TIBANE LABS
APPLIED CRYPTOGRAPHY & SECURITY RESEARCH

TECHNICAL SECURITY ADVISORY

Anatomy of a Predictable-Nonce Key Disclosure

How a single dropped secret turned every Ed25519 signature produced by the affected SecondFi (formerly Yoroi) build into a published private key.

SUBJECT	CURVE	CLASS
SecondFi / Yoroi signing flaw	Ed25519 (EdDSA)	CWE-323 nonce reuse / predictability

DATE	AUTHOR
25 June 2026	Tibane Labs

⚠ If you used the SecondFi / Yoroi wallet, read this first

IN PLAIN LANGUAGE — NO CRYPTOGRAPHY REQUIRED

COMPROMISED

If your wallet signed any transaction on or after ≈ 8 June 2026 — even once — that wallet is compromised. Signing once published enough for anyone to reconstruct your private key from public blockchain data; it does not matter how small the amount. (Transactions before that date used a correct signer and are not exposed by this flaw.)

ACT NOW

If your coins are still sitting there, move them to a different wallet immediately. You are racing anyone else who has already computed your key — speed matters.

SAFE — FOR NOW

The only wallets still safe are those that have never ever sent a transaction. Merely receiving coins is fine — your key is only exposed when you send. But such a wallet becomes exposed the instant it sends, so move everything out in a single transfer to a safe wallet and never use the old address again.

Important: restoring your recovery phrase into a different app does **not** fix this — the weakness is in the key itself — and your destination must not be another SecondFi / Yoroi-generated address.

SUMMARY

01 The flaw in one paragraph

The SecondFi wallet signed Cardano transactions with a custom Ed25519 implementation whose per-signature **nonce** was derived as `SHA-512(message)` — omitting the per-key secret that Ed25519 requires. Because a Cardano transaction body is public the instant it reaches the chain, the nonce became public too. In EdDSA a known nonce is algebraically equivalent to a published private key: the signing equation collapses to one linear equation in one unknown. Consequently **any address that signed even a single transaction with the affected build (which shipped 2026-06-08) leaked its private key** — with no reuse, special conditions, or user error required. Signatures produced before that date used a correct signer and are unaffected. This advisory documents the mechanism only; loss accounting is covered separately.

◆ CORE FINDING

The vulnerability is not weak random number generation and not classic nonce reuse. It is nonce predictability: a deterministic nonce keyed solely on public data. Recovery needs exactly one ordinary signature. We confirmed this end-to-end against live Cardano mainnet signatures.

BACKGROUND

02 How Ed25519 signing is supposed to work

Ed25519 was designed deliberately to remove the nonce from the hands of the implementer. There is **no API to supply or generate a nonce** — the derivation is sealed inside the signing routine. A 32-byte seed is expanded once with SHA-512 and split into two halves with distinct, permanent roles:

Figure 1 — Correct Ed25519 key expansion. The prefix is a dedicated secret whose only job is to make the nonce unguessable.

With the secret `prefix` in the hash, the nonce `r` is deterministic and unpredictable: it changes per message, never repeats, and cannot be computed by an outsider. The signature is the pair `(R, s)` where:

$$R = r \cdot B \qquad s = (r + H \cdot a) \bmod L$$

$$H = \text{SHA-512}(R \,||\, A \,||\, M) \bmod L, \quad A = a \cdot B \ \text{(public key)}, \quad B = \text{base point}, \quad L = \text{group order}$$

✓ THE DESIGN INTENT

Two properties protect the key: the nonce is **deterministic** (defends against bad RNGs and reuse) and **keyed by a secret** (defends against prediction). Both are mandatory. RFC 8032 builds them into the algorithm; RFC 6979 retrofits the same idea onto ECDSA.

ROOT CAUSE

03 The bug: a nonce with no secret

SecondFi's June rewrite replaced its established native signer with a hand-rolled JavaScript one — the third-party SDK `@stashers.io/trantor`. In re-implementing Ed25519 signing for Cardano's BIP32-Ed25519 extended keys, it reintroduced the one thing the standard had sealed shut: it derived the nonce from the message alone. (Provenance and deployment are detailed in our companion supply-chain advisory.)

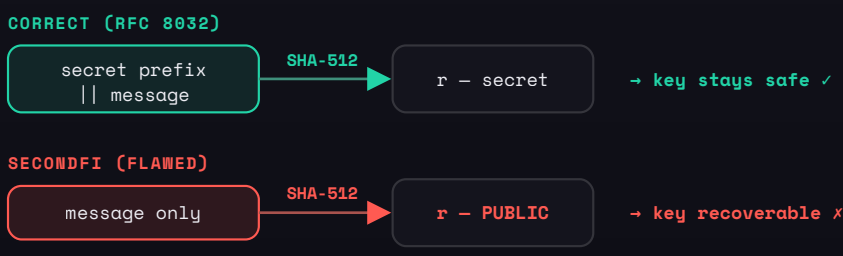

Figure 2 — The single-ingredient difference. Dropping the secret prefix leaves a signature that still verifies perfectly — and discloses the key.

```
RFC 6979 (ECDSA):   k = HMAC( private_key, H(message) )
RFC 8032 (Ed25519): r = SHA-512( secret_prefix || message )
SecondFi:           r = SHA-512(                  message )  ← secret dropped
```

The result still looks textbook-deterministic and verifies correctly, so it survives casual review. But the nonce is now a pure function of public data.

EXPLOITATION

04 Why a known nonce is a published key

The second half of an EdDSA signature is a single linear congruence:

$$s = r + H \cdot a \pmod{L}$$

One equation. The only unknown is the private scalar a.

Everything except a is public: s and R are in the signature, $H = \text{SHA-512}(R||A||M)$ is computable, and the message M is the transaction. The nonce r is the one value meant to be secret. Once it is predictable, solve for a directly:

$$r = \text{SHA-512}(M) \bmod L \quad \Longrightarrow \quad a = (s - r) \cdot H^{-1} \bmod L$$

Recovery from a **single** signature — no reuse, no second message.

✗ WHY THIS IS WORSE THAN CLASSIC REUSE

The well-known "same nonce on two signatures" attack needs a key to sign twice carelessly. Here the nonce is public from the first signature a key makes with the affected build, so a single ordinary transaction discloses the key — and nothing a user does afterwards, including importing the seed into a different wallet, can undo it. (Exposure is bounded to the build's active window — ≈ 2026-06-08 onward — not to a wallet's entire history.)

DETECTION

05 The on-chain fingerprint & authorship oracle

Because the buggy nonce is fully determined by the message, an affected signature satisfies a test that needs only its R and the transaction hash — no private key, no second signature:

$$\text{SHA-512}(M) \cdot B == R \quad \rightarrow \quad \text{signature is vulnerable}$$

This single check has a powerful side effect. The fingerprint marks signatures produced by the affected build; an attacker re-signing with a recovered key via correct code does not reproduce it. Empirically — from the drain-transaction classification — it acts as an **authorship oracle**, distinguishing spends made with the affected build from clean-signed ones, and so flags a likely unauthorized spend as a clean-signed transaction from an otherwise-fingerprinted key.

SIGNATURE ON A VICTIM KEY	FINGERPRINT SHA-512(M)·B==R	AUTHOR
Owner's spends via the affected build	present	Legitimate owner
The draining transaction	absent	Attacker (clean signer)

VERIFICATION

06 Confirmed against Cardano mainnet

We reproduced the entire chain of reasoning on live data. For one wallet's transaction, the predicted nonce reconstructs the exact on-chain R, and the same nonce is shared by two independent keys of that wallet — proof the nonce carries no per-key secret:

• VERIFIED — PREDICTABLE NONCE REPRODUCES ON-CHAIN R

```
tx (message M)   4655145484f7a0f83ddea7c2c52c7ac1f86f9fc7a99ef04f78a7ab177ce02203
r = SHA-512(M)   798f3314eb34ae00bccc959a8e548e7532cc9691e194a0059d0acc7fa8cb510b
r·B (computed)   5666e273d85d13f73294ba22d1022b0308c232ed572e2c0175da8865d538c5c0
R (on-chain)     5666e273d85d13f73294ba22d1022b0308c232ed572e2c0175da8865d538c5c0  ✓ match
shared by keys   02f680bd…  and  ef65c949…  (two keys, one nonce)
```

We then recovered the private scalars from single signatures and verified each by regenerating the public key ($a \cdot B == A$) — definitive proof of compromise. The scalars are held privately and were never used to construct a transaction.

• VERIFIED — SINGLE-SIGNATURE KEY RECOVERY (A·B == A)

```
key hash  c5b419954a1c451ef413a7cf24981f2600dc66d7806d7e9562873530  ✓ recovered & verified
key hash  55079bbf8860b687e48893c3ccceaa4cba1381e6e95f6fe41f8b83b4  ✓ recovered & verified
scan      40-tx hub sample → 40 of 41 distinct signing keys carry the fingerprint
```

⚠ EXPOSURE WINDOW (MEASURED ON-CHAIN)

Across a sample of affected wallets, predictable-nonce signatures appear on-chain **only between 2026-06-08 and 2026-06-25** — none earlier, despite years of prior, correctly-signed history. The flawed signer was therefore introduced ≈ 2026-06-08 (it shipped in app build 10.0.3), and only transactions signed on or after that date are exposed by this flaw.

Methodology and tooling (Koios CBOR retrieval, witness extraction with exact body-hash verification, recovery with self-tests, and a no-false-positive check against correct RFC 8032 signatures) accompany this advisory and regenerate every value above.

GUIDANCE

07 Remediation & the broader lesson

For the implementation

- Sign exclusively through a vetted primitive (`libsodium crypto_sign`, audited BIP32-Ed25519) that never exposes the nonce.
- Never hand-roll nonce derivation; the secret prefix must enter the hash.
- Add a regression test asserting $\text{SHA-512}(M) \cdot B \neq R$ for every signature produced.
- Audit all key-derivation paths — especially index-0 — for dropped or zeroed secrets.

For affected keys

- Treat any address that signed with the affected build (≈ 2026-06-08 onward) as **permanently compromised**; the exposure is at the key level.
- Re-importing the recovery phrase elsewhere does not help — the key is already public-derivable.
- Funds can only be protected by moving to a freshly generated key from known-good software, accepting the old key is burned.

◆ THE LESSON

Ed25519 intentionally offers no nonce knob; the absence is a safety feature shaped by ECDSA's history (the PlayStation 3 fixed-`k` break, the 2013 Android `SecureRandom` Bitcoin thefts). The moment an engineer needs to write a nonce derivation for Ed25519, they have already left the safe path. This incident is, in essence, the signature of a hand-rolled implementation standing in for a sealed one.

Prepared by **Tibane Labs** — applied cryptography & security research · tibane.net. This document analyses the attack mechanism only; impact and recovery accounting are addressed in companion material. Cryptographic claims were verified against Cardano mainnet via the public Koios API on 25 June 2026.