Nick Clark · Published July 2, 2026 · PDF

How to Authenticate Users Without Passwords or Stored Keys

If you build agents, edge devices, or delay-tolerant links, you eventually hit the same wall: every party you authenticate needs a password to protect or a long-lived key to store, register, and eventually rotate. This guide describes an architectural alternative in which identity is a validated sequence of derived hashes rather than a stored secret. It is an architecture disclosed in United States Patent Application 19/388,580, not a shipping library, and it centers on the Keyless Identity inventive step.

What You Are Building

You are building an authentication layer where no party holds a password to verify and no party stores a long-lived private key. Instead, each participant, whether a user's device, an edge node, or an autonomous agent, proves who it is by demonstrating that its current identity value is a valid continuation of a previously trusted one. The receiver checks that continuity locally, using only state it already holds.

This is the problem a lot of teams actually have when they type "how to authenticate users without passwords or stored keys" into a search box. You may be running software on devices that cannot safely retain a secret, agents that spawn and mutate too fast to enroll in a registry, or links (mesh, opportunistic, spaceborne) where a live handshake with a certificate authority is impossible. The architecture below, disclosed

in United States Patent Application 19/388,580, is aimed squarely at those environments: it treats identity as memory-resolved behavioral continuity rather than a static credential.

Why the Obvious Approaches Fall Short

The two conventional answers each work well inside their intended envelope, and it is worth being precise about where that envelope ends.

Passwords move a shared secret from the user to a verifier. That secret has to be transmitted, hashed, and stored somewhere, which creates a standing target and forces you to manage reset flows, breach response, and phishing. Passwordless schemes built on FIDO or WebAuthn remove the shared secret but replace it with a device-held private key: better, but still a persistent key that must live somewhere and be enrolled per relying party.

Public key infrastructure replaces shared secrets with asymmetric keypairs and signature validation. Per the background of the filed disclosure, PKI typically depends on centralized trust anchors, global registries, and persistent key material, and exposes users and devices to key compromise, metadata correlation, certificate revocation failure, and susceptibility to quantum cryptographic attacks. None of that makes PKI wrong; it makes it a poor fit when there is no reachable authority, no durable place to keep a key, or no synchronized registry to consult. In ephemeral or cognition-native systems such as distributed AI agents or stateless substrates, the disclosure notes that maintaining static credentials is impractical or infeasible.

The structural gap is the same in every case: both approaches root trust in a stored artifact. If your deployment cannot keep that artifact safe, reachable, or fresh, you need identity that is derived rather than stored.

The Architecture

The disclosed approach expresses identity as a **trust slope**: the cumulatively validated sequence of Dynamic Agent Hashes (DAH) or Dynamic Device Hashes (DDH) formed by successive, verifiable identity mutations. Each step is computed from the immediately prior step plus a source of non-exported unpredictability, so a receiver can evaluate continuity and provenance locally, without centralized authorities, long-lived keypairs, or synchronized registries.

The update rule. Each successor identity is a hash of the prior identity, a fresh unpredictability contribution, a volatile (non-repeating) salt, and a domain-separating tag. The disclosure gives two interchangeable sources for the unpredictability contribution:

- A **static hardware anchor** (for example a TPM, TEE, or SoC identifier) combined with a per-epoch volatile salt via a key derivation, expressed in the spec as `DAHt = H(DAHt−1 || KDF(HWID, saltt) || tag)`. This suits constrained devices.
- A **local state vector** of device-observable signals (monotonic counters, high-resolution timing deltas, scheduler jitter, I/O micro-jitter, and similar), normalized, projected, and passed through a strong extractor to yield a bounded token, expressed as `DAHt = H(DAHt−1 || Ext(Xt) || saltt || tag)`. This suits richer platforms.

A hybrid embodiment hashes both sources into the same step. Critically, the salt is non-repeating at the device and epoch level, so successors stay unpredictable even when the hardware anchor is constant, and an attacker lacking the device's local state or volatile salt cannot feasibly synthesize valid successors.

Stability tuning. For the local-state source, the feature map applies normalization, clipping, signed random projections with a public seed, and a locality-sensitive binarization, so small fluctuations in the local state produce a stable token while a

genuine role or zone change flips a controlled subset of bits. That is what lets ordinary measurement noise pass without breaking authentication, while real context changes intentionally force a new identity.

Transiently derived symmetric keys. There is no key exchange. To send a protected message, the sender derives a symmetric key by applying a key-derivation function to the recipient's current DAH or DDH plus a domain-separating context, then applies authenticated encryption. The message carries the sender's current DAH in the transport header and embeds a second copy of the sender's DAH inside the ciphertext. Per the disclosure, the message itself does not include the symmetric key.

Two-stage validation. The receiver first screens the header DAH against its last trusted successor with a fast continuity check, rejecting off-slope traffic before decryption. If that passes, it derives the symmetric key from its own current identity and decrypts; successful decryption proves the payload was encrypted for the recipient's correct memory-resolved identity. It then extracts the embedded sender DAH and validates it against the reconstructed sender slope. The message is accepted only on success of both stages, which resists malformed traffic before decryption and man-in-the-middle substitution after it.

Replay and spoof resistance. Acceptance is bound to monotonic progression along the slope. A presented value equal to a previously accepted successor, or one that regresses behind the last trusted state, is rejected as replay or regression; policy may also require advancing an epoch counter or a minimum inter-step interval.

Delayed and sparse validation. Because each successor depends only on the immediately prior value and the disclosed per-step materials, a receiver that lacks a recent anchor can replay intervening steps from its last trusted value using a bounded set of mutation proofs the sender supplies, or request a bounded checkpoint. This is what makes the model work over high-latency, intermittent, disconnected, or spaceborne links: validation is deferred and bounded rather than live. An append-only

lineage folds each step into a cumulative chain hash with periodic anchors, so long histories are verifiable from sparse storage and any omission or reordering diverges the terminal value.

Recovery without stored secrets. If a device loses its lineage, it reseeds a new initial identity and requests attestations from previously trusted peers; once a quorum policy is met, the aggregated attestations form a recovery token that is appended into lineage as a successor anchor, reattaching the device to the trust graph without any persistent credential.

How to Approach the Build

You are implementing this yourself. The steps below are the order the disclosure's mechanisms compose in; the sketches are illustrative and faithful to the spec, not a package to install.

1. **Choose your unpredictability source.** If your targets have a TPM, TEE, or SoC identifier, start with the hardware-anchor source. If they do not but expose rich local signals, use the local-state source. Support both if your fleet is mixed; the hybrid step is just both inputs hashed together.
2. **Implement the update rule as the one primitive everything else calls.** Fix a single hash and a single domain-separating tag scheme up front.

```
# illustrative, spec-faithful pseudocode
def next_identity(prev, salt, tag, hwid=None, lsv=None):
    if hwid is not None:
        contrib = kdf(hwid, salt)
    else:
        contrib = extract(project_and_binarize(lsv))  # stability-tuned
    return H(prev || contrib || salt || tag)
```

Generate `salt` as non-repeating per step. Never export the raw local state vector or the hardware secret.

3. **Establish the slope root.** Compute `DAH0` / `DDH0` from your chosen source and a volatile salt, optionally binding a semantic context code (role, zone) into the first step so identity carries provenance from epoch zero.
4. **Build the receiver's continuity check.** Store the last trusted successor per sender. On a presentation, reconstruct the expected successor neighborhood from that stored value and test whether the claim is a valid on-slope successor under your policy bounds. For the local-state source, that bound is an acceptance radius over extractor outputs; for the hardware-anchor source, it is salt freshness and cadence. Reject regressions and repeats.
5. **Wire the two-stage message path.** On send: derive the symmetric key from the recipient's current identity, authenticated-encrypt the payload, embed your own current DAH inside the ciphertext, and put your header DAH outside. On receive: screen the header, decrypt with a key derived from your own current identity, then validate the embedded sender DAH against the sender slope. Make failure at either stage a deterministic reject.
6. **Add the drift and rekey fallbacks.** When a sender lacks the recipient's current identity, derive from the most recent trusted recipient anchor and, on decryption failure, run a short challenge-response rekey or a checkpoint request bounded to a fixed retry window (the disclosure caps retries to avoid oracle leakage).
7. **Add lineage, checkpoints, and anchors for offline operation.** Fold each step into a cumulative chain hash and emit a periodic anchor every J steps. Have senders ship bounded per-step proofs so a disconnected receiver can replay from its last anchor to the presented identity.
8. **Add rotation and recovery last.** Implement a staleness monitor that reseeds a new entropy anchor with a forward link binding the old terminal value to the new initial identity, and a quorum recovery path for devices that lose state.

Define your continuity policy, acceptance radius, cadence bounds, checkpoint interval, and quorum thresholds explicitly. The disclosure treats these as tunable policy, not fixed constants, and your security depends on the min-entropy of the per-step contribution, so budget that deliberately.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to `npm install`, no SDK, and nothing here that "just works" out of the box; you implement the update rule, the two-stage path, the lineage, and every policy threshold yourself. The pseudocode above is illustrative, not a reference build.

It is a method disclosed in a patent filing, not a benchmarked or production-proven product. The filing does not report throughput, latency, or false-reject numbers, and neither does this guide, so you must measure your own implementation. In particular, your acceptance radius and stability tuning for the local-state source will need empirical calibration to keep genuine successors passing while off-manifold claims fail.

The model also does not remove all trust assumptions. Its security rests on the unpredictability of per-step inputs and the preimage resistance of your chosen hash and extractor; the disclosure frames offline next-step forgery probability as roughly `2^-lambda` (about `2^-lambda/2` against Grover-style quantum search) where lambda is the extracted min-entropy, so weak entropy or a weak hash breaks it. It presumes a reachable set of previously trusted peers for quorum recovery, and it does not, by itself, speak PKI. Interoperating with legacy PKI counterparties requires the disclosed segregated fallback adapter, which is deliberately isolated so that no PKI artifact ever enters slope formation. Where you genuinely have a durable secure element, a reachable certificate authority, and synchronized registries, conventional PKI may remain the simpler choice; this architecture earns its keep specifically where those are absent.

Disclosure Scope

The approach described here is disclosed in United States Patent Application 19/388,580. This guide is educational: it explains an architecture and how a developer might approach building it, and it is not a warranty, a guarantee of security or fitness, or an offer of software. Every description of how the approach works is drawn from that filing; nothing here should be read as a benchmarked result, a shipping product, or a promise of performance. You are responsible for your own implementation, its cryptographic parameter choices, and its validation.

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)