Remote ATtestation ProcedureS
Internet-Draft
Intended status: Standards Track
Expires: 10 October 2025

F. D'Amato
AMD
A. Draper
Altera
N. Smith
Intel Corporation

8 April 2025

Evidence Transformations
draft-ietf-rats-evidence-trans-01

Abstract

Remote Attestation Procedures (RATS) enable Relying Parties to assess the trustworthiness of a remote Attester to decide if continued interaction is warrented. Evidence structures can vary making appraisals challenging for Verifiers. Verifiers need to understand Evidence encoding formats and some of the Evidence semantics to appraise it. Consequently, Evidence may require format transformation to an internal representation that preserves original semantics. This document specifies Evidence transformation methods for DiceTcbInfo, concise evidence, and SPDM measurements block structures. These Evidence structures are converted to the CoRIM internal representation and follow CoRIM defined appraisal procedures.

Table of Contents

1. Introduction

Remote Attestation Procedures (RATS) [RFC9334] enable Relying Parties to assess the trustworthiness of a remote Attester to decide if continued interaction is warrented. Evidence structures can vary making appraisals challenging for Verifiers. Verifiers need to understand Evidence encoding formats and some of the Evidence semantics to appraise it. Consequently, Evidence may require format transformation to an internal representation that preserves original semantics. This document specifies Evidence transformation methods

for DiceTcbInfo [DICE.Attest], concise evidence [TCG.CE], and SPDM measurements block [SPDM] structures. These Evidence structures are converted to the CoRIM internal representation (Section 2.1 [I-D.ietf-rats-corim]) and follow CoRIM defined appraisal procedures (Section 8 [I-D.ietf-rats-corim]).

1.1. Terminology

This document uses terms and concepts defined by the RATS architecture. For a complete glossary. See Section 4 of [RFC9334]. Addintional RATS architecture and terminology is found in [I-D.ietf-rats-endorsements]. RATS architecture terms and concepts are always referenced as proper nouns, i.e., with Capital Letters. Additional terminology from CoRIM [I-D.ietf-rats-corim], [DICE.CoRIM], CBOR [STD94], CDDL [RFC8610] and COSE [STD96] may also apply.

In this document, Evidence structures are expressed in their respective "external representations". There are many possible Evidence structures including those mentioned above.

The CoRIM specification defines an "internal representation" for Evidence (Section 8.2.1.3 [I-D.ietf-rats-corim]). This document defines mapping operations that convert from an external representation to an internal representation. The conversion steps are also known as "transformation".

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Verifier Reconciliation

This document assumes the reader is familiar with Verifier reconciliation as described in Section 2 of [I-D.ietf-rats-corim]. It describes how a Verifier should process the CoRIM to enable CoRIM authors to convey their intended meaning and how a Verifier reconciles its various inputs. Evidence is one of its inputs. The Verifier is expected to create an internal representation from an external representation. By using an internal representation, the Verifier processes Evidence inputs such that they can be appraised consistently.

This document defines format transformations for Evidence in DICE [DICE.Attest], SPDM [SPDM], and concise evidence [TCG.CE] formats that are transformed into a Verifier’s internal representation. This

document uses the CoMID internal representation (Section 8.2.1 of [I-D.ietf-rats-corim]) as the transformation target. Other internal representations are possible but out of scope for this document.

3. Transforming DICE Certificate Extension Evidence

This section defines how Evidence from an X.509 certificate [RFC5280] containing a DICE certificate extension [DICE.Attest] is transformed into an internal representation that can be processed by Verifiers.

Verifiers supporting the DICE certificate Evidence extensions SHOULD implement this transformation.

3.1. DiceTcbInfo Transformation

This section defines transformation methods for DICE certificate extensions DiceTcbInfo, DiceMultiTcbInfo, and DiceMultiTcbInfoComp.

These extensions are identified by the following object identifiers:

* tcg-dice-TcbInfo - "2.23.133.5.4.1"

* tcg-dice-MultiTcbInfo - "2.23.133.5.4.5"

* tcg-dice-MultiTcbInfoComp - "2.23.133.5.4.8"

Each DiceTcbInfo entry in a MultiTcbInfo is converted to a CoRIM ECT (see Section 8.2.1 of [I-D.ietf-rats-corim]) using the transformation steps in this section. Each DiceMultiTcbInfo entry is independent of the others such that each is transformed to a separate ECT entry. A list of Evidence ECTs (i.e., ae = [+ ECT]) is constructed using CoRIM attestation evidence internal representation (see Section 8.2.1.1 of [I-D.ietf-rats-corim]). Each DiceMultiTcbInfoComp entry is converted to a DiceMultiTcbInfo entry then processed as a DiceMultiTcbInfo.

For each DiceTcbInfo (DTI) entry in a DiceMultiTcbInfo allocate an ECT structure.

Step 1. An ae entry is allocated.

Step 2. The cmtype of the ECT is set to evidence.

Step 3. The DiceTcbInfo (DTI) entry populates the ae ECT.

i The DTI entry populates the ae ECT environment-map

copy(DTI.type, ECT.environment.environment-map.class-map.class-id). The binary representation of DTI.type MUST be equivalent to the binary representation of class-id without the CBOR tag.

copy(DTI.vendor, ECT.environment.environment-map.class-map.vendor).

copy(DTI.model, ECT.environment.environment-map.class-map.model).

copy(DTI.layer, ECT.environment.environment-map.class-map.layer).

copy(DTI.index, ECT.environment.environment-map.class-map.index).

ii The DTI entry populates the ae ECT elemenet-list.

copy(DTI.version, ECT.element-list.element-map.measurement-values-map.version-map.version).

copy(DTI.svn, ECT.element-list.element-map.measurement-values-map.svn).

copy(DTI.vendorInfo, ECT.element-list.element-map.measurement-values-map.raw-value).

Foreach FWID in FWIDLIST: *copy*(DTI.FWID.digest, ECT.element-list.element-map.measurement-values-map.digests.digest.val).

Foreach FWID in FWIDLIST: *copy*(DTI.FWID.hashAlg, ECT.element-list.element-map.measurement-values-map.digests.digest.alg).

iiiThe DTI entry populates the ae ECT elemenet-list.flags. Foreach _f_ in DTI.OperationalFlags and each _m_ in DTI.OperationalFlagsMask:

If _m_.notConfigured = 1 AND _f_.notConfigured = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-configured = FALSE).

If _m_.notConfigured = 1 AND _f_.notConfigured = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-configured = TRUE).

If _m_.notSecure = 1 AND _f_.notSecure = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-secure = FALSE).

If _m_.notSecure = 1 AND _f_.notSecure = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-secure = TRUE).

If _m_.recovery = 1 AND _f_.recovery = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-recovery = FALSE).

If _m_.recovery = 1 AND _f_.recovery = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-recovery = TRUE).

If _m_.debug = 1 AND _f_.debug = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-debug = FALSE).

If _m_.debug = 1 AND _f_.debug = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-debug = TRUE).

If _m_.notReplayProtected = 1 AND _f_.notReplayProtected = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-replay-protected = FALSE).

If _m_.notReplayProtected = 1 AND _f_.notReplayProtected = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-replay-protected = TRUE).

If _m_.notIntegrityProtected = 1 AND _f_.notIntegrityProtected = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-integrity-proteccted = FALSE).

If _m_.notIntegrityProtected = 1 AND _f_.notIntegrityProtected = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-integrity-proteccted = TRUE).

If _m_.notRuntimeMeasured = 1 AND _f_.notRuntimeMeasured = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-runtime-meas = FALSE).

If _m_.notRuntimeMeasured = 1 AND _f_.notRuntimeMeasured = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-runtime-meas = TRUE).

If _m_.notImmutable = 1 AND _f_.notImmutable = 1;
set(ECT.element-list.element-map.measurement-values-map.flags.is-immutable = FALSE).

If _m_.notImmutable = 1 AND _f_.notImmutable = 0;
set(ECT.element-list.element-map.measurement-values-map.flags.is-immutable = TRUE).

If _m_.notTcb = 1 AND _f_.notTcb = 1; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-tcb = FALSE).

If _m_.notTcb = 1 AND _f_.notTcb = 0; *set*(ECT.element-list.element-map.measurement-values-map.flags.is-tcb = TRUE).

Step 4. The ECT.authority field is set up based on the signer of the certificate containing DTI as described in Section 3.4.

The completed ECT is added to the ae list.

3.2. DiceUeid Transformation

This section defines the transformation method for the DiceUeid certificate extension. This extension is identified by the following object identifier:

* tcg-dice-Ueid - "2.23.133.5.4.4"

Step 1. An ae entry is allocated.

Step 2. The cmtype of the ECT is set to evidence.

Step 3. The DiceUeid entry populates the ae ECT environment-map.instance-id.tagged-ueid-type. The CBOR tag #6.550 is prepended to the DiceUeid OCTET STRING then copied to ae.environment-map.instance-id.

Step 4. The ECT.authority field is set up based on the signer of the certificate containing DiceUeid as described in Section 3.4.

The completed ECT is added to the ae list.

3.3. DiceConceptualMessageWrapper Transformation

This section defines the transformation method for the DiceConceptualMessageWrapper certificate extension. This extension is identified by the following object identifier:

* tcg-dice-Ueid - "2.23.133.5.4.9"

The DiceConceptualMessageWrapper entry OCTET STRING may contain a CBOR array, JSON array, or CBOR tagged value. If the entry contains a CBOR tag value of #6.571 or #6.1668557429, or a Content ID of 10571, or a Media Type of "application/ce+cbor", the contents are transformed according to Section 4.

3.4. Authority field in DICE/SPDM ECTs

The ECT authority field is an array of $crypto-keys-type-choice values.

When adding Evidence to the ACS, the Verifier SHALL add the public key representing the signer of that Evidence (for example the DICE certificate or SPDM MEASUREMENTS response) to the ECT authority field. The Verifier SHALL add the authority of the signers of each certificate in the certificate path of the end-entity signing key to the ECT authority list. Having each authority in a certificate path in the ECT authority field lets conditional endorsement conditions match multiple authorities or match an authority that is scoped more broadly than the immediate signer of the Evidence artifact.

Each signer authority value MUST be represented using tagged-cose-key-type.

4. Transforming TCG Concise Evidence

This section defines how Evidence from TCG [TCG.CE] is transformed into an internal representation that can be processed by Verifiers.

Verifiers supporting the TCG Concise Evidence format SHOULD implement this transformation.

Concise evidence may be recognized by any of the following registered types:

CBOR tag	C-F ID	TN Tag	Media Type
#6.571	10571	#6.1668557429	"application/ce+cbor"

Table 1

A Concise Evidence entry is converted to a CoRIM ECT (see Section 8.2.1 of [I-D.ietf-rats-corim]) using the transformation steps in this section. A list of Evidence ECTs (i.e., ae = [+ ECT]) is constructed using CoRIM attestation evidence internal representation (see Section 8.2.1.1 of [I-D.ietf-rats-corim]). The

Concise Evidence scheme uses CoRIM CDDL definitions to define several Evidence representations called _triples_. Cases where Concise Evidence CDDL is identical to CoRIM CDDL the transformation logic uses the structure names in common.

4.1. Transforming the ce.evidence-triples

The ce.evidence-triples structure is a list of evidence-triple-record. An evidence-triple-record consists of an environment-map and a list of measurement-map. For each evidence-triple-record an ae ECT is constructed.

Step 1. An ae ECT entry is allocated.

Step 2. The cmtype of the ECT is set to evidence.

Step 3. The Concise Evidence (CE) entry populates the ae ECT environment fields.

copy(CE.evidence-triple-record.environment-map, ECT.environment.environment-map).

i For each ce in CE.[+ measurement-map]; and each ect in ECT.[+ element-list]:

copy(ce.mkey, ect.element-map.element-id)

copy(ce.mval, ect.element-map.element-claims‘)

Step 4. The signer of the envelope containing CE is copied to the ECT.authority field as described in Section 3.4. For example, a CE may be wrapped by an EAT token [I-D.ietf-rats-eat] or DICE certificate [DICE.Attest]. The signer identity MUST be expressed using $crypto-key-type-choice. A profile or other arrangement is used to coordinate which $crypto-key-type-choice is used for both Evidence and Reference Values.

Step 5. If CE has a profile, the profile is converted to a $profile-type-choice then copied to the ECT.profile‘ field.

The completed ECT is added to the ae list.

4.2. Transforming the ce.identity-triples

The ce.identity-triples structure is a list of ev-identity-triple-record. An ev-identity-triple-record consists of an environment-map and a list of $crypto-key-type-choice. For each ev-identity-triple-record an ae ECT is constructed where the $crypto-key-type-choice values are copied as ECT Evidence measurement values. The ECT internal representation accommodates keys as a type of measurement. In order for the $crypto-key-type-choice keys to be verified a CoRIM identity-triples claim MUST be asserted.

Step 1. An ae ECT entry is allocated.

Step 2. The cmtype of the ECT is set to evidence.

Step 3. The Concise Evidence (CE) entry populates the ae ECT environment fields.

copy(CE.ce-identity-triple-record.environment-map, ECT.environment.environment-map).

copy(_null_, ECT.element-list.element-map.element-id).

i For each cek in CE.[+ $crypto-key-type-choice]; and each ect in ECT.element-list.element-map.element-claims.intrep-keys.[+ typed-crypto-key]:

copy(cek, ect.key)

set(&(identity-key: 1), ect.key-type)

Step 4. The signer of the envelope containing CE is copied to the ECT.authority field. For example, a CE may be wrapped by an EAT token [I-D.ietf-rats-eat] or DICE certificate [DICE.Attest]. The signer identity MUST be expressed using $crypto-key-type-choice. A profile or other arrangement is used to coordinate which $crypto-key-type-choice is used for both Evidence and Reference Values.

Step 5. If CE has a profile, the profile is converted to a $profile-type-choice then copied to the ECT.profile’ field.

The completed ECT is added to the ae list.

4.3. Transforming the ce.attest-key-triples

The ce.attest-key-triples structure is a list of ev-attest-key-triple-record. An ev-attest-key-triple-record consists of an environment-map and a list of $crypto-key-type-choice. For each ev-attest-key-triple-record an ae ECT is constructed where the $crypto-key-type-choice values are copied as ECT Evidence measurement values. The ECT internal representation accommodates keys as a type of measurement. In order for the $crypto-key-type-choice keys to be verified a CoRIM attest-key-triples claim MUST be asserted.

Step 1. An ae ECT entry is allocated.

Step 2. The cmtype of the ECT is set to evidence.

Step 3. The Concise Evidence (CE) entry populates the ae ECT environment fields.

copy(CE.ce-attest-key-triple-record.environment-map, ECT.environment.environment-map).

copy(_null_, ECT.element-list.element-map.element-id).

i For each cek in CE.[+ $crypto-key-type-choice]; and each ect in ECT.element-list.element-map.element-claims.intrep-keys.[+ typed-crypto-key]:

copy(cek, ect.key)

set(&(attest-key: 0), ect.key-type)

Step 4. The signer of the envelope containing CE is copied to the ECT.authority field. For example, a CE may be wrapped by an EAT token [I-D.ietf-rats-eat] or DICE certificate [DICE.Attest]. The signer identity MUST be expressed using $crypto-key-type-choice. A profile or other arrangement is used to coordinate which $crypto-key-type-choice is used for both Evidence and Reference Values.

Step 5. If CE has a profile, the profile is converted to a $profile-type-choice then copied to the ECT.profile‘ field.

The completed ECT is added to the ae list.

5. DMTF SPDM Structure Definitons

This section defines how a Verifier shall parse a DMTF Measurement Block.

DMTF Measurement Block Definition:

* Byte Offset 0: DMTFSpecMeasurementValueType
  - Bit 7 = 0b Digest / 1b Raw bit stream
  - Bit [6:0] = Indicate what is measured
    - 0x0 Immutable Rom
    - 0x1 Mutable FW
    - 0x2 HW Config
    - 0x3 FW config
    - 0x4 Freeform Manifest
    - 0x5 Structured Representation of debug and device mode
    - 0x6 Mutable FW Version Number
    - 0x7 Mutable FW Secure Version Number
    - 0x8 Hash-Extend Measurement (new in SPDM 1.3)
    - 0x9 Informational (new in SPDM 1.3)
    - 0xA Structured Measurement Manifest (new in SPDM 1.3)
* Byte Offset 1: DMTFSpecMeasurementValueSize
* Byte Offset 3: DMTFSpecMeasurementValue

Structured Manifest Block Definition (only for >=SPDM 1.3):

* Byte Offset 0: Standard Body or Vendor Defined Header (SVH)
* Byte Offset 2 + VendorIdLen: Manifest

Standard Body or Vendor Defined Header (SVH) Definition (only for >=SPDM 1.3):

* Byte Offset 0: ID
* Byte Offset 1: VendorIdLen
* Byte Offset 2: VendorId

DMTF Header for Concise Evidence Manifest Block:

If SPDM Version 1.2:

* DMTFSpecMeasurementValueType = 0x84 (Raw Bit / Freeform Manifest)

* DMTFSpecMeasurementValueSize = Size of tagged-spdm-toc CBOR Tag

* DMTFSpecMeasurementValue = tagged-spdm-toc CBOR Tag

if SPDM >=Version 1.3:

* DMTFSpecMeasurementValueType = 0x8A (Raw Bit / Structured Manifest)

* DMTFSpecMeasurementValueSize = Size of Structured Manifest

* DMTFSpecMeasurementValue = Structured Manifest

SVH for Concise Evidence Manifest Block:

* ID = 0xA (IANA CBOR)

* VendorIdLen = 2

* VendorId = 0x570 #6.570(spdm-toc-map)

Structured Manifest Block Definition for Concise Evidence:

* SVH = SVH for Concise Evidence Manifest Block

* Manifest = tagged-spdm-toc CBOR Tag Payload

DMTF Header for CBor Web Token (CWT):

If SPDM Version 1.2:

* DMTFSpecMeasurementValueType = 0x84 (Raw Bit / Freeform Manifest)

* DMTFSpecMeasurementValueSize = Size of CWT

* DMTFSpecMeasurementValue = CWT # COSE_Sign1

if SPDM = Version 1.3:

* DMTFSpecMeasurementValueType = 0x8A (Raw Bit / Structured Manifest)

* DMTFSpecMeasurementValueSize = Size of Structured Manifest

* DMTFSpecMeasurementValue = Structured Manifest

SVH for CBor Web Token (CWT):

* ID = 0xA (IANA CBOR)

* VendorIdLen = 2

* VendorId = 0x18 #6.18

Structured Manifest Block Definition for CBor Web Token (CWT):

* SVH = SVH for CBor Web Token (CWT)

* Manifest = COSE_Sign1 Payload

6. Transforming SPDM Measurement Block Digest

Step 1. if DMTFSpecMeasurementValueType is in range [0x80 - 0x83]:

copy(SPDM.MeasurementBlock.DMTFSpecMeasurementValue , ECT.environment.measurement-map.mval.digests).

Step 2. if DMTFSpecMeasurementValueType is in range [0x88]:

copy(SPDM.MeasurementBlock.DMTFSpecMeasurementValue , ECT.environment.measurement-map.mval.integrity-registers).

7. Transforming SPDM Measurement Block Raw Value

Step 1. if DMTFSpecMeasurementValueType is in range [0x7]:

copy(SPDM.MeasurementBlock.DMTFSpecMeasurementValue , ECT.environment.measurement-map.mval.svn).

8. Transforming SPDM RATS EAT CWT

The RATS EAT CWT shall be reported in any of the assigned Measurement Blocks range [0xF0 - 0xFC] The Concise Evidence CBOR Tag is serialized inside eat-measurements (273) claim ($measurements-body-cbor /= bytes .cbor concise-evidence-map) Subsequently the transformation steps defined in Section 4.

9. Transforming SPDM Evidence

This section defines how Evidence from SPDM [SPDM] is transformed into an internal representation that can be processed by Verifiers.

Verifiers supporting the SPDM Evidence format SHOULD implement this transformation. SPDM Responders SHALL support a minimum version of 1.2

Theory of Operations:

* The SPDM Requestor SHALL retrieve the measurement Manifest at Block 0xFD (Manifest Block) and send its payload to the Verifier
  - The Verifier SHALL decode the payload as a tagged-spdm-toc CBOR tag.
  - The Verifier SHALL extract the tagged-concise-evidence CBOR TAG from the tagged-spdm-toc CBOR tag

Theconcise-evidence has a format that is similar to CoRIM triples-map (their semantics follows the matching rules described above).

* For every spdm-indirect measurement the Verifier shall ask the SPDM Requestor to retrieve the measurement block indicated by the index
  - if the index is in range [0x1 - 0xEF] (refer to #Transforming SPDM Measurement Block Digest)
  - if the index is in range [0xF0 - 0xFC] (refer to #Transforming SPDM RATS EAT CWT])

The TCG DICE Concise Evidence Binding for SPDM specification [TCG.CE] describes a process for converting the SPDM Measurement Block to Concise Evidence. Subsequently the transformation steps defined in Section 4.

The keys provided in the ECT.authority field SHOULD include the key which signed the SPDM MEASUREMENTS response carrying the Evidence and keys which authorized that key as described in Section 3.4.’’’

10. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalogue of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as Evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

11. Security and Privacy Considerations

Evidence appraisal is at the core of any RATS protocol flow, mediating all interactions between Attesters and their Relying Parties. The Verifier is effectively part of the Attesters' and Relying Parties' trusted computing base (TCB). Any mistake in the appraisal process could have security implications. For instance, it could lead to the subversion of an access control function, which creates a chance for privilege escalation.

Therefore, the Verifier' s code and configuration, especially those of the CoRIM processor, are primary security assets that must be built and maintained as securely as possible.

The protection of both the Attester and Verifier systems should be considered throughout their entire lifecycle, from design to operation. This includes the following aspects:

* Minimizing implementation complexity (see also Section 6.1 of [I-D.ietf-rats-endorsements]);

* Using memory-safe programming languages;

* Using secure defaults;

* Minimizing the attack surface by avoiding unnecessary features that could be exploited by attackers;

* Applying the principle of least privilege to the system's users;

* Minimizing the potential impact of security breaches by implementing separation of duties in both the software and operational architecture;

* Conducting regular, automated audits and reviews of the system, such as ensuring that users' privileges are correctly configured and that any new code has been audited and approved by independent parties;

* Failing securely in the event of errors to avoid compromising the security of the system.

The appraisal process should be auditable and reproducible. The integrity of the code and data during execution should be made an explicit objective, for example ensuring that the appraisal functions are computed in an attestable trusted execution environment (TEE).

The integrity of public and private key material and the secrecy of private key material must be ensured at all times. This includes key material carried in attestation key triples and key material used to assert or verify the authority of triples (such as public keys that identify trusted supply chain actors). For more detailed information on protecting Trust Anchors, refer to Section 12.4 of [RFC9334].

The Verifier should use cryptographically protected, mutually authenticated secure channels to all its trusted input sources (i.e., Attesters, Endorsers, RVPs, Verifier Owners). The Attester should use cryptographically protected, mutually authenticated secure channels to all its trusted input sources (i.e., Verifiers, Relying Parties). These links must reach as deep as possible - possibly terminating within the Attesting Environment of an Attester or within the appraisal session context of a Verifier - to avoid man-in-the-middle attacks. Also consider minimizing the use of intermediaries: each intermediary becomes another party that needs to be trusted and therefore factored in the Attesters and Relying Parties' TCBs. Refer to Section 12.2 of [RFC9334] for information on Conceptual Messages protection.

12. IANA Considerations

There are no IANA considerations.

13. References

13.1. Normative References

[DICE.Attest]
Trusted Computing Group (TCG), "DICE Attestation Architecture", Version 1.2, Revision 1 , January 2025, <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-Version-1.2-rc-1_9January25.pdf>.

[DICE.CoRIM]
Trusted Computing Group (TCG), "DICE Endorsement Architecture for Devices", Version 1.0, Revision 0.38 , November 2022, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-Endorsement-Architecture-for-Devices-V1-R38_pub.pdf>.

[I-D.ietf-rats-corim]
Birkholz, H., Fossati, T., Deshpande, Y., Smith, N., and W. Pan, "Concise Reference Integrity Manifest", Work in Progress, Internet-Draft, draft-ietf-rats-corim-07, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-corim-07>.

[I-D.ietf-rats-endorsements]
Thaler, D., Birkholz, H., and T. Fossati, "RATS Endorsements", Work in Progress, Internet-Draft, draft-ietf-rats-endorsements-06, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-endorsements-06>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[SPDM] Distributed Management Task Force, "Security Protocol and Data Model (SPDM)", Version 1.3.0 , May 2023, <https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.3.0.pdf>.

[TCG.CE] Trusted Computing Group, "TCG DICE Concise Evidence Binding for SPDM", Version 1.00, Revision 0.54 , January 2024, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Concise-Evidence-Binding-for-SPDM-Version-1.0-Revision-54_pub.pdf>.

13.2. Informative References

[I-D.ietf-rats-eat] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-31, 6 September 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-31>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[STD96] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

Contributors

The authors would like to thank the following people for their valuable contributions to the specification.

Henk Birkholz

Email: henk.birkholz@ietf.contact

Yogesh Deshpande

Email: yogesh.deshpande@arm.com

Thomas Fossati

Email: Thomas.Fossati@linaro.org

Dionna Glaze

Email: dionnaglaze@google.com

Acknowledgments

The authors would like to thank James D. Beaney, Francisco J. Chinchilla, Vincent R. Scarlata, and Piotr Zmijewski for review feedback.

Authors' Addresses

Fabrizio D'Amato
AMD
Email: fabrizio.damato@amd.com

Andrew Draper
Altera
Email: andrew.draper@altera.com

Ned Smith
Intel Corporation
Email: ned.smith@intel.com