Remote ATtestation ProcedureS
Internet-Draft
Intended status: Informational
Expires: 14 December 2025

M. Poirier
T. Fossati
Linaro

12 June 2025

An EAT Profile for Device Attestation
draft-poirier-rats-eat-da-00

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 the 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 consumed by 3rd party attestation services external to the TVM, there is a need to standardise the representation of 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 agents which 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, but other bus architecture and protocols are expected to be supported as the technology gains wider adoption. As such we focus on the formalization of an Evidence format for SPDM compliant devices while leaving room for the definition of other Evidence format such as CXL and CHI. This list is by no means exhaustive and is expected to expand.

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 Attestation Claims

The Device Attestation claim is the encompassing envelope for the individual device claims to be presented. It can be used as a standalone entity but typically enclosed in a wider platform specific attestation token. The Device attestation claim 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. 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.

```
da-token = {
  &(eat_profile: 265) => "tag:linaro.org,2025:device#1.0.0"
  &(eat_nonce: 10) => bytes .size 64 ; same as realm nonce
  &(eat_submods: 266) => {
    + device-name => $device-claims
  }
}

device-name = text .regexp "dev-[A-Za-z0-9]+"

$device-claims /= #6.1000000(spdm-claims)
$device-claims /= #6.1000001(cxl-claims)
$device-claims /= #6.1000002(chi-claims)
$device-claims /= #6.1000003(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 measurements and a certificates section. 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-claims = {
  &(measurements: 1) => {
    + block-id => spdm-measurement
    ? "signature" => spdm-measurement-blocks-signature
  }
  &(certificates: 2) => spdm-certificates
}

block-id = 1..239
```

3.1.1. Measurement Claims

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. Measurement Claims Signature

SPDM compliant devices can optionally support the capability to sign measurements. Included in the measurement claim signature are all the elements needed by a third party entity to reconstruct the original measurement log signed by the device. Those elements include L1 (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.

```
   ;
   ; 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
   ; 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-measurement-blocks-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, ; 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-slots => cert-chain
}

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

default-cert-slot = 0
aux-cert-slots = 1..7
```

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.

```
pcie-legacy-claims = {
  &(legacy-header: 1) => pcie-type-0-1-config-space
  ? $$pcie-legacy-claim-extension
}

pcie-type-0-1-config-space = {
  &(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
}
```

4. Collated CDDL

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

   da-token = {
     &(eat_profile: 265) => "tag:linaro.org,2025:device#1.0.0",
     &(eat_nonce: 10) => bytes .size 64,
     &(eat_submods: 266) => {+ device-name => $device-claims},
   }
   device-name = text .regexp "dev-[A-Za-z0-9]+"
   $device-claims /= #6.1000000(spdm-claims) / #6.1000001(cxl-claims) \
               / #6.1000002(chi-claims) / #6.1000003(pcie-legacy-claims)
   spdm-claims = {
     &(measurements: 1) => {
           + block-id => spdm-measurement,
           ? "signature" => spdm-measurement-blocks-signature,
   },
     &(certificates: 2) => spdm-certificates,
   }
   block-id = 1 .. 239
   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-slots => cert-chain,
   }
   cert-chain = bytes
   default-cert-slot = 0
   aux-cert-slots = 1 .. 7
   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-measurement-blocks-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,
  }
  cxl-claims = {}
  chi-claims = {}
  pcie-legacy-claims = {
    &(legacy-header: 1) => pcie-type-0-1-config-space,
    ? $$pcie-legacy-claim-extension,
  }
  pcie-type-0-1-config-space = {
    &(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,
  }
```

5. Security Considerations

TODO Security

6. IANA Considerations

6.1. New CWT Claims Registrations

TODO IANA CWT allocations

6.2. New CBOR Tags Registrations

TODO IANA CBOR Tag allocations

7. Normative References

[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>.

[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>.

[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>.

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: {
       "dev-a": 1000000({
         / measurements / 1: {
           1: {
             / component-type /  1: 2, / hardware config /
             / raw-measurement / 3: h'4f6d616861'
           }
         },
         / certificates / 2: {
           / device certs / 0: h'\
                              676f616e6e6174726164974696f6e6d6f6e676572'
           / no aux certs /
         }
       }),
       "dev-b": 1000000({
         / measurements / 1: {
           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 / 2: {
           / device certs / 0: h'61746865697A656178696C6C6172',
           / aux certs (slot=2) / 2: h'23451576923AE99106783948598A'
         }
       })
     }
   }
```

Acknowledgments

TODO acknowledge.

Authors' Addresses

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

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