-sovereign בספרות האנגלית—. בחירת המילה היא מכוונת ומתארת את המצב בדיוק רב. המשתמש הוא ריבון על זהותו במובן כמעט ימי-ביניימי: אף מלך, אף מנפיק, אף רשות מרכזית אינם מעניקים אותה; גם אף אחד מהנזכרים לעיל אינו יכול *לשלול* אותה. אך גם, כמו המונרך מימי הביניים, המשתמש נושא בתוצאה המלאה *של* טעויותיו: אין עוצר *שיקבל* החלטות במקומו אם יאבד את החותם.

לבחירה בין זהות המנוהלת על ידי צד *שלישי* לבין זהות ריבונית עצמית אין תשובה אוניברסלית נכונה אחת. עבור חשבון בפורום חסר חשיבות, זהות מנוהלת היא כנראה פרופורציונלית לסיכון. עבור זהות מקצועית החותמת על מסמכים מחייבים מבחינה משפטית, עבור זהות כלכלית השומרת על חסכונות אישיים, עבור זהות *של* תקשורת מקצועית עם לקוחות *שהפקידו* מידע רגיש, העניין משתנה. *שם* השאלה מפסיקה להיות «האם זה נוח?» והופכת ל-«מי, מלבד עצמי, מחזיק בכוח לפעול *בשמי*, ובאילו נסיבות?».

היכן מופיע מנגנון זה במערכות אמיתיות

פרוטוקול BIP39 נולד בעולם ה-Bitcoin ב-2013 והתפשט במהירות לכל המערכת האקולוגית *של* המטבעות הקריפטוגרפיים: כל ארנק רציני מקבל היום ביטוי BIP39 *של* *שתים* עשרה או עשרים וארבע מילים כגיבוי לזהות הכלכלית *של* *מחזיקו*. *מחזיק* למטבעות הקריפטוגרפיים, אותו קונספט בסיסי — זוג קריפטוגרפי המוכיח מחברות ללא מתווך — מופיע במערכות אחרות עם תחביר *שונה*. מפתחות SSH *שמנהל* מערכות *משתמש* בהם כדי לגשת *לשרתים* *שלו* הם מקרה קלאסי: מפתח פרטי *שמנהל* המערכת *שומר* במכונה *שלו* ומפתח ציבורי *שמועתק* לכל *שרת*; אף ישות הדומה *לשירות* מרכזי אינה מתערבת. פרוטוקול Signal *משתמש* ב-Ed25519 עם חומר מפתח קבוע במכשיר; תקני eIDAS האירופיים, בחלק *של* החתימה המוסמכת *שלהם*, נשענים על אותו עיקרון קריפטוגרפי, עם ההבדל *שהמפתח* נשמר על ידי ספק *שירותי* אמון מוסמך במקום על ידי *המשתמש*.

Solo2, הפלטפורמה המוציאה לאור *של* פרסום זה, *משתמשת* בביטוי BIP39 *של* עשרים וארבע מילים כזהות *של* כל *משתמש*. *המשתמש*, בעת יצירת החשבון *שלו*, רואה את המילים פעם אחת. הן אינן נשמרות בשום *שרת* *של* Solo2 או *של* אף אחד אחר: אם *המשתמש* רושם אותן ושומר עליהן, הוא *שומר* על זהותו לנצח. אם הוא מאבד אותן, הוא מאבד אותן. זוהי התוצאה העקבית *של* ארכיטקטורה *שאין* בה מפעיל באמצע: אם Solo2 הייתה יכולה להחזיר את הזהות *למשתמש* *שאיבד* אותה, היא הייתה יכולה גם לתת אותה *למי* *שיפעיל* *לחץ* על Solo2 כדי *שיתנו* לו אותה.

לקורא המקצועי

ארבעה *שיקולים* *למי* *ששוקל* *לאמץ* זהות קריפטוגרפית ריבונית עצמית (autosoberana) בהקשר מקצועי:

1. הביטוי הוא הזהות. *שמירה* פיזית — נייר, מספר עותקים במקומות *שונים*, בסופו *של* דבר מתכת חרוטה *לשימוש* לטווח ארוך — מציעה יותר ערבויות מאשר *שמירה* דיגיטלית, *שמוסיפה* *שטח* תקיפה מבלי להפחית את הסיכון לאובדן.

2. אין שחזור. עיצוב התהליך תוך הנחה שיום אחד העותק העיקרי יאבד הוא הרבה יותר נבון מאשר לגלות זאת ביום שבו הוא אובד. עותק שני מופרד גיאוגרפית פותר כמעט את כל התרחישים.
3. זה לא אותו דבר כמו תעודת eIDAS מוסמכת. עבור חתימה מוסמכת באיחוד — שטרות נוטריוניים, הליכים מסוימים מול המנהל — החקיקה דורשת ספק מוסמך ששומר על המפתח. זהות קריפטוגרפית ריבונית עצמית משמשת לתקשורת מקצועית וחתימת מסמכים בעלת ערך ראייתי, אך אינה מחליפה באופן אוטומטי את התעודה המוסמכת במקרים בהם התקן דורש זאת.
4. אם הזהות אמורה לעבור — ירושה, ירושה מקצועית, סגירת פעילות — כדאי להכין את ההליך מראש, לא בדיעבד. הליכים פורמליים עם מעטפות חתומות בשעווה (lacre), הוראות למוציא לפועל של צוואה, הפקדה במשרד נוטריון, הם הסדרים קלאסיים התואמים לחלוטין את האופי הקריפטוגרפי של הנכס.

מאמר זה סוגר את השלישייה המושגית שפתחה את המחזור — hash, הצפנה, זהות —. שלושת הרעיונות נבנים זה על גבי זה: ה-hash נותן את הטביעה הבלתי משתנה, ההצפנה נותנת את הסודיות ללא צד שלישי נאמן, הזהות נותנת את המחברות ללא צד שלישי מעניק. שלושתם חולקים תכונה שאינה אידיאולוגית גם כן: הם מעבירים, ממי שמנהל שירות למי שמשתמש בו, יכולות טכניות שבאופן מסורתי שכנו אצל המפעיל. הם מעבירים איתם גם אחריות. דיבור בכנות על כל אחד משלושתם מחייב דיבור גם על השניים האחרים.

מקורות וקריאה נוספת

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, הצעת שיפור של Bitcoin מ-2013. תקן דה פקטו לביטויי שחזור בתעשיית הקריפטו.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), כולל Ed25519. IETF, ינואר 2017. מפרט נורמטיבי של סכימת החתימה המשמשת בחלק גדול מהתעשייה העכשווית.
- PKCS #5: Password-Based Cryptography Specification, גרסה 2.0. IETF RFC 2898, ספטמבר 2000. מגדיר את אלגוריתם PBKDF2 המשמש בגזירת BIP39 מביטוי לזרע (seed).
- תקנת (האיחוד האירופי) 910/2014 (eIDAS) והתפתחותה על ידי תקנת (האיחוד האירופי) 2024/1183 (eIDAS 2) — מסגרת אירופית לזהות אלקטרונית וחתימה מוסמכת. משטר שונה מהריבוני העצמי, אך נתמך מושגית על ידי אותם פרימיטיבים קריפטוגרפיים.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). טקסט קנוני על העקרונות והמחויבויות של המודל הריבוני העצמי, מוקדם יותר אך רלוונטי להבנת משפחת הפתרונות העכשוויים.

← הקודם המודל העסקי כאות לאמון הבא → Self-hosting כפרקטיקה מקצועית

קריאות אחרונות

- מחשבה · 29 ביוני 2026 אתה לא אנונימי
- הרהור · 27 במאי 2026 מה שחתימה לא יכולה לתקן
- ניתוח · 26 במאי 2026 פרטיות אמיתית מול מדומה: השאלות שכדאי לשאול את עצמך

קחו את המאמר הזה אתכם לכל מקום שתצטרכו.

Markdown ↓ · טקסט פשוט ↓ · PDF ↓

הקובץ יורד למכשיר שלכם. משם תוכלו לשמור אותו, לייבא אותו ל-Solo2 או לשתף אותו היכן שתרצו. Cuadernos לא מחליטה על היעד עבורכם.

חותם שעווה · SHA-256 32c7e97ba46353a94dbd05453843f57256b317f25446fb1a7448693a5713a13d

Cuadernos Lacre · פרסום של Menzuri Gestión S.L. ·
נכתב על ידי R.Eugenio · נערך על ידי צוות Solo2.

אתר זה אינו משתמש בעוגיות. כל מה שהדפדפן שלכם טוען נכתב או מפוקח על ידינו ומאוחסן בשרתים האירופיים שלנו: מונה הביקורים האנונימי (Umami, באירוח עצמי) ומינימום ה-JavaScript הדרוש לבורר השפה ולהעדפת ערכת הנושא הבהירה/כהה שלכם, הנשמרת במכשיר שלכם עצמו. ללא משאבים מחברות חיצוניות, ללא עוקבים, ללא פרופילציה, ללא שיתוף נתונים. אם תרצו לעקוב אחרינו: RSS.