



# IETF 125 Shenzhen - RATS WG

Henk Birkholz

(Jean-Pierre Fiset, Mike Ounsworth, Hannes Tschofenig, Monty Wiseman,  
Ned Smith)

Evidence Encoding for Hardware Security Modules

<https://datatracker.ietf.org/doc/draft-ietf-rats-pkix-key-attestation/>



# Update since November 2025

- Title change
  - from “PKIX Evidence for Remote Attestation of HSMs”
  - to “Evidence Encoding for HSMs”
- Many changes based on discussion with Dave Thaler
  - Attestation Service -> Attesting Environment
  - Renamed “attribute” to “claim”
- Modified diagram that introduces Presenter role
- General care and maintenance
  - 10 opened issues; 34 closed issues
- We are continuing bi-weekly meetings inviting those who are interested
- Hackathon 125: develop test vectors

# Hackathon 125

- During the Hackathon, developed sample data
  - Python code was generated from feeding draft specification to Claude (AI)
  - Some corrections had to be done manually, but they were minor
- Evidence verifier was written in goLang
  - Developed as well using “cobe vibing”
- We now have test vectors that are included in annex to the specification

# Hackathon 125 - Glimpse into Process



```
draft-ietf-rats-  
pkix-key-  
attestation-03.tx  
t
```

TXT

Write a reference implementation in python of this IETF internet draft, particularly the ASN.1 module in Section 8. Use pyasn1 and pyasn1-alt-modules.

Viewed 5 files >

Now I have a thorough understanding of the spec. Let me also check the key entity claims:

[Read key entity claims section >](#)

Now I have everything I need. Let me write the full reference implementation:

# Hackathon 125 - Glimpse into Process (cont'd)

The code came out really clean and almost entirely correct:

```
259 class TbsEvidence(univ.Sequence):
260     """
261     TbsEvidence ::= SEQUENCE {
262         version          INTEGER,
263         reportedEntities SEQUENCE SIZE (1..MAX) OF ReportedEntity
264     }
265
266     Per spec: version MUST be 1.
267     """
268     componentType = namedtype.NamedTypes(
269         namedtype.NamedType("version", univ.Integer()),
270         namedtype.NamedType("reportedEntities", ReportedEntitiesSeq()),
271     )
272
273
274
275
276
277
278
279
280
281
282
283
284 # -----
285 # 5. Assemble TbsEvidence and sign
286 # -----
287 tbs = build_tbs([platform_entity, key_entity, transaction_entity])
288 signature, _ = sign_evidence_ec(tbs, ak)
289
290 # OID for ecdsa-with-SHA256 (RFC 5758)
291 ecdsa_with_sha256 = univ.ObjectIdentifier((1, 2, 840, 10045, 4, 3, 2))
292
293 # Use key ID = SHA-1 of SPKI (common practice)
294 key_id = hashlib.sha1(ak_spki_der).digest()
295 sid = build_signer_id_keyid(key_id)
296 sig_block = build_signature_block(sid, ecdsa_with_sha256, signature)
297
```

# Hackathon 125 - We now have auto-generating Samples in the draft

## Appendix A. Samples

A reference implementation of this specification can be found at <https://github.com/ietf-rats-wg/key-attestation>

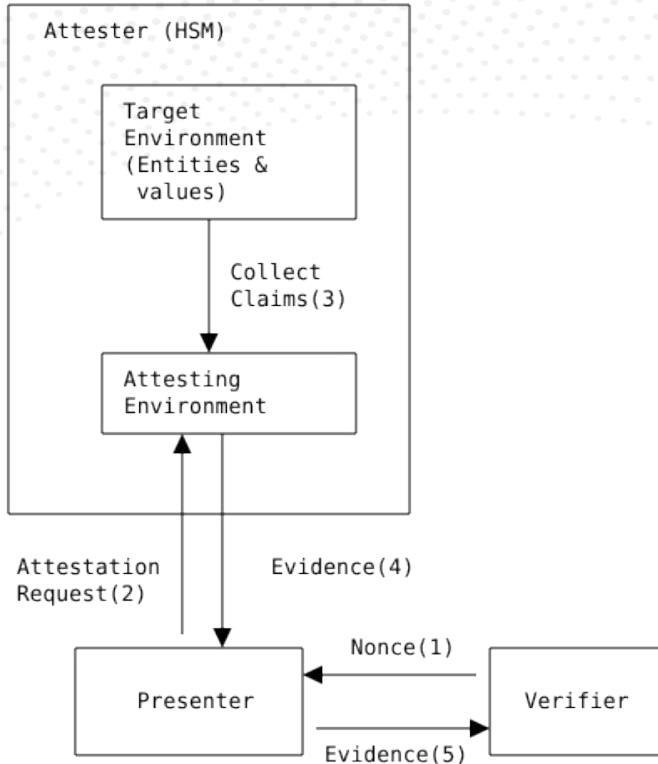
It produces the following sample Evidence:

```
-----BEGIN EVIDENCE-----
MIIGHDCCaICAQEwggGbmIGkBgYqA4dnAAAwgZkwEwYHKg0HZI
GgYHKg0HZwEAAYMPMjAyNTAzMTQxMjAwMdBaMGYGByoDh2cBA
AgEGCCqGSM49AwEHA0IABCEp/+t1k8NtMyyZTX0x6bDb1c2p
gI3Ac6cI7vqNVm84aAKVyG69wHB1RXWlqR0N/CIwCWYgKg0HZI
AIEJQWntZSDB3JwMBMGByoDh2cBAQKACEhTTS05MDAwMBAGB
MAwGByoDh2cBAQuCAf8wDAYHKg0HZwEBDYQBAzA0BgcqA4dnA
ZwACMHMwEgYHKg0HZwECAIEHa2V5LTAwMTAMBgcqA4dnAQICg
Af8wDAYHKg0HZwECA4IB/zAMBgcqA4dnAQIFggH/MCUGByoDh
AgQGBioDh2cCBgYgKg0HZwIIMIICfjCCAnowggIhoBYEFI1Fd
tbbIooICBTCCAgEwggGooAMCAQICFFCeay6JyU4vr+w9yZTK2
BAMCMEMxEjAQBqNVBAoMCWllldGYtcmF0czEdMBSGA1UECwwUc
```

This sample data embeds the AK and Intermediate CA certificate, but a Verifier will require the root CA certificate to verify the signature.

```
-----BEGIN CERTIFICATE-----
MIIB7TCCAZ0gAwIBAgIUJ7ZeDJCpIUX0NZyTsEf4yJpKuIkwCgYIKoZIzj0EAwIw
RDESMBAGA1UECgwJalV0Zi1yYXRzMR0wGwYDVQQwLDBRwa2l4LWtleS1hdHRlc3Rh
dGlvbWJEPMA0GA1UEAwGUm9vdENBMB4XDTI2MDMxNTA1MzI0MloXDTM2MDMxMjA1
Mz00MlowRDESMBAGA1UECgwJalV0Zi1yYXRzMR0wGwYDVQQwLDBRwa2l4LWtleS1h
dHRlc3RhZGlvdWJEPMA0GA1UEAwGUm9vdENBMBFkwEwYHKoZIzj0CAQYIKoZIzj0D
AQcDQgAEb1Ro6rb2fpcoSDt9e207EbZKP4z0fqZuHIN4HujPr/98zj+jiq/x51JR
hyfNfBB84Nag7u4X1KD0xQc+ha99PKNjMGEwDwYDVR0TAQH/BAUwAwEB/zAdBgNV
HQ4EFgQUyYn1sFj3x1wIpZD3nFBMkhGCXwwwHwYDVR0jBBgwFoAUyYn1sFj3x1wI
pZD3nFBMkhGCXwwwDgYDVR0PAQH/BAQDAgKEMAoGCCqGSM49BAMCA0gAMEUCIQDq
i8a/ODw7ZFtT2FAuhhihgbe/QRkbgHg4ioQNU6MegIqVvu6zI44i3Jqs3M5BS3
Ticc+cNRtFZGxcwccgJwQxk=
-----END CERTIFICATE-----
```

# Diagram introducing Presenter



- This diagram was modified to better represent the concept in RATS
- “Verifier” role was added
- “Nonce” generated by Verifier

# Information Structure

- Target Environment is portioned into “entities”
  - Platform
  - Keys
  - Transaction
- Each entity is a collection of claims
  - A claim has a type and a value
- With the development of test vectors, the structure has solidified and is less likely to change.

# Challenges

- **ASN.1 ANY**
  - The current revision of the draft uses a CHOICE for claim value
  - ANY defined by claimType would be desired
  - ANY has fallen out of favour with IETF specifications
- **ASN.1 Class**
  - Would be an alternative
  - No open tool yet available

```
ReportedClaim ::= SEQUENCE {  
    claimType      OBJECT IDENTIFIER,  
    value          ClaimValue OPTIONAL  
}
```

```
ClaimValue ::= CHOICE {  
    bytes          [0] IMPLICIT OCTET STRING,  
    utf8String    [1] IMPLICIT UTF8String,  
    bool           [2] IMPLICIT BOOLEAN,  
    time          [3] IMPLICIT GeneralizedTime,  
    int           [4] IMPLICIT INTEGER,  
    oid           [5] IMPLICIT OBJECT IDENTIFIER  
}
```

# Upcoming Work

- Add entity type for User Modules (usermods)
  - Look into EAT and other specifications
- Complete Extended Key Usage for X.509 (dependency)
  - draft-jpfi-set-lamps-attestationkey-eku
- Complete a functional implementation
  - An effort was started at Hackathon 125 to generate reference vectors
  - Start modifying existing implementations to align with this specification
- Reach WGLC

# References

Datatracker:

<https://datatracker.ietf.org/doc/draft-ietf-rats-pkix-key-attestation/>

GitHub:

<https://github.com/ietf-rats-wg/key-attestation>