Remote ATtestation ProcedureS
Internet-Draft
Intended status: Informational
Expires: 9 July 2026

T. Fossati
Linaro
Y. Deshpande
Arm Ltd
H. Birkholz
Fraunhofer SIT
5 January 2026

A CoRIM Profile for Arm's Platform Security Architecture (PSA) Endorsements
draft-fdb-rats-psa-endorsements-09

Abstract

PSA Endorsements comprise reference values, endorsed values, cryptographic key material and certification status information that a Verifier needs in order to appraise Attestation Evidence produced by a PSA device. This memo defines PSA Endorsements as a profile of the CoRIM data model.

Discussion Venues

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

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

Source for this draft and an issue tracker can be found at https://github.com/thomas-fossati/corim-psa.

Status of This Memo

Table of Contents

1. Introduction

PSA Endorsements include reference values, endorsed values, cryptographic key material and certification status information that a Verifier needs in order to appraise attestation Evidence produced by a PSA device [PSA-TOKEN]. This memo defines PSA Endorsements as a profile of the CoRIM data model [CoRIM].

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.

An understanding of the [CoRIM] data model is a prerequisite.

The reader is also assumed to be familiar with the terms defined in Section 2.1 of [PSA-TOKEN] and in Section 4 of [RATS-ARCH].

3. PSA Endorsements

PSA Endorsements describe an attesting device in terms of the hardware and firmware components that make up its PSA Root of Trust (RoT). This includes the identification and expected state of the device as well as the cryptographic key material needed to verify Evidence signed by the device's PSA RoT. Additionally, PSA Endorsements can include information related to the certification status of the attesting device.

There are three basic types of PSA Endorsements:

* Reference Values (Section 3.3), i.e., measurements of the PSA RoT firmware;

* Attestation Verification Keys (Section 3.4), i.e., cryptographic keys that are used to verify signed Evidence produced by the PSA RoT, along with the identifiers that bind the keys to their device instances;

* Certification Claims (Section 3.5), i.e., metadata that describe the certification status associated with a PSA device;

There is a fourth PSA Endorsement type that aims at covering more advanced Verifier use cases (e.g., the one described in Section 7 of [TEEP]):

* Software Relations (Section 3.6), used to model upgrade and patch relationships between software components.

3.1. PSA Endorsement Profile

PSA Endorsements are carried in one or more CoMIDs inside a CoRIM.

The profile attribute in the CoRIM MUST be present and MUST be the URI tag:arm.com,2025:psa#1.0.0 as shown in Figure 1.

```
/ corim-map / {
  / corim.profile / 3: 32("tag:arm.com,2025:psa#1.0.0")
  / ... /
}
```

Figure 1: CoRIM profile for PSA Endorsements version 1.0.0

3.2. PSA Endorsements to PSA RoT Linkage

Each PSA Endorsement - be it a Reference Value, Attestation Verification Key or Certification Claim - is associated with an immutable PSA RoT. The linkage between a PSA Endorsement and its PSA RoT is made by means of the unique PSA RoT identifier known as Implementation ID (see Section 3.2.2 of [PSA-TOKEN]).

To encode an Implementation ID, the tagged-bytes variant of the $class-id-type-choice is used, as described in Figure 2. The length of the byte string MUST be exactly 32.

```
; from draft-tschofenig-rats-psa-token
psa-implementation-id-type = bytes .size 32

tagged-implementation-id-type = #6.560(psa-implementation-id-type)
```

Figure 2: PSA Platform Implementation ID encoding

Besides, a PSA Endorsement can be associated with a specific instance of a certain PSA RoT - as is the case for Attestation Verification Keys. The Instance ID (see Section 3.2.1 of [PSA-TOKEN]) provides a unique identifier for a given PSA RoT.

To encode an Instance ID, the tagged-ueid-type variant of the $instance-id-type-choice is used, as described in Figure 3. The first byte MUST be 0x01 (RAND) followed by the 32-byte unique instance identifier.

```
inst-id-tagged-ueid = #6.550(eat-ueid-rand-type)

eat-ueid-rand-type = bytes .join eat-ueid-rand-fmt

eat-ueid-rand-fmt = [
  ; the type byte is 0x01
  ueid-rand-typ
  bytes .size 32
]

ueid-rand-typ = h'01'
```

Figure 3: PSA RoT Instance ID encoding

PSA Attestation Verification Keys are associated with a PSA RoT instance by means of the Instance ID and the corresponding Implementation ID. These identifiers are typically found in the subject of a CoMID triple, encoded in an environment-map as shown in Figure 4.

```
/ environment-map / {
  / comid.class / 0 : {
    / comid.class-id / 0 :
      / tagged-bytes / 560(
        h'61636d652d696d706c656d656e746174
          696f6e2d69642d303030303030303031'
      )
  },
  / comid.instance / 1 :
    / tagged-ueid-type / 550(
      h'01
        4ca3e4f50bf248c39787020d68ffd05c
        88767751bf2645ca923f57a98becd296'
    )
}
```

Figure 4: Example PSA RoT Identification

3.3. Reference Values

Reference Values carry measurements and other metadata associated with the updatable firmware in a PSA RoT. When appraising Evidence, the Verifier compares Reference Values against the values found in the Software Components of the PSA token (see Section 3.4.1 of [PSA-TOKEN]).

Each measurement is encoded in a measurement-map of a CoMID reference-triple-record. Since a measurement-map can encode one or more measurements, a single reference-triple-record can carry as many measurements as needed, provided they belong to the same PSA RoT identified in the subject of the triple.

A single reference-triple-record can completely describe the PSA RoT measurements.

Each PSA Software Component (i.e., the psa-software-component defined in Section 4.4.1 of [PSA-TOKEN]) is encoded in a measurement-values-map as defined in Figure 5.

```
psa-swcomp-measurement-values-map = {
  ? &(version: 0) => psa-swcomp-version-map
  &(digests: 2) => psa-swcomp-digests-type
  ? &(name: 11) => psa-swcomp-name
  &(cryptokeys: 13) => [ psa-swcomp-signer-id ]
}

psa-swcomp-version-map = {
  &(version: 0) => text
}

psa-swcomp-digests-type = [ + psa-digest ]

psa-digest = [
  alg: text
  val: psa-hash-type
]

psa-hash-type = bytes .size 32 / bytes .size 48 / bytes .size 64

psa-swcomp-name = text

psa-swcomp-signer-id = #6.560(psa-hash-type)
```

Figure 5: PSA Software Component encoding

version (key 0): A version-map with its version field containing the version (key 4) of the psa-software-component. The version-scheme field of the version-map MUST NOT be present. The version field is optional.

digests (key 2): Each array element encodes the "measurement value" (key 2) and "measurement-desc" (key 6) of the psa-sw-component in the val and alg entries, respectively. The alg entry MUST use the text encoding. The digests array MUST contain at least one entry

and MAY contain more than one entry if multiple digests (obtained with different hash algorithms) of the same measured component exist. If multiple entries exist, they MUST have different alg values. The digests field is mandatory.

name (key 11): A text value containing the "measurement-type" (key 1) of the psa-sw-component. The name field is optional.

cryptokeys (key 13): An array with _only one_ entry using the tagged-bytes variant of the $crypto-key-type-choice. The entry contains the "signer-id" (key 5) of the psa-sw-component. The cryptokeys field is mandatory.

Each measurement-values-map for a PSA RoT software component is wrapped in a measurement-map with an mkey using the text variant of the $measured-element-type-choice. The value of the mkey MUST be "psa.software-component". The authorized-by field of the measurement-map MUST NOT be present. See Figure 6 for the related CDDL definitions.

```
psa-swcomp-measurement-map = {
  &(mkey: 0) => "psa.software-component"
  &(mval: 1) => psa-swcomp-measurement-values-map
}
```

Figure 6: PSA RoT Software Component measurement-map

The complete example of a Reference Value CoMID Triple that encodes multiple psa-sw-component is given Figure 7.

```
/ concise-mid-tag / {
  / comid.tag-identity / 1 : {
    / comid.tag-id / 0 : h'3f06af63a93c11e4979700505690773f'
  },
  / comid.triples / 4 : {
    / comid.reference-triples / 0 : [
      [
        / environment-map / {
          / comid.class / 0 : {
            / comid.class-id / 0 :
              / tagged-impl-id-type / 560(
                h'61636d652d696d706c656d656e746174
                  696f6e2d69642d303030303030303031'
              )
          }
        },
        [
          / measurement-map / {
```

```
              / comid.mkey / 0 : "psa.software-component",
              / comid.mval / 1 : {
                / comid.digests / 2 : [
                  / hash-alg-id / "sha-256",
                  / hash-value /  h'9a271f2a916b0b6ee6cecb2426f0b320
                                    6ef074578be55d9bc94f6f3fe3ab86aa'
                ],
                / name / 11 : "BL",
                / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                            e2dc5641419c3d3060cfe32238
                                            c0fa973f7aa3') ]
              }
            },
            / measurement-map / {
              / comid.mkey / 0 : "psa.software-component",
              / comid.mval / 1 : {
                / comid.digests / 2 : [
                  / hash-alg-id / "sha-256",
                  / hash-value /  h'53c234e5e8472b6ac51c1ae1cab3fe06
                                    fad053beb8ebfd8977b010655bfdd3c3'
                ],
                / name / 11 : "PRoT",
                / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                            e2dc5641419c3d3060cfe32238
                                            c0fa973f7aa4') ]
              }
            }
          ]
        ]
      ]
    }
  }
```

Figure 7: Example Reference Value

3.4. Attestation Verification Keys

An Attestation Verification Key carries the verification key associated with the Initial Attestation Key (IAK) of a PSA device. When appraising Evidence, the Verifier can use the Implementation ID and Instance ID claims (see Section 3.2) to look up the verification key that it SHALL use to check the signature on the Evidence. This allows the Verifier to prove (or disprove) the Attester's claimed identity.

Each verification key is provided alongside the corresponding device Instance and Implementation IDs (and, possibly, a product identifier) in an attest-key-triple-record. Specifically:

* The Instance and Implementation IDs are encoded in the environment-map as shown in Figure 4;

* The IAK public key uses the tagged-pkix-base64-key-type variant of the $crypto-key-type-choice.

The IAK public key is a SubjectPublicKeyInfo [RFC5280] using the encoding defined in Section 13 of [RFC7468]. There MUST be only one key in an attest-key-triple-record.

The example in Figure 8 shows the PSA Endorsement of type Attestation Verification Key carrying a secp256r1 EC public IAK associated with Instance ID 4ca3...d296.

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

/ concise-mid-tag / {
  / comid.tag-identity / 1 : {
    / comid.tag-id / 0 : h'3f06af63a93c11e4979700505690773f'
  },
  / comid.triples / 4 : {
    / comid.attest-key-triples / 3 : [
      [
        / environment-map / {
          / comid.class / 0 : {
            / comid.class-id (implementation id) / 0 :
              / tagged-bytes / 560(
                h'61636d652d696d706c656d656e746174
                  696f6e2d69642d303030303030303031'
              )
          },
          / comid.instance / 1 :
            / tagged-ueid-type (instance id) / 550(
              h'01
                4ca3e4f50bf248c39787020d68ffd05c
                88767751bf2645ca923f57a98becd296'
            )
        },
        [
          / tagged-pkix-base64-key-type / 554(
            "-----BEGIN PUBLIC KEY-----\\
nMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEIShnxS4rlQiwpCCpBWDzlNLfqiG911FP\\
n8akBr+fh94uxHU5m+Kijivp2r2oxxN6MhM4tr8mWQli1P61xh3T0ViDREbF26DGO\\
       nEYfbAjWjGNN7pZf+6A4OTHYqEryz6m7U\n-----END PUBLIC KEY-----\n"
          )
        ]
      ]
    ]
  }
}
```

Figure 8: Example Attestation Verification Key

3.5. Certification Claims

PSA Certified [PSA-CERTIFIED] defines a certification scheme for the PSA ecosystem. A product - either a hardware component, a software component, or an entire device - that is verified to meet the security criteria established by the PSA Certified scheme is warranted a PSA Certified Security Assurance Certificate (SAC). A SAC contains information about the certification of a certain product (e.g., the target system, the attained certification level, the test

lab that conducted the evaluation, etc.), and has a unique Certificate Number.

The linkage between a PSA RoT -- comprising the immutable part as well as zero or more of the mutable components -- and the associated SAC is provided by a Certification Claim, which binds the PSA RoT Implementation ID and the software component identifiers with the SAC unique Certificate Number. When appraising Evidence, the Verifier can use the Certification Claims associated with the identified Attester as ancillary input to the Appraisal Policy, or to enrich the produced Attestation Result.

A Certification Claim is encoded as a conditional-endorsement-triple-record.

The SAC is encoded in a psa-cert-num that extends the measurement-values-map. See Figure 9.

```
$$measurement-values-map-extension //= (
  &(psa-cert-num: 100) => psa-cert-num-type
)

psa-cert-num-type = text .regexp "[0-9]{13} - [0-9]{5}"
```

Figure 9: Example Certification Triple

The conditional-endorsement-triple-record is constructed as follows:

* The Implementation ID of the immutable PSA RoT to which the SAC applies is encoded as a tagged-bytes in the environment-map of the stateful-environment-record; as shown in Figure 2

* Any software component that is part of the certified PSA RoT is encoded as a reference value (see Section 3.3) in the measurement-map of the stateful-environment-record;

* The unique SAC Certificate Number is encoded as psa-cert-num (key 100) in the measurement-values-map.

The example in Figure 10 shows a Certification Claim that associates Certificate Number 1234567890123 - 12345 to Implementation ID acme-implementation-id-000000001 and a single "PRoT" software component with version "1.3.5".

```
/ concise-mid-tag / {
  / comid.tag-identity / 1 : {
    / comid.tag-id / 0 : h'dbb0508ac658421c99c904124bab59ca'
  },
  / comid.triples / 4 : {
    / comid.conditional-endorsement-triple / 9 : [
      [
        / stateful-environment-record / [
        / environment-map / {
          / comid.class / 0 : {
            / comid.class-id / 0 :
              / tagged-bytes / 560(
                h'61636d652d696d706c656d656e746174
                  696f6e2d69642d303030303030303031'
              )
          }
        },
        / measurement-map / {
          / comid.mkey / 0 : "psa.software-component",
          / comid.mval / 1 : {
            / comid.digests / 2 : [
              / hash-alg-id / "sha-256",
              / hash-value /  h'53c234e5e8472b6ac51c1ae1cab3fe06
                                fad053beb8ebfd8977b010655bfdd3c3'
            ],
            / name / 11 : "PRoT",
            / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                        e2dc5641419c3d3060cfe32238
                                        c0fa973f7aa4') ]
          }
        }
      ],
      / measurement-values-map / {
         / psa.cert-num / 100 : "1234567890123 - 12345"
      }
    ]
  ]
  }
}
```

Figure 10: Example Certification Claim

3.6. Software Upgrades and Patches

In order to model software lifecycle events such as updates and patches, this profile defines a new triple that conveys the following semantics:

* SUBJECT: a software component

* PREDICATE: (non-critically / critically) (updates / patches)

* OBJECT: another software component

The triple is reified and used as the object of another triple, psa-swrel-triple-record, whose subject is the embedding environment.

```
comid.psa-swrel-triples = TBD2

$$triples-map-extension //= (
  comid.psa-swrel-triples => [ + psa-swrel-triple-record ]
)

psa.updates = 1
psa.patches = 2

psa-swrel-rel = [
  type: psa.updates / psa.patches
  security-critical: bool ; true means it’s a fix for a security bug
]

sw-rel = [
  new: comid.measurement-map ; the "new" firmware
  rel: psa-swrel-rel         ; patches/updates and the security flag
  old: comid.measurement-map ; the "old" firmware
]

psa-swrel-triple-record = [
  environment-map
  sw-rel
]
```

An example of a security critical update involving versions "1.2.5" and "1.3.0" of software component "PRoT" within the target environment associated with Implementation ID acme-implementation-id-000000001 is shown in Figure 11.

```
/ concise-mid-tag / {
  / comid.tag-identity / 1 : {
    / comid.tag-id / 0 : h’3f06af63a93c11e4979700505690773f’
  },
  / comid.triples / 4 : {
    / comid.psa-swrel-triples / 5 : [
      [
        / environment-map / {
          / comid.class-id / 0 :
```

```
            / tagged-impl-id-type / 560(
              h'61636d652d696d706c656d656e746174
                696f6e2d69642d303030303030303031'
            )
          },

          / sw-rel / [
            / new / {
              / comid.mval / 1 : {
              / comid.ver / 0 : {
                / comid.version / 0 : "1.3.0",
              },
                / comid.digests / 2 : [
                  / hash-alg-id / "sha-256",
                  / hash-value /  h'53c234e5e8472b6ac51c1ae1cab3fe06
                                    fad053beb8ebfd8977b010655bfdd3c3'
                ],
                / name / 11 : "PRoT",
                / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                            e2dc5641419c3d3060cfe32238
                                            c0fa973f7aa4') ]
              }
            },
            / rel / [
              / type / 1, / psa.updates /
              / security-critical / true
            ],

            / old / {
              / comid.mval / 1 : {
                    / comid.ver / 0 : {
                    / comid.version / 0 : "1.2.5",
                  },
                / comid.digests / 2 : [
                  / hash-alg-id / "sha-256",
                  / hash-value /  h'53c234e5e8472b6ac51c1ae1cab3fe06
                                    fad053beb8ebfd8978b010655bfdd3c3'
                ],
                / name / 11 : "PRoT",
                / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                            e2dc5641419c3d3060cfe32238
                                            c0fa973f7ad4') ]
              }
            }
          ]
        ]
      ]
    }
```

```
}
```

Figure 11: Example Critical Software Upgrade

4. Security Considerations

// TODO

5. IANA Considerations

5.1. CoMID Codepoints

5.1.1. CoMID Triples Map Extension

IANA is requested to register the following codepoints to the "CoMID Triples Map" registry.

Index	Item Name	Specification
50	comid.psa-swrel-triples	RFCthis

Table 1: PSA CoMID Triples

5.1.2. CoMID Measurement Values Map Extension

Key	Item Name	Item Type	Specification
100	comid.psa-cert-num	psa-cert-num	Section 3.5 of RFCthis

Table 2: Measurement Values Map Extensions

Acknowledgements

// TODO

References

Normative References

[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-09, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-corim-09>.

[PSA-TOKEN] Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", RFC 9783, DOI 10.17487/RFC9783, June 2025, <https://www.rfc-editor.org/rfc/rfc9783>.

[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>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/rfc/rfc7468>.

[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>.

Informative References

[PSA-CERTIFIED] "PSA Certified", 2021, <https://www.psacertified.org>.

[RATS-ARCH] 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>.

[TEEP] Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", RFC 9397, DOI 10.17487/RFC9397, July 2023, <https://www.rfc-editor.org/rfc/rfc9397>.

Authors' Addresses

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

Yogesh Deshpande
Arm Ltd
Email: yogesh.deshpande@arm.com

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@sit.fraunhofer.de