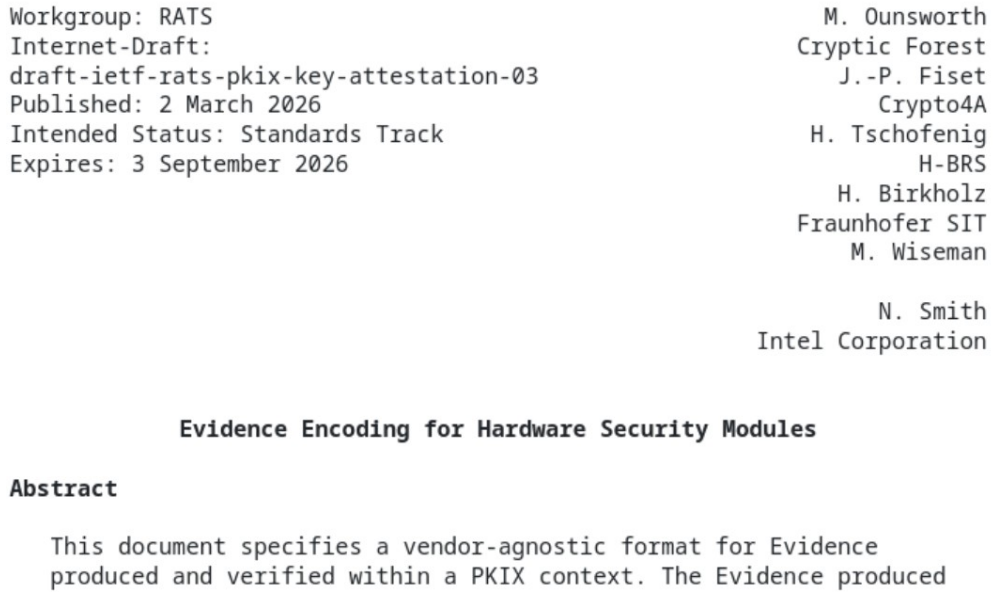
RATS PKIX Evidence Reference Implementations

IETF 125
14–15 March 2026
Shenzhen, China

Hackathon Plan

- The authors of this draft decided that we needed a reference implementation to produce test vectors.



Workgroup: RATS
Internet-Draft:
draft-ietf-rats-pkix-key-attestation-03
Published: 2 March 2026
Intended Status: Standards Track
Expires: 3 September 2026

M. Ounsworth
Cryptic Forest
J.-P. Fiset
Crypto4A
H. Tschofenig
H-BRS
H. Birkholz
Fraunhofer SIT
M. Wiseman

N. Smith
Intel Corporation

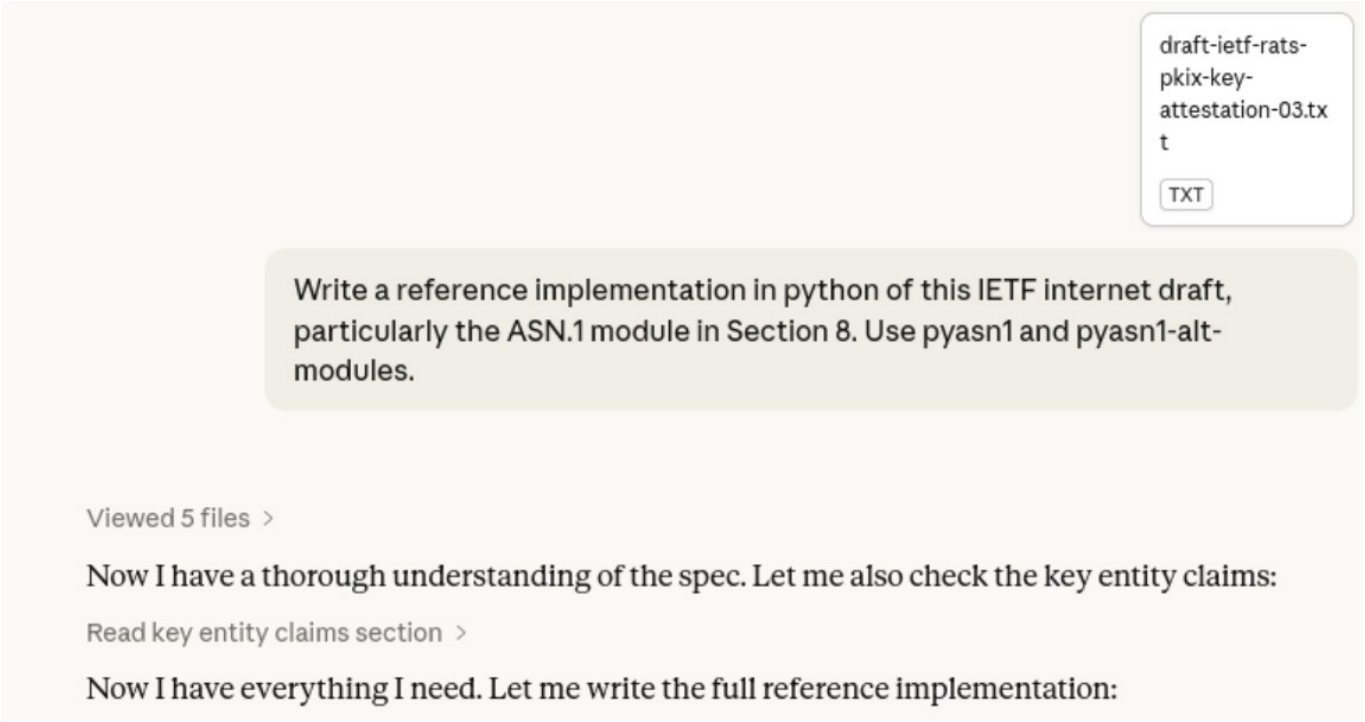
Evidence Encoding for Hardware Security Modules

Abstract

This document specifies a vendor-agnostic format for Evidence produced and verified within a PKIX context. The Evidence produced

What got done

- So … we fed the draft to Claude:



What got done

- The code came out really clean and almost entirely correct:

```
class TbsEvidence(univ.Sequence):
    """
    TbsEvidence ::= SEQUENCE {
        version          INTEGER,
        reportedEntities SEQUENCE SIZE (1..MAX) OF ReportedEntity
    }

    Per spec: version MUST be 1.
    """
    componentType = namedtype.NamedTypes(
        namedtype.NamedType("version", univ.Integer()),
        namedtype.NamedType("reportedEntities", ReportedEntitiesSeq()),
    )
```

```
    # ------------------------------------------------------------------
    # 5.  Assemble TbsEvidence and sign
    # ------------------------------------------------------------------
    tbs = build_tbs([platform_entity, key_entity, transaction_entity])
    signature, _ = sign_evidence_ec(tbs, ak)

    # OID for ecdsa-with-SHA256 (RFC 5758)
    ecdsa_with_sha256 = univ.ObjectIdentifier((1, 2, 840, 10045, 4, 3, 2))

    # Use key ID = SHA-1 of SPKI (common practice)
    key_id = hashlib.sha1(ak_spki_der).digest()
    sid = build_signer_id_keyid(key_id)
    sig_block = build_signature_block(sid, ecdsa_with_sha256, signature)
```

What got done

- 3 hours later: we had a working implementation generating sample data for the draft.
  (actually 2: one in Python and one in Go to cross-test)
- **This is a great way to check that your spec is clear and implementable** since the quality of the code is a direct reflection of the quality of your spec.
- The shiny new reference implementations live with the draft source:
  - https://github.com/ietf-rats-wg/key-attestation

Wrap Up

Team members:

Mike Ounsworth, JP Fiset, Hannes Tschofenig

Collaboration entirely over Gather