RATS
Internet-Draft
Intended status: Informational
Expires: 9 July 2026

Y. Deshpande
Arm Ltd
T. Fossati
Linaro

5 January 2026

A CoRIM Profile for Arm's Confidential Computing Architecture (CCA) Endorsements
draft-ydb-rats-cca-endorsements-03

Abstract

Arm Confidential Computing Architecture (CCA) Endorsements comprise reference values and cryptographic key material that a Verifier needs to appraise Attestation Evidence produced by an Arm CCA system.

This memo defines CCA Endorsements as a profile of the CoRIM data model.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/yogeshbdeshpande/draft-cca-rats-endorsements.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 July 2026.

Copyright Notice

Table of Contents

1. Introduction

Arm Confidential Computing Architecture (CCA) Endorsements comprise reference values and cryptographic key material that a Verifier needs to appraise Attestation Evidence produced by an Arm CCA system [I-D.ffm-rats-cca-token].

This memo defines CCA Endorsements as a profile of the CoRIM data model [I-D.ietf-rats-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.

The reader is assumed to be familiar with the terms and concepts introduced in [I-D.ffm-rats-cca-token] and in Section 4 of [RFC9334].

3. Arm CCA Endorsements

The Arm CCA Attester is a layered Attester comprising separate yet linked Platform and Realm Attesters. For the details, see Section 3 of [I-D.ffm-rats-cca-token]. Appraising Arm CCA Evidence requires Endorsements for both the Platform and Realm. This document outlines the Platform and Realm Endorsements in Section 3.1 and Section 3.2, respectively.

3.1. Arm CCA Platform Endorsements

There are two types of CCA Platform Endorsements:

* Reference Values (Section 3.1.3), i.e., measurements of the CCA Platform firmware.

* Attestation Verification Keys (Section 3.1.4), i.e., cryptographic keys that can be used to verify Evidence produced by the CCA Platform, along with the identifiers that link the keys to their platform instances.

3.1.1. Arm CCA Platform Endorsement Profile

Arm CCA Platform Endorsements are carried in one or more CoMIDs within a CoRIM.

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

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

Figure 1: CoRIM profile for CCA Platform Endorsements version 1.0.0

3.1.2. Arm CCA Platform Endorsements linkage to CCA Platform

Each CCA Platform Endorsement, be it a Reference Value or an Attestation Verification Key, is associated with a unique identifier known as CCA Platform Implementation ID (see Section 4.4.2 of [I-D.ffm-rats-cca-token]). The Implementation ID uniquely identifies a given implementation of a CCA Platform and it is used by the Endorser or Reference Value Provider as an anchor to which Reference Values and Attestation Verification Keys for a CCA Platform are linked.

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.

```
impl-id-tagged-bytes = #6.560(arm-platform-implementation-id-type)

arm-platform-implementation-id-type = bytes .size 32
```

Figure 2: CCA Platform Implementation ID encoding

Besides, a CCA Endorsement can be associated with a specific _instance_ of a certain CCA Platform implementation - as is the case of Attestation Verification Keys. The Instance ID (see Section 4.4.1 of [I-D.ffm-rats-cca-token]) provides a unique identifier for a given CCA Platform instance.

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: CCA Platform Instance ID encoding

CCA Attestation Verification Keys are associated with a CCA Platform 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 (implementation id) / 0 :
      / tagged-bytes / 560(
        h'61636d652d696d706c656d656e746174
          696f6e2d69642d303030303030303031'
      )
  },
  / comid.instance / 1 :
    / tagged-ueid-type (instance id) / 550(
      h'01
        4ca3e4f50bf248c39787020d68ffd05c
        88767751bf2645ca923f57a98becd296'
    )
}
```

Figure 4: Example CCA Platform Identification

Together, they are interpreted as a unique identifier of the CCA Platform.

3.1.3. Reference Values

Reference Values carry measurements and other metadata associated with the updatable firmware of the CCA Platform. CCA Platform is a collective term used to identify all the hardware and firmware components that comprise a CCA system. Specifically these include the following:

* CCA system security domain

* Monitor security domain

* Realm Management Security domain

When appraising Evidence, the Verifier compares Reference Values against:

* The values found in the Software Components of the CCA Platform token (see Section 4.6 of [I-D.ffm-rats-cca-token]).

* The value set in the platform configuration of the CCA Platform token (see Section 4.5.3 of [I-D.ffm-rats-cca-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 CCA Platform identified in the subject of the triple. A single reference-triple-record MUST completely describe the CCA Platform measurements.

3.1.3.1. CCA Platform Software Components

Each CCA Platform software component (called arm-platform-sw-component in Section 4.6.1 of [I-D.ffm-rats-cca-token]) is encoded in a measurement-values-map as defined in Figure 5.

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

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

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

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

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

cca-swcomp-name = text

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

Figure 5: CCA Platform Software Component encoding

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

digests (key 2): Each array element encodes the "measurement value"

(key 2) and "hash algorithm identifier" (key 6) of the arm-platform-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. This field is mandatory.

name (key 11): A text value containing the "component type" (key 1) of the arm-platform-sw-component. This 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 arm-platform-sw-component. This field is mandatory.

Each measurement-values-map for a CCA Platform 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 "cca.software-component". The authorized-by field of the measurement-map MUST NOT be present. Find the related CDDL definitions in Figure 6.

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

Figure 6: CCA Platform Software Component measurement-map

3.1.3.2. CCA Platform Configuration

The CCA Platform configuration describes the set of chosen implementation options of the CCA Platform. For example, this may include a description of the level of physical memory protection provided.

CCA Platform configuration is vendor-specific variable-length data. Only some of the data may be security-relevant. For these reasons, it is represented in a raw-value of the measurement-values-map, using the tagged-masked-raw-value variant of the $raw-values-type-choice. Refer to Section 5.1.4.1.4.6 of [I-D.ietf-rats-corim] for the details about the comparison algorithm.

```
cca-config-measurement-values-map = {
  &(raw-value: 4) => cca-tagged-masked-raw-value
}

cca-config-tagged-masked-raw-value = #6.563([
  value: bytes
  mask: bytes
])
```

Figure 7: CCA Platform Configuration measurement-map

The measurement-values-map for a CCA Platform configuration 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 "cca.platform-config". There MUST be only one measurement-map with mkey "cca.platform-config" in the triple.

The authorized-by field of the measurement-map MUST NOT be present. Find the related CDDL definitions in Figure 8.

```
cca-config-measurement-map = {
  &(mkey: 0) => "cca.platform-config"
  &(mval: 1) => cca-config-measurement-values-map
}
```

Figure 8: CCA Platform Software Component measurement-map

3.1.3.3. CoMID Example

An example CoMID containing one Reference Values triple with the expected values for both software components and platform configuration is given in Figure 9.

```
/ 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-bytes / 560(
                h'61636d652d696d706c656d656e746174
                  696f6e2d69642d303030303030303031'
              )
          }
```

```
          },
          [
            / measurement-map / {
              / comid.mkey / 0 : "cca.software-component",
              / comid.mval / 1 : {
                / comid.digests / 2 : [
                  / hash-alg-id / "sha-256",
                  / hash-value /  h'9a271f2a916b0b6ee6cecb2426f0b320
                                    6ef074578be55d9bc94f6f3fe3ab86aa'
                ],
                / name / 11 : "RSE_BL1_2",
                / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                            e2dc5641419c3d3060cfe32238
                                            c0fa973f7aa3') ]
              }
            },
            / measurement-map / {
              / comid.mkey / 0 : "cca.software-component",
              / comid.mval / 1 : {
                / comid.digests / 2 : [
                  / hash-alg-id / "sha-256",
                  / hash-value /  h'53c234e5e8472b6ac51c1ae1cab3fe06
                                    fad053beb8ebfd8977b010655bfdd3c3'
                ],
                / name / 11 : "RSE_BL2",
                / cryptokeys / 13 : [ 560(h'5378796307535df3ec8d8b15a2
                                            e2dc5641419c3d3060cfe32238
                                            c0fa973f7aa3') ]
              }
            },
            / measurement-map / {
              / comid.mkey / 0 : "cca.platform-config",
              / comid.mval / 1 : {
                / comid.raw-value / 4 : / tagged-masked-raw-value / 563([
                  / value / h'cfcfcfcf',
                  / mask /  h'ffffffff'
                ])
              }
            }
          ]
        ]
      ]
    }
  }
```

Figure 9: Example CCA Platform Reference Values

3.1.4. Attestation Verification Keys

An Attestation Verification Key contains the public key associated with the CCA Platform Attestation Key (CPAK). When appraising Platform Evidence, the Verifier uses the Implementation ID and Instance ID claims found in the Platform Token to identify the key that it shall use to verify the signature on the CCA Platform token. This allows the Verifier to prove (or disprove) the Attester's claimed identity.

Each verification key is provided with the corresponding CCA Platform Instance and Implementation IDs in an attest-key-triple-record. Specifically:

* The Instance and Implementation IDs are encoded in the environment-map as described in Section 3.1.2;

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

The CPAK 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 10 shows the CCA Endorsement of type Attestation Verification Key carrying a secp256r1 EC public CPAK 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 10: Example CCA Platform Attestation Verification Key

3.2. Arm CCA Realm Endorsements

Arm CCA provides confidential computing environments, known as Realms, that enable application workloads requiring confidential execution to operate in isolation from the host hypervisor and any other concurrent workload. Arm CCA allows the initial and run-time state of a Realm to be attested (Section 4.8 of [I-D.ffm-rats-cca-token]).

Realm Endorsements consist of Reference Values (Section 3.2.3), which are measurements of the configuration and contents of a Realm at the time of its activation, along with measurements of the software operating within the Realm, which can be extended throughout the Realm's lifetime.

Unlike the Platform, Realm Attestation Verification Key Endorsements are not necessary as the key material needed to verify the Realm Evidence is inline in the CCA Token (Section 3.2 of [I-D.ffm-rats-cca-token]).

3.2.1. Realm Endorsements linkage to Realm

Realms do not have _explicit_ class or instance identifiers. However, the Realm Initial Measurement (RIM) is unique and stable enough to serve as an identifier for the Realm Target Environment. Therefore, this profile employs an environment map with a class identifier that uses the tagged bytes variant of the $class-id-type-choice to encode the RIM value (Figure 11).

```
/ environment-map / {
  / comid.class / 0 : {
    / comid.class-id / 0 :
      / RIM as tagged-bytes / 560(
        h'311314ab73620350cf758834ae5c65d9
          e8c2dc7febe6e7d9654bbe864e300d49'
      )
  }
}
```

Figure 11: CCA Realm Identification

3.2.2. Arm CCA Realm Endorsement Profile

Arm CCA Realm Endorsements are carried in a CoMID within a CoRIM.

The profile attribute in the CoRIM MUST be present and MUST be the URI tag:arm.com,2025:cca_realm#1.0.0 as shown in Figure 12.

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

Figure 12: CoRIM profile for CCA Realm endorsements version 1.0.0

3.2.3. Reference Values

Reference Values carry measurements and other metadata associated with the CCA Realm.

Realm Reference Values comprise:

1. Realm Initial Measurements (RIM)

2. Realm Extended Measurements (REMs)

3. Realm Personalization Value (RPV)

All Realm Reference Values are carried in a reference-triple-record whose environment-map is as described in Section 3.2.1 The triple includes as many measurement-maps as needed to fully describe the Realm.

The measurement-map contents depend on the type of Reference Value. For all, the mkey uses the text variant of the $measured-element-type-choice. The value of the mkey MUST be "cca.rim" for the RIM measurement, "cca.rpv" for the RPV measurement, and "cca.rem0".."cca.rem3" for the REM measurements. The authorized-by field of the measurement-map MUST NOT be present.

RIM and REMs are encoded as digests (key 2).

RPV is encoded using a raw-value (key 4) using the tagged bytes variant of the $raw-value-type-choice.

All the Realm Reference Values are optional except RIM, which is mandatory.

3.2.3.1. CoMID Example

An example CoMID containing one Reference Values triple with the expected values for a Realm is given in Figure 13.

```
/ 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 :
```

```
            / RIM as tagged-bytes / 560(
              h'311314ab73620350cf758834ae5c65d9
                e8c2dc7febe6e7d9654bbe864e300d49'
            )
          }
        },
        / Realm measurements /
        [
          / measurement-map (RIM) / {
            / comid.mkey / 0 : "cca.rim",
            / comid.mval / 1 : {
              / comid.digests / 2 : [
                "sha-256",
                h'311314ab73620350cf758834ae5c65d9
                  e8c2dc7febe6e7d9654bbe864e300d49'
              ]
            }
          },
          / measurement-map (REM[0]) / {
            / comid.mkey / 0 : "cca.rem0",
            / comid.mval / 1 : {
              / comid.digests / 2 : [
                "sha-256",
                h'24d5b0a296cc05cbd8068c5067c5bd47
                  3b770dda6ae082fe3ba30abe3f9a6ab1'
              ]
            }
          },
          / measurement-map (REM[1]) / {
            / comid.mkey / 0 : "cca.rem1",
            / comid.mval / 1 : {
              / comid.digests / 2 : [
                "sha-256",
                h'788fc090bfc6b8ed903152ba8414e73d
                  af5b8c7bb1e79ad502ab0699b659ed16'
              ]
            }
          },
          / measurement-map (REM[2]) / {
            / comid.mkey / 0 : "cca.rem2",
            / comid.mval / 1 : {
              / comid.digests / 2 : [
                "sha-256",
                h'dac46a58415dc3a00d7a741852008e9c
                  ae64f52d03b9f76d76f4b3644fefc416'
              ]
            }
          },
```

```
            / measurement-map (REM[3]) / {
              / comid.mkey / 0 : "cca.rem3",
              / comid.mval / 1 : {
                / comid.digests / 2 : [
                  "sha-256",
                  h'32c6afc627e55585c03155359f331a0e
                    225f6840db947dd96efab81be2671939'
                ]
              }
            },
            / measurement-map (RPV) / {
              / comid.mkey / 0 : "cca.rpv",
              / comid.mval / 1 : {
                / comid.raw-value / 4 : 560(
                  h'54686520717569636b2062726f776e20
                    666f78206a756d7073206f7665722031
                    33206c617a7920646f67732e54686520
                    717569636b2062726f776e20666f7820'
                )
              }
            }
          ]
        ]
      ]
    }
  }
```

Figure 13: CCA realm identifiers

4. Security Considerations

// TODO

5. IANA Considerations

This document makes no requests to IANA.

6. Acknowledgements

// TODO

7. References

7.1. Normative References

[I-D.ffm-rats-cca-token]
Frost, S., Fossati, T., and G. Mandyam, "Arm's Confidential Compute Architecture Reference Attestation Token", Work in Progress, Internet-Draft, draft-ffm-rats-cca-token-02, 2 September 2025, <https://datatracker.ietf.org/doc/html/draft-ffm-rats-cca-token-02>.

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

[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>.

7.2. Informative References

[CCA-ARCH] Arm, "Learn the architecture - Introducing Arm Confidential Compute Architecture", March 2025, <https://developer.arm.com/documentation/den0125/0400>.

[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>.

Contributors

Simon Frost
Arm Limited
Email: Simon.Frost@arm.com

Sergei Trofimov
Arm Limited
Email: Sergei.Trofimov@arm.com

Authors' Addresses

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

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