CA-less Mutual Co-Signing of Documents over a Unidirectional Visual Channel with Transported Hardware Attestation

Dmytro Diikun*
Independent Researcher, Ukraine
diikunapp@gmail.com
ORCID: 0009-0007-5303-3358
Also available at Zenodo: doi:10.5281/zenodo.22055260

August 2026 (preprint)

Abstract

We describe and analyze a protocol for mutual co-signing of a document by two mobile devices that (i) communicate only over a one-way, lossy, low-bandwidth optical channel (an animated on-screen code read by the counterparty's camera), (ii) use no intermediary server on the trust path, and (iii) use no certificate authority. Trust in each party's public key is instead grounded in a hardware attestation token produced by the platform secure element, transported in full over the visual channel by a rateless (fountain) code and cryptographically bound into the co-signature. The core technical contribution is a *two-stage hash anchor* that removes the circular signing dependency inherent to interactive co-signing: the first party commits to the document before the identity of the second party is known, and the second party's identity is later bound to that commitment without invalidating the first signature. We give a threat model, define four security properties—anchor binding, co-signature inseparability, attestation-bound key provenance, and post-signing tamper evidence—and reduce them to standard assumptions (collision resistance of H and

*Aspects of the construction described here are the subject of a pending patent application by the author.

EUF-CMA security of the underlying signature scheme), with the secure element modeled as an ideal signing oracle. We report a working instantiation on iOS/Android using ECDSA P-256 in the Secure Enclave/StrongBox, SHA-256, Apple App Attest / Play Integrity, and an LT-style fountain code, together with an independent third-party verifier that recomputes all anchors and checks both signatures fully offline.

Keywords. digital signatures · hardware attestation · certificate-less trust · offline protocols · fountain codes · secure element · signature chaining.

1 Introduction

Problem. Two people meet in person, each holding a phone, and want to co-sign a document so that a third party can later verify both signatures and the fact that each signing key lives in genuine device hardware. The setting we target has three hard constraints simultaneously: no network at signing time; no server that the signed data passes through; and no certificate authority (CA) to vouch for either public key. The only channel is optical: one phone displays a machine-readable code, the other reads it with its camera. The channel is one-way per turn, lossy (dropped frames), and capacity-limited.

Why this is not trivial. Two difficulties compound.

1. *Circular dependency.* In naive interactive co-signing, each party wants to sign "the final document," but the final document includes the other party's identity and signature, which do not yet exist. Party A cannot sign until B is known; B cannot be bound until A has signed. We break this with a two-stage hash anchor (§5.2).

2. *Key trust without a CA.* With no CA and no network, a receiver cannot fetch or validate a certificate chain for the sender's key online. We replace certificate-based trust with *transported hardware attestation*: the full platform attestation token, which cryptographically ties the public key to key generation inside a genuine secure element, is carried over the optical channel itself and bound into the co-signature (§5.4). Because a full attestation object exceeds the capacity of a single static optical code, this is made practical only by rateless framing (§5.5).

Contributions.

- A two-stage hash-anchor construction that provably removes the circular signing dependency while keeping the first signature stable (§5.2, §6.1).
- A signature-chaining rule that makes the second signature inseparable from a *specific* first signature, preventing "re-pasting" a signature onto a different document or a different first signature (§5.3, §6.2).
- A CA-less key-provenance mechanism based on transporting the full hardware-attestation token point-to-point over the visual channel and binding its hash into the signed anchor, with an offline verification path when the attestation scheme offers a certificate chain to a manufacturer root and a detachable third-party path otherwise (§5.4, §6.3).
- An engineering instantiation and a standalone verifier that reproduces all anchors and both signatures with no network access (§7).

Scope. This paper is about a cryptographic protocol. It makes no claim about the legal status of its output in any jurisdiction, and we do not claim the scheme is a substitute for a qualified or PKI signature; the absence of a CA is a design goal with explicit trade-offs, discussed in §8. Binding a key to a natural person is an application-layer step outside the offline core and is deliberately excluded from the protocol and its analysis.

2 Related Work

Our construction sits at the intersection of five lines of work. For each we state what it provides and how our setting differs; the contribution is the *combination* (§2), not any single ingredient.

Multi-signatures and co-signing. Multi-signature schemes let a group of signers produce a single *compact* signature on a *common* message; Bellare and Neven [1] give a scheme secure in the plain public-key model, and Schnorr-based constructions such as MuSig [2] refine round complexity. These target signature *aggregation* under a PKI or the plain-public-key model and are interactive protocols run over a network. Our setting differs on three axes. First, we do not aggregate: the two parties produce two *distinct* signatures over *role-separated* messages (m_A, m_B), and it is their *chaining* that carries security (§5.3). Second, signing is *ordered and asymmetric*: the initiator commits before the responder's identity exists, which

our two-stage anchor (§5.2) makes possible. Third, key trust is established without certificates and without a network at signing time.

Certificate-less and identity-based public-key cryptography. Al-Riyami and Paterson [3] introduced certificate-less public-key cryptography (CL-PKC) to remove certificates while avoiding identity-based key escrow. CL-PKC still relies on a key-generation centre (KGC) holding a master secret. Our scheme also dispenses with certificates, but its trust root is different in kind: the *hardware manufacturer's attestation root*, reached and checked without any network at signing time (§5.4). We do not replace a CA with a KGC; we replace certificate-based key trust with transported hardware attestation.

Remote and hardware attestation. Platform attestation services—Apple App Attest / DeviceCheck [9], Google Play Integrity [10], Android Key Attestation [11], and, in the anonymous setting, TPM-based Direct Anonymous Attestation [4]—let a relying party check that a key was generated inside genuine device hardware. In standard deployment the token is verified *server-side*. Our use is different: the *full* token is transported peer-to-peer over the visual channel and *bound into the mutual signature* (§5.4), so a third party can assess key provenance either fully offline (when the token carries a certificate chain to a pinned manufacturer root, as with App Attest) or as a detachable step against the manufacturer's service (as with Play Integrity)—in both cases without a CA.

Rateless codes and animated visual channels. Fountain codes—LT [5], Raptor [6], and RaptorQ [7]—recover a k-symbol payload from any sufficiently large set of encoded symbols over an erasure channel [8], with no feedback channel. Combining them with animated on-screen barcodes to move data screen-to-camera is established in practice (e.g. TXQR [12], Cimbar [13]). In all of these, the animated fountain stream is a *one-way data pipe*; none binds the transport to a mutual signing act or to hardware attestation. We use rateless framing for one structural reason: a full attestation token exceeds the capacity of a single static optical code, so transporting it point-to-point—and thereby achieving CA-less provenance at all—is only practical with a rateless code (§5.5). We make no optimality claim for our specific degree rule and defer optimal rateless design to the LT/Raptor line (§5.5, §8).

Content provenance and capture-time signing. Content-authenticity frameworks, notably C2PA / Content Credentials [14], attach a signed manifest asserting an asset's origin and edit history. Two differences are salient. First, C2PA is *one-sided* capture/edit provenance, not a mutual co-signature between two independent parties. Second, C2PA hard bindings hash the *asset bytes*, so any re-encoding invalidates the binding; we deliberately anchor to the *canonical textual* content and *exclude* display-file bytes (§5.6, §5.6), so the display artifact can be regenerated without breaking either signature.

The combination. To our knowledge, no prior work co-signs a document *mutually*, *offline*, with *certificate-less* key trust established by a *transported hardware-attestation token* carried over a *rateless visual channel*, and with a third party able to reproduce both signatures and both provenance checks without network access. Each ingredient is individually known; the novelty we claim is their composition and the two mechanisms that make it work—the two-stage anchor that removes the circular signing dependency, and the signature chaining that makes the second signature inseparable from a specific first signature.

3 Preliminaries and Notation

Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be a collision-resistant hash function (instantiated as SHA-256). We write $x \parallel y$ for concatenation and use two non-printing control bytes as field delimiters: $\langle \mathsf{RS} \rangle$ (ASCII 30, "record separator", `0x1E`) between top-level fields, and $\langle \mathsf{US} \rangle$ (ASCII 31, "unit separator", `0x1F`) inside a compound field. All strings are UTF-8 encoded before hashing.

A *secure-element signature scheme* $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a standard signature scheme (instantiated as ECDSA over NIST P-256 with SHA-256, DER-encoded, Base64-serialized [15, 16]) with the additional property that sk is generated inside, and never leaves, a hardware secure element; the signing operation is gated by a user-presence factor (biometric). We model the secure element as an oracle $\mathcal{O}_{\mathrm{SE}}$ that holds sk and, on input m, returns $\mathsf{Sign}(sk, m)$ only after a presence check; sk is never exposed. This matches an EUF-CMA signing oracle.

An *attestation scheme* $\mathsf{Att} = (\mathsf{Gen}, \mathsf{AttVer})$ produces, for a secure-element public key pk and a challenge c, a token τ such that $\mathsf{AttVer}(pk, c, \tau)$ accepts only if pk was generated inside a genuine secure element of a device of the attested platform, and c was bound at generation time. We denote by $\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{Att}}$ the advantage of a PPT adversary against this soundness property.

Remark 1 (Production vs. development attestation)**.** The soundness assumption is meaningful only for *production* attestation. A development-environment token (distinguishable by its attestation metadata, e.g. the App Attest environment / `aaguid`) must not be treated as the full hardware guarantee. On a device with no secure element the platform path is unavailable; the implementation then emits a reduced-assurance marker ($isReal = \text{false}$) that is visible to any verifier and carries no hardware-origin claim (§7).

4 System Model and Threat Model

4.1 Parties and channel

Two mobile devices, D_A (initiator) and D_B (responder), each with a secure element, a camera, and a display. The primary channel is optical: $D_A \to D_B$ (A's animated code read by B), then $D_B \to D_A$ (B's animated code read by A). No server carries signed data; no CA is contacted.

An optional *online variant* exists in which a relay forwards frames between remote devices. The relay is explicitly *not on the trust path*: every datum it forwards is already committed and signed before it reaches the relay, and the third-party verification recomputes all anchors from the committed fields, so a malicious relay can at most cause a denial of service, never a change of accepted content. The same non-network channels (NFC, Bluetooth/BLE, ultrasound, direct device-to-device links, or machine-readable data embedded in a carrier file) may substitute for the optical channel; the unifying property is that the self-contained structure travels without signed data passing through any server.

4.2 Adversary

We consider four adversaries, analyzed separately.

$\mathcal{A}_{\text{net}}$ — **hostile relay/channel.**
Controls any relay or optical path; may drop, reorder, replay, or inject frames. Goal: alter the accepted content, or make a verifier accept a document neither party signed.

$\mathcal{A}_{\text{repaste}}$ — **signature re-paster.**
A malicious counterparty or channel observer who has seen valid (pk_A, σ_A) and/or B's signature and tries to transplant a signature onto a different document or a different first signature.

$\mathcal{A}_{\text{key}}$ — **key-provenance forger.**
Tries to make a verifier accept a key as hardware-backed when it is not (e.g. a software key on an emulator), or to replay another device's token for a fresh session.

$\mathcal{A}_{\text{post}}$ — **post-signing tamperer.**
Given the fully signed document, tries to change any field (amount, party, time, geolocation, evidence) without detection.

4.3 Trust assumptions

H is collision-resistant; Sig is EUF-CMA-secure; the secure element does not leak sk. The attestation manufacturer root(s) are authentic and are embedded (pinned) in the verifier out of band. *Out of scope, stated explicitly:* coercion or duress of a present user; malware with full control of a rooted/jailbroken device that can itself drive the biometric prompt; and the legal weight of the output. The implementation carries a `duressFlag`; it is a UX signal only, is *not* integrity-protected, and is outside the cryptographic threat model.

5 Construction

We present the scheme bottom-up: canonical encoding (§5.1), the two-stage anchor (§5.2), signature chaining (§5.3), attestation transport and binding (§5.4), rateless framing (§5.5), and the composite verification hash (§5.6). The analysis assumes only that the canonicalization enc is injective and domain-separated by a scheme tag ver (§5.1); the two serializations we use both satisfy this (Lemma 1). The worked example in Appendix A is synthetic and reproducible.

5.1 Canonical field encoding and separator-injection defense

A fixed, ordered field vector $\vec{C}$ is serialized to a canonical UTF-8 string $\mathsf{enc}(\vec{C})$ that is *injective* (distinct vectors yield distinct byte strings) and *domain-separated* by a scheme tag ver. We use two standard injective serializations behind the tag. In the *control-byte-escaped* form, every value passes through an escaping map $f(\cdot)$ that rewrites any embedded $\langle\mathsf{RS}\rangle$ (the top-level separator) to $\langle\mathsf{US}\rangle$, and compound sub-fields use $\langle\mathsf{US}\rangle$; this is the form of the composite record (§5.6). In the *length-prefixed* (TLV) form, every value is emitted as $\langle\text{byte-length}\rangle : \langle\text{value}\rangle \setminus\text{n}$, so a separator inside a value cannot

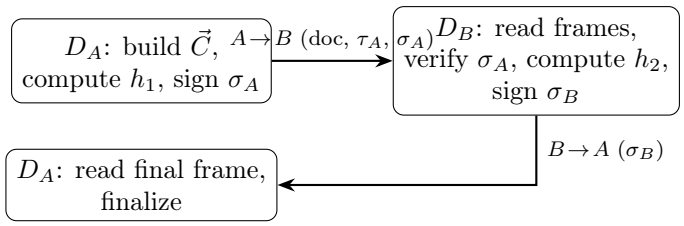

Figure 1: Two optical turns. h_1 is committed before B exists; h_2 binds B without invalidating σ_A.

forge a field boundary regardless of its content. Both prevent the concatenation collisions in which two distinct field vectors serialize to the same byte string (Lemma 1); the length-prefixed form is injective unconditionally, the escaped form on the admissible domain (Remark 2). We write $\mathsf{enc}(\vec{C})$ for either serialization of an ordered content vector $\vec{C}$.

5.2 Two-stage hash anchor

Let $C = \mathsf{enc}(\vec{C})$ denote the canonical content string containing the document terms and *only* party A's identity fields (no B fields, no signatures, no public keys). Define the first anchor

$$h_1 \;=\; H(\mathsf{ver} \,\|\, \langle \mathsf{RS} \rangle \,\|\, C)\,.$$

Crucially, h_1 is computable by D_A *before* B's identity is known. When B joins, define the second anchor as a deterministic function of h_1 and B's identity fields id_B (nickname, avatar index, and an optional verified-contact hash):

$$h_2 \;=\; H\big(\mathsf{ver} \,\|\, \langle \mathsf{RS} \rangle \,\|\, h_1 \,\|\, \mathsf{enc}(id_B)\big),$$

computed with the same injective canonicalization as h_1. The scheme tag ver—a short literal fixing the canonicalization, such as the deployed `v81`/`v82` identifiers—domain-separates schemes and rules out cross-scheme collisions. Because h_2 is a deterministic function of h_1, the first signature (below) remains valid after B joins—no re-signing round is needed, so the exchange collapses from three rounds to two optical turns (Figure 1).

5.3 Signatures and chaining

With document identifier id, the two signed messages are

$$m_A = \text{“A:”} \,\|\, id \,\|\, \text{“:”} \,\|\, h_1, \qquad m_B = \text{“B:”} \,\|\, id \,\|\, \text{“:”} \,\|\, h_2 \,\|\, \text{“:”} \,\|\, pk_A \,\|\, \text{“:”} \,\|\, \sigma_A,$$

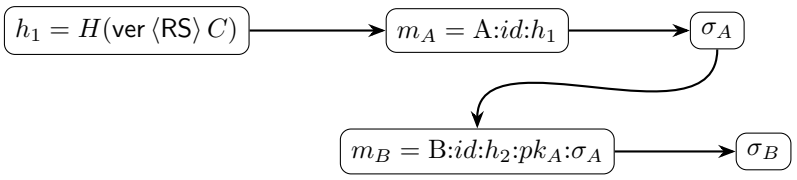

Figure 2: Signature chaining: σ_A and pk_A are inputs to m_B, so σ_B is inseparable from this specific σ_A.

and $\sigma_A = \mathsf{Sign}(sk_A, m_A)$, $\sigma_B = \mathsf{Sign}(sk_B, m_B)$, each produced inside the respective secure element under biometric gating. The domain-separating prefixes "A:"/"B:" prevent cross-role message collisions. Because m_B contains *both* pk_A and σ_A, the second signature is bound to one specific, already-existing first signature (Figure 2).

5.4 Transported hardware attestation and key provenance

D_A obtains a full attestation token τ_A binding pk_A to secure-element key generation, with a fresh single-use challenge c committing to the key: the platform attests a client-data hash $cdh = H(c \,\|\, H(pk_A))$. The *full* τ_A (not merely a boolean "is-real") is placed in the transfer structure; only its context-separated digest $H(\text{“}\mathsf{ctx\_device\_A}\text{:”} \,\|\, \tau_A)$ enters the signed anchor set (§5.6). Verification has two modes: (i) *offline chain mode*—the token carries a certificate chain to a manufacturer root pinned in the verifier (e.g. Apple App Attest on iOS), checked locally with no network; and (ii) *detachable service mode*—the token is verifiable only against the manufacturer's service (e.g. Google Play Integrity on Android), so the offline phase transports and binds τ and the hardware-origin check completes as a detachable third-party step. Soundness and no-replay are proved in §6.3.

5.5 Rateless framing of the transfer structure

The self-contained transfer structure (content fields, h_1, σ_A, pk_A, full τ_A, scheme id, session id) is compressed and split into k base fragments (≤ 675 bytes each). Base frames carry one fragment; fountain frames carry the XOR of a pseudo-random subset chosen by a linear congruential generator (multiplier 1664525, increment 1013904223, modulus 2^{32}, seeded by the frame sequence number), with redundancy factor 3 and degree $(\text{seq} \bmod k) + 1$. The decoder peels fragments and confirms completion via a short checksum (a prefix of $H(\text{payload})$). Because the code is rateless, no back-channel is

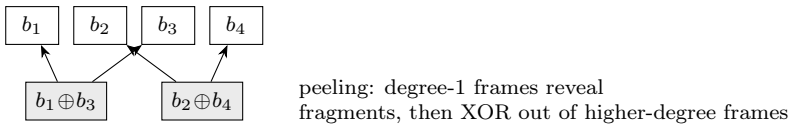

Figure 3: Fountain framing and peeling decode over the lossy optical channel.

needed to request lost frames (Figure 3). We emphasize that the specific LCG-XOR degree rule is an *engineering choice*, not an optimized construction: we claim only that it decodes reliably at the payload sizes in use. Optimal rateless design (Robust-Soliton LT, Raptor/RaptorQ) is orthogonal and applies unchanged as a drop-in replacement.

5.6 Composite verification hash

For archival and third-party checking, a composite hash *finalHash* is computed over an ordered 39-field record in the control-byte-escaped form of §5.1 (both parties' public keys, signatures, biometric-backing flags, device-token hashes, timestamps, geolocation, evidence hashes, identity hashes, and h_1), with the B-block included only when $\sigma_B \neq \varepsilon$. Every value is passed through the escaping map f; the separator $\langle \mathsf{RS} \rangle$ is written after every field. The exact field order is reproduced verbatim by the reference verifier and listed in Appendix B.

Time-source field. One canonical field records the time source as a short string of the form $\langle \text{ISO-8601} \rangle \mid \langle \text{source-id} \rangle$, where the source identifier ranks, in decreasing order of trust, a relay-authenticated timestamp, an NTP-synchronized time, and the local device clock (marked as such). The security-relevant fact is only that this string is included verbatim in the canonical text and therefore covered by h_1: by Theorem 3, any retroactive substitution of the source (e.g. silently downgrading to the local clock, or claiming a higher-trust source) changes the recomputed anchor and is detected. We make no non-repudiation claim for the time source itself; a higher-trust timestamp is a strengthening input, not part of the offline security core.

Anchor detached from display bytes. The cryptographic anchor is the canonical *text* (h_1, *finalHash*), *not* the bytes of any display file (e.g. PDF). Non-deterministic rendering bytes are deliberately excluded from the canonical representation, the signed messages, and the transfer structure, so the display artifact can be regenerated at will without breaking any signature.

Generality and variants. The construction is stated for two parties over an optical channel, but the same mechanisms apply to a family of instantiations. *Number of parties:* the two-party case is the base of a sequential N-party co-signing ($N \geq 2$), where the i-th signature is taken over an anchor depending on the i-th party's identity fields and chained to the public keys and signatures of all previous parties; the one-sided (N=1) case is a special case. *Channel:* any non-network carrier of the self-contained structure works—animated optical code, NFC, Bluetooth/BLE, ultrasound, a direct device-to-device link, or machine-readable data embedded in a carrier file—the unifying property being that signed data never passes through a server. *Primitives:* H may be any collision-resistant hash (SHA-256, SHA-512, SHA-3, BLAKE2/BLAKE3); Sig any EUF-CMA scheme with a secure-element key (ECDSA P-256/secp256k1, Ed25519, RSA); the secure element any of Secure Enclave, StrongBox, TEE, TPM, or smart card; and the attestation any scheme exposing key-in-hardware provenance (App Attest/DeviceCheck, Play Integrity, Android Key Attestation, FIDO, TPM quote). *Framing:* the rateless code may be LT, Raptor/RaptorQ, an online code, or a fixed multi-frame scheme recoverable from a subset. A trusted timestamp (e.g. RFC 3161) is an optional, detachable strengthening, not part of the offline core. All statements below depend only on the abstract properties (collision resistance of H, EUF-CMA of Sig, soundness of Att, injectivity of enc), so they carry over to every instantiation in this family unchanged.

6 Security Analysis

We define and prove four properties. Throughout, the secure element is modeled as an ideal signing oracle (§3), and "accept" means the third-party verifier's check passes:

$$\mathsf{Vrfy}^*(R) = 1 \iff \mathsf{Vrfy}(pk_A, m_A, \sigma_A) = 1 \ \wedge \ \mathsf{Vrfy}(pk_B, m_B, \sigma_B) = 1,$$

where $R = (id, C, id_B, pk_A, \sigma_A, pk_B, \sigma_B)$ and m_A, m_B are derived as in §5.3. Attestation checks are orthogonal and treated in §6.3.

6.1 Encoding injectivity and anchor binding

Lemma 1 (Encoding injectivity)**.** *Fix a field schema of arity n (an ordered list of n slots, each either a simple slot or a compound slot of two sub-fields). Let* enc *apply the escaping map f to every field value (replacing every* $\langle\mathsf{RS}\rangle$ *byte inside a value by* $\langle\mathsf{US}\rangle$*), join the n resulting tokens with* $\langle\mathsf{RS}\rangle$*, and join*

the two sub-fields of a compound slot with $\langle\mathsf{US}\rangle$*; the byte* $\langle\mathsf{RS}\rangle$ *appears in the output only as a top-level separator. Then, restricted to inputs whose sub-field values contain no* $\langle\mathsf{US}\rangle$ *byte in a position that would forge a sub-boundary,* enc *is injective: for any two field vectors* $\vec{v} \neq \vec{w}$ *of the same schema,* $\mathsf{enc}(\vec{v}) \neq \mathsf{enc}(\vec{w})$.

Proof. The arity n and the type of each slot are fixed and public, so a decoder knows where boundaries are expected; we show enc is decodable, which implies injectivity. Given $s = \mathsf{enc}(\vec{v})$, split s on $\langle\mathsf{RS}\rangle$. By construction $\langle\mathsf{RS}\rangle$ occurs only between top-level tokens (inside every value f has rewritten each $\langle\mathsf{RS}\rangle$ to $\langle\mathsf{US}\rangle$), so the split yields exactly the n tokens $f(v_1), \ldots, f(v_n)$ in order. For a compound slot, split its token on $\langle\mathsf{US}\rangle$; the schema fixes the slot as compound and forbids a spurious $\langle\mathsf{US}\rangle$, recovering the ordered pair. Thus $\mathsf{enc}(\vec{v})$ determines $(f(v_1), \ldots, f(v_n))$ componentwise. If $\vec{v} \neq \vec{w}$ differ in component i with $f(v_i) \neq f(w_i)$, the encodings differ. The remaining case $v_i \neq w_i$ but $f(v_i) = f(w_i)$ requires values differing only in $\langle\mathsf{RS}\rangle$-vs-$\langle\mathsf{US}\rangle$ bytes; on the admissible domain f is the identity there, so $f(v_i) = f(w_i) \Rightarrow v_i = w_i$, a contradiction. Hence $\mathsf{enc}(\vec{v}) \neq \mathsf{enc}(\vec{w})$. □

Remark 2 (Domain caveat, stated honestly)**.** Injectivity holds on the admissible domain, i.e. where field values do not themselves contain a raw $\langle\mathsf{RS}\rangle$ that a caller expected to survive as data. In practice the hashed fields are UUIDs, base64/hex digests, ISO-8601 timestamps, and short human strings, none of which legitimately contain the control bytes $\langle\mathsf{RS}\rangle$ (`0x1E`) or $\langle\mathsf{US}\rangle$ (`0x1F`); the scheme tag ver additionally domain-separates the scheme. The length-prefixed (TLV) serialization is a standard technique that removes the caveat entirely: it is injective unconditionally. When it is used for the content anchors, Lemma 1 holds without restriction; the escaped form is retained for the composite record and for verifying documents signed under it. Both coexist behind the scheme tag.

Definition 1 (Anchor binding)**.** An adversary wins if it outputs either (i) two distinct A-content vectors $\vec{C} \neq \vec{C}'$ with $h_1(\vec{C}) = h_1(\vec{C}')$, or (ii) two distinct bindings $(h_1, id_B) \neq (h_1', id_B')$ with $h_2 = h_2'$. Its advantage is $\mathrm{Adv}^{\mathrm{bind}}$.

Theorem 1 (Anchor binding)**.** *If* H *is collision-resistant and* enc *is injective on the admissible domain (Lemma 1), then* $\mathrm{Adv}^{\mathrm{bind}} \leq \mathrm{Adv}^{\mathrm{cr}}_H$.

Proof. Case (i): $h_1(\vec{C}) = h_1(\vec{C}')$ means $H(\mathsf{ver} \,\|\, \langle\mathsf{RS}\rangle \,\|\, \mathsf{enc}(\vec{C})) = H(\mathsf{ver} \,\|\, \langle\mathsf{RS}\rangle \,\|\, \mathsf{enc}(\vec{C}'))$. Since $\vec{C} \neq \vec{C}'$, Lemma 1 gives $\mathsf{enc}(\vec{C}) \neq \mathsf{enc}(\vec{C}')$, so the two hash inputs are distinct: a collision of H. Case (ii): $h_2 = h_2'$ means $H(h_1 \,\|\, \langle\mathsf{RS}\rangle \,\|\, e) =$

$H(h_1' \,\|\, \langle \mathsf{RS} \rangle \,\|\, e')$ where e, e' are the (escaped, $\langle \mathsf{RS} \rangle$-separated) encodings of id_B, id_B'. If $(h_1, id_B) \neq (h_1', id_B')$ then the fixed-length h_1 prefix and the injective encoding of the identity fields make the two inputs distinct: a collision. In either case the reduction outputs the colliding pair. □

6.2 Co-signature inseparability

We formalize that σ_B cannot be detached from the specific first signature it endorses and re-attached to a different one.

Definition 2 (Co-signature inseparability)**.** The challenger runs $(pk_B, sk_B) \leftarrow \mathsf{KGen}(1^\lambda)$ inside an ideal secure element, gives pk_B to $\mathcal{A}$, and keeps a set Q. On a query $(id, C, id_B, pk_A, \sigma_A)$ with $\mathsf{Vrfy}(pk_A, m_A, \sigma_A) = 1$, the oracle $\mathcal{O}_B$ computes h_2, forms m_B, returns $\sigma_B \leftarrow \mathsf{Sign}(sk_B, m_B)$, and records $(id, h_2, pk_A, \sigma_A) \in Q$. $\mathcal{A}$ may generate its own A-side keys and signatures. It outputs $R^* = (id^*, C^*, id_B^*, pk_A^*, \sigma_A^*, pk_B, \sigma_B^*)$; let $h_2^* = h_2(h_1(C^*), id_B^*)$. $\mathcal{A}$ wins if $\mathsf{Vrfy}^*(R^*) = 1$ and $(id^*, h_2^*, pk_A^*, \sigma_A^*) \notin Q$.

Theorem 2 (Inseparability)**.** *If* Sig *is EUF-CMA-secure then for every PPT* $\mathcal{A}$, $\mathrm{Adv}^{\mathrm{insep}}_{\mathcal{A}} \leq \mathrm{Adv}^{\text{euf-cma}}_{\mathcal{B},\mathsf{Sig}}$.

Proof. $\mathcal{B}$ receives a challenge key pk^* with signing oracle $\mathcal{O}_{\mathrm{Sign}}$ and sets $pk_B := pk^*$ (a perfect simulation, since sk_B is used only through the ideal secure element). It answers each $\mathcal{O}_B$ query by forming m_B and calling $\sigma_B \leftarrow \mathcal{O}_{\mathrm{Sign}}(m_B)$, logging m_B and recording Q. On a winning R^*, $\mathcal{B}$ outputs (m_B^*, σ_B^*) with $m_B^* =$ "B:" $\|\, id^* \,\|$ ":" $\|\, h_2^* \,\|$ ":" $\|\, pk_A^* \,\|$ ":" $\|\, \sigma_A^*$. Validity: $\mathsf{Vrfy}(pk^*, m_B^*, \sigma_B^*) = 1$ by assumption. It remains to show m_B^* was never queried. The template "B:" id ":" h_2 ":" pk_A ":" σ_A is uniquely decodable: id is a UUID containing no ":", h_2 is a fixed-length hex digest, pk_A occupies two known-length base64 tokens, and σ_A is the final base64 token; base64 and hex alphabets exclude ":". Hence m_B^* determines $(id^*, h_2^*, pk_A^*, \sigma_A^*)$ uniquely; if it equalled a queried m_B then the corresponding recorded tuple would equal $(id^*, h_2^*, pk_A^*, \sigma_A^*) \in Q$, contradicting the winning condition. So m_B^* is fresh and (m_B^*, σ_B^*) is a valid forgery. □

Remark 3 (Why the chaining fields are load-bearing)**.** The extraction used only that m_B commits injectively to $(id, h_2, pk_A, \sigma_A)$. Dropping pk_A or σ_A from m_B would let $\mathcal{A}$ present the same σ_B under a different σ_A^* without leaving Q, and the contradiction step fails. The chain pk_A, σ_A inside m_B is exactly what buys inseparability.

6.3 Attestation-bound key provenance

Recall $\mathsf{Att} = (\mathsf{Gen}, \mathsf{AttVer})$ from §3 and the challenge binding $cdh = H(c \,\|\, H(pk_A))$ with c fresh, single-use, and session-scoped (§5.4).

Definition 3 (Provenance soundness)**.** A PPT $\mathcal{A}_{\text{key}}$ outputs (pk^*, c^*, τ^*). Let $\mathsf{Genuine}(pk^*)$ be the event that pk^* was generated inside a genuine secure element under Att's soundness model, and $\mathcal{C}$ the set of issued (fresh, single-use) challenges. $\mathcal{A}_{\text{key}}$ wins if $\mathsf{AttVer}(pk^*, c^*, \tau^*) = 1$ and either (a) $\neg\mathsf{Genuine}(pk^*)$, or (b) $c^* \notin \mathcal{C}$ or the accepted $cdh^* \neq H(c^* \,\|\, H(pk^*))$.

Proposition 1 (Provenance / no replay)**.** *If* Att *is sound and* H *is collision-resistant, then* $\mathrm{Adv}^{\text{prov}}_{\mathcal{A}} \leq \mathrm{Adv}^{\text{snd}}_{\mathsf{Att}} + \mathrm{Adv}^{\text{cr}}_{H}$.

Proof. Suppose $\mathcal{A}_{\text{key}}$ wins, so $\mathsf{AttVer}(pk^*, c^*, \tau^*) = 1$. *Case (a):* $\neg\mathsf{Genuine}(pk^*)$ is exactly a break of Att-soundness; the reduction outputs (pk^*, c^*, τ^*), bounded by $\mathrm{Adv}^{\text{snd}}_{\mathsf{Att}}$. *Case (b):* by Att-soundness the bound client-data hash is the value fixed at generation. If $cdh^* = H(c \,\|\, H(pk))$ for some *issued* $(c, pk) \neq (c^*, pk^*)$, then $H(c \,\|\, H(pk)) = H(c^* \,\|\, H(pk^*))$ on distinct preimages—an H-collision, bounded by $\mathrm{Adv}^{\text{cr}}_{H}$. Otherwise no issued pair explains cdh^*, so AttVer accepted a challenge never bound to any issued attestation—again a soundness break. Freshness and single use of c rule out cross-session reuse. Summing gives the bound. □

Remark 4 (Trust root)**.** Proposition 1 reduces provenance to attestation soundness and H; it does *not* eliminate a trusted party. It replaces CA trust with the hardware-manufacturer attestation root, and a compromise of that root breaks provenance—a deliberate trade (§8). Attestation binds the key to genuine device hardware, not to a natural person.

6.4 Post-signing tamper evidence

Write $\vec{C}$ for the vector of A-content fields, $C = \mathsf{enc}(\vec{C})$, $h_1 = H(\mathsf{ver} \,\|\, \langle \mathsf{RS} \rangle \,\|\, C)$, and $m_A =$ "A:" $\|\, id \,\|$ ":" $\|\, h_1$.

Definition 4 (Content tamper resistance)**.** The challenger generates (pk_A, sk_A) in an ideal secure element. On a content vector $\vec{C}$ (with id id) the oracle returns $\sigma_A \leftarrow \mathsf{Sign}(sk_A, m_A)$ and records $\vec{C} \in Q_A$. $\mathcal{A}$ outputs $(\vec{C}^*, \sigma^*_A)$ and wins if $\mathsf{Vrfy}(pk_A, m^*_A, \sigma^*_A) = 1$ and $\vec{C}^* \notin Q_A$.

Theorem 3 (Tamper evidence)**.** *If* Sig *is EUF-CMA-secure and* H *is collision-resistant, then* $\mathrm{Adv}^{\text{tamper}}_{\mathcal{A}} \leq \mathrm{Adv}^{\text{euf-cma}}_{\mathcal{B}_1,\mathsf{Sig}} + \mathrm{Adv}^{\text{cr}}_{\mathcal{B}_2,H}$.

Proof. Let $\mathcal{A}$ win with $(\vec{C}^*, \sigma_A^*)$, $\vec{C}^* \notin Q_A$. Split on m_A^*. *Event E_1: m_A^* was never returned by the oracle.* $\mathcal{B}_1$ sets $pk_A := pk^*$, answers signing queries via its oracle, and outputs (m_A^*, σ_A^*), a fresh valid EUF-CMA forgery; thus $\Pr[\mathcal{A} \text{ wins} \wedge E_1] \leq \mathrm{Adv}^{\text{euf-cma}}_{\mathcal{B}_1}$. *Event E_2: m_A^* equals some queried m_A for $\vec{C} \in Q_A$.* The template "A:" id ":" h_1 is uniquely decodable, so $m_A^* = m_A$ forces $h_1^* = h_1$, i.e. $H(\mathsf{ver} \,\|\, \langle \mathsf{RS} \rangle \,\|\, C^*) = H(\mathsf{ver} \,\|\, \langle \mathsf{RS} \rangle \,\|\, C)$. Since $\vec{C}^* \neq \vec{C}$ (one is in Q_A, the other is not), Lemma 1 gives $C^* \neq C$, so the inputs differ: a collision. $\mathcal{B}_2$ (holding sk_A, simulating exactly) outputs the colliding pair; thus $\Pr[\mathcal{A} \text{ wins} \wedge E_2] \leq \mathrm{Adv}^{\text{cr}}_{\mathcal{B}_2}$. E_1, E_2 are exhaustive and disjoint. □

Remark 5 (Coverage)**.** $\vec{C}$ comprises the document terms and A-identity fields of §5.1; every one is inside h_1 and hence under σ_A. The 39-field *finalHash* record (Appendix B) additionally binds both signatures, both public keys, the biometric-backing flags, the device-token hashes, and h_1, so the argument extends to the B-block once σ_B is present (via Theorem 2 for the chain and this theorem for the content). Fields deliberately excluded—the non-deterministic rendering bytes of any display file (§5.6)—are, by design, not covered.

7 Implementation

- *Primitives.* ECDSA P-256 (`prime256v1`), SHA-256; public keys serialized as B64(X):B64(Y); signatures DER + Base64. Secure element: Apple Secure Enclave / Android StrongBox-Keystore. The signing path admits *no* software key: a device without a secure element yields an error, not a silent software fallback, and every signing site enforces a hardware gate (*backingType* $\neq$ HW $\Rightarrow$ abort). A strict DER decoder rejects non-canonical encodings and out-of-range r, s; low-S signatures are normalized (Secure Enclave emits high-S), and the public point is checked to lie on P-256.

- *Attestation.* Apple App Attest (iOS) and Google Play Integrity (Android); production environment only ($allow\_dev$ = false), cf. Remark 1. App Attest tokens (iOS) admit offline chain-to-Apple-root verification (mode i) and Play Integrity (Android) is manufacturer-service-verified (mode ii). The reference deployment verifies both server-side at onboarding, via a challenge c with client-data hash $cdh = H(c \,\|\, H(pk))$, and records a server-signed (ECDSA P-256) attestation that any party can re-check with the server's public key; an App Attest token additionally remains offline-verifiable by a third party that pins Apple's root.

- *Transport.* Animated optical code with the LT-style fountain code of §5.5; NFC, BLE, ultrasound, and file-embedded channels are supported alternatives.

- *Verifier.* A standalone verifier recomputes h_1, h_2, and *finalHash* and checks σ_A, σ_B with no network access. A public instance that performs the recomputation locally in the browser is available at `https://pakto.pro/en/verify`. The protocol core is implemented twice (Dart on the client, Python in the independent verifier), and byte-for-byte agreement of all digests across the two implementations was confirmed, so verification does not depend on a single implementation.

8 Limitations and Trade-offs

- *No CA is a trade, not a free win.* Key trust rests entirely on the attestation manufacturer root and its soundness; a break there breaks provenance. We do not remove a trusted party—we relocate it from a CA to the hardware manufacturer.

- *Provenance is per-device, not per-natural-person.* Attestation binds a key to genuine hardware, not to an individual. Binding a key to a real-world identity is an application-layer step outside the offline core and is not analyzed here; it must not be read as a cryptographic guarantee of the core.

- *Attestation strength is platform-dependent.* StrongBox, TEE, and software backings differ in assurance, and Android is fragmented; production vs. development attestation matters (Remark 1).

- *Injectivity is domain-restricted.* Lemma 1 holds on the admissible domain; a fully unconditional statement requires TLV encoding (Remark 2).

- *Fountain parameters are an engineering choice.* No optimality is claimed for the custom degree rule (§5.5).

- *Out of scope.* Duress/coercion of a present user; malware on a rooted device able to drive the biometric prompt; the `duressFlag` is not integrity-protected.

- *No external audit yet.* The implementation has not undergone an independent third-party security audit; we regard such an audit as a prerequisite to any high-assurance deployment.

9 Conclusion

We showed that mutual co-signing survives the removal of all standard infrastructure—no network, no server on the trust path, and no CA—provided one handles the circular signing dependency with a two-stage hash anchor, makes the second signature inseparable from a specific first signature by chaining, and grounds key trust in a hardware-attestation token transported over the channel itself. The four properties reduce to collision resistance of H and EUF-CMA security of the signature scheme, and an independent verifier reproduces every anchor and both signatures offline. The construction does not replace PKI; it maps out how much verifiability remains when the infrastructure is taken away.

A Worked Example (redacted)

All values below use *synthetic, non-personal* inputs and can be recomputed with the reference verifier. The content anchor uses the length-prefixed serialization (§5.1): each field is emitted as ⟨byte-length⟩:⟨value⟩\n, after the tag `PAKTO-v82\n` and a domain label.

```
preimage (head):  PAKTO-v82\n 11:contentHash\n
36:11111111-2222-3333-4444-555555555555\n 7:General\n ... 5:alice\n
1:1\n
contentHash     = 8271868f458a447f70bd54eda563b7347c0ea330fd1a77e24535fd719332384e
contentHashFull = 7ca362ee0b7663f64480b746d4835044f815c2be7e2abed681ed0de84cb4120b
m_A = A:11111111-2222-3333-4444-555555555555:8271868f...
m_B = B:11111111-2222-3333-4444-555555555555:7ca362ee...:<pkA>:<sigA>
```

Here $\sigma_A = \mathsf{Sign}(sk_A, m_A)$ and $\sigma_B = \mathsf{Sign}(sk_B, m_B)$ are produced in the respective secure elements. Only device-token *hashes* enter the composite record (Appendix B); the full attestation tokens travel in the transfer structure (§5.5).

B Composite `finalHash` field order (escaped record)

The composite hash is $\mathit{finalHash} = H\big(\mathrm{utf8}(F_0 \langle\mathsf{RS}\rangle F_1 \langle\mathsf{RS}\rangle \cdots \langle\mathsf{RS}\rangle F_{38} \langle\mathsf{RS}\rangle)\big)$, i.e. ⟨RS⟩ (`0x1E`) is written after every field including the last. Each F_i is passed through the escaping map f of §5.1 unless noted. Compound profile fields use ⟨US⟩ (`0x1F`) between nickname and avatar index. The B-block (indices 28–35) is included iff $\sigma_B \neq \varepsilon$; indices 36–38 always follow.

This composite record uses the control-byte-escaped serialization and carries its own scheme tag in field 0; the *signed* content anchors h_1, h_2 use the length-prefixed (TLV) serialization of Appendix A. The example column is illustrative and abbreviated: it fixes the *field order*, not a numeric artifact reproduced elsewhere in this paper.

#	Field	Fallback / rule	Example (abbrev.)
0	schema_prefix	literal scheme tag	`<scheme-tag>`
1	contractId	—	`11111111-...`
2	contractType	—	
3	contractCategory	—	
4	title	—	*(value)*
5	details	—	*(value)*
6	amount	(empty → 0 bytes)	(empty)
7	currency	—	`EUR`
8	contractLanguage	—	`en`
9	jurisdiction	(empty → `nojuris`)	
10	createdAt	ISO-8601 UTC	`2026-01-01T00:00:00Z`
11	tsaTime	ISO-8601 UTC	`2026-01-01T00:00:00Z`
12	timeSource	—	`LOCAL:WARNING`
13	publicKeyA	`B64(X):B64(Y)`	*(base64:base64)*
14	signatureA	DER+B64	*(DER+base64)*
15	biometricBackedA	(empty → `SW`)	`HW`
16	deviceTokenHashA	H(ctx_device_A: $\|\| \tau_A$); (empty → `nodeviceA`)	*(hash)*
17	geoHash	(empty → `nogeo`)	*(redacted)*
18	geoCoordinates	(empty → `nocoords`)	(test)
19	tgPhoneHashA	(empty → `notg`)	`notg`
20	emailA	(empty → `noemail`)	`noemail`
21	identityRnokppHashA	(empty → `noidentity`)	*(redacted)*
22	identityQualifiedA	`qualified`/`notqualified`	`qualified`
23	tgPhoneHashB	(empty → `notg`)	`notg`
24	evidenceHashesJson	(empty → `nophoto`)	`["bc118af04b..."]`
25	photoHashA	(empty → `nophoto`)	`nophoto`
26	photoHashB	(empty → `nophoto`)	`nophoto`
27	tsaHashA	(empty → `notsa`)	`notsa`
B-block, indices 28–35, present iff $\sigma_B \neq \varepsilon$:			
28	publicKeyB	`B64(X):B64(Y)`	*(base64:base64)*
29	signatureB	DER+B64	*(DER+base64)*
30	biometricBackedB	(empty → `SW`)	`HW`
31	deviceTokenHashB	H(ctx_device_B: $\|\| \tau_B$); (empty → `nodeviceB`)	*(hash)*
32	tsaTokenHashB	(empty → `notsaB`)	`notsaB`
33	emailB	(empty → `noemail`)	`noemail`
34	identityRnokppHashB	(empty → `noidentity`)	`noidentity`

#	Field	Fallback / rule	Example (abbrev.)
35	identityQualifiedB	`qualified`/`notqualified`	`notqualified`
always present:			
36	contentHash ($= h_1$)	(empty → `nocontent`)	`8271868f...`
37	profileA	nick ⟨US⟩ avatar; (empty → `noprofile`)	`alice <US> 1`
38	profileB	nick ⟨US⟩ avatar; (empty → `noprofile`)	`<US> 0`

For a complete document the reference verifier recomputes *finalHash* from these fields and compares it to the stored value.

References

[1] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 2006*, pp. 390–399. ACM, 2006. doi:10.1145/1180405.1180453

[2] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography* 87(9):2139–2164, 2019. doi:10.1007/s10623-019-00608-x (IACR ePrint 2018/068).

[3] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *ASIACRYPT 2003*, LNCS 2894, pp. 452–473. Springer, 2003. doi:10.1007/978-3-540-40061-5_29 (IACR ePrint 2003/126).

[4] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM CCS 2004*, pp. 132–145. ACM, 2004. doi:10.1145/1030083.1030103

[5] M. Luby. LT codes. In *43rd IEEE FOCS 2002*, pp. 271–280. IEEE, 2002. doi:10.1109/SFCS.2002.1181950

[6] A. Shokrollahi. Raptor codes. *IEEE Trans. Information Theory* 52(6):2551–2567, 2006. doi:10.1109/TIT.2006.874390

[7] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder. RaptorQ forward error correction scheme for object delivery. IETF RFC 6330, 2011. doi:10.17487/RFC6330

[8] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM 1998*, pp. 56–67. ACM, 1998. doi:10.1145/285237.285258

[9] Apple Inc. Establishing your app's integrity — DeviceCheck / App Attest. Apple Developer Documentation. https://developer.apple.com/documentation/devicecheck.

[10] Google. Play Integrity API. Android Developers Documentation. https://developer.android.com/google/play/integrity.

[11] Google. Verifying hardware-backed key pairs with key attestation. Android Developers Documentation. https://developer.android.com/privacy-and-security/security-key-attestation.

[12] I. Danylenko (divan). TXQR: transfer data via animated QR codes (fountain-coded). Open-source software, 2018. https://github.com/divan/txqr.

[13] *libcimbar*: Color Icon Matrix Barcodes with fountain coding. Open-source software. https://github.com/sz3/libcimbar.

[14] Coalition for Content Provenance and Authenticity. C2PA Technical Specification (Content Credentials), v2.x, 2024–2026. https://spec.c2pa.org/.

[15] National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS PUB 186-5, 2023. doi:10.6028/NIST.FIPS.186-5

[16] Standards for Efficient Cryptography Group (SECG). SEC 1: Elliptic Curve Cryptography, Version 2.0, 2009.