Nick Clark · Published July 2, 2026 · PDF

How to Encrypt Messages Without PKI or a Key Exchange

If you are building for stateless workers, disconnected edge nodes, or agents that cannot hold a long-lived private key, PKI and interactive key exchange become the hard part. This guide describes an architecture for deriving a symmetric key from a recipient's current dynamic identity, with no keypair, no certificate authority, and no handshake on the common path. It describes an architecture disclosed in United States Patent Application 19/388,580, the home of the Keyless Identity inventive step, not a shipping library.

What You Are Building

You want two parties to exchange encrypted messages, but you cannot lean on the usual machinery. There is no certificate authority you trust, no registry to look up a recipient's public key, and no reliable moment to run a Diffie-Hellman handshake because the recipient may be offline, ephemeral, or memory-constrained when you send. This is the situation for serverless functions with no durable storage, delay-tolerant and mesh links, IoT endpoints, and distributed agents that spin up and vanish.

The goal is confidentiality plus sender authentication where the sender derives a symmetric key from the recipient's *current identity state* and sends a self-contained message. The recipient reproduces the same key from its own local identity state and

decrypts. No key travels on the wire, no keypair is persisted, and no external authority is consulted on the common path.

This guide describes the architecture disclosed in United States Patent Application 19/388,580. It is a design you implement yourself, not a package you install.

Why the Obvious Approaches Fall Short

The standard approaches are sound where their assumptions hold; the problem is the assumptions.

PKI with certificates. Public key infrastructure binds a persistent public key to an identity through a certificate authority. It works well for the web, but it requires a long-lived private key on each party, a trust anchor both sides accept, and a working revocation story. In decentralized or memory-constrained settings there may be no acceptable central anchor, no durable place to keep a private key, and no live path to a revocation service when you need it. The patent's background section names these same structural constraints: persistent key material, centralized trust anchors, and revocation dependence.

Interactive key exchange. An ephemeral Diffie-Hellman handshake avoids long-lived encryption keys, but it is interactive: both parties must be online at the same time to complete the exchange before the first protected message. That round trip is exactly what you do not have with a store-and-forward relay, a spaceborne link, or a recipient that is not currently running.

Pre-shared symmetric keys. A shared secret removes the handshake but reintroduces a static credential to provision, store, and rotate across every pair of parties. That does not scale to large, fluid populations of short-lived agents, and a single leaked key compromises every message under it.

The structural gap is the same across all three: each treats identity as a *static credential* that must be held, exchanged, or certified. The architecture below treats identity as a moving, locally reconstructable value, which removes both the persistent secret and the handshake from the common path.

The Architecture

The core idea is that a party's identity is not a fixed key but a *trust slope*: a cumulatively validated sequence of ephemeral hashes, each one a verifiable successor of the last. The spec calls these a Dynamic Agent Hash (DAH) for an agent and a Dynamic Device Hash (DDH) for a device. A symmetric key is derived transiently from the recipient's *current* dynamic identity, so it exists only for as long as that identity step does and never needs to be sent.

1. Identity as a successor chain. Each identity step is computed from the immediately prior step plus a fresh source of local unpredictability. The spec's update rule is $DAH_t = H(DAH_{t-1} \| Ext(X_t) \| salt_t \| tag)$, where X_t is derived from a locally observed state vector, $Ext(\cdot)$ is a strong randomness extractor, $salt_t$ is a non-repeating volatile salt, and tag is a domain separator. A hardware-anchor variant substitutes $KDF(HWID, salt_t)$ for the extractor term, and a hybrid concatenates both. Because each step binds to non-exported unpredictability (the local state vector, or a hardware anchor combined with the salt), an attacker who lacks the device's local state cannot feasibly synthesize valid successors. This is the mechanism that lets identity move without a keypair.

2. Deriving the message key from the recipient's identity. Per Section 3 of the spec (FIG. 4), the sender applies a key-derivation function to the recipient's current DDH or DAH together with a domain-separating context string, and uses the resulting symmetric key for authenticated encryption of the payload. The recipient, holding its own current identity, runs the corresponding derivation and reproduces the same key to

decrypt. The spec is explicit that the message does not include the symmetric key. That is the property you are after: the key is a function of an identity state both sides can independently compute.

3. Two-stage binding at transport and payload layers. The sender places its current dynamic hash (`DAH_t`) in the transport header for fast, stateless screening, and *also* embeds a copy of its current sender hash (`DAH_S`) inside the encrypted payload. The recipient does two checks: first a fast continuity check that the header hash is a valid on-slope successor of the last trusted state (this rejects malformed traffic *before* decryption); then, after decrypting, it validates the embedded sender hash against the reconstructed sender slope. The spec states the message is accepted only on successful validation of both. Binding identity at both layers is what prevents a man-in-the-middle from swapping the payload after transport-level screening.

4. Replay and spoof resistance from monotonic progression. Acceptance is bound to forward movement along the slope. A presented hash that equals a previously accepted value for the same sender and context, or that regresses behind the last trusted state, is rejected as replay or regression (spec Section 4). Off-slope claims are treated as probable spoofs. There is no nonce database to consult and no authority to ask; the check is local against retained slope state.

5. A bounded fallback for identity drift, not a standing handshake. When the sender lacks the recipient's current identity, it encrypts under the most recent trusted recipient anchor and sends. On decryption failure, the spec provides a bounded fallback: either a short challenge-response rekey scoped to the recipient's current epoch, or a checkpoint request that returns a bounded proof window sufficient to advance to the recipient's current identity. Retries are capped by policy to a fixed attempt window to avoid oracle leakage. Crucially, this is an off-path recovery step, not a handshake required before every message.

6. Why this is post-quantum aligned. The spec grounds security in the unpredictability of per-step inputs and the preimage resistance of the hashes and extractors, rather than on the number-theoretic hardness that Shor's algorithm targets. It states that with `λ` bits of min-entropy per step, an offline next-step forgery succeeds with probability approximately `2^(−λ)`, degrading only quadratically to about `2^(−λ/2)` under Grover-style search, and suggests 256 to 512-bit extractor and digest sizes as conservative margins. This is a claimed structural property of the design, not a benchmark.

How to Approach the Build

The following is an ordered path a developer would take to implement the architecture. The interface sketches are illustrative and faithful to the spec; they are not a library.

Step 1: Choose your unpredictability source. Decide per device class. Constrained devices that expose a hardware identifier (TPM, TEE, SoC ID) use the hardware-anchor path: `KDF(HWID, salt_t)`. Richer platforms build a *local state vector* from device-observable signals (monotonic counters, high-resolution timing deltas, scheduler jitter, I/O inter-arrival micro-jitter) and run it through a strong extractor. You can hybridize by hashing both into the same step. The salt must be non-repeating per device and epoch.

Step 2: Make the local state vector stable enough to be reproducible. This is the subtle part. The recipient must reproduce the *same* identity value the sender keyed against, so small measurement fluctuations must not change the token while genuine role or context changes must. The spec's approach: normalize and clip signals to bounded ranges, project to a fixed dimension via signed random projections with a public seed, and apply a locality-sensitive binarization so that minor fluctuations yield stable `X_t` but real changes flip a controlled subset of bits. Budget real engineering time here; get this wrong and you get spurious decryption failures.

Step 3: Implement the update rule and a slope store. Wrap the successor computation:

```
# illustrative, faithful to spec Section 2, not a drop-in library
def next_identity(prev, x_t, salt, tag):
    token = extractor(project(x_t))        # Ext(X_t)
    return H(prev + token + salt + tag)     # DAH_t
```

Each party keeps an append-only record of validated steps and records a *mutation class* per step (role update, delegation, policy commit) for provenance. Receivers store the last trusted successor per sender.

Step 4: Implement send. Derive the message key from the recipient's current identity, encrypt with an authenticated cipher, and embed your own current hash in the ciphertext:

```
# illustrative
key = KDF(recipient_current_DAH, context_string)
payload = AEAD_encrypt(key, plaintext + embed(sender_current_DAH))
message = Header(sender_current_DAH) + payload   # key is NOT in message
```

Step 5: Implement receive as two stages. First screen the header hash for on-slope continuity against your last trusted state for that sender; reject or defer before touching the ciphertext. Then derive the decryption key from *your own* current identity, decrypt, extract the embedded sender hash, and validate it against the sender's reconstructed slope. Accept only if both stages pass. Record rejections with an explicit reason (continuity violation, neighborhood mismatch, salt staleness, replay).

Step 6: Add the bounded fallback and sparse recovery. Implement the capped rekey handshake and the checkpoint request so a sender working from a stale recipient anchor can advance to the current identity. For memory-constrained or intermittently connected nodes, add the delayed-validation path: senders ship bounded per-step proofs (extractor tokens and/or keyed derivations with per-step salts) so a receiver can replay intervening steps from its last periodic anchor without live synchronization. Keep failure responses opaque so they do not leak rekey status.

Step 7: Add rotation and replay policy. Enforce non-reuse within a policy horizon, a minimum inter-step interval, and (per the spec) a two-epoch acceptance window with per-sender rate limits keyed to header continuity. Rotate entropy anchors on staleness, recording a forward link so verifiers can bridge old and new epochs.

What This Does Not Give You

This is an architecture disclosed in a patent filing, not a drop-in library, an SDK, or a downloadable package. There is nothing to `npm install`. You implement every component described above yourself, and the security you get depends on how well you do it, especially the local-state-vector stability and the min-entropy `λ` of your per-step contribution.

It has not been benchmarked or productized here, and no performance numbers are claimed beyond the forgery-probability relationships the spec derives from min-entropy. Treat those as the design's stated security argument, not measured results from a running system.

It is not a general replacement for PKI. If your counterparty is a browser talking to a public web server, PKI is the right tool and this is not. The architecture targets decentralized, stateless, memory-constrained, and disconnected environments where a persistent keypair or a synchronous handshake is impractical. For legacy

interoperability the spec keeps a strictly segregated fallback adapter that uses conventional PKI signatures, walled off from the trust slope; that is a bridge, not the core design.

Finally, correctness hinges on both parties independently reconstructing the same identity state. If the recipient's identity has drifted and no fallback or checkpoint can bridge it, a message will not decrypt. Sizing the acceptance windows, salt cadence, and checkpoint frequency for your environment is design work you own.

Disclosure Scope

The approach described in this guide is disclosed in United States Patent Application 19/388,580. This guide is educational: it explains the architecture and how a developer might approach building it. It is not a warranty, a specification of a shipping product, or an offer of software, and it does not grant any license. Every mechanism described above traces to that filing; where the filing does not state a parameter, guarantee, or measurement, this guide does not claim one. Readers implementing the architecture are responsible for their own security review and for any patent, export, and compliance considerations that apply to their use.

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)