The Custody–Privacy Gap: What a Ten-Dollar Hardware Wallet Reveals About Self-Custody on Public Ledgers

Arnav Panjla[1] and Sahitya Shankar[1]
[1]Blockchain Society, IIT Delhi
arnavpanjla@gmail.com, yraretil@gmail.com

Abstract. Every self-custody hardware wallet is sold on one promise: your private key never leaves the device. Noir Wallet, an open-source Ethereum signer built on a $3–5 STM32F401CCU6 microcontroller (WeAct BlackPill board) and a $1–$2 ATECC608A secure element, is used here as a working case study to test a second, quieter promise that hardware wallets do not make but are often assumed to deliver: privacy. We trace the signing pipeline of this ten-dollar device stage by stage — secure element, firmware, USB HID transport, host software, broadcast node, public ledger — and show that hardware-anchored key custody leaves every one of these stages except the first fully observable. Address reuse, transaction-graph clustering, and host-side metadata survive perfect key custody untouched. We use a real firmware bug found on this device, a secure element whose random number generator produces a fixed diagnostic pattern until its configuration zone is permanently locked, as a concrete illustration of what a verifiable hardware security claim actually looks like, as opposed to a marketing one. We then map where privacy-enhancing techniques such as stealth addresses, zero-knowledge proofs of authorization, and selectively disclosable credentials would need to sit on top of this hardware to close the gap, and argue, in line with Ledger's own hardware-anchored trust model, that the cost of that root of trust should not be the reason self-custody privacy stays a privilege of a few hundred dollars.
Keywords: self-custody, secure elements, hardware wallets, zero-knowledge proofs, transaction privacy, chain surveillance, ATECC608A

1. Introduction

Public ledgers make ownership pseudonymous and the flow of funds globally visible [1]. A hardware wallet solves a narrower problem: it keeps the signing key off a compromised host. It says nothing about the transaction after it is signed. Marketing copy, and sometimes users, treat "self-custody" and "private" as synonyms. They are not. A hardware-signed payment from a reused address is as clusterable as one signed in a browser. The hardware protects the key, not what the key signs or where the signed artifact travels.

We investigate this gap using Noir Wallet, an open-source Ethereum hardware wallet built on an STM32F4 (WeAct BlackPill, ARM Cortex-M4F) for roughly $10, as a concrete substrate rather than a hypothetical one. Using its firmware, its secure element, and a real bring-up bug, we ask: (1) where a hardware-wallet pipeline leaks privacy, (2) what a secure element can verifiably guarantee, and (3) whether the cost of a hardware root of trust is why privacy-preserving self-custody stays out of reach. Section 6 is the cost point. Section 5 compares this prototype to published Ledger and Trezor architectures on two physical attacks the $10 build does not close.

2. Background

Address-reuse and multi-input clustering collapse pseudonyms into real-world clusters without breaking any cryptography [1]. The same class of heuristic applies to Ethereum through reuse and interaction graphs. Three primitives sit above a hardware signer, and only their cost on 64 KB SRAM matters here. ERC-5564 stealth addresses need elliptic-curve derivation, which this MCU already does [3]. Selective disclosure (W3C credentials / SD-JWT) lets a holder prove one claim without dumping the rest [4]. Full zk-SNARK shielding, as in Zerocash, hides origin, destination, and amount [2] — and needs working memory this board does not have (Section 7). OFAC's 2022 Tornado Cash action is why this paper treats selective disclosure as the more defensible near-term layer [6].

3. Methodology

This is a design-analysis paper, not a controlled lab experiment. The method is a structured architectural review of Noir Wallet's signing pipeline, firmware, and secure-element behavior, plus a comparison to published commercial architectures, in four steps:

1. **Decompose** the signing pipeline into discrete stages, from key generation inside the secure element to broadcast on the public chain.
2. **Classify** each stage's exposure: what an on-path or on-chain observer can see at that stage, independent of whether the key itself is ever compromised.
3. **Measure** the unlocked ATECC608A `Random` output as a Shannon-entropy object (Section 4.3), us-

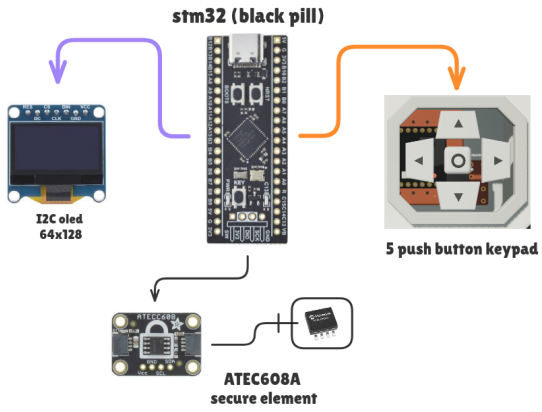

Figure 1: Noir Wallet hardware layout: STM32F401CCU6 MCU (BlackPill), SSD1306 OLED, ATECC608A secure element, and push-button input, all on a single I^2C/GPIO bus.

ing the documented diagnostic stream rather than a claimed post-lock capture. Configuration lock is one-way; this build has not been locked.

4. **Compare** the same pipeline against published Ledger and Trezor hardware designs (Section 5), then map remaining leaks to protocol-layer mitigations (Section 7).

The object of the model is the custody/privacy split, not key extraction — the threat hardware wallets are already sold against.

4. Case Study: Noir Wallet

4.1 Architecture

Noir Wallet signs Ethereum transactions offline on an STM32F401CCU6 microcontroller (WeAct BlackPill board, ARM Cortex-M4F at 84 MHz, 64 KB SRAM, 256 KB flash). Transaction fields are shown on a 128×64 OLED for human verification; a 4-digit PIN is required before any sign; the host talks USB HID, the same transport MetaMask already speaks. An ATECC608A on the same I^2C bus as the display holds keys and is the entropy source. The 64 KB SRAM figure is the constraint Section 7 uses: it is enough for signing, not for a proving system. Figure 1 shows the layout; Figures 2–4 are the physical build.

4.2 What the secure element actually guarantees

The ATECC608A is not a generic EEPROM; it is a cryptographic co-processor with hardware-enforced primitives [5]:

- **Non-exportable key storage** across 16 slots: a private key generated inside the chip can sign and derive, but the raw key material is never returned over the bus.
- **Hardware ECDSA/ECDH** on NIST P-256, so signing happens inside the secure boundary rather than in MCU RAM, which is a far larger and more frequently patched attack surface.
- **Hardware SHA-256/HMAC**, usable for firmware or command authentication independent of the MCU's own (much weaker) software stack.
- **An on-chip TRNG** specified against NIST SP 800-90A/B/C, with runtime health testing of the entropy source.
- **Monotonic counters and a one-way configuration lock**, which is the mechanism Section 4.3 turns into a worked example.

The important property for this paper is that every one of these is independently testable. A vendor claim like "bank-grade security" cannot be falsified by an outsider; a claim like "the private key never appears outside the secure element" can be checked by instrumenting the I^2C bus and confirming the key never transits it. This distinction, verifiable versus asserted, is what Section 8 returns to under the Ledger Lens.

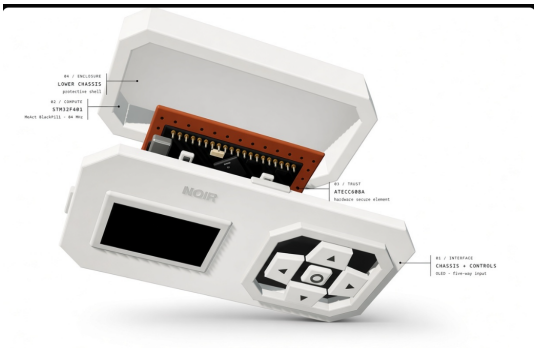

Figure 2: Enclosure CAD model.

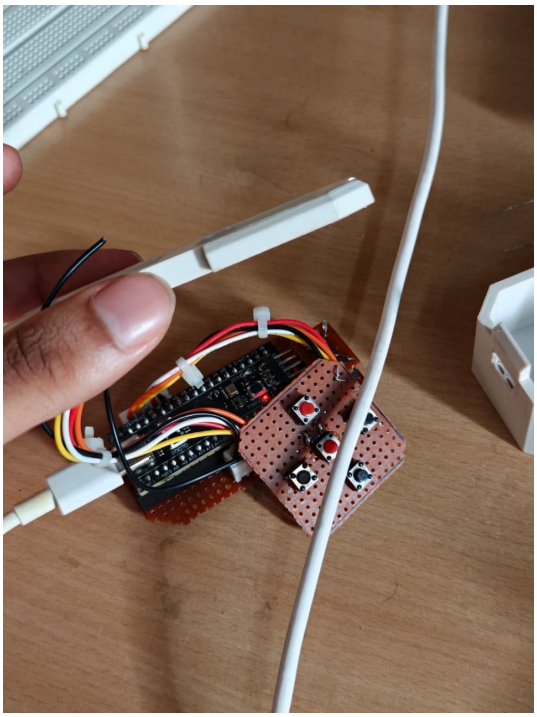

Figure 3: Assembled internals: STM32F401CCU6, button matrix, OLED header.

Figure 4: The open enclosure with OLED powered on, showing the physical arrangement referenced throughout Section 4.

Section 5 is the complementary point: the same bus test also shows what the chip does *not* hide.

4.3 A worked example: the TRNG that wasn't random

While bringing up this exact chip, we found that on an unconfigured, unlocked ATECC608A, the `Random` command (opcode `0x1B`) does not return entropy at all: it returns a fixed diagnostic pattern (`FF FF 00 00...`), by design, to make bring-up testing deterministic. Microchip confirms this behavior applies until the configuration zone is permanently locked. On this unit it produced a fixed 12-word BIP-39 mnemonic every time, since the "random" seed was not random. The fix is a single one-way command, `Lock` (`0x17`, mode `0x80`) on the configuration zone, after which the same command returns genuine entropy.

This is offered deliberately as a small, unglamorous bug rather than a success story. It illustrates exactly what "hardware-anchored trust" should mean in a paper rather than a datasheet: a property that can fail silently, that failed in a specific, reproducible, and now-documented way on this device, and that has a specific, verifiable fix. The current shipped firmware still uses a demo-grade entropy backstop (an MCU cycle-counter XOR) rather than the locked TRNG, and this is stated plainly rather than papered over: it is not a NIST-certified source. A paper making unfalsifiable security claims about its own hardware would be exactly the kind of assertion this section argues against.

Making "random" measurable. The bug is also why RNG claims need a repeated-invocation metric. Treated as one 32-byte draw, the pattern `FF FF 00 00...` still contains two symbols and can look "mixed." Across calls it does not: every invocation returns the same string, so the Shannon entropy of any fixed byte

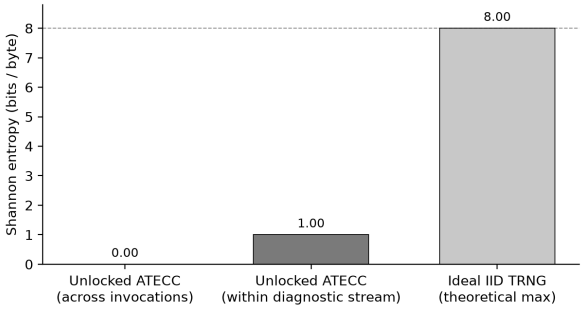

Figure 5: Shannon entropy of the unlocked ATECC608A diagnostic RNG versus the 8-bit/byte ceiling of an ideal IID byte source. Left and center bars are computed from the documented `FF FF 00 00` stream (10,000 concatenated bytes). The right bar is theoretical, not a locked-device measurement.

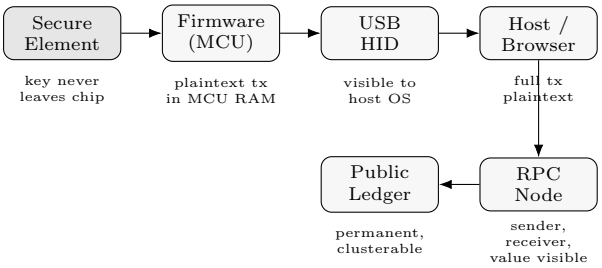

Figure 6: Exposure across the Noir Wallet signing pipeline. Only the leftmost stage is protected by the secure element; every stage after it is observable by an on-path party or, once broadcast, by anyone.

offset is $H = -\sum_i p_i \log_2 p_i = 0$ bits. Concatenating the documented diagnostic stream into 10,000 bytes yields only the symbols `0x00` and `0xFF` at equal frequency, so within-stream entropy is exactly 1.00 bit/byte. An ideal IID uniform byte source sits at 8 bits/byte. Figure 5 plots those three numbers. The right-hand bar is a theoretical ceiling, not a measured post-lock capture: lock is one-way, and this firmware has not taken that step. NIST SP 800-90B's Repetition Count Test and Adaptive Proportion Test are built to catch exactly this class of stuck source [7].

4.4 Where privacy actually leaks

Figure 6 is the same pipeline scored for information, not for key theft. Only the secure element is dark. Past that hardware boundary the USB HID path and host see the unsigned plaintext, the RPC node sees sender, receiver, and value, and the settled transaction is public and clusterable [1]. The interesting line is the chip edge. The hops after that are the usual public-ledger story and are not retold here.

PIN and offline signing are custody: stolen-device lockout, and no key on the host. They do not stop that host from reading the composed transaction, reusing an address, or logging metadata. That is the air-gapped-signer pattern used by essentially every hardware wallet, not a Noir-only bug (Sections 7 and 8).

Table 1 summarizes this at the level of custody model

Table 1: Custody model versus privacy guarantee.

Model	Key custody	Transaction privacy
Hot wallet	Weak (host RAM)	None
Hardware wallet	Strong (secure element)	None
Hardware wallet + stealth address	Strong	Recipient unlinkability only
Hardware wallet + ZK proof	Strong	Origin/amount hidden, needs off-chip proving

rather than pipeline stage.

5. Hardware Threat Model and V1 Mitigations

Section 4 established that the ATECC608A can keep a private key off the wire. That is not the same as a trustworthy display, and it is not the same as a confidential bus. This section names two physical attacks that commercial designs spend their architecture on, and that Noir Wallet V0 fails. Neither attack extracts the key. Both break the user's ability to know what was signed, or who else saw the signing session.

5.1 Display substitution (compromised MCU)

On this board the STM32 owns the OLED and also owns the I^2C commands sent to the ATECC608A. A malicious or replaced firmware image can show "send 0.1 ETH to Alice" on the screen and submit a different digest to the secure element. The ATECC608A will sign whatever hash it is given. It has no path to the pixels. That is a failure of What-You-See-Is-What-You-Sign (WYSIWYS), not of key custody. A user who reads the OLED carefully still authorizes the attacker's payment. The PIN does not help: it only unlocks signing, it does not bind the digest to the pixels.

Ledger's published architecture inverts that trust: application logic and secrets run on an ST33-class secure element; an STM32 MCU is an untrusted proxy for USB and peripherals; display updates are issued by the SE over the SEPROXYHAL link rather than composed by the MCU as the source of truth [8]. Ledger's own write-up of a Nano X MCU debug issue states the same split: the MCU is not trusted, and it is not supposed to be able to change what the secure display shows [9]. This paper does not claim that split is "mathematically impossible" to break. It claims the trust boundary is in a different place than on Noir Wallet, where the MCU is both the display driver and the SE client.

5.2 Plaintext I^2C sniffing

The OLED (`0x3C`) and the ATECC608A (`0x60`) share one unencrypted I^2C bus. A clip-on logic analyzer on SDA/SCL can record opcodes, slot indices, the digest sent for `Sign`, and the bytes written into the display buffer. The raw key still does not appear. Transaction metadata and screen contents do. Isolation between the two devices is a firmware convention, not a physical one: both sit on the same two header pins. The PIN on this firmware is checked on the MCU, so a tap of the ATECC lines does not automatically yield the PIN; a tap of the OLED lines does yield whatever the user was shown. Encrypted I/O on the ATECC608A would close the SE half of that tap; it would not close the OLED half.

Trezor Safe 3 and Safe 5 add an Infineon OPTIGA Trust M (V3) beside the main MCU and use it for PIN-gated secrets, genuineness, and entropy [10]. That part supports a shielded I^2C session (AES-128-CCM) between host MCU and secure element [11]. A bus tap of that channel sees ciphertext. It does not, by itself, encrypt the OLED. The relevant comparison for Noir Wallet is therefore narrower than a slogan: commercial designs encrypt or relocate the SE link; this prototype leaves that link in the clear, even though the ATECC608A datasheet already lists an encrypted I/O mode that this build has not turned on [5].

5.3 What V1 has to change

V0 protects remote key exfiltration. It does not protect local confidentiality or display integrity. The next revision does not need a closed ST33 to close the cheapest of these gaps: enable the ATECC608A I/O-protection secret so SE traffic is not plaintext; put the OLED on a separate bus or treat its framebuffer as attacker-visible; keep signing gated on a button event the user can correlate with the screen. Encrypted I/O is already in the ATECC608A datasheet [5]; it is a configuration bit this build has not set, not a new chip. A secure-element-owned display path, Ledger's published model, is the stronger WYSIWYS end-state and is also the expensive one. The open question for Section 8 is whether the encrypted-bus half of that stack can stay in the $10 class.

6. Cost as a Privacy Variable: A Ten-Dollar Root of Trust

Section 4.2 described a set of hardware guarantees, non-exportable keys, hardware ECDSA, an auditable TRNG, that are the same class of primitive found in commercial hardware wallets costing $60–$250. Noir Wallet builds those guarantees for roughly $10 in total: an STM32F401CCU6 board ($3–5), an SSD1306 OLED ($2–3), the ATECC608A secure element itself ($1–2), and a handful of push buttons and passives ($1).

The marginal cost of hardware-anchored key custody is not why self-custody stays rare. A $1–2 secure element

buys the Section 4.2 guarantees — non-exportable keys, hardware ECDSA, an auditable TRNG — on a $10 board or a $150 device. If later privacy tooling needs that anchor, it should be treated as cheap and reproducible, not as a branded premium. Section 5 is the limit of the claim: the $10 BOM buys the key, not display integrity and not a confidential bus. This is not a product pitch, and it is not a claim about Common Criteria, supply-chain assurance, or tamper packaging, all of which commercial devices do and this prototype does not.

7. Where Privacy-Enhancing Technology Plugs In

Mapping Section 4.4's leak points onto Section 2's primitives gives a rough ordering by feasibility on constrained hardware like an STM32F401CCU6 (84 MHz Cortex-M4F, 64 KB SRAM, 256 KB flash):

Stealth addresses. ERC-5564 is elliptic-curve scalar multiplication this MCU already performs for ECDSA, or can hand to the ATECC608A ECC engine [3]. Recipient unlinkability does not touch key custody.

Selective disclosure. Hold a signed credential; disclose one SD-JWT claim at spend time [4]. The signature stays on-chip. Disclosure logic can live on the MCU or a companion app.

Full ZK shielding. Zerocash-class circuits are millions of R1CS constraints and large-integer modular arithmetic [2]. That is a memory ceiling: 64 KB SRAM cannot hold the witness. The Cortex-M4F floating-point unit does not change it, because proving is field arithmetic, not floats. Near-term design: generate the proof on a paired phone, keep only the authorizing signature on the secure element. The proving host then sees the private inputs even if the chain does not.

8. Ledger Lens: Hardware Trust Is Necessary, Not Sufficient

Ledger's own position — self-custody, hardware-anchored trust — is the right frame for this paper's claim, and the critique should be stated in those terms, not around them. Hardware-anchored trust answers "can I prove this key was never exposed?" It does not answer, and was never designed to answer, "can I prove I sent value without revealing who I am or how much?" Those are different guarantees on different mechanisms. Section 4 showed the first one is testable on this chip: dump `Random` before lock, get a constant; lock, or refuse to ship. A vendor sentence that cannot be falsified is not that kind of claim.

The honest version of the Ledger Lens for a privacy paper is this: any credible privacy-preserving wallet design still needs a hardware root of trust underneath it, because a signature or a ZK-proof authorization is only as trustworthy as the key that produced it, and software-only key storage has a long, well-documented history of exfiltration. Section 5 adds the physical half of that sentence. A $10 ATECC608A stops remote key theft. It does not stop a malicious STM32 from lying about the screen, and it does not stop a logic analyzer from reading the bus. Ledger's published answer to the first problem is an SE-owned display path [8, 9]. Trezor Safe 3/ 5's published answer to the second, on the SE link, is a shielded MCU–SE channel [10, 11]. Those are real engineering costs, not packaging. The unresolved question this paper wants on the table is narrower than "why does a hardware wallet cost $150?": does *true* self-custody — key isolation plus WYSIWYS plus a non-plaintext SE bus — have to stay at that price, or can an open $10 board take the encrypted-I^2C step without buying a closed display-controller SE? Hardware-anchored trust remains necessary. On this prototype it is also not sufficient, and the part that is cheap is only the key.

9. Limitations and Discussion

This is a design analysis of one open-source device, not an empirical study across hardware wallets, and it makes no claim that Noir Wallet firmware is production-secure. The TRNG finding in Section 4.3 is an unlocked, demo-grade state, not a post-lock measurement. Figure 5's 8-bit bar is a ceiling, not a captured locked-TRNG dataset. The mapping in Section 4.4 and the two attacks in Section 5 are threat models, not logic-analyzer captures, and they omit power and EM side channels against the ATECC608A. The cost argument in Section 6 is bill-of-materials only.

Section 5 already carries the I^2C and display-integrity limits. Section 7 is feasibility only: no stealth-address firmware and no proving pipeline were implemented. A phone-assisted ZK split is named as future work, not as a result.

10. Conclusion

Using a ten-dollar open-source hardware wallet as the object of study, this paper traced where custody ends and privacy exposure begins: at the secure-element boundary. The unlocked ATECC608A `Random` output — Shannon entropy 0 bits across invocations, 1 bit/byte inside the diagnostic stream — is what a verifiable hardware claim looks like. Section 5 named the two physical gaps V0 does not close: an MCU that can lie about the screen, and a plaintext I^2C bus. Stealth addresses and selective disclosure fit the same SRAM budget; full ZK shielding does not. Hardware-anchored trust remains the foundation. On this board it is cheap for the key, and still insufficient for display integrity, bus confidentiality, and on-chain privacy.

Originality Statement

This paper is the original work of the listed author(s). All external claims are cited to their sources. AI tools were used as a working aid (research synthesis, drafting assistance); no section was generated end-to-end by an AI sys-

tem without author review and rewriting. *[Signature / date placeholder — complete before submission.]*

References

[1] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, S. Savage, "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names," *Proc. Internet Measurement Conference (IMC)*, ACM, 2013, pp. 127–140.

[2] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, "Zerocash: Decentralized Anonymous Payments from Bitcoin," *IEEE Symposium on Security and Privacy*, 2014.

[3] T. Wahrstätter, M. Solomon, B. DiFrancesco, V. Buterin, "ERC-5564: Stealth Addresses," Ethereum Improvement Proposals, 2022. `https://eips.ethereum.org/EIPS/eip-5564`

[4] World Wide Web Consortium, "Securing Verifiable Credentials using JOSE and COSE," W3C Recommendation, 2025. `https://www.w3.org/TR/vc-jose-cose/`

[5] Microchip Technology Inc., "ATECC608A: CryptoAuthentication Device Data Sheet," Doc. DS40001977A, 2019.

[6] U.S. Department of the Treasury, Office of Foreign Assets Control, "U.S. Treasury Sanctions Notorious Virtual Currency Mixer Tornado Cash," Press Release, Aug. 8, 2022. `https://home.treasury.gov/news/press-releases/jy0916`

[7] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, M. Boyle, "Recommendation for the Entropy Sources Used for Random Bit Generation," NIST Special Publication 800-90B, Jan. 2018.

[8] Ledger, "Hardware architecture," Ledger Developer Portal. `https://developers.ledger.com/docs/device-app/explanation/ledger-os/hardware-architecture`

[9] Ledger, "Enhancing the Ledger Nano X's Security," Ledger blog, 2020. `https://www.ledger.com/enhancing-the-ledger-nano-xs-security`

[10] Trezor, "Secure Elements in Trezor Safe devices." `https://trezor.io/learn/security-privacy/how-trezor-keeps-you-safe/secure-elements-in-trezor-safe-devices`

[11] Infineon Technologies, "OPTIGA Trust M: Shielded Connection," Knowledge Base Article. `https://community.infineon.com/t5/Knowledge-Base-Articles/OPTIGA-Trust-M-Shielded-connection/ta-p/354380`