Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 August 2026

N. Ritz
Independent

1 February 2026

Interactive DPoP
draft-ritz-idpop-00

Abstract

This document describes IDPoP, an extension to DPoP [RFC9449] that uses a key derivation scheme to separate access control from identity. It mitigates credential exfiltration risks by requiring fresh hardware attestation to unseal identity keys via an interactive challenge.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 August 2026.

Table of Contents

1. Introduction

Standard DPoP binds tokens to keys but recent events have increased risks to exfiltration attacks where autonomous systems (such as agentic AI) leak valid credentials via prompt injection. Likewise, when malware compromises a device filesystem, attackers steal credentials and impersonate users until explicit revocation.

This proposal introduces a derivation scheme separating *access control* (encapsulate/decapsulate) from *identity* (signing).

2. Three-Factor Interface Model

Factors:

* *PF (Provisioning Factor):* Represents authorization intent (client_id + scope + audience).

* *EF (Evidence Factor):* Confidential attestation proof (TPM quote, YubiKey HMAC-secret, biometric), never transmitted.

* *VF (Vault Factor):* High-entropy secret, encapsulated by Evidence derived key.

Two-Stage Derivation:

1. Access gating: K_kem = HKDF(salt=PF, ikm=EF, info="seal")

2. Identity derivation: K_attest = HKDF(salt=PF, ikm=VF, info="dpop")

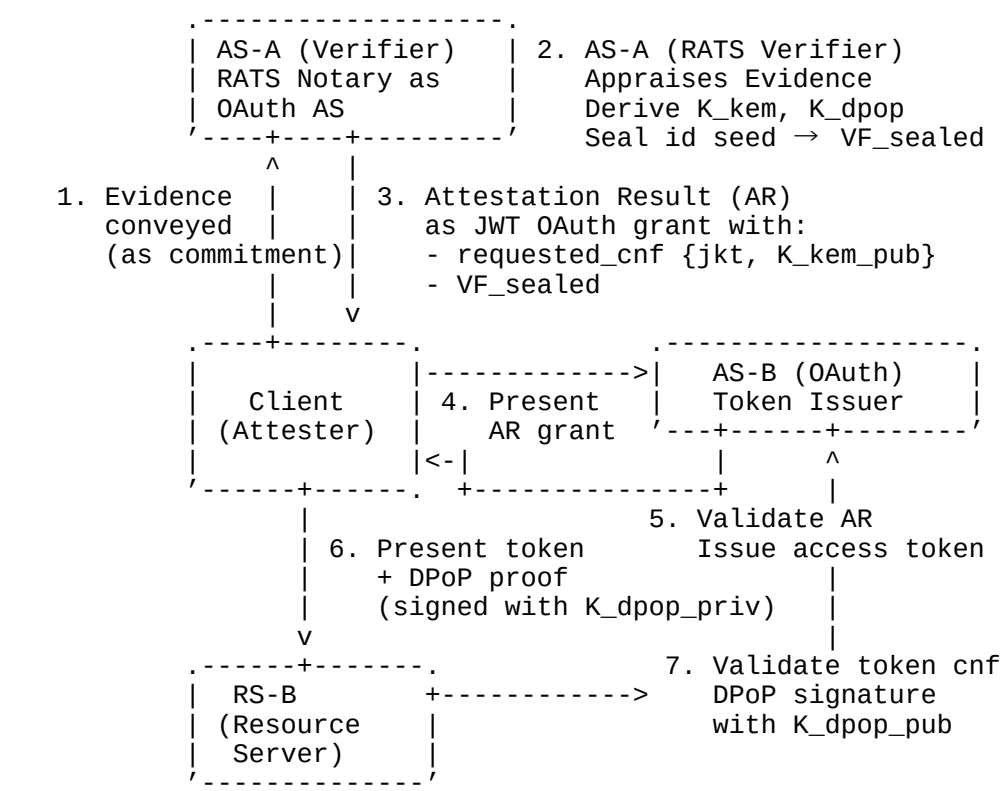

Figure 1: OAuth Identity Chaining Flow with RATS Verifier as AS-A and "IDPoP"-aware OAuth AS-B as delegated notary

_NOTE: K_kem controls access to VF (identity key seed material). K_attest *is* the identity. Compromise of the identity key alone is insufficient—attacker needs to demonstrate access to notorized Evidence (EF) to derive K_kem and unseal._

3. Algorithms

3.1. Algorithm 1: DeriveKEM (Client)

Input: PF, EF
Output: (K_kem_priv, K_kem_pub)

```
K_kem_priv = HKDF(salt=PF, ikm=EF, info="seal")
K_kem_pub  = X25519.Public(K_kem_priv)
```

3.1.1. Algorithm 2: EncapsulateSecret (AS-A)

```
   Input:  PF, K_kem_pub (from client attestation)
   Output: (VF_sealed, K_dpop_pub)

   VF = Random(32)
   VF_sealed = HPKE.Seal(recipient_pk=K_kem_pub, info=PF, plaintext=VF)

   K_dpop_priv = HKDF(salt=PF, ikm=VF, info="dpop")
   K_dpop_pub  = Ed25519.Public(K_dpop_priv)
```

3.1.2. Algorithm 3: DeriveIdentity (Client)

```
Input:  PF, K_kem_priv, VF_sealed
Output: K_dpop_priv

VF = HPKE.Unseal(recipient_sk=K_kem_priv, info=PF, ciphertext=VF_sealed)
K_dpop_priv = HKDF(salt=PF, ikm=VF, info="dpop")
```

Convergence: Client derives K_dpop_priv from (PF, VF) only after unsealing the encapsulated VF secret.

4. OAuth Identity Chaining Flow

Setup (Client → AS-A):

1. Client generates attestation evidence (EF)

2. Client: K_kem_priv = DeriveKEM(PF, EF)

3. Client sends attestation to AS-A with K_kem_pub

Notarization (RATS Verifier as AS-A):

1. AS-A Appraises attestation Evidence

2. AS-A: (VF_sealed, K_dpop_pub) = EncapsulateSeed(PF, K_kem_pub)

3. AS-A creates JWT grant to AS-B:

```
      {
        "iss": "https://as-a.example",
        "sub": "client_id",
        "aud": "https://as-b.example",
        "iat": 1706745600,
        "exp": 1706745660,
        "jti": "grant-7f3a9c",
        "requested_cnf": {
          "jkt": "0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I",
          "K_kem_pub": {
            "kty": "OKP",
            "crv": "X25519",
            "x": "l8tFrhx-34tV3hRICRDY9zCkDlpBhF42UQUfWVAWBFs"
          }
        }
      }
```

Token Issuance (AS-B):

1. AS-B validates grant, issues DPoP-bound access token:

```
      {
        "access_token": "eyJ...(JWT with cnf.jkt inside)...",
        "token_type": "DPoP",
        "expires_in": 3600
      }
```

The access token JWT contains:

```
      {
        "iss": "https://as-b.example",
        "sub": "client_id",
        "aud": "https://rs-b.example",
        "iat": 1706745600,
        "exp": 1706749200,
        "cnf": {
          "jkt": "0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I", // Validates the DPoP sig
nature (Standard)
          "K_kem_pub": { // Enables the Liveness Challenge (IDPoP Extension)
            "kty": "OKP",
            "crv": "X25519",
            "x": "l8tFrhx-34tV3hRICRDY9zCkDlpBhF42UQUfWVAWBFs"
          }
        }
      }
```

5. Interactive Challenge & Proofs

Interactive Challenge (AS-B → Client):

1. Client sends request with DPoP proof (no nonce or cached nonce)

2. AS-B decides: "I need hardware liveness proof"

3. AS-B → 401 Unauthorized

```
WWW-Authenticate: DPoP error="use_dpop_nonce"
DPoP-Nonce: <Base64Url(HPKE.AuthSeal(nonce, K_kem_pub, AS_B_privkey))>
```

Client Proof:

1. VF = HPKE.Unseal(K_kem_priv, VF_sealed) [proves fresh EF]

2. Client extracts sealed nonce from header

3. Client: nonce = HPKE.AuthUnseal(nonce_sealed, K_kem_priv, AS_B_pubkey)

4. Client: K_dpop_priv = HKDF(PF, VF, "dpop")

5. Client creates new DPoP proof with unsealed nonce and retries:

```
{
  "typ": "idpop+jwt", // Proposed new type
  "alg": "EdDSA",
  "jwk": { // The Public Identity Key (K_dpop)
    "kty": "OKP",
    "crv": "Ed25519",
    "x": "11qYAYKxCrfVS_7TyWQHOg7hcvPapiMlrwIaaPcHURo"
  }
}
{
  "jti": "e1j3V_bKic8-LAEB",
  "htm": "POST",
  "htu": "https://rs-b.example/api",
  "iat": 1706745600,
  "ath": "fUHyO2r2Z3DZ53EsNrWBb0xWXoaNy59IiKCAqksmQEo",
  "nonce": "kH8Ws3xYnE2f7d9pQ1vR5jL4mT6oU0aC8bN3gZ7yXsA"
}
```

Verification (AS-B or RS-B):

1. Verify DPoP signature with K_dpop_pub (from token’s cnf.jkt)

2. Verify proof.nonce == SHA256(expected_nonce)

3. Accept request

Note: The interactive challenge (Steps 8-10) introduces a round-trip. To mitigate latency, this challenge MAY be performed once per session or upon detecting high-risk context. The resulting nonce acts as a session binding signal for subsequent DPoP proofs, reducing overhead for high-frequency API calls.

6. Security Properties

1. *No durable secrets at AS-A.* Post-issuance compromise of AS-A yields no client key material. RATS Verifier (AS-A) computes K_dpop_priv transiently, extracts K_dpop_pub, discards both K_dpop_priv and secret seed material (VF).

2. *Sealed grant confidentiality.* Interception without K_kem_priv yields nothing. VF_sealed is HPKE-encrypted to K_kem_pub.

3. *Attestation-bound unsealing.* Stolen VF_sealed without device access is unusable. K_kem_priv = HKDF(PF, EF). Unsealing VF requires fresh hardware attestation.

4. *Key separation.* K_kem controls access. K_dpop asserts identity. Compromise of signing key doesn't grant unsealing; compromise of unsealing requires live EF.

5. *Convergent derivation.* No key transport. Client derives K_dpop from (PF, VF) independently.

7. Security Considerations

7.1. Authorization Server as Root of Trust

The architecture described in this document leverages the Authorization Server (AS-A in the RATS flow, or the OAuth AS) as a Trusted Third Party (TTP). This aligns with the standard OAuth 2.0 threat model [RFC6819], where the AS is already trusted to generate access tokens, sign ID tokens, and manage client credentials.

While IDPoP introduces the generation and sealing of the Identity Key seed (VF) by the AS, this does not introduce a new root of trust. An AS capable of issuing a valid access_token can already impersonate the user to the Resource Server. Therefore, entrusting the AS with the transient generation of the VF seed is consistent with its existing role. The security goal of IDPoP is not to protect the client _from_ the AS, but to protect the client's credentials from exfiltration _after_ issuance.

7.2. Encapsulated Secret Transport

This specification utilizes a Key Transport pattern where the Identity Key material (VF) is generated by the server and transmitted to the client in an HPKE-sealed envelope (VF_sealed). While Key Transport is sometimes disfavored in comparison to Key Agreement, it is necessary here to bind the identity to the specific hardware attestation presented by the client.

The security of this transport relies on *Hardware Binding*: The VF is sealed to K_kem_pub, which is derived from the client's hardware-bound Evidence Factor (EF). Even if VF_sealed is intercepted, it remains "inert" (unusable) without the corresponding hardware-bound private key K_kem_priv.

7.3. "Harvest Now, Decrypt Later" and Forward Secrecy

A prominent concern with any encrypted transport of secrets is the "Harvest Now, Decrypt Later" (HNDL) strategy, where an adversary records encrypted traffic today to decrypt it in the future once quantum computing breaks current asymmetric primitives (e.g., X25519).

In the context of IDPoP, the impact of HNDL is significantly mitigated by the *ephemeral nature of the Identity Key*:

* *Short-Lived Credential Utility:* The VF seed is used solely to derive the DPoP signing key (K_dpop). This key is only useful for signing DPoP proofs for the validity period of the associated access token (defined by exp).

* *Obsolescence upon Decryption:* If an adversary successfully breaks the KEM algorithm years in the future and recovers a historical VF, the associated access token will have long since expired. Unlike confidentiality keys used to encrypt persistent data (where future decryption is catastrophic), recovering an authentication key after the session window has closed yields no advantage, provided the AS enforces standard expiration (exp) and replay protection (jti).

7.3.1. Post-Quantum Agility

To further mitigate HNDL risks and ensure long-term resistance, implementations SHOULD support cryptographic agility in the HPKE configuration.

* *Hybrid KEMs:* Implementations can adopt hybrid KEMs such as *X-Wing* (combining X25519 and ML-KEM-768) to provide post-quantum resistance while maintaining classical security guarantees.

* *Algorithm Negotiation:* The alg parameter in the attestation request or metadata allows the client and AS to negotiate Quantum-Resistant KEMs (e.g., ML-KEM) as they become standardized, ensuring the VF_sealed remains secure against future quantum cryptanalysis.

7.4. DPoP Proof Replay and Pre-Computation

To prevent an attacker with temporary access to the signing key (or the ability to predict nonces) from pre-computing proofs, this specification mandates the use of HPKE-sealed nonces.

* *Liveness Requirement:* By sealing the nonce to the client’s K_kem_pub, the AS ensures that only a client with active access to the hardware-bound K_kem_priv can recover the nonce and sign the proof.

* *Mitigation of Malware:* Malware capable of stealing a cached K_dpop from memory cannot pre-generate valid proofs for future requests because it cannot decrypt the future nonces without invoking the hardware anchor.

8. Normative References

[I-D.ietf-oauth-identity-chaining]
Schwenkschuster, A., Kasselman, P., Burgin, K., Jenkins, M. J., and B. Campbell, "OAuth Identity and Authorization Chaining Across Domains", Work in Progress, Internet-Draft, draft-ietf-oauth-identity-chaining-06, 12 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-chaining-06>.

[RFC6819] Lodderstedt, T., Ed., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", RFC 6819, DOI 10.17487/RFC6819, January 2013, <https://www.rfc-editor.org/rfc/rfc6819>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

Acknowledgments

TODO

Author’s Address

Nathanael Ritz
Independent
Email: ietf@nritz.com