Nick Clark · Published July 2, 2026 · PDF

How to Authenticate Devices on a Jammed or Disconnected Network

If your devices lose their link to a certificate authority or key server, most authentication stacks stall: they cannot check a signature chain or a revocation list when the network is jammed, delayed, or dark. This guide describes an architectural approach for authenticating devices from locally retained material and delayed proofs, so a receiver can decide whether to trust a peer offline and finish verifying later. It is grounded in the Keyless Identity inventive step disclosed in United States Patent Application 19/388,580, and it describes an architecture you implement yourself, not a shipping library.

What You Are Building

You are building a way for two devices to authenticate each other when they cannot reach any shared online service at the moment they need to talk. Think of a drone mesh under active jamming, a sensor field on an intermittent backhaul, ground stations exchanging with a spacecraft across minutes of one-way delay, or edge nodes on an opportunistic link that appears for seconds and vanishes.

The search intent behind "how to authenticate devices on a jammed or disconnected network" is almost always the same underlying frustration: the authentication logic works fine on a lab bench with full connectivity, then falls over the moment the device

cannot contact a certificate authority (CA), an OCSP responder, or a key-distribution server. You want a receiver to be able to make a trust decision from what it already holds locally, accept or reject the peer deterministically, and settle any deferred verification once a link returns, all without a live third party in the loop.

This guide describes the architecture disclosed in the filing above. You will do the implementation.

Why the Obvious Approaches Fall Short

The standard tools are good tools, described here neutrally so you can see exactly where the gap is.

PKI and certificate chains. Public key infrastructure binds an identity to a long-lived public key, signed by a CA. Verification is local in principle: you can check a signature against a cached root. The friction is everything around it. Revocation checking (CRL or OCSP) typically wants a live fetch, certificates expire on a wall-clock schedule that a long-disconnected node cannot renew, and the trust model assumes a reachable hierarchy of anchors. The spec notes plainly that PKI "typically requires centralized trust anchors, global registries, and persistent key material," which is what makes it awkward for decentralized or intermittently connected environments.

Pre-shared keys. You can ship every device a symmetric key and skip the network entirely. This authenticates fine offline, but a single compromised device exposes the group, rotation is a logistics problem when devices are unreachable, and a captured long-lived secret is a standing liability.

Persistent keypairs held on device. Same offline benefit, but the private key is now a durable target: key compromise, side-channel extraction, and (over a long enough horizon) exposure to quantum attacks on the underlying hardness assumptions.

The structural gap common to all three: they either need a live authority at authentication time, or they lean on a long-lived secret that has to survive on the device. Neither fits a link that is down for hours and a device you cannot fully trust to keep a static secret safe.

The Architecture

The disclosed approach removes both the live authority and the persistent keypair. Identity is expressed not as a static credential but as a **trust slope**: a cumulatively validated sequence of dynamic hashes, each one a verifiable successor of the last. The core idea is that a receiver can check *continuity* locally, and can defer the parts of that check it cannot complete offline.

Dynamic hashes derived from local unpredictability. Each device advances a Dynamic Device Hash (DDH); each agent advances a Dynamic Agent Hash (DAH). A successor is computed by hashing the prior value together with a fresh unpredictability contribution, a non-repeating volatile salt, and a domain-separating tag. The spec gives the update rule in two interchangeable forms: `DAHt = H(DAHt−1 || Ext(Xt) || saltt || tag)` when the unpredictability comes from a locally observed state vector run through a strong extractor, or `DAHt = H(DAHt−1 || KDF(HWID, saltt) || tag)` when it comes from a static hardware anchor (TPM, TEE, or SoC identifier) plus a volatile salt. A hybrid hashes both sources into the same step. The point for offline work: the unpredictability is *non-exported*. An attacker who lacks the device's local state or salt cannot feasibly synthesize valid successors, and no registry lookup is involved in forming or checking one.

Trust-slope continuity as the check. A receiver stores a previously trusted step for a given peer and evaluates a presented successor against policy-defined continuity criteria: is it a valid descendant, does it advance monotonically, is the salt fresh relative to expected cadence, does it fall inside the stability-tuned acceptance neighborhood for

the local-state case. Acceptance requires forward movement along the slope; a value equal to a previously accepted successor, or one that regresses behind the stored reference, is rejected as replay or regression. All of this reads only local state.

Two-stage message binding. When a message is sent, the sender places its current dynamic hash in the transport header (for fast, stateless screening before any decryption) and *also* embeds the same value inside the encrypted payload. Payload keys are symmetric and derived transiently from the recipient's current identity via a key-derivation function over the recipient DAH/DDH and a domain-separating context; the message never carries the key. The receiver screens the header, derives its own key from its current identity, decrypts, then validates the embedded sender hash against the sender's stored slope. Both stages must pass. This is entirely offline once both sides hold the relevant slope state.

Delayed validation with bounded proof windows. This is the mechanism that makes the disconnected case work. When a receiver lacks a recent anchor for the sender, the sender can attach, for each missing step in the transmission window, the per-step materials needed to deterministically recompute the successor: an extractor token derived from the local state vector with a per-step salt, or a keyed derivation from the hardware anchor with a per-step salt, plus an optional reference to the last periodic anchor. On reconnect (or immediately, from cached anchors), the receiver replays the intervening steps by iteratively applying the update rule from its last trusted value forward to the presented identity, and accepts when the recomputed terminal value matches and opens against a trusted anchor. The spec calls out exactly the target environments: "delay-tolerant, mesh, opportunistic, or spaceborne links."

Sparse checkpoints for constrained devices. A device need not retain every step. It keeps selected identities plus periodic checkpoints that summarize validated mutations up to an epoch, and a slope proof supplies only the per-step materials for the missing interval, letting a verifier replay forward from a checkpoint. Checkpoint cadence is a policy knob that trades storage against replay effort.

Quorum recovery after state loss. If a device loses its lineage entirely (memory loss, corruption), it reseeds a fresh identity and requests attestations from previously trusted peers. Peers that find the request consistent with their retained checkpoints and anchors issue signed attestations; these are aggregated under a quorum policy into a recovery token that reattaches the device to the trust graph, with a forward link recorded so downstream verifiers bridge the pre-loss and post-loss segments. No central authority is contacted.

Header rotation and entropy-anchor rotation preserve freshness over long deployments, recording forward links so verifiers can reconcile old and new epochs under policy.

How to Approach the Build

The sketches below are illustrative and deliberately faithful to the spec's update rule. They are not a package you can install. You supply the hash, the extractor/KDF, the transport, and the policy.

1. **Choose your unpredictability source.** Constrained device exposing a hardware identifier: use the hardware-anchor form. Richer platform: build a local state vector from device-observable signals (monotonic counters, timing deltas, scheduler jitter, I/O micro-jitter, process histograms) and run it through a strong extractor. Or hybridize. This choice drives everything downstream.
2. **Implement the update rule and slope store.**

```
# illustrative, not a library
def next_hash(prev, entropy_contrib, salt, tag):
    return H(prev || entropy_contrib || salt || tag)   # salt is non-repea
```

For the local-state path, `entropy_contrib = Ext(project(local_state_vector))`; for the hardware path, `entropy_contrib = KDF(HWID, salt)`. Persist the slope append-only, and record a mutation class per step so provenance survives.

3. **Tune continuity policy before you tune anything else.** For local-state identity, this is a stability-tuned acceptance neighborhood (the spec describes locality-sensitive binarization and a Hamming-ball radius `r` calibrated to intra-role variation) so small measurement fluctuations do not cause spurious rejections while genuine role changes intentionally flip bits. For hardware-anchor identity, it is salt-freshness and cadence bounds. Set the replay horizon and the monotonicity rule here too.
4. **Build the two-stage message path.** Header = current dynamic hash. Payload = authenticated encryption under a key derived from the recipient's current identity, with the sender's current hash embedded inside. Receiver screens header, derives key from its own identity, decrypts, validates embedded hash. Make failure responses opaque so they do not leak rekey status.
5. **Add delayed validation for the disconnected case.** Have senders attach per-step proof materials plus an anchor reference for the transmission window. Have receivers implement forward replay from their last trusted value, and a checkpoint-request path for when their stored state predates the referenced anchor or the proof set is insufficient. Bound retries to a fixed policy window to avoid oracle leakage.
6. **Add sparse checkpointing** so memory-constrained nodes keep only selected identities and anchors and reconstruct on demand.
7. **Add quorum recovery** for devices that may lose state in the field, defining the quorum threshold via an explicit policy (count-based or trust-weighted).
8. **Decide legacy interoperability, if any.** The spec describes a segregated fallback adapter that speaks PKI to legacy counterparties while keeping that material strictly isolated from slope formation; if you need it, keep the isolation boundary strict and fail closed on any cross-contamination.

What This Does Not Give You

This is an architecture disclosed in a patent filing, not a drop-in library, an SDK, or benchmarked, production-proven software. There is nothing to `npm install`. You implement the hash, extractor or KDF, transport, key derivation, continuity policy, checkpointing, and quorum logic yourself, and you own the security review of that implementation.

It does not eliminate every online interaction. Bootstrapping a *first* trust relationship still requires that the two parties share some prior context: a receiver validates a successor against a step it already trusts, and quorum recovery needs peers that were previously trusted. This buys you offline operation *after* an initial relationship exists; it does not conjure trust between two devices that have never had any.

Delayed validation defers verification, it does not skip it. A message can be provisionally screened offline, but final acceptance may wait on a bounded proof or checkpoint that arrives later, and until then the receiver holds, not blindly trusts. The spec gives an entropy-based security estimate (`2^-lambda` offline forgery probability, `2^-lambda/2` under generic quantum search) whose real-world strength depends entirely on the min-entropy your unpredictability source actually delivers after extraction; a weak local state vector or a predictable salt undermines the whole model. No performance or throughput numbers are claimed here because the filing does not state them.

Where it does not apply: environments with no shared prior context and no path to ever establish one, or where you cannot obtain a genuinely non-exportable, high-entropy per-step contribution on the device.

Disclosure Scope

The approach described in this guide is disclosed in United States Patent Application 19/388,580. This guide is educational: it explains an architecture so a skilled developer can understand and implement it, and it is neither a warranty nor an offer of software. Every mechanism described above (the trust-slope update rule, two-stage message binding, delayed validation with bounded proof windows, sparse checkpoint recovery, and quorum-based reauthorization) traces to that filing; nothing here should be read as a claim that a shipping product, benchmark, or production deployment exists. You are responsible for your own implementation and its security review.

Keyless Identity (/keyless-identity)

All 40 steps → (/inventive-steps)

Identity from accumulated continuity. Post-quantum by construction.

U.S. 19/388,580 (/patents/19-388580)

PRIMARY TECHNICAL DISCLOSURE

- Stateless Device Pseudonymity and Secure Messaging in Cognition-Native Systems (/articles/stateless-device-pseudonymity-and-secure-messaging-in-cognition-native-systems)

SECONDARY TECHNICAL

- Trust Slope as Identity Primitive: Cumulative Hash Chains Replace Static Credentials (/articles/keyless-identity/trust-slope-identity)
- Dual-Source Identity Derivation: Hardware Anchors and Local State Vectors Combined Per Epoch (/articles/keyless-identity/dual-source-derivation)
- Stateless Symmetric Encryption: Session Keys Derived From Current Identity State (/articles/keyless-identity/stateless-encryption)
- Two-Stage Message Authentication: Transport Continuity Screening Before Semantic Validation (/articles/keyless-identity/two-stage-auth)
- Agent-Substrate Slope Entanglement: Binding Every Mutation Step to Its Execution Host (/articles/keyless-identity/slope-entanglement)

- Append-Only Mutation Lineage Log: Forward-Secure Identity Transition Chains (/articles/keyless-identity/mutation-lineage-log)
- Cumulative Slope Validation Across Substrates: Multi-Node Provenance Verification (/articles/keyless-identity/cross-substrate-validation)
- Quorum-Based Identity Recovery: Peer Attestation After Memory Loss (/articles/keyless-identity/quorum-recovery)
- Entropy Anchor Rotation: Proactive Identity Reseeding With Forward Links (/articles/keyless-identity/anchor-rotation)
- Biometric-Assisted Reseeding: Privacy-Preserving Fuzzy Extractors for Anchor Rotation (/articles/keyless-identity/biometric-reseeding)
- Delayed Slope Validation: Bounded Proof Windows for Disconnected Environments (/articles/keyless-identity/delayed-validation)
- Sparse Trust Slope Recovery: Compact Checkpoints for Resource-Constrained Devices (/articles/keyless-identity/sparse-checkpoints)
- Predictive Identity Validation: Drift Detection Before Full Discontinuity (/articles/keyless-identity/predictive-drift)
- Legacy PKI Fallback: Session-Scoped Adapters With Strict Isolation Boundaries (/articles/keyless-identity/pki-fallback)
- Post-Quantum Alignment: Hash-Based Security Without Vulnerable Hardness Assumptions (/articles/keyless-identity/post-quantum)
- Hardware-Anchor Embodiment of the Continuity-Identity Processor (/articles/keyless-identity/continuity-identity-processor-ic)

APPLICATIONS · GENERAL

- Keyless Workload Identity for Serverless Functions: Authenticating Ephemeral FaaS Without a Persistent Keypair (/articles/keyless-identity/serverless-workload-identity)
- Verifiable Agent Identity Without Credentials: Cryptographic Lineage for Distributed AI Agents (/articles/keyless-identity/trust-slope-entanglement)
- Post-Quantum Identity Migration Without a Public-Key Rebuild (/articles/keyless-identity/post-quantum-migration)
- IoT Device Authentication at Fleet Scale Without Keys or Certificates (/articles/keyless-identity/iot-authentication)
- Keyless Financial Identity Verification Without Credential Databases (/articles/keyless-identity/financial-identity-verification)
- Patient Matching Without a National Identifier: Keyless Identity for Cross-Institutional Patient Continuity (/articles/keyless-identity/healthcare-patient-continuity)
- Supply Chain Authentication Without PKI (/articles/keyless-identity/supply-chain-authentication)

- Keyless Smart Building Access Control: Credential-Free Entry Through Behavioral Continuity (/articles/keyless-identity/smart-building-access)
- Stopping Relay Attacks and Fob-Sharing: Binding Vehicle Access to the Driver, Not the Key (/articles/keyless-identity/vehicle-operator-identity)
- Refugee Identity Without Documents: A Keyless, Database-Free Approach (/articles/keyless-identity/refugee-identity)
- Licensing Keyless Identity at the Silicon Layer: Component-Level IP for Verifiable Device Provenance (/articles/keyless-identity/silicon-vendor-licensing)
- How Do Agents Prove Identity Without a Static Secret or an Issuer? The Market Is Converging on Keyless Continuity (/articles/keyless-identity/agent-identity-convergence)
- Drone Swarm Identity Under Jamming: Keyless Authentication When There Is No Certificate Authority to Reach (/articles/keyless-identity/drone-swarm-jammed-identity)
- Forced Off the Keys: Why the Post-Quantum Migration Points Past PKI to Keyless (/articles/keyless-identity/post-quantum-forcing-function)
- Authenticating Spaceborne and Interplanetary Links: Keyless Identity for Delay-Tolerant Networks With No Reachable Certificate Authority (/articles/keyless-identity/spaceborne-dtn-authentication)
- Authenticating Federated Learning Nodes Without Keypairs or a Certificate Authority (/articles/keyless-identity/federated-learning-node-authentication)

APPLICATIONS · SPECIFIC

- Okta Alternative for Keyless Identity: Federated IdP vs Credential-Free Continuity (/articles/keyless-identity/okta)
- Auth0 Alternative: Keyless Identity Beyond Stored Credentials (/articles/keyless-identity/auth0)
- YubiKey Alternative for Keyless Identity: Beyond the Stored Private Key (/articles/keyless-identity/yubico)
- CLEAR Alternative: Biometric Identity Without a Stored Template Database (/articles/keyless-identity/clear)
- Worldcoin Scans Irises to Prove Humanity. The Proof Depends on a Central Enrollment System. (/articles/keyless-identity/worldcoin)
- Jumio Alternative for Continuity-Based Identity: Keyless Identity Beyond Document Verification (/articles/keyless-identity/jumio)
- Microsoft Entra Alternative: Keyless Identity Beyond Stored Credentials (/articles/keyless-identity/microsoft-entra)
- Ping Identity vs Keyless Identity: A Post-Quantum Alternative to PKI-Bound Federation (/articles/keyless-identity/ping-identity)

- OneLogin Alternative: Keyless Identity Beyond the Stored SSO Credential (/articles/keyless-identity/onelogin)
- Duo Security Made MFA Ubiquitous. The Second Factor Is Still a Credential. (/articles/keyless-identity/duo-security)
- Thales HSM vs Keyless Identity: Protecting Keys or Eliminating Them? (/articles/keyless-identity/thales-hsm)
- Entrust Alternative: Keyless Identity Beyond Certificate-Based Trust (/articles/keyless-identity/entrust)
- DigiCert Alternative for Keyless Identity: Beyond Certificate-Chain Trust (/articles/keyless-identity/digicert)
- Let's Encrypt Alternative: Keyless Identity Beyond the Certificate Model (/articles/keyless-identity/lets-encrypt)
- Qorvo Secure Element Alternative: Continuity Identity Above the Root of Trust (/articles/keyless-identity/qorvo-secure-element)
- NXP EdgeLock Secure Element Alternative: Continuity Above Key Custody (/articles/keyless-identity/nxp-trust-anchor)
- Infineon OPTIGA Alternative: Keyless Continuity Identity Beyond Stored-Key Roots (/articles/keyless-identity/infineon-secure-microcontroller)
- Microchip Trust Platform Alternative: Keyless Identity Above the Secure Element (/articles/keyless-identity/microchip-trust-platform)
- Indicio SSI alternative: keyless identity beyond wallet-held keys (/articles/keyless-identity/indicio-ssi)
- Sovrin Foundation Alternative: Keyless Identity Beneath Self-Sovereign Identity (/articles/keyless-identity/sovrin-foundation)
- W3C DIDs vs keyless identity: who holds the controller's keys? (/articles/keyless-identity/w3c-dids)
- W3C Verifiable Credentials and Keyless Holder Binding (/articles/keyless-identity/w3c-verifiable-credentials)
- Keycard Alternative: Issuer-Free Agent Identity Beyond Token-Based IAM (/articles/keyless-identity/keycard)
- Aembit Alternative: Carried Continuity Beyond External Attestation (/articles/keyless-identity/aembit)
- Astrix Security Alternative: Keyless Identity Beneath the NHI Governance Overlay (/articles/keyless-identity/astrix-security)
- Oasis Security Alternative: Keyless Non-Human Identity Beyond External Lifecycle Anchoring (/articles/keyless-identity/oasis-security)

- Token Security Alternative: NHI Catalog vs. Keyless Cryptographic Continuity (/articles/keyless-identity/token-security)
- Entro Security Alternative: Secret Discovery vs. Keyless Secret Elimination (/articles/keyless-identity/entro-security)
- SPIFFE/SPIRE alternative: workload identity without a certificate authority or persistent keypair (/articles/keyless-identity/spiffe-spire)
- HashiCorp Vault vs a keyless trust slope: where does the identity of the caller come from? (/articles/keyless-identity/hashicorp-vault)

Keyless Identity overview → (/keyless-identity)