Remote ATtestation ProcedureS
Internet-Draft
Intended status: Informational
Expires: 29 November 2026

M. Poirier
Linaro
H. Birkholz
Fraunhofer SIT
T. Fossati
Linaro

28 May 2026

# An EAT Profile for Trustworthy Device Assignment
## draft-poirier-rats-eat-da-09

## Abstract

In confidential computing, device assignment (DA) is the method by which a device (e.g., network adapter, GPU), whether on-chip or behind a PCIe Root Port, is assigned to a Trusted Virtual Machine (TVM). For the TVM to trust an assigned device, the device must provide the TVM with attestation Evidence confirming its identity and the state of its firmware and configuration.

Since Evidence claims can be processed by 3rd party entities (e.g., Verifiers, Relying Parties) external to the TVM, there is a need to standardize the representation of DA-related information in Evidence to ensure interoperability. This document defines an attestation Evidence format for DA as an EAT (Entity Attestation Token) profile.

## About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://rats-device-attestation.github.io/draft-poirier-rats-eat-da/draft-poirier-rats-eat-da.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-poirier-rats-eat-da/.

Discussion of this document takes place on the Remote ATtestation ProcedureS Working Group mailing list (mailto:rats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rats/. Subscribe at https://www.ietf.org/mailman/listinfo/rats/.

Source for this draft and an issue tracker can be found at https://github.com/rats-device-attestation/draft-poirier-rats-eat-da.

## Status of This Memo

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

## Copyright Notice



## Table of Contents

## 1. Introduction

In confidential computing, device assignment (DA) is the method by which a device (e.g., network adapter, GPU), whether on-chip or behind a PCIe Root Port, is assigned to a Trusted Virtual Machine (TVM). Most confidential computing platforms (e.g., Arm CCA, AMD SEV-SNP, Intel TDX) provide DA capabilities. Such capabilities prevent execution environments or software components that are untrusted by the TVM (including other TVMs and the host hypervisor) from accessing or controlling a device that has been assigned to the TVM. This includes, for example, protection of device MMIO interfaces and device caches. From a trust perspective, DA allows a device to be included in the TVM's Trusted Computing Base (TCB). For the TVM to trust the device, the device must provide the TVM with attestation Evidence confirming its identity and the state of its firmware and configuration.

This document defines an attestation Evidence format for DA as an EAT [RFC9711] profile. The format is designed to be generic, extensible and architecture-agnostic. Ongoing work on DA concentrates on PCIe devices that support the SPDM protocol [SPDM]. As such, this document focuses on establishing the overall framework and formalizing an Evidence format for SPDM-compliant devices. This format is based on the information provided by the SPDM protocol without imposing additional security constraints. It is incumbent upon other entities to describe, select and enforce those additional security constraints based on operational requirements.

Since other bus architectures and protocols are expected to be supported as the technology gains wider adoption, provisions have been made for the definition of other Evidence formats such as Compute Express Link (CXL) and the Coherent Hub Interface (CHI). This list is by no means exhaustive and is expected to expand. Section 5 outlines the requirements for incorporating new bus technologies into the DAT framework. Lastly, live migration of a TVM from one host to another is currently not addressed by the SPDM specification and therefore not covered herein.

## 2. Conventions and Definitions

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.

## 3. Device Assignment Token (DAT) Claims

The Device Assignment Token (DAT) is the encompassing envelope for the individual device claims to be presented. A DAT can be used as a standalone entity but can also be embedded in a larger, platform-specific attestation token. A DAT consists of an EAT profile identifier, a nonce and an EAT submodule (Section 4.2.18 of [RFC9711]) that contains any number of individual device claims. Each individual device claim is the combination of a device name and a standard claims format based on the bus or protocol the device supports. The syntax of the device name depends on the type of bus or protocol used. Each name consists of two parts joined by a semicolon: a namespace and a bus-specific name. See Section 3.1.6 for SPDM devices, and Section 3.2.1 for legacy PCIe devices. As previously mentioned, this draft currently defines the claims set for SPDM compliant devices and PCIe legacy devices that do not support the SPDM protocol. Careful condideration was also given to the overall design in order to leave room for future expansion.

```
dat = {
  &(eat_profile: 265) => "tag:linaro.org,2025:device#1.0.0"
  &(eat_nonce: 10) => bytes .size 64
  &(eat_submods: 266) => {
    + device-name => $device-claims-set
  }
}

device-name = text

$device-claims-set /= spdm-claims
$device-claims-set /= pcie-legacy-claims
```

### 3.1. SPDM Claims

A SPDM claim instance is expected to be present for each SPDM compatible device to be attested. Each instance consists of a measurements section, a certificates section, or both. These can be supplemented with two additional sections: (1) a challenge for Component Mesaurement and Authentication (CMA) scenarios and (2) a device interface report that contains information from the TEE Device Information Security Protocol (TDISP) Device Interface Report. A challenge needs certificate information from the certificate section and as such, can only be present if certificates are included in the SPDM artifacts. TDISP messages are embedded in the VENDOR_DEFINED_REQUEST and VENDOR_DEFINED_RESPONSE messages of the SPDM protocol. Optionally, the Negotiated State preamble (version, capabilities and algorithms) bytes can be included to present the full negotiated state between the SPDM requester and responder.

```
spdm-claims = {
  &(eat_profile: 265) => "tag:linaro.org,2025:device-spdm#1.0.0"
  spdm-artefacts
  ? &(vca: 3804) => bytes
}

spdm-artefacts //= (
  &(measurements: 3802) => spdm-measurements
  &(certificates: 3803) => spdm-certificates
  ? &(challenge: 3807) => spdm-signature
  ? &(device-interface-report: 3808) => tdisp-device-interface-report
)

spdm-artefacts //= (
  &(measurements: 3802) => spdm-measurements
  ? &(device-interface-report: 3808) => tdisp-device-interface-report
)

spdm-artefacts //= (
  &(certificates: 3803) => spdm-certificates
  ? &(challenge: 3807) => spdm-signature
  ? &(device-interface-report: 3808) => tdisp-device-interface-report
)
```

#### 3.1.1. Measurements Claim

There can be up to 239 measurements per device with the entire measurement log optionally signed by the certificate populated in one of the 8 certificate slots. It should be noted that measurements formalized herein follow the DMTF measurement specification.

```
spdm-measurements = {
  + block-id => spdm-measurement
  ? "signature" => spdm-signature
}

block-id = 1..239
```

##### 3.1.1.1. Measurement

SPDM measurements start with a component type that reflects one of the 10 categories defined by the SPDM specification. Following is the measurement itself represented by either a raw bitstream or a digest. The size of the digest value is derived from the measurement hash algorithm conveyed by the SPDM ALGORITHMS message response.

```
spdm-measurement = {
  &(component-type: 1) => component-type
  measurement
}

measurement //= ( &(digest-measurement: 2) => digest-measurement )
measurement //= ( &(raw-measurement: 3) => raw-measurement )

component-type /= &(immutable-rom: 0)
component-type /= &(mutable-firmware: 1)
component-type /= &(hardware-config: 2)
component-type /= &(firmware-config: 3)
component-type /= &(freeform-measurement-manifest: 4)
component-type /= &(device-mode: 5)
component-type /= &(mutable-firmware-version: 6)
component-type /= &(mutable-firmware-svn: 7)
component-type /= &(hash-extend-measurement: 8)
component-type /= &(informational: 9)
component-type /= &(structured-measurement-manifest: 10)

raw-measurement = bytes
digest-measurement = digest

digest = [
  alg: uint / text
  val: bytes
]
```

### 3.1.2. SPDM Challenge Claim

SPDM compliant devices can optionally support the capability to authenticate responders through the challenge-response protocol and sign measurements. Included in the signature are all the elements needed by a third party entity to reconstruct the original transcript or measurement log signed by the device. Those elements include M1 for challenge signatures or L1 for measurement signatures (see CDDL below), the combined SPDM prefix, the hash algorithm used to generate a digest of the measurement log and nonces provided by the requester and responder. The slot number of the leaf certificate used to sign the measurement log is also provided.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

;
; What follows is based on SPDM v1.3.2 (DSP0274_1.3.2.pdf)
;

;
```

```
   ; Algorithms currently supported by SPDM.
   ; See "MeasurementHashAlgo", table 21, page 79.
   ;
   hash-algorithm-type /= &(tpm_alg_sha_256: 0)
   hash-algorithm-type /= &(tpm_alg_sha_384: 2)
   hash-algorithm-type /= &(tpm_alg_sha_512: 4)
   hash-algorithm-type /= &(tpm_alg_sha3_256: 8)
   hash-algorithm-type /= &(tpm_alg_sha3_384: 16)
   hash-algorithm-type /= &(tpm_alg_sha3_512: 32)
   hash-algorithm-type /= &(tpm_alg_sm3_256: 64)

   ;
   ; See signature generation and verification algorithms for
   ; CHALLENGE_AUTH message on page 108.
   ;
   ; M1 is _one_ of the following:
   ;
   ; M1 /= GET_VERSION , GET_CAPABILITIES , NEGOTIATE_ALGORITHMS , \
                 GET_DIGESTS , GET_CERTIFICATE , CHALLENGE (A1, B1, C1)
   ; M1 /= GET_VERSION , GET_CAPABILITIES , NEGOTIATE_ALGORITHMS , \
                                   GET_DIGESTS , CHALLENGE (A1, B3, C1)
   ; M1 /= GET_VERSION , GET_CAPABILITIES , NEGOTIATE_ALGORITHMS , \
                               GET_CERTIFICATE , CHALLENGE (A1, B4, C1)
   ; M1 /= GET_VERSION , GET_CAPABILITIES , NEGOTIATE_ALGORITHMS , \
                                                 CHALLENGE (A1, B2, C1)
   ; M1 /= GET_DIGESTS , GET_CERTIFICATE , CHALLENGE (A2, B1, C1)
   ; M1 /= GET_DIGESTS , CHALLENGE (A2, B3, C1)
   ; M1 /= GET_CERTIFICATE , CHALLENGE (A2, B4, C1)
   ; M1 /= CHALLENGE (A2, B2, C1)
   ;
   ; See signature generation and verification algorithms for
   ; MEASUREMENTS messages on page 126.
   ;
   ; L1 = Concatenate(VCA, GET_MEASUREMENTS_REQUEST1,
   ;                MEASUREMENTS_RESPONSE1, ...,
   ;                GET_MEASUREMENTS_REQUESTn-1,
   ;                MEASUREMENTS_RESPONSEn-1,
   ;                GET_MEASUREMENTS_REQUESTn, MEASUREMENTS_RESPONSEn)
   ;
   spdm-signature = {
      &(slot: 1) => 0..7, ; Slot of the certificate chain used to
                          ; authenticate the measurement.  Default
                          ; should be 0.
      &(requester-nonce: 2) => bytes .size 32,
      &(responder-nonce: 3) => bytes .size 32,
      &(combined-spdm-prefix: 4) => bytes .size 100,
      &(IL1: 5) => bytes, ; M1 or L1 (see comment above)
      &(base-hash-algo: 6) => hash-algorithm-type,
```

```
      &(signature: 7) => bytes
   }
```

### 3.1.3. Certificate Claims

According to the specification, SPDM compliant devices should support at most 8 slots, with slot 0 populated by default. Slot 0 SHALL contain a certificate chain that follows the Device certificate model or the Alias certificate model. Regardless of the certificate model used, a certificate chain comprises one or more DER-encoded X.509 v3 certificates [RFC5280]. The certificates MUST be concatenated with no intermediate padding.

```
spdm-certificates = {
  default-cert-slot => cert-chain
  ? aux-cert-slot-1 => cert-chain
  ? aux-cert-slot-2 => cert-chain
  ? aux-cert-slot-3 => cert-chain
  ? aux-cert-slot-4 => cert-chain
  ? aux-cert-slot-5 => cert-chain
  ? aux-cert-slot-6 => cert-chain
  ? aux-cert-slot-7 => cert-chain
}

; ASN.1 DER-encoded certificates concatenated with no intermediate
; padding.
cert-chain = bytes

default-cert-slot = 0

aux-cert-slot-1 = 1
aux-cert-slot-2 = 2
aux-cert-slot-3 = 3
aux-cert-slot-4 = 4
aux-cert-slot-5 = 5
aux-cert-slot-6 = 6
aux-cert-slot-7 = 7
```

### 3.1.4. TDISP Device Interface Report

A TDISP Device Interface Report can only be obtained if the device interface has transitioned to the CONFIG_LOCK or RUN state of the TDISP state machine.

It begins with various bitfields indicating the state and characteristics of the PCIe device interface. Next are 3 register fields pertaining to MSI-X (Message Signalled Interrupts), LNR (Lightweight Notification Requester) and TPH (TLP Processing Hints)

capabilities. MMIO ranges are assigned from PCIe BAR(s) and provide information about the memory areas a device is working with. More information on the MMIO range bitfields and the ones defined as part of the device interface field (above) can be found in the TDISP section of the PCI Express specification. The last field is device-specific and optionally included to convey additional configuration information about the device.

```
tdisp-device-interface-report = {
  ? &(interface-info: 1) => interface-info-bits
  ? &(msi-x-message-control: 2) => bytes .size 2
  ? &(lnr-control: 2) => bytes .size 2
  ? &(tph-control: 3) => bytes .size 4
  ? &(mmio-ranges: 4) => mmio-ranges
  ? &(device-specific-info: 5) => bytes
}

interface-info-bits = bytes .bits interface-info-flags
interface-info-flags = &(bit0: 0,
                         bit1: 1,
                         bit2: 2,
                         bit3: 3,
                         bit4: 4,
                         bit5: 5,
                        )

mmio-ranges = {
  + &(mmio-range: 1) => mmio-range
}

mmio-range = {
  &(first-4k-page: 1) => bytes .size 8
  &(number-of-4k-pages: 2) => bytes .size 4
  &(attributes: 3) => range-attributes
}

range-attributes = {
  &(range-attribute-bits: 1) => range-attribute-bits
  &(range-attribute-range-id: 2) => bytes .size 2
}

range-attribute-bits = bytes .bits range-attributes-flags
range-attributes-flags = &(bit0: 0,
                           bit1: 1,
                           bit2: 2,
                           bit3: 3,
                          )
```

#### 3.1.5. Negotiated State Preamble (Version, Capabilities and Algorithms)

The Negotiated State Preamble (i.e., vca) claim contains the concatenation of messages GET_VERSION, VERSION, GET_CAPABILITIES, CAPABILITIES, NEGOTIATE_ALGORITHMS, and ALGORITHMS last exchanged between the SPDM Requester and Responder.

#### 3.1.6. Submodule Naming

The namespace used for SPDM submodules is "spdm".

The name associated with an SPDM submodule is extracted from the leaf certificate of the relevant device.

* If the leaf certificate contains a Subject Alternative Name of type DMTFOtherName, the submodule name is the value contained in ub-DMTF-device-info. For example: "spdm:ACME:WIDGET:0123456789".

* Otherwise, the submod name is the string representation of the certificate Subject, as described in [RFC4514]. For example: "spdm:C=CA,O=ACME,OU=Widget,CN=0123456789".

### 3.2. PCIe Legacy Device Claims

The definition of a device claims set for PCIe legacy devices that do not implement the extensions needed to attest for their provenance and configuration is provided, making it is possible to keep using current assets as secures ones are being provisioned. This legacy device claims set simply mirrors the type 0/1 common registers of the PCIe configuration space, mandating only that the vendor and device identification code be provided. Other fields of the configuration space header may optionally be included should they add value. A binary format of the PCIe configuration space is made available for processing by existing PCIe configuration space tools. Implementers may optionally choose to include both text and binary versions should there be a use case to support this representation.

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   pcie-legacy-claims = {
     &(eat_profile: 265) => "tag:linaro.org,2025:device-pcie-legacy#1.0\
                                                                    .0"
     pcie-legacy-artefacts
     ? $$pcie-legacy-claim-extension
   }


   pcie-legacy-artefacts //= (
     &(artefacts-text: 3805) => pcie-type-0-1-config-space-text
     &(artefacts-bytes: 3806) => pcie-type-0-1-config-space-bytes
   )

   pcie-legacy-artefacts //= (
     &(artefacts-text: 3805) => pcie-type-0-1-config-space-text
   )

   pcie-legacy-artefacts //= (
     &(artefacts-bytes: 3806) => pcie-type-0-1-config-space-bytes
   )

   pcie-type-0-1-config-space-bytes = bytes .size 256

   pcie-type-0-1-config-space-text = {
     &(vendorID: 1) => bytes .size 2
     &(deviceID: 2) => bytes .size 2
     ? &(command: 3) => bytes .size 2
     ? &(status: 4) => bytes .size 2
     ? &(revisionID: 5) => bytes .size 1
     ? &(classCode: 6) => bytes .size 3
     ? &(cacheLineSize: 7) => bytes .size 1
     ? &(latencyTimer: 8) => bytes .size 1
     ? &(headerType: 9) => bytes .size 1
     ? &(BITS: 10) => bytes .size 1
   }
```

#### 3.2.1. Submodule Naming

The namespace used for legacy PCIe submodules is "legacy-pcie".

The name is any arbitrary string chosen by the implementation. For example, "legacy-pcie:0000:01:02.0" where "0000" is the domain, "01" the PCI bus id, "02" the device on the bus and "0" the device function.

# 4. DAT EAT Profile

## 4.1. Encoding

A DAT is encoded in CBOR [STD94]. The CBOR representation of a DAT MUST be "valid" according to the definition in Section 1.2 of [STD94]. Only definite-length strings, arrays, and maps are allowed. Since a DAT emitter may be found in a constrained environment, it may not be able to emit CBOR preferred serializations (Section 4.1 of [STD94]). Therefore, the Verifier MUST be a variation-tolerant CBOR decoder.

## 4.2. Cryptographic Protection

Cryptographic protection can be obtained by wrapping the dat claims-set in a COSE Web Token (CWT) [RFC8392]. In this case, the signature structure MUST be a tagged (18) COSE_Sign1. Alternatively, a DAT can be part of a Conceptual Message Wrapper (CMW) [I-D.ietf-rats-msg-wrap] collection. In this case, the DAT claims-set can be a UCCS [RFC9781] and the protection is provided by the signed CMW.

The flexibility provided by the COSE [RFC9052] format should be sufficient to adapt to the level of cryptographic agility required for specific use cases. It is RECOMMENDED that commonly adopted algorithms, such as those discussed in [RFC9053], are used. While receivers are expected to accept a wide range of algorithms, Attesters will produce DAT using only one such algorithm.

## 4.3. Use with Conceptual Message Wrappers

When used in a CMW, the collector will wrap the serialised COSE_Sign1 or UCCS with the appropriate media type or CoAP Content-Format defined in [RFC9782].

## 4.4. Freshness Model

DAT supports the freshness models for attestation Evidence based on nonces and epoch IDs (see Section 10.2 and Section 10.3 of [RFC9334]) using the eat_nonce claim to convey the nonce or epoch ID supplied by the Verifier. No further assumptions are made about the specific remote attestation protocol.

Note that the use of epoch IDs is subject by the type restrictions imposed by the eat_nonce syntax. For use in DAT, the epoch ID must be encodable as an opaque binary string of between 8 and 64 octets; an Epoclet can be used for this purpose (see [I-D.ietf-rats-epoch-markers]).

### 4.5. Synopsis

Table 1 presents a concise view of the requirements described in the preceding sections.

| Issue | Profile Definition |
|---|---|
| CBOR/JSON | CBOR MUST be used |
| CBOR Encoding | Definite length maps and arrays MUST be used |
| CBOR Encoding | Definite length strings MUST be used |
| CBOR Serialization | Variant serialization MAY be used |
| COSE Protection | COSE_Sign1 MUST be used (directly or via CMW) |
| Algorithms | [RFC9053] SHOULD be used |
| Detached EAT Bundle Usage | Detached EAT bundles MUST NOT be sent |
| Verification Key Identification | Any identification method listed in Appendix F.1 of [RFC9711] |
| Freshness | nonce or epoch ID based (Section 10.2 and Section 10.3 of [RFC9334]) |
| Claims | Those defined in Section 3. As per general EAT rules, the receiver MUST NOT error out on claims it does not understand. |

Table 1: DAT Profile Synopsis

## 5. Extending the DAT Framework

An extension to the DAT framework that introduces support for a new bus technology MUST provide the following information in a public document (e.g., an Internet-Draft):

* A precise definition of the new claims-set,

* A naming convention for the submod map entry,

* The registration of any new claims with IANA.

### 5.1. Claims-set Definition

The new claims-set MUST be specified clearly and unambiguously, ideally using CDDL, with a separate prose description of each claim. The claims-set MUST include a suitable eat_profile value.

See Section 3.1 for the blueprint.

### 5.2. Naming Conventions for the submod Key

A new claims-set MUST define a suitable naming convention for the submod keys associated with it. When creating this convention, ensure that it does not clash with any existing ones.

See Section 3.1.6 for the blueprint.

### 5.3. Claims Registrations

A new claims-set can reuse any number of already registered claims. If the claims-set needs to define new claims to express the desired semantics, and if these claims have generally applicable semantics, they SHOULD be registered with IANA.

See Section 9.1 for the blueprint.

## 6. Collated CDDL

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

dat = {
  &(eat_profile: 265) => "tag:linaro.org,2025:device#1.0.0",
  &(eat_nonce: 10) => bytes .size 64,
  &(eat_submods: 266) => {+ device-name => $device-claims-set},
}
device-name = text
$device-claims-set /= spdm-claims / pcie-legacy-claims
spdm-claims = {
  &(eat_profile: 265) => "tag:linaro.org,2025:device-spdm#1.0.0",
  spdm-artefacts,
  ? &(vca: 3804) => bytes,
}
spdm-artefacts //= ((
        &(measurements: 3802) => spdm-measurements,
        &(certificates: 3803) => spdm-certificates,
        ? &(challenge: 3807) => spdm-signature,
        ? &(device-interface-report: 3808) => tdisp-device-interface\
```

```
                                                                -report,
        ) // (
          &(measurements: 3802) => spdm-measurements,
          ? &(device-interface-report: 3808) => tdisp-device-interface\
                                                                -report,
        ) // (
          &(certificates: 3803) => spdm-certificates,
          ? &(challenge: 3807) => spdm-signature,
          ? &(device-interface-report: 3808) => tdisp-device-interface\
                                                                -report,
        ))
    spdm-measurement = {
      &(component-type: 1) => component-type,
      measurement,
    }
    measurement //= (&(digest-measurement: 2) => digest-measurement // &\
                                (raw-measurement: 3) => raw-measurement)
    component-type /= &(immutable-rom: 0) / &(mutable-firmware: 1) / &(\
    hardware-config: 2) / &(firmware-config: 3) / &(freeform-measurement\
    -manifest: 4) / &(device-mode: 5) / &(mutable-firmware-version: 6) \
    / &(mutable-firmware-svn: 7) / &(hash-extend-measurement: 8) / &(\
              informational: 9) / &(structured-measurement-manifest: 10)
    raw-measurement = bytes
    digest-measurement = digest
    digest = [
      alg: uint / text,
      val: bytes,
    ]
    spdm-certificates = {
      default-cert-slot => cert-chain,
      ? aux-cert-slot-1 => cert-chain,
      ? aux-cert-slot-2 => cert-chain,
      ? aux-cert-slot-3 => cert-chain,
      ? aux-cert-slot-4 => cert-chain,
      ? aux-cert-slot-5 => cert-chain,
      ? aux-cert-slot-6 => cert-chain,
      ? aux-cert-slot-7 => cert-chain,
    }
    cert-chain = bytes
    default-cert-slot = 0
    aux-cert-slot-1 = 1
    aux-cert-slot-2 = 2
    aux-cert-slot-3 = 3
    aux-cert-slot-4 = 4
    aux-cert-slot-5 = 5
    aux-cert-slot-6 = 6
    aux-cert-slot-7 = 7
    spdm-measurements = {
```

```
    + block-id => spdm-measurement,
    ? "signature" => spdm-signature,
  }
  block-id = 1 .. 239
  hash-algorithm-type /= &(tpm_alg_sha_256: 0) / &(tpm_alg_sha_384: 2\
  ) / &(tpm_alg_sha_512: 4) / &(tpm_alg_sha3_256: 8) / &(\
  tpm_alg_sha3_384: 16) / &(tpm_alg_sha3_512: 32) / &(tpm_alg_sm3_256\
                                                                  : 64)
  spdm-signature = {
    &(slot: 1) => 0 .. 7,
    &(requester-nonce: 2) => bytes .size 32,
    &(responder-nonce: 3) => bytes .size 32,
    &(combined-spdm-prefix: 4) => bytes .size 100,
    &(IL1: 5) => bytes,
    &(base-hash-algo: 6) => hash-algorithm-type,
    &(signature: 7) => bytes,
  }
  pcie-legacy-claims = {
    &(eat_profile: 265) => "tag:linaro.org,2025:device-pcie-legacy#1.0\
                                                                  .0",
    pcie-legacy-artefacts,
    ? $$pcie-legacy-claim-extension,
  }
  pcie-legacy-artefacts //= ((
          &(artefacts-text: 3805) => pcie-type-0-1-config-space-text,
          &(artefacts-bytes: 3806) => pcie-type-0-1-config-space-bytes,
        ) // &(artefacts-text: 3805) => pcie-type-0-1-config-space-\
  text // &(artefacts-bytes: 3806) => pcie-type-0-1-config-space-bytes)
  pcie-type-0-1-config-space-bytes = bytes .size 256
  pcie-type-0-1-config-space-text = {
    &(vendorID: 1) => bytes .size 2,
    &(deviceID: 2) => bytes .size 2,
    ? &(command: 3) => bytes .size 2,
    ? &(status: 4) => bytes .size 2,
    ? &(revisionID: 5) => bytes .size 1,
    ? &(classCode: 6) => bytes .size 3,
    ? &(cacheLineSize: 7) => bytes .size 1,
    ? &(latencyTimer: 8) => bytes .size 1,
    ? &(headerType: 9) => bytes .size 1,
    ? &(BITS: 10) => bytes .size 1,
  }
  tdisp-device-interface-report = {
    ? &(interface-info: 1) => interface-info-bits,
    ? &(msi-x-message-control: 2) => bytes .size 2,
    ? &(lnr-control: 2) => bytes .size 2,
    ? &(tph-control: 3) => bytes .size 4,
    ? &(mmio-ranges: 4) => mmio-ranges,
    ? &(device-specific-info: 5) => bytes,
```

```
}
interface-info-bits = bytes .bits interface-info-flags
interface-info-flags = &(
  bit0: 0,
  bit1: 1,
  bit2: 2,
  bit3: 3,
  bit4: 4,
  bit5: 5,
)
mmio-ranges = {+ &(mmio-range: 1) => mmio-range}
mmio-range = {
  &(first-4k-page: 1) => bytes .size 8,
  &(number-of-4k-pages: 2) => bytes .size 4,
  &(attributes: 3) => range-attributes,
}
range-attributes = {
  &(range-attribute-bits: 1) => range-attribute-bits,
  &(range-attribute-range-id: 2) => bytes .size 2,
}
range-attribute-bits = bytes .bits range-attributes-flags
range-attributes-flags = &(
  bit0: 0,
  bit1: 1,
  bit2: 2,
  bit3: 3,
)
```

## 7. Security Considerations

As this specification reuses the EAT specification [RFC9711], it also reuses the CWT specification [RFC8392]. The security and privacy considerations of these specifications therefore apply here too. In particular, the considerations discussed in Sections 9.1 (Claim Trustworthiness), 9.4 (Multiple EAT Consumers) and 9.5 (Detached EAT Bundle Digest Security Considerations) of [RFC9711] apply fully.

When DAT is an UCCS, the considerations in [RFC9781] also apply.

## 8. Privacy Considerations

A DAT can include a great deal of detail about the execution environment associated with the TVM and, therefore, the workload being executed within it. This can provide insight into the type of computation being carried out by the workload. It can also enable tracking of a given workload across multiple TVM instances in both the temporal and spatial dimensions.

A DAT is usually one component of a composite evidence payload. In such cases, multiple Verifiers may be involved in the appraisal process. The differential encryption considerations discussed in Section 9.4 (Multiple EAT Consumers) of [RFC9711] therefore apply.

# 9. IANA Considerations

## 9.1. New CWT Claims Registrations

IANA is requested to register the following claims in the "CBOR Web Token (CWT) Claims" registry [IANA.cwt].

### 9.1.1. SPDM Measurements Claim

* Claim Name: spdm-measurements
* Claim Description: SPDM Measurements
* JWT Claim Name: N/A
* Claim Key: 3802
* Claim Value Type(s): map
* Change Controller: IETF
* Specification Document(s): Section 3.1.1 of RFCthis

### 9.1.2. SPDM Certificates Claim

* Claim Name: spdm-certificates
* Claim Description: SPDM Certificates
* JWT Claim Name: N/A
* Claim Key: 3803
* Claim Value Type(s): map
* Change Controller: IETF
* Specification Document(s): Section 3.1.3 of RFCthis

### 9.1.3. SPDM VCA Claim

* Claim Name: spdm-vca

* Claim Description: SPDM Version, Capabilities and Algorithms
* JWT Claim Name: N/A
* Claim Key: 3804
* Claim Value Type(s): bytes
* Change Controller: IETF
* Specification Document(s): Section 3.1.5 of RFCthis

#### 9.1.4. PCIe Legacy Device Text Claim

* Claim Name: pcie-legacy-device-text
* Claim Description: PCIe Legacy Device Textual Representation
* JWT Claim Name: N/A
* Claim Key: 3805
* Claim Value Type(s): map
* Change Controller: IETF
* Specification Document(s): Section 3.2 of RFCthis

#### 9.1.5. PCIe Legacy Device Binary Claim

* Claim Name: pcie-legacy-device-binary
* Claim Description: PCIe Legacy Device Binary Representation
* JWT Claim Name: N/A
* Claim Key: 3806
* Claim Value Type(s): bytes
* Change Controller: IETF
* Specification Document(s): Section 3.2 of RFCthis

#### 9.1.6. SPDM Challenge Claim

* Claim Name: spdm-challenge

* Claim Description: SPDM Challenge signature block
* JWT Claim Name: N/A
* Claim Key: 3807
* Claim Value Type(s): map
* Change Controller: IETF
* Specification Document(s): Section 3.1.2 of RFCthis

#### 9.1.7. TDISP Device Interface Report

* Claim Name: tdisp-device-interface-report
* Claim Description: TDISP Device Interface Report
* JWT Claim Name: N/A
* Claim Key: 3808
* Claim Value Type(s): map
* Change Controller: IETF
* Specification Document(s): Section 3.1.4 of RFCthis

## 10. References

### 10.1. Normative References

[I-D.ietf-rats-epoch-markers]
Birkholz, H., Fossati, T., Pan, W., Mihalcea, I., and C. Bormann, "Epoch Markers", Work in Progress, Internet-Draft, draft-ietf-rats-epoch-markers-04, 18 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-epoch-markers-04>.

[I-D.ietf-rats-msg-wrap]
Birkholz, H., Smith, N., Fossati, T., Tschofenig, H., and D. Glaze, "RATS Conceptual Messages Wrapper (CMW)", Work in Progress, Internet-Draft, draft-ietf-rats-msg-wrap-23, 11 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-msg-wrap-23>.

[IANA.cwt] IANA, "CBOR Web Token (CWT) Claims", <https://www.iana.org/assignments/cwt>.

[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>.

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, DOI 10.17487/RFC4514, June 2006, <https://www.rfc-editor.org/rfc/rfc4514>.

[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>.

[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>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC9052] 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>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[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>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/rfc/rfc9711>.

[RFC9781] Birkholz, H., O'Donoghue, J., Cam-Winget, N., and C. Bormann, "A Concise Binary Object Representation (CBOR) Tag for Unprotected CBOR Web Token Claims Sets (UCCS)", RFC 9781, DOI 10.17487/RFC9781, May 2025, <https://www.rfc-editor.org/rfc/rfc9781>.

[RFC9782] Lundblade, L., Birkholz, H., and T. Fossati, "Entity Attestation Token (EAT) Media Types", RFC 9782, DOI 10.17487/RFC9782, May 2025, <https://www.rfc-editor.org/rfc/rfc9782>.

[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>.

## 10.2. Informative References

[SPDM] DMTF, "Security Protocol and Data Model (SPDM) Specification Version: 1.3.2", 21 August 2024, <https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.3.2.pdf>.

# Appendix A. Examples

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   {
     / profile / 265: "tag:linaro.org,2025:device#1.0.0",
     / nonce / 10: h'\
   f9efc3341597f75f8d94432ad39566a8c5704b2004ba001c094f475bfc057f9f25d7\
          aa40cd86cd30ebaae746fb19f008c1e6a1f23ad6a178e18dceda918f7f6e',
     / submods / 266: {
       "spdm:ACME:WIDGET-A:0123456789": {
         / profile / 265: "tag:linaro.org,2025:device-spdm#1.0.0",
         / measurements / 0x0eda: {
           1: {
             / component-type /  1: 2, / hardware config /
             / raw-measurement / 3: h'4f6d616861'
           }
         },
         / certificates / 0x0edb: {
           / device certs / 0: h'\
                             676f616e6e6174726164974696f6e6d6f6e676572'
           / no aux certs /
         }
       },
       "spdm:C=CA,O=ACME,OU=Widget-B,CN=9876543210": {
         / profile / 265: "tag:linaro.org,2025:device-spdm#1.0.0",
         / measurements / 0x0eda: {
           1: {
             / component-type / 1: 1, / mutable firmware /
             / digest-measurement / 2: [
               / alg / 1,
               / val / h'6b656e6e656c6c79'
             ]
           },
           6: {
             / component-type / 1: 2, / hardware config /
             / digest measurement / 2: [
               / alg / 0,
               / val / h'756e646572637279'
             ]
           }
         },
         / certificates / 0x0edb: {
           / device certs / 0: h'61746865697A656178696C6C6172',
           / aux certs (slot=2) / 2: h'23451576923AE99106783948598A'
         }
       }
     }
   }
```

## Appendix B. Example Composite Device

Figure 1 shows an example of the composite device described in Section 3.3 of [RFC9334] within a confidential computing environment. In this setup, a Trusted Virtual Machine (TVM) executes on a Confidential Platform, which provides the confidential computing environment. One or more devices (e.g., a GPU) are assigned to the TVM.

Within the TVM, a Lead Attester agent, e.g., a userland daemon, can collect Evidence from the Confidential Platform, as well as from all the assigned devices, using the relevant ABI offered by the guest OS kernel.

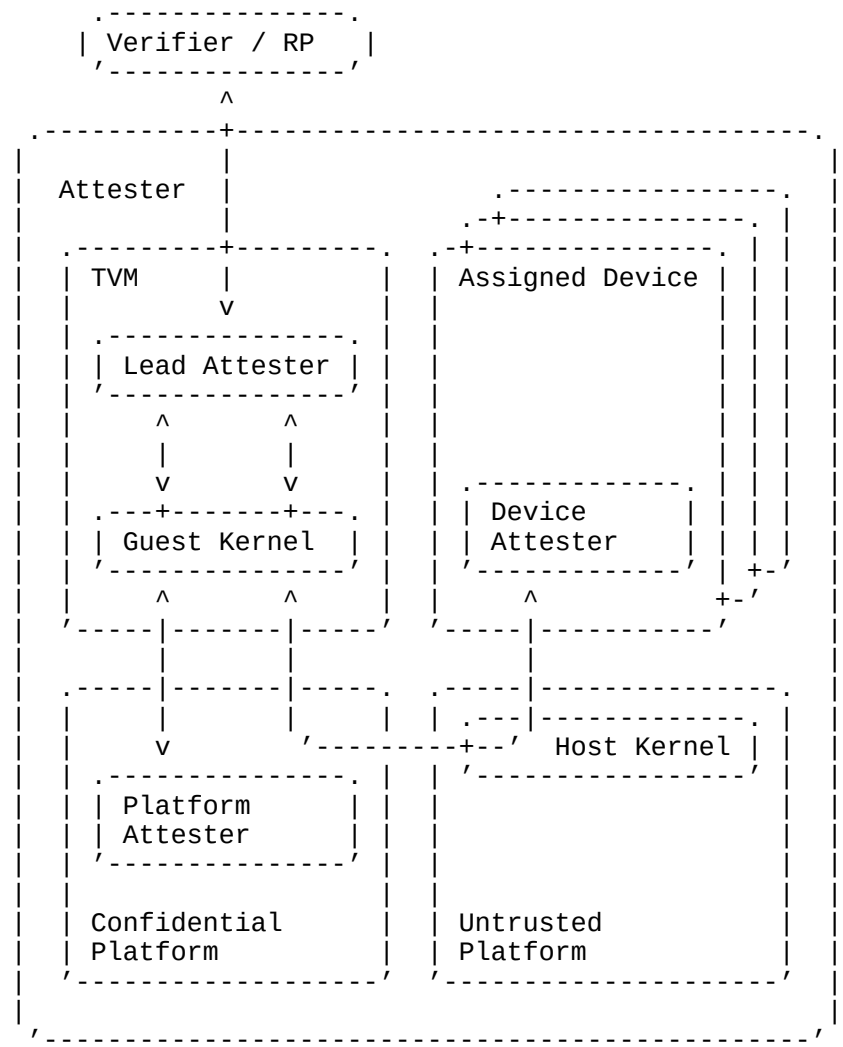


Figure 1: Confidential VM with Trusted Device(s)

When a challenger (i.e., a Verifier or a Relying Party) requests Evidence from the TVM, the Lead Attester broadcasts the received nonce to all the sub-Attesters, obtains Evidence from each of them and assembles the composite Evidence using a CMW Collection (Section 3.3 of [I-D.ietf-rats-msg-wrap]). It then signs the composite Evidence using its key material as shown in Figure 2.

The claims obtained by the assigned devices are repackaged into DAT submods, which are then signed as part of the CMW collection using the Lead Attester key.

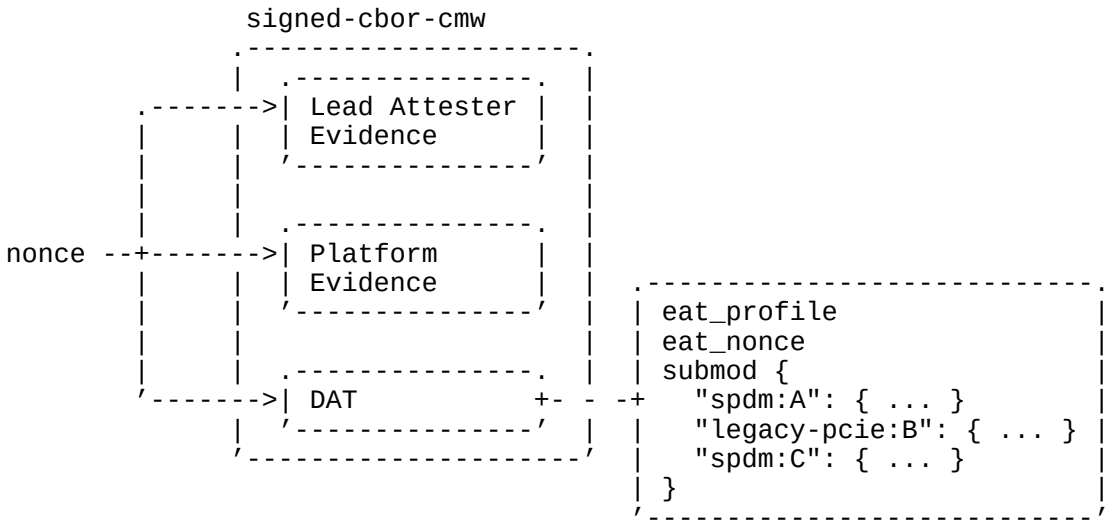


Figure 2: Composite Attestation Evidence

The Veraison project's ratsd daemon is an example of this behaviour.

## Acknowledgments

Thank you Basma El Gaabouri, James Bottomley, Jon Lange, Lukas Wunner, Roksana Golizadeh Mojarad, Simon Frost and Yousuf Sait for your comments and suggestions.

## Authors' Addresses

Mathieu Poirier
Linaro
Email: mathieu.poirier@linaro.org

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@ietf.contact

Thomas Fossati
Linaro
Email: thomas.fossati@linaro.org