Warden: Event-Gated Threshold Conditional-Decryption for Guaranteed-but-Deferred Encrypted Delivery

Sandeep Nandal*

Draft v1 · June 2026

Status disclosure (read first). Warden is **deployed on a public testnet (Base Sepolia) and is pre-audit.** This paper presents a *design, threat model, and early deployment experience*, not an audited production system. The current testnet federation — five nodes, but all operated by a single party (BytesBrains, on one infrastructure provider) — therefore has, by construction, **no timing-security** (see §6, W4); it exercises the protocol, integration, and liveness, not the trust model. Independent operators are actively being recruited (§9). We state this plainly because, for the audience this work addresses, calibrated honesty about limitations is the contribution's credibility, not a footnote to it.

Abstract

Time-lock encryption lets a sender make a ciphertext readable only after a predetermined time, and recent constructions such as *tlock* realize it efficiently over a threshold-BLS randomness beacon (drand). But an important class of applications needs a *conditional* rather than a *temporal* trigger, over a *decades-long* horizon, with the data author *absent* at release time: a dead-man's-switch that delivers a sealed message to designated recipients only if its owner stops checking in — for digital estates, personal-safety triggers, and "your work survives you" press-freedom scenarios.

We present **Warden**, an event-gated threshold conditional-decryption network. A federation of independent operators holds shares of an identity-based master key (Boneh–Franklin IBE over threshold BLS, established by distributed key generation) and releases a per-message partial decryption key only when an on-chain condition — a resettable smart-contract flag — is observed true at finalized block depth. Payloads are *double-wrapped*: an outer condition-gate the federation can open, and an inner recipient-key layer it cannot, so operators learn release *timing* but never *content*.

We formalize the resettable-condition trigger and the owner-absent, decades-horizon threat model; describe master-key permanence across operator churn via resharing; give an honest security analysis (threshold timing guarantees, finality/reorg leakage, collusion and traitor-tracing); and report a Rust implementation deployed as a five-node testnet federation that runs end-to-end on Base Sepolia, integrated with the live Maktub protocol and a mobile client — a single-trust-domain deployment that exercises the protocol and its integration, with independent-operator security

*BytesBrains Pte Ltd, sandeep@nandal.in. Solo authorship reflects the work as done; a co-author position is open to an academic collaborator who contributes to the formal peer-reviewed version.

as the explicit next step. Warden reuses an audited cryptographic substrate (drand's threshold-BLS/IBE/DKG stack) and contributes the *system, threat model, and conditional-trigger design* — not new cryptography. It is intended to run as token-free public-good infrastructure in the drand / League-of-Entropy tradition. The precise contribution claims and explicit non-claims are enumerated in §1.3 and §1.4.

---

1. Introduction

1.1 The gap

Time-lock encryption (TLE) lets a sender seal a message that becomes readable only after a chosen point in the future, without trusting any single party to keep time. The deployed state of the art is **tlock** [Gailly–Melissaris–Romailler 2023], which observes that drand's threshold-BLS randomness beacon is *already* an identity-based-encryption key-generator: the beacon's signature on a future round number *is* the IBE decryption key for that round. A sender encrypts "to round R"; once the League-of-Entropy federation publishes round R's signature, anyone can decrypt — offline, with no interaction at decryption time.

This is powerful but fixes the trigger to a **clock**. A large and practically important class of applications needs the release to be gated not on *time* but on a *condition*, and over horizons and absence assumptions that the timelock literature does not target:

- **Conditional, not temporal.** "Release if and only if event E has happened," where E is some externally observable fact — not "release at time T."
- **Resettable.** The condition is not monotone-by-fiat from the sender's side: the owner actively keeps it *false* by checking in, and it flips *true* only when they stop. This is the dead-man's-switch shape, and it is the inverse of a timelock (where the countdown is unstoppable once started).
- **Decades-horizon and owner-absent.** The ciphertext may sit dormant for years or decades, and — definitionally — the author is *not present* at release time to participate, re-key, or intervene. Most timelock and threshold work assumes short horizons and a live participant.

1.2 Motivating application: Maktub / Veil

Warden is motivated by **Maktub**, a decentralized conditional-execution protocol on the Base L2 whose single primitive is: *if the owner does not check in within a chosen interval, deliver an encrypted payload to designated recipients.* Maktub's on-chain contract (`MaktubCore`, "Beat") is immutable and holds no plaintext; it merely records, per *beat*, whether the owner has gone silent (an `executed` status the contract latches to `true` once a timer lapses and an executor calls it).

The capability this enables for end users — *sealed until you go silent; revocable while you are present; end-to-end encrypted to the recipient* — is called **Veil**. Veil's three named use cases are equal: digital-estate inheritance (seed phrases, passwords, documents to heirs), personal-safety triggers (hikers, field workers, solo travelers), and press-freedom **"your work survives you"** (material a journalist prepared is delivered to recipients they designated if they become unable to deliver it themselves — framed as *work survival*, e.g. accident or illness, **never** as adversarial-state anonymity).

Maktub's contract enforces the *timer and the trigger*. It does **not** by itself enforce the *time-confidentiality* half of Veil: nothing in a public smart contract can hold a payload unreadable-until-trigger, because anything the contract can read, the world can read. That half is exactly what Warden supplies. `MaktubCore` requires **no modification** to support Warden — Warden only *reads* the beat's status.

1.3 Contribution

We claim the following, precisely:

1. **A conditional, resettable trigger.** Release is gated on *mutable on-chain contract state* (a flag the owner resets by checking in), evaluated at finalized block depth — not a fixed time or round. This problem shape is under-explored relative to tlock and time-lock puzzles, and the *combination* it sits in is not served by deployed conditional-decryption systems (§2.6).
2. **An owner-absent, decades-horizon threat model**, with master-key permanence maintained across operator churn by *resharing* (the master public key — and hence the decryptability of all dormant ciphertexts — survives a completely changed operator set).
3. **A double-wrap construction** that cleanly separates *timing control* (the threshold federation) from *content confidentiality* (a recipient-held key, pure mathematics): operators can open the outer condition-gate but **never** see plaintext.
4. **An honest, extensive limitation analysis** (W1–W11): early-release via collusion, liveness/delay, finality/reorg leakage, operator/infra capture, master-key permanence as a double-edged sword, the non-post-quantum horizon, condition-evaluation trust tiers, the open-request metadata channel, cross-sender identity collision, share-key registration, and request-channel denial-of-service.
5. **An open-source (MIT) implementation and live testnet deployment:** a single Rust core driving a node daemon, a WASM web client, and an FFI mobile client, running end-to-end as a five-node federation on Base Sepolia and integrated with the Maktub protocol (a single-trust-domain testnet; independent operators are being recruited — §9).

The contribution is a **systems / applied** one. Gen-1 Warden reuses drand's audited threshold-BLS + Boneh–Franklin IBE + DKG + resharing essentially verbatim and replaces only the *trigger* (clock → on-chain condition) and the engineering around it. We are explicit about this because aiming such work at cryptographic-theory venues would be both wrong and counterproductive: there is no new primitive here, and the genuinely novel surface (a resettable-condition release gate with a finality policy) is a systems contribution.

1.4 Explicit non-claims

The same honesty discipline that governs all Maktub copy governs this paper. We do **not** claim:

- That timing/revocation is **trustless**, **absolute**, or **mathematically guaranteed**. It rests on a **threshold honesty assumption** (no early release unless `t` operators collude; no denial unless fewer than `t` stay honest and live — §6.1) *and* on operators correctly observing the on-chain condition. The trustless ideal is witness encryption (§2.4), which is not deployable today; Warden is the best deployable approximation.
- That Warden provides **metadata privacy or anonymity.** The existence of a sealed message, its release timing, its public condition, and (for Veil) its `beatId`, recipient set, and sender-recipient relationship are **public on-chain by design.** Warden hides *content until the trigger* — never *that something exists.*

- That gen-1 is **post-quantum.** It is pairing-based (BLS12-381); the realistic horizon for the threshold layer is on the order of years, and no deployable post-quantum threshold-IBE replacement exists. The envelope is algorithm-versioned so the cryptography can migrate (§10).

1.5 Roadmap of the paper

§2 surveys the cryptographic background and positions Warden against timelock, time-lock puzzles, witness encryption, and signature-based witness encryption. §3 formalizes the resettable-condition trigger and the threat model. §4 gives the system design (double-wrap envelope, condition model, network planes, finality policy). §5 gives the cryptographic construction (DKG, encryption, threshold release, resharing). §6 is the honest security analysis (W1–W11) — the credibility core. §7 describes the implementation; §8 the testnet deployment experience. §9 covers federation governance and sustainability; §10 limitations and future work; §11 ethics and responsible disclosure; §12 concludes.

---

2. Background and Related Work

2.1 Threshold cryptography and distributed key generation

A *(t, n)*-threshold scheme splits a secret among `n` parties such that any `t` can jointly use it but any `t-1` learn nothing. Shamir secret sharing [Shamir 1979] is the foundation; Feldman [1987] and Pedersen [1991] add *verifiability* (VSS), letting parties detect a cheating dealer. **Distributed key generation** (DKG) [Gennaro et al. 1999] removes the dealer entirely: `n` parties jointly produce a public key and individual private-key shares such that the full private key is *never assembled by anyone.* drand uses Pedersen's DKG built from parallel Feldman VSS, and contributes a hardened orchestration layer around it (a durable state machine, explicit joiner/remainer/leaver lifecycle).

Threshold BLS [Boneh–Lynn–Shacham 2001; Boldyreva 2003] is the release primitive Warden inherits. A BLS signature on message `m` under secret key `sk` is `sk·H(m)` for a hash-to-curve `H`. Splitting `sk` into Shamir shares `sk_i`, each party emits a partial `sig_i = sk_i·H(m)`; any `t` partials Lagrange-combine *in the exponent* into the full signature `sk·H(m)`. Signatures are deterministic (hence unbiasable), which is what makes the drand beacon usable as a key generator.

2.2 Identity-based encryption and timelock (tlock)

Boneh–Franklin IBE [BF 2001] lets a sender encrypt to an arbitrary *identity string* using only a master public key; the matching decryption key `msk·H1(id)` is generated later by the key authority. The Boneh–Franklin ciphertext `(U, V, W)` with the Fujisaki–Okamoto transform [FO 1999] achieves CCA security in the random-oracle model.

tlock [Gailly–Melissaris–Romailler 2023] makes the crucial observation: a threshold-BLS beacon *is* an IBE private-key generator that does not know it is one. Set the IBE identity to a future round number; the beacon's BLS signature over that round (`msk·H1(round)`) *is exactly* the IBE decryption key for that identity. No extra protocol is needed — when the beacon naturally publishes the round signature, the timelock opens. tlock requires the beacon's **unchained** mode (the message signed for a future round is predictable: `H(round)` alone, not chained on the previous signature),

and it packages the result as a hybrid age-encryption envelope with a `-> tlock {round} {chainHash}` stanza.

The single most important fact Warden inherits from this line: nothing in the IBE math requires the identity to be a round number. *The identity is whatever string the network agrees to sign.* Warden swaps `H(round)` for `H(condition)` and the time-elapsed trigger for an on-chain-boolean trigger; the entire pairing/IBE/threshold/envelope layer is reused unchanged (§5, [drand-analysis §5, §8]).

drand itself is run by the **League of Entropy**, a federation of more than a dozen independent organizations (founded 2019 by Protocol Labs, Cloudflare, EPFL, Kudelski, and the University of Chile, among others; membership has since rotated) — the operational template for Warden's public-good federation (§9).

2.3 Time-lock puzzles and VDFs

A different lineage realizes delay through *sequential computation*: time-lock puzzles [Rivest–Shamir–Wagner 1996] and verifiable delay functions [Wesolowski 2019; Pietrzak 2019] force a solver to perform an inherently sequential amount of work before a value is recoverable. These are *self-contained* (no committee) but pay for it: the "time" is wall-clock compute that must actually be expended, the horizon is hard to calibrate across hardware generations, and — decisively for our setting — they are **temporal, not conditional**, and assume a present solver. They do not fit a decades-dormant, owner-absent, condition-gated release. Warden is a committee-release design, not a sequential-work one; we note the contrast rather than compete on it.

2.4 Witness encryption — the trustless ideal, and why not yet

Witness encryption [Garg–Gentry–Sahai–Waters 2013] encrypts to an NP statement such that anyone holding a *witness* can decrypt and no one else can. Applied to our setting, the on-chain proof that `executed(beatId)==true` would *itself* be the decryption key — **no federation at all**, hence genuinely trustless timing. This is the endgame. It is not deployable: general WE rests on indistinguishability obfuscation or similarly idealized assumptions, and even recent **WE-from-SNARKs/KZG** results [e.g. ePrint 2025/1364] that target concrete blockchain state remain theoretical (they require Base's `executed` flag expressed as a SNARK/KZG statement under idealized assumptions). Warden treats WE as the gen-3 successor (§10) and keeps its envelope `alg`-versioned so a WE outer layer can drop in with no change to recipients or to Maktub.

Signature-based witness encryption (SWE) [McFly, ePrint 2022/433; compact SWE 2024/1477; eWEB 2025/1064] is a nearer approximation: encrypt against a committee's verification keys plus a *tag*, such that the committee's threshold *signature on the tag* is the decryption witness — users never interact with the committee, and *no per-item decryption key and no master secret are ever held.* This "no master secret" property is, on paper, the longevity-optimal design for a dead-man's-switch, and it runs on the same BLS12-381 substrate. Warden targets SWE for gen-2 (§10) but does not build on it today: the only implementations are dormant research prototypes, ciphertext is linear in committee size (heavy on mobile), and the compact variant requires iO.

2.5 Removing DKG, and detecting collusion

Two recent advances bear directly on Warden's two worst liabilities (DKG complexity; undetectable collusion). **Silent threshold encryption** [ePrint 2024/263] derives the joint public key as a deterministic function of operators' *locally published* keys — no interactive DKG ceremony. **Silent threshold traitor-tracing** (ST3) [ePrint 2025/342; CCS 2025] adds *public tracing of a colluding committee with no trusted authority*. The latter is significant for us: gen-1 Warden cannot detect silent early-reveal collusion (weakness W1, §6), and traitor-tracing would make such collusion *publicly attributable*, converting an undetectable attack into a detectable, deterrable one. We treat traitor-tracing as a **layer** to adopt when it has audited code, independent of the base scheme (§8 of the architecture decision).

2.6 Deployed and adjacent conditional-decryption systems

Warden is not the first system to release decryption capability on a condition; positioning it honestly against deployed and academic neighbours sharpens the actual delta.

- **TACo (Threshold Access Control), Threshold Network / NuCypher** — a productized threshold network that releases decryption rights when access *conditions*, including on-chain state, are satisfied. This is the closest *productized* competitor (now under World Ethical Data Foundation stewardship, with a stable relaunch slated for 2026); its conditions are point-in-time access checks rather than latching triggers. Warden's distinguishing requirements are the *monotone-latching* condition discipline (§3.1), *permanence across disjoint operator sets via resharing* over a decades horizon (§5.4), and an explicit *finality policy* for the chain read (§4.4).
- **Shutter Network** — threshold encryption with "keypers" releasing decryption keys on a trigger (originally anti-front-running for mempools). It shares the threshold-release shape but targets short-horizon, per-epoch keys, not dormant, owner-absent, resettable-condition ciphertexts.
- **Calypso** [Kokoris-Kogias et al. 2021] — on-chain secret management that releases threshold-encrypted secrets when an on-chain access policy is met. Direct academic prior art for "decrypt when a blockchain condition holds"; Warden differs in the owner-absent dead-man's-switch trigger, the decades-permanence requirement, and the multi-platform client.
- **Encryption to the Future / YOSO** [Campanelli et al. 2022; Gentry et al.] — sending ciphertexts to *future* committees selected over time; it motivates the owner-absent, operator-churn setting that resharing addresses here.

The point is not that conditional decryption is new — it is not — but that the specific *combination* in §2.7 is not served by these systems. *(The precise competitive deltas, especially against TACo's evolving condition language, should be re-verified against each system's current capabilities before any future peer-reviewed version.)*

2.7 How Warden differs

Against all of the above, Warden's distinguishing combination is: **(a)** a *resettable on-chain condition* as the release trigger (vs. fixed round/time in tlock and time-lock puzzles); **(b)** an explicit *decades-horizon, owner-absent* threat model with permanence-through-resharing; and **(c)** a deployed multi-platform system (node + WASM + mobile FFI) with a finality policy for the chain read — a security-critical surface that has no analog in drand because drand has a clock, not a chain to observe. None of these is new *cryptography*; together they are a new *system*.

3. Problem Statement and Threat Model

3.1 The resettable-condition trigger

Let `C` be a **condition**: a canonical, public, deterministic predicate over finalized on-chain state. The release predicate Warden enforces is:

> *Release the decryption key for ciphertext* `e` *if and only if* `C(e)` *is observed* true *at finalized block depth.*

Two properties make `C` suitable, both load-bearing (§4.2):

- **Determinism.** Every honest operator, reading finalized chain state at the agreed confirmation depth, must compute the *same* yes/no. A threshold scheme cannot release if honest parties disagree on whether the condition holds.
- **Monotonicity (latching).** Once `C` becomes true it stays true. Because a released key cannot be un-released, the trigger must not be allowed to flip back. The canonical instance — `executed(beatId)==true` for a Maktub Beat — is naturally latching: the contract never un-executes. Non-monotone raw predicates (e.g. `price > X`) are rejected unless *anchored* into a permanent form ("became true at/by block B", or an event-was-emitted form).

The *resettable* character lives entirely on the **sender's** side and entirely *before* the trigger: the owner keeps `C` false by checking in, and revocation is the owner constructing `C` so that it can be made **permanently unsatisfiable** (Maktub's `deactivate(beatId)` ensures `executed` can never become true → Warden never releases → the ciphertext is permanent gibberish). From Warden's perspective the *observed* predicate is still monotone; resettability is a property of how the application drives the underlying contract state.

3.2 Actors

- **Author / owner** — encrypts the payload offline and drives the condition (checks in, or revokes). Absent at release time by assumption.
- **Recipient(s)** — hold the private key that the inner layer is encrypted to; the only parties who can ever read content.
- **Operator federation** — `n` independent organizations, each holding one share of the master key; each runs a node that observes the chain and serves partial decryptions.
- **Chain** — the public ledger (Base) whose finalized state defines the condition.
- **Adversary** — see §3.4.

3.3 Security goals

- **G1 — Time-confidentiality.** Before `C` holds, the payload is unreadable by *everyone, including the intended recipient.*
- **G2 — Delivery availability.** After `C` holds, the payload is readable by the intended recipient, and remains so indefinitely (release is idempotent and retryable forever). In the current client-requested mode (§5.3), release is *recipient-initiated* — a recipient must pull and combine — so this is availability-on-demand, not push delivery; the autonomous mode (§5.3) pre-produces partials node-side, but a recipient still fetches and decrypts.

- **G3 — Content confidentiality.** Only the intended recipient ever reads plaintext — *including* against a fully-colluding federation.

3.4 Adversary model and assumptions

We assume:

- **Threshold-bounded corruption** for timing. Confidentiality-before-trigger (G1) holds as long as fewer than `t` operators collude; availability-after-trigger (G2) holds as long as at least `t` operators are honest and live. With `t ≈ ⌈2n/3⌉` these bounds trade off (raising `t` strengthens G1 but weakens G2-liveness, and vice versa) — the standard threshold tension, stated precisely in §6.1; it is *not* an "honest majority". **G3 rests on neither** (it is pure mathematics, §6).
- **Correct finalized-state observation:** honest operators read true finalized chain state (not an eclipsed or spoofed view). This is part of the trust model and is *new relative to drand*, which has no chain to observe (§4.4, §6/W3, §6/W4).
- **Recipient key secrecy:** the recipient's private key is not compromised (otherwise G3 is moot — but only for that one message).
- **Independence of operators:** the security parameter is not raw node count but *genuine independence* (distinct organizations, jurisdictions, and infrastructure), so that no single subpoena, coercion, or failure reaches `t` (§6/W4, §9).

The two-axis trust split is the heart of the model and is stated in full in §6.1.

---

4. System Design

4.1 The double-wrap envelope

Warden's defining design choice is that the federation's power is *confined to timing* and can never extend to content. This is achieved by nesting two independent gates over a single random content key `K`:

```
K          = random symmetric content key
inner      = AEAD(payload, K)                        ← the content
K_wrapped  = ECIES(K, recipientPub)                  ← recipient-gated.  PURE MATH.
outer    = Warden-IBE(K_wrapped, H(condition))   ← condition-gated.  Federation can open this.
```

Warden's release only ever opens **outer**, which reveals `K_wrapped` — *an ECIES ciphertext*, not `K` and not the payload. Only the recipient's private key turns `K_wrapped` into `K`. Therefore **both** the condition (to open `outer`) **and** the recipient key (to open `K_wrapped`) are required, and **even a fully-colluding federation that releases every outer key learns only release timing, never content** (this is goal G3). Content confidentiality is independent of Warden entirely (given recipient-key secrecy, §3.4).

As implemented. Because the Boneh–Franklin IBE plaintext block is a fixed 32 bytes, the outer layer is realized as a hybrid: `outer.ibe = IBE(obk, H(condition))` gates a random 32-byte `obk`, and `outer.seal = AEAD_obk(K_wrapped)` seals the ECIES-wrapped key under `obk`. This is logically identical to "IBE-gate `K_wrapped`". Hardening from a security review: both AEAD layers bind `domain ‖ network ‖ H(condition)` as associated data (tamper-evident and domain-separated), and

the payload is **bucket-padded** to hide its length. AEAD is ChaCha20-Poly1305 [RFC 8439]; the recipient layer is ECIES over secp256k1 — chosen so recipients reuse their existing Ethereum wallet keypair rather than manage a second key — with HKDF info bound to `ephPub ‖ recipientPub` (SEC1 shared-info). (The IBE/threshold layer is on BLS12-381; the two curves serve the two different layers.) **No recipient metadata is stored in the envelope** — the recipient is implicit in who can ECIES-open `K_wrapped`. All domain tags and pad buckets are provisional and will be frozen with cross-language test vectors before mainnet.

The envelope is `alg`-versioned (`warden-v1`); future schemes (SWE, witness encryption) replace only the `outer/condition` handling, leaving `inner` and the integration unchanged, and old envelopes remain decryptable forever (§10). An optional age-stanza form (`-> warden <network-hash> <H(condition)>`) provides interoperability with the tlock/age ecosystem, mirroring tlock's stanza with *round* → *H(condition)* and *chainHash* → Warden network hash.

4.2 The condition model

The IBE identity a payload is encrypted to is the hash of a **canonical, versioned condition specification**, serialized with **RFC 8785 (JSON Canonicalization Scheme)** so the bytes are byte-identical across the Rust, TypeScript, and Dart implementations:

```
identity = H( "warden-cond-v1" ‖ jcs_rfc8785(condition) )
```

When asked for its partial on `identity`, a node is given the condition `C` (public metadata in the envelope), checks `H(C) == identity`, independently evaluates `C` against finalized chain state, and returns its partial *iff* `C` holds. This **binds the condition cryptographically to the ciphertext**: conditions cannot be swapped after encryption. The condition is *public* — it is metadata, not a secret (for Veil it reveals `beatId` and that release gates on the beat's status, both already public on-chain — no new leakage).

The schema (`cond-v1`) supports exactly one `type` per condition plus a mandatory `meta`:

- `contract` — `{ chain, address, fn, args, word?, test, meta }`
- `block` — `{ chain, field: "number"|"timestamp", cmp, value, meta }`
- `event` — `{ chain, address, sig, args, meta }`
- `all | any | not | threshold` — `{ of: [...sub-conditions], k (threshold only), meta }`

The Veil release condition for a real Maktub Beat illustrates a subtlety that the canonicalization must survive: `MaktubCore` exposes **no `executed(uint256)` getter** — execution status is field 7 (the 8th element) of the `getHeartbeat(uint256)` return tuple — so the condition selects it with `word: 7`:

```
{
  "type": "contract", "chain": 84532,
  "address": "0xb603C96D089F64Ac487EE0bdaE97D49848F86133",
  "fn": "getHeartbeat(uint256)", "args": ["123"], "word": 7,
  "test": { "cmp": "==", "value": true },
  "meta": { "finality": 32, "tier": 1 }
}
```

Two serialization rules are load-bearing because the bytes are hashed into the identity, and a mismatch yields a *different* identity that silently breaks decryption: (i) **all `uint256` args/values are decimal strings**, never JSON numbers, to avoid JavaScript's $2^{53} - 1$ precision loss and to

guarantee byte-identical serialization across platforms; (ii) `word` is **omitted from the canonical form when 0**, so single-value-getter conditions hash identically to the pre-`word` schema.

Trust tiers. *Tier 1* — on-chain state (`contract`/`block`/`event`, finalized) — is deterministic and is what v1 ships. *Tier 2* — external/oracle data — adds a data-source trust and a determinism risk; it is opt-in and later-phase, and operators *declare* which tiers they evaluate so clients can decide (§6/W7). The same condition machinery generalizes well beyond Veil — time capsules (`block.timestamp >= T`), multi-trigger delivery (`any(executed, deadline)`), milestone escrow, DAO-vote-gated reveals, or any third-party EVM contract getter — which is the adoption story (§9).

4.3 Network planes

Warden inherits drand's hard-won plane separation. drand's v2.0 post-mortem documented a reshare-downgrade attack that arose precisely from *merging* operator-control RPC with node-to-node RPC on one service; the fix split them, and Warden adopts the separation from the start:

- **Control plane** — local-only operator management (`wardenctl` → daemon over a localhost port). **Never internet-exposed.**
- **Private plane** — node-to-node messaging for DKG and resharing, authenticated and behind TLS (terminated by a reverse proxy; the daemon does not terminate TLS itself).
- **Public plane** — clients fetch partial decryptions over HTTP/relay. Read-only; partials are publicly verifiable against the master public key, so the relay need not be trusted.

4.4 Finality policy

A node must decide *when* an on-chain condition is "final enough" to release on. This is the genuinely new, security-critical surface Warden adds over drand, because **a reorg that reverts the condition after a release is an irreversible confidentiality leak** (§6/W3). Policy:

- Read at the chain's **finalized/safe head**, or at least the chain's finality depth (for Base, prefer L1-finalized state). The current implementation reads at the `finalized` tag.
- The **minimum finality floor is a federation-wide consensus parameter — identical for every node, not configured locally.** Per-node floors would make nodes disagree on whether a condition is final and stall release (a node with a higher local floor would block the threshold). `C.meta.finality` may override the floor *upward only*, never below it.
- The residual reorg risk for the chosen depth is documented, not hidden.

---

5. Cryptographic Construction

The cryptography is reused from drand near-verbatim; what follows states it precisely and marks the one new ingredient (the condition gate).

5.1 Setup (key generation)

- **Mainnet — real DKG, no dealer.** All operators run one DKG ceremony (Pedersen/Feldman VSS, drand-derived), producing one **master public key** `P_pub` and one **share `sk_i` per operator**; the master private key `msk` is never assembled. The threshold targets a BFT-style `t ≈ ⌈2n/3⌉` (minimum `⌊n/2⌋+1`). The deployment inherits drand's

lifecycle (joiners/remainers/leavers), strict control-/node-plane separation, and **`OldThreshold` downgrade protection** on every reshare (the v2.0 attack class).

- **Testnet — trusted dealer.** A one-off script generates `msk`, Shamir-splits it into `n` proper field-element shares, and injects one per node. The full key briefly exists in the dealer — **testnet-only, never mainnet** — and the script outputs `P_pub` (client config) and optionally each node's share-public-key (for partial verification).

5.2 Encryption (client, offline, no network interaction)

```
K          = random symmetric content key
inner      = AEAD(payload, K)
K_wrapped  = ECIES(K, recipientPub)
identity   = H("warden-cond-v1" ‖ jcs_rfc8785(condition))
outer      = IBE_encrypt(K_wrapped, identity, P_pub)      // BF-IBE (U, V, W)
envelope   = { alg: "warden-v1", network, condition, outer, inner }
```

The Boneh–Franklin ciphertext (with the FO transform for CCA security) is, following tlock:

```
U = r·G
V = σ  XOR  H2( e(Q_id, P_pub)^r )      where Q_id = H1(identity)
W = M  XOR  H4(σ)                        with r = H3(σ ‖ M)   (Fujisaki–Okamoto)
```

The pairing is written for the concrete BLS12-381 placement used by tlock/drand quicknet: the identity hashes into G1 (`Q_id = H1(identity) ∈ G1`), the master public key lives in G2 (`P_pub ∈ G2`), and partials/decryption keys are in G1, with `e: G1 × G2 → G_T`. Because the BF plaintext block is a fixed 32 bytes, `outer` is realized as the hybrid of §4.1 — `IBE(obk)` gates a random 32-byte key that AEAD-seals the larger `K_wrapped` — which is logically identical to IBE-gating `K_wrapped` directly, but is *necessary* rather than cosmetic since `K_wrapped` exceeds one BF block.

5.3 Release (the condition gate — the new component)

Each node, on a request for `identity` carrying condition `C`:

1. Verify `H(C) == identity` (binding).
2. Verify `C.type`/`tier` is supported by this node.
3. **Evaluate `C` against finalized chain state** (read the named chain at no fewer than `C.meta.finality` confirmations, never below the federation floor).
4. If `C` holds, return the partial `sig_i = sk_i · H1(identity)` (publicly verifiable against node `i`'s share-public-key); otherwise return `ConditionNotMet`.

The client collects `t` valid partials, Lagrange-combines them *in the exponent* into the IBE decryption key `outerKey = msk · H1(identity)`, IBE-decrypts `outer` to recover `K_wrapped`, has the recipient ECIES-open `K_wrapped` to `K`, and AEAD-decrypts `inner` to the plaintext. Decryption (given a met condition) requires no further interaction beyond gathering partials.

Identity domain-separation (security-critical). Unlike drand round numbers, which are globally unique by construction, condition-derived identities are *attacker-influenceable*. To prevent cross-item key reuse and grinding, the hash-to-curve `H1` must be strongly domain-separated and must bind the chain id, contract address, and full condition. A partial minted for one condition must never open a ciphertext for another (§6/W7).

Idempotent and retryable. Because the condition is monotone and chain state is permanent, release can be requested at any later time and always succeeds once met; nodes may cache released partials and serve them forever. A temporarily-offline network therefore causes *delay, not loss* (§6/W2). The current implementation ships **client-requested** signing (`POST /partial` evaluates and signs on demand); an **autonomous** mode (nodes watch the contract and pre-produce partials when the condition flips, for better liveness) is the mainnet target.

5.4 Resharing — the permanence mechanism

When operators join or leave (or for periodic share refresh), the federation runs **resharing**: each old node deals its existing share as the *free coefficient* of a new polynomial; Lagrange interpolation gives the new node set shares of the **same** `msk` — so `P_pub` is unchanged and **all historical ciphertexts remain decryptable**. Old and new operator sets may be entirely disjoint. `OldThreshold` MUST be specified (downgrade protection). Resharing *can* change membership, every share (rotation), and the threshold; it *cannot* change `P_pub` (and hence the decryptability of past ciphertexts). This is precisely the **permanence-with-churn** property that no off-the-shelf vendor offered and that the decades-horizon requirement demands.

5.5 Verifiability

Every partial `sig_i` is a BLS signature verifiable against node `i`'s published share-public-key; the combined key verifies against `P_pub`. Clients reject invalid partials before combining (relay-trust-free, exactly as drand clients verify beacons against the chain public key). A partial released for a condition that is *not* met is therefore **publicly provable misbehavior**, which can feed an eviction/slashing process in tokenized deployments (not required for the public-good federation, §9).

---

6. Security Analysis

This section is the credibility core. We state the trust split exactly and then walk the weaknesses one by one; for this audience, the rigor of the limitation analysis *is* the contribution's value.

6.1 The two-axis trust split

Property	Rests on	Strength
Content confidentiality (only the recipient ever reads plaintext — G3)	the recipient's own key (ECIES *inner* layer)	**Pure math.** Even a fully-colluding federation cannot read content.
Timing + revocation (unreadable until the condition holds — G1/G2)	a **threshold-bounded** federation (fewer than `t` collude) *and* nodes correctly observing the on-chain condition	**Threshold**, not trustless.

The double-wrap is what confines the federation's power to *timing*: a colluding federation can release the outer key early, or refuse it, but never read content.

The timing axis has two distinct thresholds, not one "majority". Confidentiality (G1) is broken only if `t` operators collude to release early; availability (G2) fails only if fewer than `t` honest operators are reachable to release at all. With `t ≈ ⌈2n/3⌉` the design tolerates up to `t−1` colluders without an early leak and up to `n−t` unavailable nodes without stalling — and the two cannot both be maximized, so the choice of `t` is an explicit confidentiality-versus-liveness trade, not a single honest-majority assumption.

6.2 Weaknesses (extensive, for the deployable design)

W1 — Early reveal by bribing a threshold. A *receiver* who corrupts `t` operators can have the outer key released before the condition holds and — with their own recipient key — read early. Bounds and defenses: only the receiver benefits (content is inner-locked, so a third-party briber gains nothing readable); the cost is corrupting `t` *independent* operators, so the defense is economic and diversity-based (a large, jurisdictionally-diverse federation and a higher `t`); the attack bites hardest on high-value beats (large-wallet inheritance) where the bribe bar equals the wallet's value. An optional trigger-gated-identity construction can force such an attack to be a *loud, detectable blanket release* rather than a quiet targeted one. As of the 2026-06 scan, **threshold traitor-tracing** (ST3, ePrint 2025/342) can make a colluding committee *publicly attributable* — turning W1 from undetectable to detectable-and-deterrable — and is the tracked mitigation (a layer to adopt when it has audited code, §10).

W2 — Liveness / denial → mostly delay, not loss. Because the ciphertext and the condition are permanent, release is retryable forever; a temporarily down or refusing network causes *delayed* delivery, not loss, and the receiver can pull at any later time. Permanent loss occurs only if *all* redundant Warden generations/networks die forever — mitigated by resharing (permanence-with-churn) plus multi-network redundancy; there is deliberately no first-party backup key (see W4). Delay severity is use-case dependent: acceptable for inheritance, harmful for urgent safety triggers, which argues for high availability there.

W3 — Finality / reorg leak (new; no drand precedent). If a node releases on an `executed==true` state that is subsequently reorged away, the reveal is irreversible. Mitigation is the finality policy of §4.4 (release only on finalized state at a conservative, federation-wide depth). This is a genuinely new, security-critical surface that Warden adds over drand precisely because Warden observes a chain rather than a clock.

W4 — Operator/infra capture, and the no-first-party-key rule. A federation entirely controlled by one party (e.g. all nodes in one cloud account) collapses to a single party: compellable and capable of early release. **Warden never holds a release key itself**, and real security requires *genuinely independent* operators. This is why the present testnet — five nodes, but a single trust domain (one operator, one provider) — has **zero timing-security and is for protocol exercise only.** A related capture vector is *condition-source integrity*: if many operators rely on the *same* public RPC provider, a compromise or **eclipse** of that provider can spoof `executed==true` and trick a threshold into early release *without breaking any cryptography.* Mainnet operators must therefore run their own chain node or use independent, authenticated, jurisdictionally-diverse RPC endpoints, and must read at the federation-wide finality floor. The drand v2.0 lesson is inherited in full: strict control-/node-plane separation and `OldThreshold` downgrade protection on every reshare, or a malicious operator can chain reshares to reduce the threshold to themselves and recover `msk`.

W5 — Master-key permanence is double-edged. `P_pub` is immutable across reshares (the permanence feature). The consequence: *any* eventual reconstruction of `msk` breaks *all* past and

future ciphertexts under that key — and Warden's ciphertexts may sit dormant for years. Reshare hygiene is non-negotiable, and a key-rotation/new-generation migration story (opt-in re-encryption, mirroring Maktub's immutable-V2 pattern) is part of the roadmap.

W6 — Not quantum-resistant. BLS/IBE are pairing-based, not post-quantum (the stated tlock horizon is on the order of five years). For payloads meant to "survive you by decades" this is a real limit to disclose to product and legal stakeholders. There is currently no deployable post-quantum replacement for threshold-IBE/BLS, so the decades claim is unbacked *at the threshold layer*; the mitigation is to keep the envelope `alg`-versioned and to post-quantum-harden the *inner* recipient layer first, where deployable PQ KEMs exist (§10).

W7 — Condition-evaluation trust (Tier-2). Tier-1 (on-chain, finalized) is deterministic. Tier-2 (oracle/API data) adds a data-source trust and a determinism risk; it stays opt-in and clearly labeled, with operators declaring supported tiers. Identity domain-separation (§5.3) must hold so that a partial for one condition cannot open another (grinding/replay).

W8 — Open-request metadata channel (client → node). Opening a Veil envelope POSTs the *condition* (which embeds `beatId`, the core address, and the chain id) to every node's `/partial`. Each node operator — and any on-path observer — therefore learns *the link between a requester IP and a beatId of interest* and the *timing* of the open attempt. The `beatId` and condition are already public on-chain; what is new is binding them to a requesting IP at open time. This is consistent with "not metadata-private" (§6.3), not a content leak (a tampered condition simply never releases — fail-closed). Mitigations for the independent-operator federation: client egress over a privacy-preserving transport (e.g. Tor or an oblivious relay) and TLS pinning / a known-key channel to deny passive collectors the linkage. With the all-ours testnet federation this is moot, but it must be addressed before independent operators run.

W9 — Cross-sender identity collision. The identity is `H("warden-cond-v1" ‖ jcs(condition))` with nothing per-message mixed in, so two independent senders who encrypt to the *same* condition share an identity and therefore share the released outer key `msk · H1(identity)`. For Veil this is benign: every beat carries a unique `beatId` in the condition `args`, so identities never collide. But for the generalized conditions of §4.2 (e.g. a popular time-capsule date used verbatim by many senders), an identical condition couples all its users — one requester's pull releases the outer key for every ciphertext under that condition. Content stays safe (the inner ECIES layer is per-recipient), but the *timing gate* is shared. Mitigation: applications that reuse generic conditions should bind a per-message salt into the condition `meta` (a node still hashes and recomputes the identity deterministically), or otherwise engineer identity uniqueness as Veil does via `beatId`.

W10 — Share-key registration / rogue-key. Public verifiability (§5.5) rests on each node's published share-public-key. For the "publicly provable misbehavior" property to hold, share-public-keys MUST be bound to the DKG transcript (or, on testnet, the dealer's commitment) and registered with proof-of-possession; otherwise an operator publishing an adversarially-chosen share-public-key could undermine partial verification, as in classic BLS rogue-key attacks. This is a requirement on the setup ceremony, stated here so it is not assumed away. (The current trusted-dealer testnet build is not exposed to this: the dealer computes every share-public-key honestly by construction; the requirement bites once nodes self-publish keys under DKG.)

W11 — Request-channel denial-of-service. Each `/partial` request triggers a finalized chain read and a BLS signing operation, so an unauthenticated public endpoint is a compute/RPC amplification vector. This is an availability concern, not a confidentiality one (an unmet or malformed condition simply yields no release — fail-closed). Mitigation: rate-limiting, lightweight request

authentication, and caching of already-released partials at the public plane.

6.3 What Warden is not trying to be

- **Not trustless.** The trustless ideal is witness encryption (the on-chain proof *is* the key, no federation); it is not deployable today. Warden is the best deployable approximation and is designed to migrate to WE via the `alg` envelope when it ships.
- **Not metadata-private.** Existence, timing, condition, and (for Veil) `beatId`, recipient set, and the sender-recipient relationship are public. Warden hides *content until the trigger*, never *that something exists*.

6.4 Honest claim language

For any derived copy: one *may* say *time-bound, revocable, end-to-end-sealed delivery enforced by an independent threshold federation — secure against early release unless* `t` *operators collude, and against denial as long as* `t` *stay honest and live — with redundancy.* One *must not* say *absolute*, *mathematically guaranteed* (for timing/revocation), *bulletproof*, *absolute anonymity*, *prevents bribery*, *cannot leak*, or *honest-majority* (the bound is the threshold `t`, not a majority).

---

7. Implementation

Warden is implemented as a single **Rust core** (`warden-core`) — chosen for memory safety, mature pairing-crypto crates, and the ability to drive one codebase to three client targets — plus the surrounding daemon, dealer, CLI, and bindings. The reference implementation is complete and deployed on testnet (§8).

Component	Crate / artifact	Role
Crypto core	`warden-core`	Threshold BF-IBE encrypt/decrypt, threshold-BLS partial/combine, the `warden-v1` double-wrap envelope, the federation file format. On `arkworks` BLS12-381. Library only.
Dealer	`warden-dealer`	Trusted-dealer ceremony CLI: materialize `msk`, Shamir-split, write `federation.json` (public) + per-node share files (secret, mode 0600). **Testnet only.**
Node	`warden-node` (`wardend`)	Condition-watcher + `POST /partial` threshold release. Reads Base Sepolia at the `finalized` tag. The security-critical evaluator is `node/src/eval.rs`.

Component	Crate / artifact	Role
Client CLI	`warden-cli` (`warden`)	`keygen` / `encrypt` (double-wrap → CID store) / `decrypt` (poll federation → combine → open, retry-until-released).
WASM	`warden-wasm`	`wasm-bindgen` bindings over core for the TypeScript SDK: `condition_identity` / `seal_gated` / `open_gated` / `combine`.
Mobile FFI	`warden-ffi`	Thin C-ABI over core for the Flutter app via `dart:ffi`. Panics are caught at the boundary and returned as `{ok:false,…}` JSON rather than aborting the host app. Cross-compiles to an iOS xcframework and 4-ABI Android `jniLibs`.

A reproducible 3-node federation is provided via `docker-compose`. The cryptographic loop is verified by the crate's own offline integration tests and by a local devnet end-to-end harness, and the full loop has been run end-to-end on live Base Sepolia (§8). The same core, compiled to native, WASM, and mobile FFI, is what closes the multi-platform-client gap (notably Flutter mobile) that disqualified off-the-shelf alternatives. Both the specification and the implementation are public and MIT-licensed (see *Artifact availability* below); the audit gate (§9, §10) governs operator recruitment and mainnet, not the openness of the source.

The reuse/replace boundary is clean: the entire IBE/threshold-BLS/DKG/resharing/age-envelope stack is reused near-verbatim from the drand lineage, while the genuinely new engineering is the round-scheduler-to-Base-condition-watcher swap, the domain-separated `H(condition)` identity, the finality handling for the chain read, and the on-demand (vs. periodic-firehose) condition-keyed release with an idempotent released-key store.

Artifact availability. Warden is developed in the open as a standalone primitive at **https://github.com/bytesbrains/warden** (MIT), with an overview and operator information at **https://warden.bytesbrains.com**. The specification (`docs/`) is public; the implementation and reproducible federation are in the repository.

---

8. Deployment Experience / Evaluation

Warden is deployed as a **five-node federation on fly.io** and runs the full loop end-to-end against **Base Sepolia**, integrated with the live Maktub protocol and a mobile client (in testing): create a beat (with a check-in interval), seal a payload to the condition `getHeartbeat(beatId).word[7] == true`, let the timer lapse so an executor flips the beat to executed, wait for the configured

confirmation depth, collect partials from the federation, combine, and decrypt. In this configuration the system operates as expected. The complementary negative path — `deactivate(beatId)` → the condition can never be satisfied → the payload is never released (permanent gibberish) — also holds. A local Hardhat devnet harness (`evm_increaseTime` to skip the timer) drives the same flow without funds and is the "Warden works for all conditions" gate.

Crucially, this deployment exercises the **protocol, its integration, and liveness — not the timing-security trust model.** All five nodes are operated by a single party (BytesBrains) on one provider (fly.io), i.e. a single trust domain; per W4 (§6) that collapses to one party and has, by construction, no early-release resistance. The missing ingredient is *independence* — distinct organizations, jurisdictions, and infrastructure — which is being actively recruited (§9). A revised version will add on-chain transaction evidence and a measurement pass.

We characterize this as **deployment experience** rather than a full performance evaluation, which is appropriate (and, at applied/systems venues, acceptable) for a pre-audit testnet system. The compute profile is favorable by construction: a single threshold-BLS partial is microseconds of work per request, and per-node resource needs are modest (low single-digit vCPU, a few GB of RAM, tens of GB of disk for the share, config, released-partial cache, and logs, over commodity always-on infrastructure with a Base RPC). A rigorous measurement pass — partial-release latency, combine latency, the finality-wait distribution, per-node footprint, and per-operator bandwidth — is future work and is the natural content for a future revision.

---

9. Federation Governance and Sustainability

Warden is intended to run as **token-free public-good infrastructure**, in the drand / League-of-Entropy tradition. Warden, and the Maktub protocol it serves, are public-good initiatives of BytesBrains Pte Ltd; Maktub is an open, unregistered on-chain protocol, and the operator *federation* is an open collaboration of independent organizations rather than a single legal entity — "foundation"/"federation" here name that collaboration, not an incorporated body. That this is *not* one entity is the point: independence across organizations is exactly the security property (§6, W4). The governance posture is deliberate and is itself a security property:

- **No token; operators credited, not paid.** The absence of a payment/staking economy keeps operators *disinterested* — neutrality and the lack of a financial stake are exactly what make a federation hard to bribe or capture, which is the property W1/W4 depend on.
- **Institutional, not personal, commitment.** The real ask of an operator is not hardware but *reliability, key-share custody, and longevity* — running for years and participating in occasional resharing ceremonies. This is why **institutions that persist** (universities especially) are the ideal operators; the effort once set up is a few hours per month of part-time sysadmin attention, with no dedicated team and no special hardware.
- **Independence is the security, not node count.** Even a small federation of genuinely independent operators (different organizations, jurisdictions, infrastructure) gives a meaningful `t`-of-`n`; the goal is diversity and longevity, not a data center. A federation that converges on one cloud account or one RPC provider has the security of one party (W4).
- **Funding line.** Audits, permanence, and continued development are funded through a treasury and grants rather than protocol fees; the honest status (testnet, pre-audit) is disclosed throughout. The decade-scale funding question — who pays for audits and continuity across

the full dormancy horizon — is openly unresolved, and the paper does not print durability claims it cannot back.

Recruiting independence (an open invitation). This independence is being actively pursued. The current five-node testnet runs within a single trust domain (BytesBrains, on one provider), and discussions are underway with universities and other long-lived institutions to take over genuinely independent shares. We extend this as an **open invitation**: any institution able to commit to reliability, key-share custody, and longevity is welcome to operate a founding node. That diversity — distinct organizations, jurisdictions, and infrastructure — not node count, is what gives Warden its timing-security (§6, W4), and turning this paper into that invitation is a deliberate part of why it is published (§11).

The federation's behavior has **no governance surface over its own cryptography**: membership churns via resharing, but the master public key, and hence the decryptability of every dormant ciphertext, is outside anyone's power to change. This mirrors the immutability invariant of the Maktub protocol layer it serves.

---

10. Limitations and Future Work

The weaknesses of §6 are the honest limitation set; the forward path addresses them through the `alg`-versioned envelope, which lets the cryptography migrate with **no change to recipients or to Maktub** and with old envelopes decryptable forever.

- **Gen-2 — signature-based witness encryption (SWE).** The "no master secret" property is longevity-optimal for a dead-man's-switch (it dissolves W5 and reshapes W1). Adopt when SWE has audited, production-grade, cost-acceptable implementations with mobile-acceptable ciphertext size.
- **Layer — threshold traitor-tracing.** Make a colluding committee publicly attributable, closing the *detectability* half of W1, when production-ready code exists. This is a layer over the base scheme, adoptable independently of gen-1/gen-2.
- **Gen-3 — witness encryption from SNARKs.** The trustless endgame: the on-chain proof *is* the key, no federation. Tracked semi-annually as it matures from theory.
- **Post-quantum migration.** No deployable PQ threshold-IBE exists; harden the *inner* recipient layer first (deployable PQ KEMs exist) and track the threshold layer. Keeping `alg`-versioning a hard requirement is the standing mitigation for W6.
- **Tier-2 conditions.** Oracle/API-gated release, opt-in and separately tiered, with the data-source trust made explicit (W7).
- **Full performance evaluation** and the autonomous (watch-and-sign) release mode for stronger liveness (§5.3, §8).

The architecture spike that chose gen-1 (DKG threshold-IBE) over the silent-setup and SWE challengers is re-run semi-annually, because the maturity of SWE and of WE-from-SNARKs is exactly what will flip the gen-2/gen-3 timing.

---

11. Ethics and Responsible Disclosure

Warden is dual-use infrastructure, and we treat the ethics of that squarely.

- **What it protects vs. what it does not.** Warden protects *delivery* (a sealed message reaches its recipients once the trigger fires) and *content* (only the recipient can read it). It does **not** protect *metadata*: existence, timing, the public condition, and the recipient/sender relationship are public on-chain by design (§6.3). We disclose this prominently rather than letting an "encrypted, decentralized" framing imply anonymity it does not provide, because a user who *believes* they are invisible when they are not is harmed by our overclaim.
- **No anonymity / press-freedom framing discipline.** The press-freedom use case is framed strictly as *work survival* ("your work survives you" — accident, illness, inability to deliver), never as a tool for adversarial-state anonymity, consistent with the protocol's honesty guardrail.
- **Pre-audit status.** The system is testnet and pre-audit; the current five-node federation is a single trust domain and so has no timing-security, and independent-operator commitments are being sought but not yet bound (§9, §10). We frame this work as *design + threat model + early deployment* precisely so that publishing it before the audit is responsible — openness supports the audit gate rather than substituting for it, and binding founding-operator commitments wait on the audit.
- **Why publishing gives nothing away.** Openness is part of Warden's trust model (no serious operator runs a black box), the codebase is MIT public-good, and the cryptographic substrate is already public (drand). Publishing the design, threat model, and deployment experience therefore costs no security; it *is* the federation-recruitment vehicle.

---

12. Conclusion

Warden takes the deployed timelock idea — a threshold-BLS federation that is, unknowingly, an IBE key generator — and changes exactly one thing: the identity is the hash of a *resettable on-chain condition* rather than a future round number, and release is gated on observing that condition true at finalized depth rather than on a clock. Around that one change it builds an honest systems story: a double-wrap envelope that confines the federation to timing and never to content; resharing that gives the master key permanence across decades of operator churn; a finality policy for the security-critical chain read that drand never needed; and a multi-platform implementation running end-to-end as a five-node testnet federation on Base Sepolia, integrated with the Maktub protocol (a single trust domain today; independent operators are being recruited). The cryptography is reused and audited; the contribution is the system, the resettable-condition threat model, and a limitation analysis (W1–W11) calibrated to claim exactly the two properties Warden keeps — guaranteed-but-deferred delivery and content confidentiality — and nothing more. The honest gap between this and the trustless ideal (witness encryption) is the roadmap, and the `alg`-versioned envelope is the bridge across it.

Appendix A — Envelope byte-format (`warden-v1`)

The wire form as implemented (`core/src/envelope.rs`); blobs are hex of the compressed-canonical / `nonce ‖ ct` bytes.

```
{
  "alg": "warden-v1",
  "network": "<warden federation / master-key id>",   // bound into both AEAD layers
  "condition": {                                       // PUBLIC; hashed into the IBE identity
    "type": "contract",
    "chain": 84532,                                    // Base Sepolia (testnet)
    "address": "0x<MaktubCore>",
    "fn": "getHeartbeat(uint256)",                     // no executed() getter…
    "args": ["<beatId>"],
    "word": 7,                                         // …executed is return field 7 (see §4.2)
    "test": { "cmp": "==", "value": true },
    "meta": { "finality": 32, "tier": 1 }
  },
  "outer": {                  // condition gate (hybrid IBE — see §4.1)
    "ibe": "<hex>",           // IBE(obk, H(condition)) — the (U,V,W) ciphertext, canonical-serialized
    "seal": "<hex>"           // nonce ‖ AEAD_obk(K_wrapped), K_wrapped = ECIES(K, recipientPub)
  },
  "inner": {                  // content layer; recipient-only. Warden never touches this.
    "ct": "<hex>"             // nonce ‖ AEAD_K(pad(payload))
  }
}
```

The identity `H("warden-cond-v1" ‖ jcs(condition))` is recomputable by any node from the public `condition`, so the envelope need not store it (`word` is omitted from the canonical form when 0). No recipient metadata is stored — the recipient is implicit in who can ECIES-open `K_wrapped`.

Appendix B — Condition schema (`cond-v1`)

One `type` per condition, plus a mandatory `meta`:

- `contract` — `{ chain, address, fn, args, word?, test: {cmp, value}, meta }`
- `block` — `{ chain, field: "number"|"timestamp", cmp, value, meta }`
- `event` — `{ chain, address, sig, args, meta }`
- `all | any | not | threshold` — `{ of: [...sub-conditions], k (threshold only), meta }`

Rules: `cmp ∈ { ==, !=, >=, <=, >, < }` (string); all `uint256` args/values are decimal strings (never JSON numbers); `word` (contract only, default 0, omitted when 0) selects the 32-byte ABI return word; serialization is RFC 8785 (JCS) — a number-vs-string mismatch yields a different identity and silently breaks decryption. The three non-negotiable constraints are determinism (finalized reads only), monotonicity/latching (non-monotone predicates rejected unless anchored), and finality/reorg safety (§3.1, §4.4). For `threshold`/compound conditions spanning multiple chains, deterministic evaluation across heterogeneous finality semantics is under-specified in v1 and is deferred to future work.

Appendix C — Parameter choices

- **Threshold:** target `t ≈ ⌈2n/3⌉` (BFT-style), minimum `⌊n/2⌋+1`.
- **Finality:** federation-wide floor (Base: prefer L1-finalized; the current implementation reads the `finalized` tag); `C.meta.finality` may raise it, never lower it.
- **Curve / scheme:** BLS12-381, Boneh–Franklin IBE over threshold BLS, RFC 9380 hash-to-curve, FO-CCA transform; AEAD = ChaCha20-Poly1305; recipient layer = ECIES over secp256k1 with HKDF/SEC1 shared-info.

References

(ePrint identifiers are given where applicable.)

- D. Boneh, M. Franklin. *Identity-Based Encryption from the Weil Pairing.* CRYPTO 2001.
- D. Boneh, B. Lynn, H. Shacham. *Short Signatures from the Weil Pairing.* ASIACRYPT 2001.
- A. Boldyreva. *Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.* PKC 2003.
- A. Shamir. *How to Share a Secret.* CACM 1979.
- P. Feldman. *A Practical Scheme for Non-interactive Verifiable Secret Sharing.* FOCS 1987.
- T. Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing.* CRYPTO 1991.
- R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems.* EUROCRYPT 1999.
- E. Fujisaki, T. Okamoto. *Secure Integration of Asymmetric and Symmetric Encryption Schemes.* CRYPTO 1999.
- N. Gailly, K. Melissaris, Y. Romailler. *tlock: Practical Timelock Encryption from Threshold BLS.* IACR ePrint 2023/189.
- drand / League of Entropy. Project documentation and the drand v2.0 post-mortem (2025).
- R. Rivest, A. Shamir, D. Wagner. *Time-lock Puzzles and Timed-release Crypto.* 1996.
- B. Wesolowski. *Efficient Verifiable Delay Functions.* EUROCRYPT 2019. / K. Pietrzak. *Simple Verifiable Delay Functions.* ITCS 2019.
- S. Garg, C. Gentry, A. Sahai, B. Waters. *Witness Encryption and its Applications.* STOC 2013.
- McFly: *Signature-Based Witness Encryption.* IACR ePrint 2022/433. / Compact SWE, ePrint 2024/1477. / eWEB (next-block SWE), ePrint 2025/1064.
- *Threshold Encryption with Silent Setup.* IACR ePrint 2024/263.
- *Silent Threshold Traitor-Tracing (ST3).* IACR ePrint 2025/342; ACM CCS 2025.
- *Witness Encryption from SNARKs / KZG.* IACR ePrint 2025/1364 (CRYPTO 2025).
- Threshold Network. *TACo — Threshold Access Control: conditional threshold decryption* (project documentation).
- Shutter Network. *Threshold encryption for front-running protection* (project documentation).
- E. Kokoris-Kogias, E. C. Alp, L. Gasser, P. Jovanovic, E. Syta, B. Ford. *Calypso: Private Data Management for Decentralized Ledgers.* VLDB 2021.
- M. Campanelli, B. David, H. Khoshakhlagh, A. Konring, J. B. Nielsen. *Encryption to the Future: A Paradigm for Sending Secret Messages to Future (Anonymous) Committees.* ASI-

ACRYPT 2022.
- RFC 9380 — *Hashing to Elliptic Curves.* / RFC 8785 — *JSON Canonicalization Scheme (JCS).* / RFC 8439 — *ChaCha20-Poly1305.* / RFC 5869 — *HKDF.*
- F. Valsorda et al. *The age file-encryption format*, v1 (`age-encryption.org/v1`; C2SP specification).
- Certicom Research. *SEC 1: Elliptic Curve Cryptography, Version 2.0* (SECG, 2009) — ECIES and EC primitives.
- Warden project: source repository — https://github.com/bytesbrains/warden (MIT); overview / operator information — https://warden.bytesbrains.com; specification (`docs/`) — https://github.com/bytesbrains/warden/tree/main/docs.