RATS Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2021

H. Birkholz
M. Eckel
Fraunhofer SIT
C. Newton
L. Chen
University of Surrey

July 08, 2020

Reference Interaction Models for Remote Attestation Procedures
draft-birkholz-rats-reference-interaction-model-03

Abstract

This document describes interaction models for remote attestation procedures (RATS). Three conveying mechanisms - Challenge/Response, Uni-Directional, and Streaming Remote Attestation - are illustrated and defined. Analogously, a general overview about the information elements typically used by corresponding conveyance protocols are highlighted. Privacy preserving conveyance of Evidence via Direct Anonymous Attestation is elaborated on for each interaction model, individually.

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 January 9, 2021.

Table of Contents

1. Introduction

Remote ATtestation procedureS (RATS, [I-D.ietf-rats-architecture]) are workflows composed of roles and interactions, in which Verifiers create Attestation Results about the trustworthiness of an Attester's system component characteristics. The Verifier's assessment in the form of Attestation Results is created based on Attestation Policies and Evidence - trustable and tamper-evident Claims Sets about an Attester's system component characteristics - created by an Attester. The roles _Attester_ and _Verifier_, as well as the Conceptual Messages _Evidence_ and _Attestation Results_ are terms defined by the RATS Architecture [I-D.ietf-rats-architecture]. This documents captures interaction models that can be used in specific RATS-related solution documents. The primary focus of this document is the conveyance of attestation Evidence. Specific goals of this document are to:

- prevent inconsistencies in descriptions of these interaction models in other documents (due to text cloning over time),
- enable to highlight an exact delta/divergence between the core set of characteristics captured here in this document and variants of these interaction models used in other specifications or solutions, and to
- illustrate the application of Direct Anonymous Attestation (DAA) for each of the interaction models described.

In summary, this document enables the specification and design of trustworthy and privacy preserving conveyance methods for attestation Evidence from an Attester to a Verifier. While the conveyance of other Conceptual Messages is out-of-scope the methods described can also be applied to the conveyance of Endorsements or Attestation Results.

2. Terminology

This document uses the terms, roles, and concepts defined in [I-D.ietf-rats-architecture]:

Attester, Verifier, Relying Party, Conceptual Message, Evidence, Endorsement, Attestation Result, Appraisal Policy, Attesting Environment, Target Environment

A PKIX Certificate is an X.509v3 format certificate as specified by [RFC5280].

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. Disambiguation

The term "Remote Attestation" is a common expression and often associated or connoted with certain properties. The term "Remote" in this context does not necessarily refer to a remote entity in the scope of network topologies or the Internet. It rather refers to a decoupled system or entities that exchange the payload of the Conceptual Message type called Evidence [I-D.ietf-rats-architecture]. This conveyance can also be "local", if the Verifier is part of the same entity as the Attester, e.g., separate system components of a Composite Device (a single RATS Entity). Examples of these types of co-located environments include: a Trusted Execution Environment (TEE), Baseboard Management Controllers (BMCs), as well as other physical or logical protected/isolated/shielded Computing Environments (e.g. embedded Secure Elements (eSE) or Trusted Platform Modules (TPM)).

4. Scope and Intent

This document focuses on generic interaction models between Attesters and Verifiers in order to convey Evidence. Complementary procedures, functions, or services that are required for a complete semantic binding of the concepts defined in [I-D.ietf-rats-architecture] are out-of-scope of this document. Examples include: identity establishment, key distribution and enrollment, time synchronization, as well as certificate revocation.

Furthermore, any processes and duties that go beyond carrying out remote attestation procedures are out-of-scope. For instance, using the results of a remote attestation that are created by the Verifier, e.g., how to triggering remediation actions or recovery processes, as well as such remediation actions and recovery processes themselves, are also out-of-scope.

The interaction models illustrated in this document are intended to provide a stable basis and reference for other solutions documents inside or outside the IETF. Solution documents of any kind can reference the interaction models in order to avoid text clones and to avoid the danger of subtle discrepancies. Analogously, deviations from the generic model descriptions in this document can be illustrated in solutions documents to highlight distinct contributions.

5. Direct Anonymous Attestation

DAA [DAA] is a signature scheme used in RATS that allows preservation of the privacy of users that are associated with an Attester (e.g. its owner). Essentially, DAA can be seen as a group signature scheme with the feature that given a DAA signature no-one can find out who the signer is, i.e., the anonymity is not revocable. To be able to sign anonymously an Attester has to obtain a credential from a DAA Issuer. The DAA Issuer uses a private/public key pair to generate a credential for an Attester and makes the public key (in the form of a public key certificate) available to the verifier to enable them to validate the DAA signature obtained as part of the Evidence.

In order to support these DAA signatures, the DAA Issuer MUST associate a single key pair with each group of Attesters and use the same key pair when creating the credentials for all of the Attesters in this group. The DAA Issuer's public key certificate for the group replaces the Attester Identity documents in the verification of the Evidence (instead of unique Attester Identity documents). This is in contrast to intuition that there has to be a unique Attester Identity per device.

This document extends the duties of the Endorser role as defined by the RATS architecture with respect to the provision of these Attester Identity documents to Attesters. The existing duties of the Endorser role and the duties of a DAA Issuer are quite similar as illustrated in the following subsections.

5.1. Endorsers

Via its Attesting Environments, an Attester can only create Evidence about its Target Environments. After being appraised to be trustworthy, a Target Environment may become a new Attesting Environment in charge of creating Evidence for further Target Environments. [I-D.ietf-rats-architecture] explains this as Layered Attestation. Layered Attestation has to start with an initial Attesting Environment (i.e., there cannot be turtles all the way down [turtles]). At this rock bottom of Layered Attestation, the Attesting Environments are called Roots of Trust (RoT). An Attester cannot create Evidence about its own RoTs by design. As a consequence, a Verifier requires trustable statements about this subset of Attesting Environments from a different source than the Attester itself. The corresponding trustable statements are called Endorsements and originate from external, trustable entities that take on the role of an Endorser (e.g., supply chain entities).

5.2. Endorsers for Direct Anonymous Attestation

In order to enable DAA to be used, an Endorser role takes on the duties of a DAA Issuer in addition to its already defined duties. DAA Issuers offer zero-knowledge proofs based on public key certificates used for a group of Attesters [DAA]. Effectively, these certificates share the semantics of Endorsements. The differences are:

- The associated private keys are used by the DAA Issuer to provide an Attester with a credential that it can use to convince the Verifier that its Evidence is valid. To keep their anonymity the Attester randomises this credential each time that it is used.
- The Verifier can use the DAA Issuer's public key certificate, together with the randomised credential from the Attester, to confirm that the Evidence comes from a valid Attester.
- A credential is conveyed from an Endorser to an Attester together with the transfer of the public key certificates from Endorser to Verifier.

The zero-knowledge proofs required cannot be created by an Attester alone - like the Endorsements of RoTs - and have to be created by a trustable third entity - like an Endorser. Due to that vast semantic overlap (XXX-mcr:explain), an Endorser in this document can convey trustable third party statements both to a Verifier and an Attester.

6. Normative Prerequisites

In order to ensure an appropriate conveyance of Evidence, the following set of prerequisites MUST be in place to support the implementation of interaction models:

Attester Identity: The provenance of Evidence with respect to a distinguishable Attesting Environment MUST be correct and unambiguous.

An Attester Identity MAY be a unique identity, it MAY be included in a zero-knowledge proof (ZKP), or it MAY be part of a group signature, or it MAY be a randomised DAA credential.

Attestation Evidence Authenticity: Attestation Evidence MUST be correct and authentic.

In order to provide proofs of authenticity, Attestation Evidence SHOULD be cryptographically associated with an identity document (e.g. an PKIX certificate or trusted key material, or a randomised

DAA credential), or SHOULD include a correct and unambiguous and stable reference to an accessible identity document.

Authentication Secret: An Authentication Secret MUST be available exclusively to an Attester's Attesting Environment.

The Attester MUST protect Claims with that Authentication Secret, thereby proving the authenticity of the Claims included in Evidence. The Authentication Secret MUST be established before RATS can take place.

Evidence Freshness: Evidence MUST include an indicator about its Freshness that can be understood by a Verifier. Analogously, interaction models MUST support the conveyance of proofs of freshness in a way that is useful to Verifiers and their appraisal procedures.

Evidence Protection: Evidence MUST be a set of well-formatted and well-protected Claims that an Attester can create and convey to a Verifier in a tamper-evident manner.

7. Generic Information Elements

This section defines the information elements that are vital to all kinds interaction models. Varying from solution to solution, generic information elements can be either included in the scope of protocol messages or can be included in their payload. Ultimately, the following information elements are required by any kind of scalable remote attestation procedure using one or more of the interaction models provided.

Attester Identity ('attesterIdentity'): _mandatory_

A statement about a distinguishable Attester made by an Endorser without accompanying evidence about its validity - used as proof of identity.

In DAA the Attester's identity is not revealed to the verifier. The Attester is issued with a credential by the Endorser that is randomised and then used to anonymously confirm the validity of their evidence. The evidence is verified using the Endorser's public key.

Authentication Secret IDs ('authSecID'): _mandatory_

A statement representing an identifier list that MUST be associated with corresponding Authentication Secrets used to protect Evidence. In DAA, Authentication Secret IDs are

represented by the Endorser (DAA issuer)'s public key that MUST be used to create DAA credentials for the corresponding Authentication Secrets used to protect Evidence.

Each Authentication Secret is uniquely associated with a distinguishable Attesting Environment. Consequently, an Authentication Secret ID also identifies an Attesting Environment. In DAA an Authentication Secret ID does not identify a unique Attesting Environment but associated with a group of Attesting Environments. This is because an Attesting Environment should not be distinguishable and the DAA credential which represents the Attesting Environment is randomised each time it used.

Handle ('handle'): _mandatory_

A statement that is intended to uniquely distinguish received Evidence and/or determine the Freshness of Evidence.

A Verifier can also use a Handle as an indicator for authenticity or attestation provenance, as only Attesters and Verifiers that are intended to exchange Evidence should have knowledge of the corresponding Handles. Examples include Nonces or signed timestamps.

Claims ('claims'): _mandatory_

Claims are assertions that represent characteristics of an Attester's Target Environment.

Claims are part Conceptual Message and are, for example, used to appraise the integrity of Attesters via a Verifiers. The other information elements in this section can be expressed as Claims in any type of Conceptional Messages.

Reference Claims ('refClaims') _mandatory_

Reference Claims are a specific subset of Appraisal Policies as defined in [I-D.ietf-rats-architecture].

Reference Claims are used to appraise the Claims received from an Attester via appraisal by direct comparison. For example, Reference Claims MAY be Reference Integrity Measurements (RIM) or assertions that are implicitly trusted because they are signed by a trusted authority (see Endorsements in [I-D.ietf-rats-architecture]). Reference Claims typically represent (trusted) Claim sets about an Attester's intended platform operational state.

Claim Selection ('claimSelection'): _optional_

A statement that represents a (sub-)set of Claims that can be created by an Attester.

Claim Selections can act as filters that can specify the exact set of Claims to be included in Evidence. An Attester MAY decide whether or not to provide all Claims as requested via a Claim Selection.

Evidence ('signedAttestationEvidence'): _mandatory_

A set of Claims that consists of a list of Authentication Secret IDs that each identifies an Authentication Secret in a single Attesting Environment, the Attester Identity, Claims, and a Handle. Attestation Evidence MUST cryptographically bind all of these information elements. The Evidence MUST be protected via the Authentication Secret. The Authentication Secret MUST be trusted by the Verifier as authoritative.

Attestation Result ('attestationResult'): _mandatory_

An Attestation Result is produced by the Verifier as the output of the appraisal of Evidence. Attestation Results include condensed assertions about integrity or other characteristics of the corresponding Attester.

8. Interaction Models

The following subsections introduce and illustrate the interaction models:

1. Challenge/Response Remote Attestation

2. Uni-Directional Remote Attestation

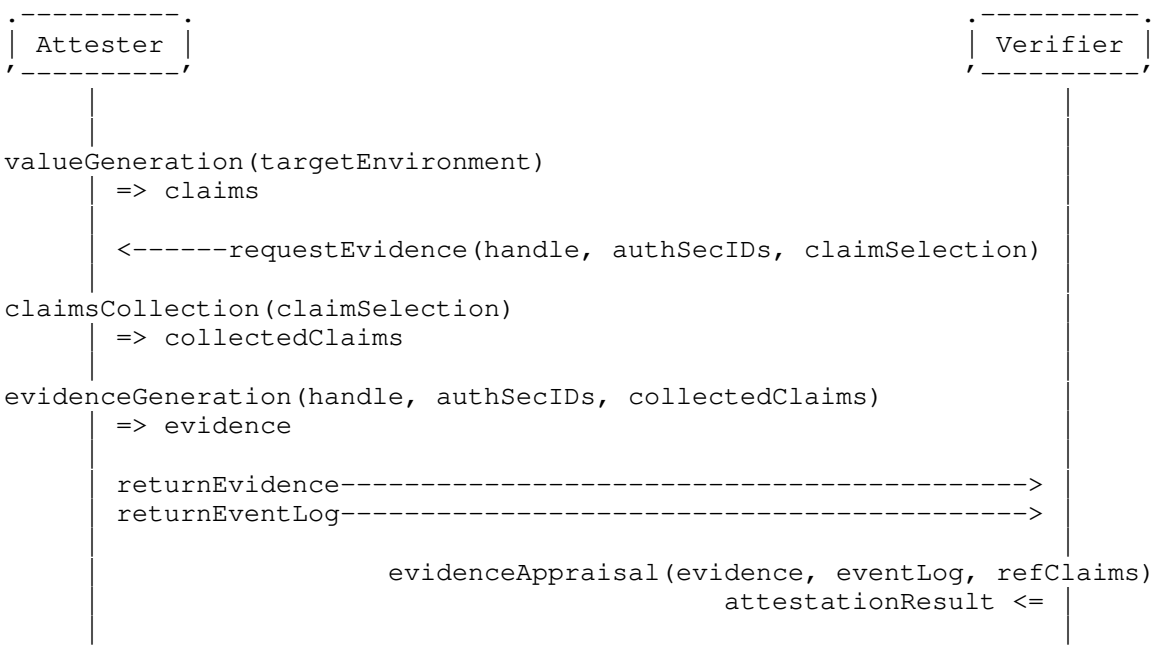
3. Streaming Remote Attestation

Each section starts with a sequence diagram illustrating the interactions between Attester and Verifier. The other roles RATS roles - mainly Relying Parties and Endorsers - are not relevant for this interaction model. While the interaction models presented focus on the conveyance of Evidence, future work could apply this to the conveyance of other Conceptual Messages, namely Attestation Results, Endorsements, or Appraisal Policies.

All interaction model have a strong focus on the use of a handle to incorporate a proof of freshness. The ways these handles are

processed is the most prominent difference between the three interaction models.

8.1. Challenge/Response Remote Attestation



This Challenge/Response Remote Attestation procedure is initiated by the Verifier, by sending a remote attestation request to the Attester. A request includes a Handle, a list of Authentication Secret IDs, and a Claim Selection.

In the Challenge/Response model, the handle is composed of qualifying data in the form of a cryptographically strongly randomly generated, and therefore unpredictable, nonce. The Verifier-generated nonce is intended to guarantee Evidence freshness.

The list of Authentication Secret IDs selects the attestation keys with which the Attester is requested to sign the Attestation Evidence. Each selected key is uniquely associated with an Attesting Environment of the Attester. As a result, a single Authentication Secret ID identifies a single Attesting Environment.

Analogously, a particular set of Evidence originating from a particular Attesting Environments in a composite device can be requested via multiple Authentication Secret IDs. Methods to acquire

Authentication Secret IDs or mappings between Attesting Environments to Authentication Secret IDs are out-of-scope of this document.

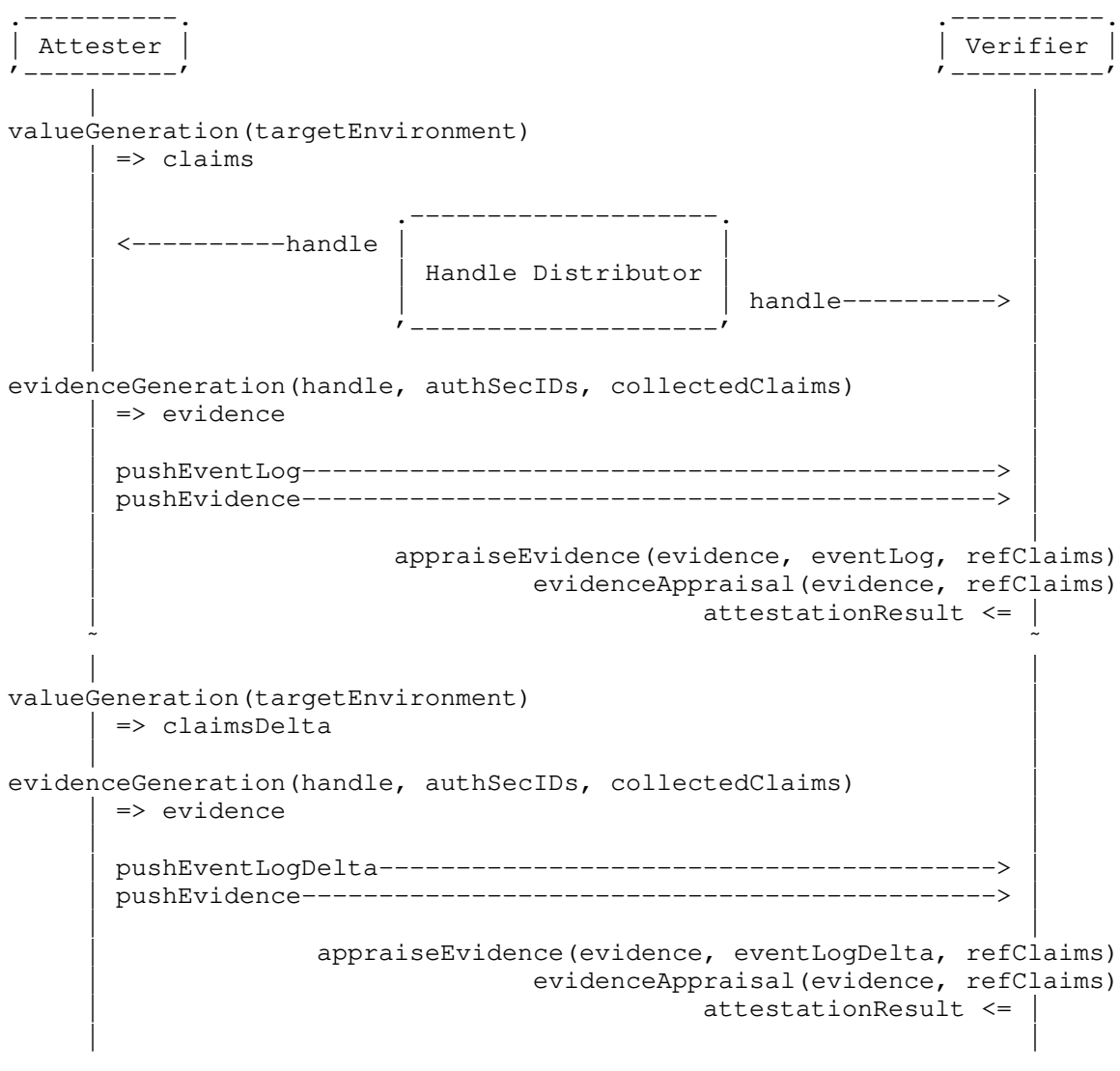
The Claim Selection narrows down the set of Claims collected and used to create Evidence to those that the Verifier requires. If the Claim Selection is omitted, then by default all Claims that are known and available on the Attester MUST be used to create corresponding Evidence. For example when performing a boot integrity evaluation, a Verifier may only be requesting a particular subset of claims about the Attester, such as Evidence about BIOS and firmware the Attester booted up, and not include information about all currently running software.

While it is crucial that Claims, the Handle, as well as the Attester Identity information MUST be cryptographically bound to the signature of Evidence, they may be presented in an encrypted form.

Cryptographic blinding MAY be used at this point. For further reference see section Section 11.

As soon as the Verifier receives signed Evidence, it validates the signature, the Attester Identity, as well as the Nonce, and appraises the Claims. Appraisal procedures are application-specific and can be conducted via comparison of the Claims with corresponding Reference Claims, such as Reference Integrity Measurements. The final output of the Verifier are Attestation Results. Attestation Results constitute new Claims Sets about an Attester's properties and characteristics that enables Relying Parties, for example, to assess an Attester's trustworthiness.

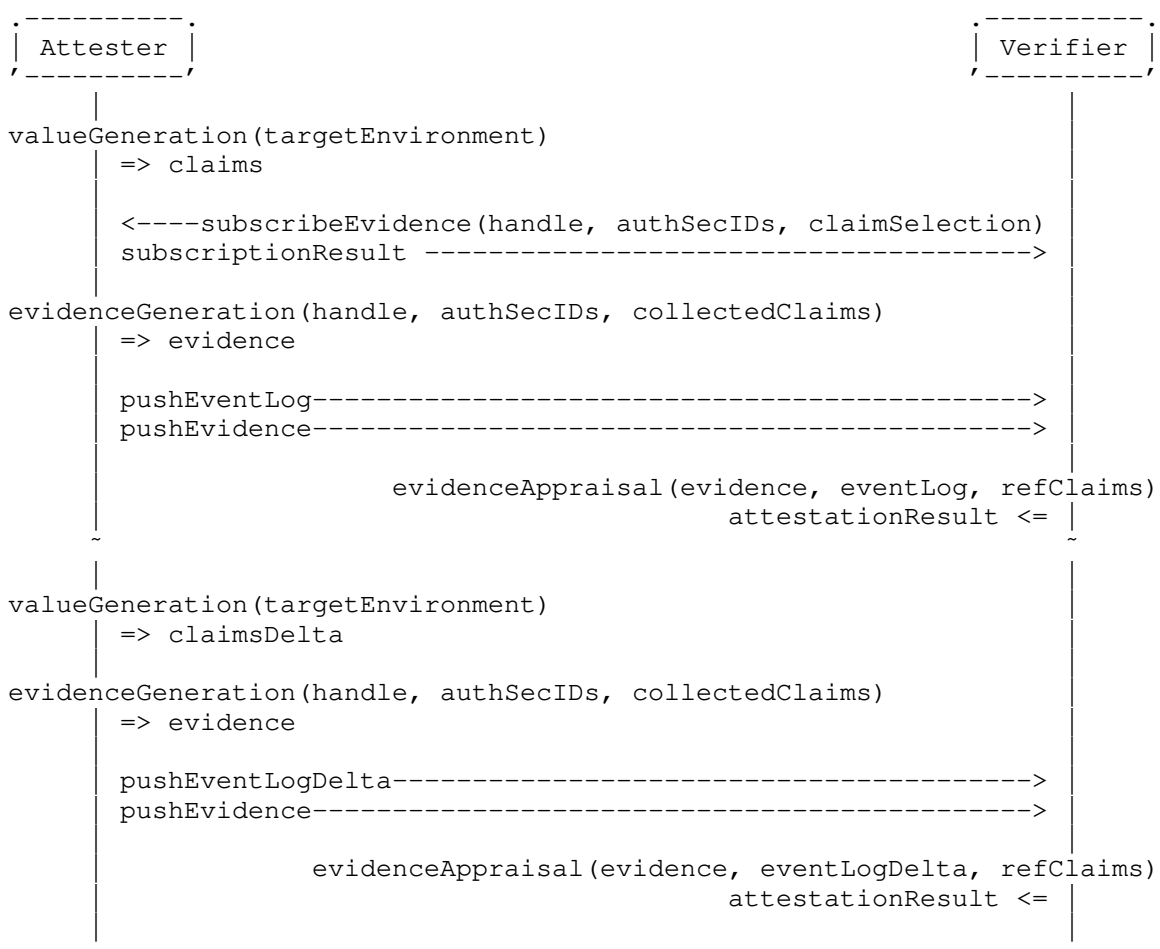
8.2. Uni-Directional Remote Attestation



Uni-Directional Remote Attestation procedures can be initiated both by the Attester and by the Verifier. Initiation by the Attester can result in unsolicited pushes of Evidence to the Verifier. Initiation by the Verifier always results in solicited pushes to the Verifier. The Uni-Directional model uses the same information elements as the Challenge/Response model. In the sequence diagram above, the Attester initiates the conveyance of Evidence (comparable with a RESTful POST operation or the emission of a beacon). While a request of evidence from the Verifier would result in a sequence diagram more similar to the Challenge/Response model (comparable with a RESTful

GET operation), the specific manner how handles are created and used always remains as the distinguishing quality of this model. In the Uni-Directional model, handles are composed of trustable signed timestamps as shown in [I-D.birkholz-rats-tuda], potentially including other qualifying data. The handles are created by an external 3rd entity - the Handle Distributor - that includes a trustworthy source of time and takes on the role of a Time Stamping Authority (TSA, as initially defined in [RFC3161]). Timstamps created from local clocks (absolute clocks using a global timescale, as well as relative clocks, such as tick-counters) of Attesters and Verifiers MUST be cryptographically bound to fresh Handles received from the Handle Distributor. This binding provides a proof of synchronization that MUST be included in every evidence created. Correspondingly, evidence created for conveyance via this model provides a proof that it was fresh at a certain point in time. Effectively, this allows for series of evidence to be pushed to multiple Verifiers, simultaniously. Methods to detect excessive time drift that would mandate a fresh Handle to be received by the Handle Distributor, as well as timing of handle distribution are out-of-scope of this document.

8.3. Streaming Remote Attestation



Streaming Remote Attestation procedures require the setup of subscription state. Setting up subscription state between a Verifier and an Attester is conducted via a subscribe operation. This subscribe operation is used to convey the handles required for Evidence generation. Effectively, this allows for series of evidence to be pushed to a Verifier similar to the Uni-Directional model. While a Handle Distributor is not required in this model, it is also limited to bi-lateral subscription relationships, in which each Verifier has to create and provide its individual handle. Handles provided by a specific subscribing Verifier MUST be used in Evidence generation for that specific Verifier. The Streaming model uses the same information elements as the Challenge/Response and the Uni-Directional model. Methods to detect excessive time drift that would mandate a refreshed Handle to be conveyed via another subscribe operation are out-of-scope of this document.

9. Additional Application-Specific Requirements

Depending on the use cases covered, there can be additional requirements. An exemplary subset is illustrated in this section.

9.1. Confidentiality

Confidentiality of exchanged attestation information may be desirable. This requirement usually is present when communication takes place over insecure channels, such as the public Internet. In such cases, TLS may be uses as a suitable communication protocol that preserves confidentiality. In private networks, such as carrier management networks, it must be evaluated whether or not the transport medium is considered confidential.

9.2. Mutual Authentication

In particular use cases mutual authentication may be desirable in such a way that a Verifier also needs to prove its identity to the Attester, instead of only the Attester proving its identity to the Verifier.

9.3. Hardware-Enforcement/Support

Depending on given usage scenarios, hardware support for secure storage of cryptographic keys, crypto accelerators, as well as protected or isolated execution environments can be mandatory requirements. Well-known technologies in support of these requirements are roots of trusts, such as Hardware Security Modules (HSM), Physically Unclonable Functions (PUFs), Shielded Secrets, or Trusted Executions Environments (TEEs).

10. Implementation Status

Note to RFC Editor: Please remove this section as well as references to [BCP205] before AUTH48.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [BCP205]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their

features. Readers are advised to note that other implementations may exist.

According to [BCP205], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

10.1. Implementer

The open-source implementation was initiated and is maintained by the Fraunhofer Institute for Secure Information Technology - SIT.

10.2. Implementation Name

The open-source implementation is named "CHAllenge-Response based Remote Attestation" or in short: CHARRA.

10.3. Implementation URL

The open-source implementation project resource can be located via: https://github.com/Fraunhofer-SIT/charra

10.4. Maturity

The code's level of maturity is considered to be "prototype".

10.5. Coverage and Version Compatibility

The current version (commit '847bcde') is aligned with the exemplary specification of the CoAP FETCH bodies defined in section Appendix A of this document.

10.6. License

The CHARRA project and all corresponding code and data maintained on github are provided under the BSD 3-Clause "New" or "Revised" license.

10.7. Implementation Dependencies

The implementation requires the use of the official Trusted Computing Group (TCG) open-source Trusted Software Stack (TSS) for the Trusted Platform Module (TPM) 2.0. The corresponding code and data is also maintained on github and the project resources can be located via: https://github.com/tpm2-software/tpm2-tss/

The implementation uses the Constrained Application Protocol [RFC7252] (http://coap.technology/) and the Concise Binary Object Representation [RFC7049] (https://cbor.io/).

10.8. Contact

Michael Eckel (michael.eckel@sit.fraunhofer.de)

11. Security and Privacy Considerations

In a remote attestation procedure the Verifier or the Attester MAY want to cryptographically blind several attributes. For instance, information can be part of the signature after applying a one-way function (e. g. a hash function).

There is also a possibility to scramble the Nonce or Attester Identity with other information that is known to both the Verifier and Attester. A prominent example is the IP address of the Attester that usually is known by the Attester itself as well as the Verifier. This extra information can be used to scramble the Nonce in order to counter certain types of relay attacks.

12. Acknowledgments

Olaf Bergmann, Michael Richardson, and Ned Smith

13. Change Log

- Initial draft -00
- Changes from version 00 to version 01:
  - Added details to the flow diagram
  - Integrated comments from Ned Smith (Intel)
  - Reorganized sections and
  - Updated interaction model
  - Replaced "claims" with "assertions"
  - Added proof-of-concept CDDL for CBOR via CoAP based on a TPM 2.0 quote operation
- Changes from version 01 to version 02:

    * Revised the relabeling of "claims" with "assertion" in alignment with the RATS Architecture I-D.

    * Added Implementation Status section

    * Updated interaction model

    * Text revisions based on changes in [I-D.ietf-rats-architecture] and comments provided on rats@ietf.org.

  o Changes from version 02 to version 00 RATS related document

    * update of the challenge/response diagram

    * minor rephrasing of Prerequisites section

    * rephrasing to information elements and interaction model section

  o Changes from version 00 to version 01

    * added Attestation Authenticity, updated Identity and Secret

    * relabeled Secret ID to Authentication Secret ID + rephrasing

    * relabeled Claim Selection to Assertion Selection + rephrasing

    * relabeled Evidence to (Signed) Attestation Evidence

    * Added Attestation Result and Reference Assertions

    * update of the challenge/response diagram and expositional text

    * added CDDL spec for CoAP FETCH operation proof-of-concept

  o Changes from version 01 to version 02

    * prepared the inclusion of additional reference models

    * update to Introduction and Scope section

    * major update to (Normative) Prerequisites

    * relabeled Attestation Authenticity to Att. Evidence Authenticity

    * relabeled Assertion term back to Claim terms

    * added BCP205 Implementation Status section related to Appendix CDDL

  o Changes from version 02 to version 03

    * major refactoring to now accommodate three interaction models

    * updated existing and added two new diagrams for models

    * major refactoring of existing and adding of new diagram description

    * incorporated content about Direct Anonymous Attestation

    * integrated comments from Michael Richardson

    * updated roster

14. References

14.1. Normative References

[BCP205] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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/info/rfc2119>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[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/info/rfc5280>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[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/info/rfc8174>.

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

14.2. Informative References

[DAA] Brickell, E., Camenisch, J., and L. Chen, "Direct Anonymous Attestation", ACM Proceedings of the 11rd ACM conference on Computer and Communications Security , page 132-145, 2004.

[I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", draft-birkholz-rats-tuda-02 (work in progress), March 2020.

[I-D.ietf-rats-architecture] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-04 (work in progress), May 2020.

[turtles] Wikipedia, "Turrles all the way down", July 2020, <https://en.wikipedia.org/wiki/Turtles_all_the_way_down>.

Appendix A. CDDL Specification for a simple CoAP Challenge/Response Interaction

The following CDDL specification is an exemplary proof-of-concept to illustrate a potential implementation of the Challenge/Response Interaction Model. The transfer protocol used is CoAP using the FETCH operation. The actual resource operated on can be empty. Both the Challenge Message and the Response Message are exchanged via the FETCH operation and corresponding FETCH Request and FETCH Response body.

In this example, evidence is created via the root-of-trust for reporting primitive operation "quote" that is provided by a TPM 2.0.

```
RAIM-Bodies = CoAP-FETCH-Body / CoAP-FETCH-Response-Body

CoAP-FETCH-Body = [ hello: bool, ; if true, the AK-Cert is conveyed
                    nonce: bytes,
                    pcr-selection: [ + [ tcg-hash-alg-id: uint .size 2, ; TPM2_AL
G_ID
                                         [ + pcr: uint .size 1 ],
                                       ]
                                     ],
                  ]

CoAP-FETCH-Response-Body = [ attestation-evidence: TPMS_ATTEST-quote,
                             tpm-native-signature: bytes,
                             ? ak-cert: bytes, ; attestation key certificate
                           ]

TPMS_ATTEST-quote = [ qualifiediSigner: uint .size 2, ;TPM2B_NAME
                      TPMS_CLOCK_INFO,
                      firmwareVersion: uint .size 8
                      quote-responses: [ * [ pcr: uint .size 1,
                                             + [ pcr-value: bytes,
                                                 ? hash-alg-id: uint .size 2,
                                               ],
                                           ],
                                         ? pcr-digest: bytes,
                                       ],
                    ]

TPMS_CLOCK_INFO = [ clock: uint .size 8,
                    resetCounter: uint .size 4,
                    restartCounter: uint .size 4,
                    save: bool,
                  ]
```

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

Christopher Newton
University of Surrey

Email: cn0016@surrey.ac.uk

Liqun Chen
University of Surrey

Email: liqun.chen@surrey.ac.uk

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 4, 2021

G. Mandyam
Qualcomm Technologies Inc.
L. Lundblade
Security Theory LLC
M. Ballesteros
J. O'Donoghue
Qualcomm Technologies Inc.

August 31, 2020

The Entity Attestation Token (EAT)
draft-ietf-rats-eat-04

Abstract

An Entity Attestation Token (EAT) provides a signed (attested) set of claims that describe state and characteristics of an entity, typically a device like a phone or an IoT device. These claims are used by a relying party to determine how much it wishes to trust the entity.

An EAT is either a CWT or JWT with some attestation-oriented claims. To a large degree, all this document does is extend CWT and JWT.

Contributing

TBD

Table of Contents

1. Introduction

Remote device attestation is a fundamental service that allows a remote device such as a mobile phone, an Internet-of-Things (IoT) device, or other endpoint to prove itself to a relying party, a server or a service. This allows the relying party to know some characteristics about the device and decide whether it trusts the device.

Remote attestation is a fundamental service that can underlie other protocols and services that need to know about the trustworthiness of the device before proceeding. One good example is biometric authentication where the biometric matching is done on the device. The relying party needs to know that the device is one that is known to do biometric matching correctly. Another example is content protection where the relying party wants to know the device will protect the data. This generalizes on to corporate enterprises that might want to know that a device is trustworthy before allowing corporate data to be accessed by it.

The notion of attestation here is large and may include, but is not limited to the following:

- Proof of the make and model of the device hardware (HW)
- Proof of the make and model of the device processor, particularly for security-oriented chips
- Measurement of the software (SW) running on the device
- Configuration and state of the device
- Environmental characteristics of the device such as its GPS location

1.1. CDDL, CWT and JWT

An EAT token is either a CWT as defined in [RFC8392] or a JWT as defined in [RFC7519]. This specification defines additional claims for entity attestation.

This specification uses CDDL, [RFC8610], as the primary formalism to define each claim. The implementor then interprets the CDDL to come to either the CBOR [RFC7049] or JSON [ECMAScript] representation. In the case of JSON, Appendix E of [RFC8610] is followed. Additional rules are given in Section 4.3.2 of this document where Appendix E is insufficient. (Note that this is not to define a general means to translate between CBOR and JSON, but only to define enough such that

the claims defined in this document can be rendered unambiguously in JSON).

1.2. Entity Overview

An "entity" can be any device or device subassembly ("submodule") that can generate its own attestation in the form of an EAT. The attestation should be cryptographically verifiable by the EAT consumer. An EAT at the device-level can be composed of several submodule EAT's. It is assumed that any entity that can create an EAT does so by means of a dedicated root-of-trust (RoT).

Modern devices such as a mobile phone have many different execution environments operating with different security levels. For example, it is common for a mobile phone to have an "apps" environment that runs an operating system (OS) that hosts a plethora of downloadable apps. It may also have a TEE (Trusted Execution Environment) that is distinct, isolated, and hosts security-oriented functionality like biometric authentication. Additionally, it may have an eSE (embedded Secure Element) - a high security chip with defenses against HW attacks that can serve as a RoT. This device attestation format allows the attested data to be tagged at a security level from which it originates. In general, any discrete execution environment that has an identifiable security level can be considered an entity.

1.3. EAT Operating Models

At least the following three participants exist in all EAT operating models. Some operating models have additional participants.

The Entity. This is the phone, the IoT device, the sensor, the sub-assembly or such that the attestation provides information about.

The Manufacturer. The company that made the entity. This may be a chip vendor, a circuit board module vendor or a vendor of finished consumer products.

The Relying Party. The server, service or company that makes use of the information in the EAT about the entity.

In all operating models, the manufacturer provisions some secret attestation key material (AKM) into the entity during manufacturing. This might be during the manufacturer of a chip at a fabrication facility (fab) or during final assembly of a consumer product or any time in between. This attestation key material is used for signing EATs.

In all operating models, hardware and/or software on the entity create an EAT of the format described in this document. The EAT is always signed by the attestation key material provisioned by the manufacturer.

In all operating models, the relying party must end up knowing that the signature on the EAT is valid and consistent with data from claims in the EAT. This can happen in many different ways. Here are some examples.

- The EAT is transmitted to the relying party. The relying party gets corresponding key material (e.g. a root certificate) from the manufacturer. The relying party performs the verification.

- The EAT is transmitted to the relying party. The relying party transmits the EAT to a verification service offered by the manufacturer. The server returns the validated claims.

- The EAT is transmitted directly to a verification service, perhaps operated by the manufacturer or perhaps by another party. It verifies the EAT and makes the validated claims available to the relying party. It may even modify the claims in some way and re-sign the EAT (with a different signing key).

All these operating models are supported and there is no preference of one over the other. It is important to support this variety of operating models to generally facilitate deployment and to allow for some special scenarios. One special scenario has a validation service that is monetized, most likely by the manufacturer. In another, a privacy proxy service processes the EAT before it is transmitted to the relying party. In yet another, symmetric key material is used for signing. In this case the manufacturer should perform the verification, because any release of the key material would enable a participant other than the entity to create valid signed EATs.

1.4. What is Not Standardized

The following is not standardized for EAT, just the same they are not standardized for CWT or JWT.

1.4.1. Transmission Protocol

EATs may be transmitted by any protocol the same as CWTs and JWTs. For example, they might be added in extension fields of other protocols, bundled into an HTTP header, or just transmitted as files. This flexibility is intentional to allow broader adoption. This flexibility is possible because EAT's are self-secured with signing

(and possibly additionally with encryption and anti-replay). The transmission protocol is not required to fulfill any additional security requirements.

For certain devices, a direct connection may not exist between the EAT-producing device and the Relying Party. In such cases, the EAT should be protected against malicious access. The use of COSE and JOSE allows for signing and encryption of the EAT. Therefore, even if the EAT is conveyed through intermediaries between the device and Relying Party, such intermediaries cannot easily modify the EAT payload or alter the signature.

1.4.2. Signing Scheme

The term "signing scheme" is used to refer to the system that includes end-end process of establishing signing attestation key material in the entity, signing the EAT, and verifying it. This might involve key IDs and X.509 certificate chains or something similar but different. The term "signing algorithm" refers just to the algorithm ID in the COSE signing structure. No particular signing algorithm or signing scheme is required by this standard.

There are three main implementation issues driving this. First, secure non-volatile storage space in the entity for the attestation key material may be highly limited, perhaps to only a few hundred bits, on some small IoT chips. Second, the factory cost of provisioning key material in each chip or device may be high, with even millisecond delays adding to the cost of a chip. Third, privacy-preserving signing schemes like ECDAA (Elliptic Curve Direct Anonymous Attestation) are complex and not suitable for all use cases.

Over time to faciliate interoperability, some signing schemes may be defined in EAT profiles or other documents either in the IETF or outside.

2. Terminology

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.

This document reuses terminology from JWT [RFC7519], COSE [RFC8152], and CWT [RFC8392].

Claim Name. The human-readable name used to identify a claim.

Claim Key. The CBOR map key or JSON name used to identify a claim.

Claim Value. The CBOR map or JSON object value representing the value of the claim.

CWT Claims Set. The CBOR map or JSON object that contains the claims conveyed by the CWT or JWT.

Attestation Key Material (AKM). The key material used to sign the EAT token. If it is done symmetrically with HMAC, then this is a simple symmetric key. If it is done with ECC, such as an IEEE DevID [IDevID], then this is the private part of the EC key pair. If ECDAA is used, (e.g., as used by Enhanced Privacy ID, i.e. EPID) then it is the key material needed for ECDAA.

3. The Claims

This section describes new claims defined for attestation. It also mentions several claims defined by CWT and JWT that are particularly important for EAT.

Note also: * Any claim defined for CWT or JWT may be used in an EAT including those in the CWT [IANA.CWT.Claims] and JWT IANA [IANA.JWT.Claims] claims registries.

- All claims are optional
- No claims are mandatory
- All claims that are not understood by implementations MUST be ignored

CDDL along with text descriptions is used to define each claim indepdent of encoding. Each claim is defined as a CDDL group (the group is a general aggregation and type definition feature of CDDL). In the encoding section Section 4, the CDDL groups turn into CBOR map entries and JSON name/value pairs.

3.1. Token ID Claim (cti and jti)

CWT defines the "cti" claim. JWT defines the "jti" claim. These are equivalent to each other in EAT and carry a unique token identifier as they do in JWT and CWT. They may be used to defend against re use of the token but are distinct from the nonce that is used by the relying party to guarantee freshness and defend against replay.

3.2. Timestamp claim (iat)

The "iat" claim defined in CWT and JWT is used to indicate the date-of-creation of the token.

3.3. Nonce Claim (nonce)

All EATs should have a nonce to prevent replay attacks. The nonce is generated by the relying party, the end consumer of the token. It is conveyed to the entity over whatever transport is in use before the token is generated and then included in the token as the nonce claim.

This documents the nonce claim for registration in the IANA CWT claims registry. This is equivalent to the JWT nonce claim that is already registered.

The nonce must be at least 8 bytes (64 bits) as fewer are unlikely to be secure. A maximum of 64 bytes is set to limit the memory a constrained implementation uses. This size range is not set for the already-registered JWT nonce, but it should follow this size recommendation when used in an EAT.

Multiple nonces are allowed to accommodate multistage verification and consumption.

3.3.1. nonce CDDL

```
nonce-type = [ + bstr .size (8..64) ]

nonce-claim = (
    nonce => nonce-type
)
```

3.4. Universal Entity ID Claim (ueid)

UEID’s identify individual manufactured entities / devices such as a mobile phone, a water meter, a Bluetooth speaker or a networked security camera. It may identify the entire device or a submodule or subsystem. It does not identify types, models or classes of devices. It is akin to a serial number, though it does not have to be sequential.

UEID’s must be universally and globally unique across manufacturers and countries. UEIDs must also be unique across protocols and systems, as tokens are intended to be embedded in many different protocols and systems. No two products anywhere, even in completely different industries made by two different manufacturers in two different countries should have the same UEID (if they are not global

and universal in this way, then relying parties receiving them will have to track other characteristics of the device to keep devices distinct between manufacturers).

There are privacy considerations for UEID's. See Section 6.1.

The UEID should be permanent. It should never change for a given device / entity. In addition, it should not be reprogrammable. UEID's are variable length. All implementations MUST be able to receive UEID's that are 33 bytes long (1 type byte and 256 bits). The recommended maximum sent is also 33 bytes.

When the entity constructs the UEID, the first byte is a type and the following bytes the ID for that type. Several types are allowed to accommodate different industries and different manufacturing processes and to give options to avoid paying fees for certain types of manufacturer registrations.

Creation of new types requires a Standards Action [RFC8126].

Type Byte	Type Name	Specification
0x01	RAND	This is a 128, 192 or 256 bit random number generated once and stored in the device. This may be constructed by concatenating enough identifiers to make up an equivalent number of random bits and then feeding the concatenation through a cryptographic hash function. It may also be a cryptographic quality random number generated once at the beginning of the life of the device and stored. It may not be smaller than 128 bits.
0x02	IEEE EUI	This makes use of the IEEE company identification registry. An EUI is either an EUI-48, EUI-60 or EUI-64 and made up of an OUI, OUI-36 or a CID, different registered company identifiers, and some unique per-device identifier. EUIs are often the same as or similar to MAC addresses. This type includes MAC-48, an obsolete name for EUI-48. (Note that while devices with multiple network interfaces may have multiple MAC addresses, there is only one UEID for a device) [IEEE.802-2001], [OUI.Guide]
0x03	IMEI	This is a 14-digit identifier consisting of an 8-digit Type Allocation Code and a 6-digit serial number allocated by the manufacturer, which SHALL be encoded as a binary integer over 48 bits. The IMEI value encoded SHALL NOT include Luhn checksum or SVN information. [ThreeGPP.IMEI]

Table 1: UEID Composition Types

UEID's are not designed for direct use by humans (e.g., printing on the case of a device), so no textual representation is defined.

The consumer (the relying party) of a UEID MUST treat a UEID as a completely opaque string of bytes and not make any use of its internal structure. For example, they should not use the OUI part of a type 0x02 UEID to identify the manufacturer of the device. Instead they should use the oemid claim that is defined elsewhere. The reasons for this are:

- UEIDs types may vary freely from one manufacturer to the next.
- New types of UEIDs may be created. For example, a type 0x07 UEID may be created based on some other manufacturer registration scheme.

- Device manufacturers are allowed to change from one type of UEID to another anytime they want. For example, they may find they can optimize their manufacturing by switching from type 0x01 to type 0x02 or vice versa. The main requirement on the manufacturer is that UEIDs be universally unique.

3.4.1. ueid CDDL

```
ueid-claim = (
     ueid => bstr .size (7..33)
)
```

3.5. Origination Claim (origination)

This claim describes the parts of the device or entity that are creating the EAT. Often it will be tied back to the device or chip manufacturer. The following table gives some examples:

Name	Description
Acme-TEE	The EATs are generated in the TEE authored and configured by "Acme"
Acme-TPM	The EATs are generated in a TPM manufactured by "Acme"
Acme-Linux-Kernel	The EATs are generated in a Linux kernel configured and shipped by "Acme"
Acme-TA	The EATs are generated in a Trusted Application (TA) authored by "Acme"

TODO: consider a more structure approach where the name and the URI and other are in separate fields.

TODO: This needs refinement. It is somewhat parallel to issuer claim in CWT in that it describes the authority that created the token.

3.5.1. origination CDDL

```
origination-claim = (
    origination => string-or-uri
)
```

3.6. OEM Identification by IEEE (oemid)

The IEEE operates a global registry for MAC addresses and company IDs. This claim uses that database to identify OEMs. The contents of the claim may be either an IEEE MA-L, MA-M, MA-S or an IEEE CID

[IEEE.RA]. An MA-L, formerly known as an OUI, is a 24-bit value used as the first half of a MAC address. MA-M similarly is a 28-bit value uses as the first part of a MAC address, and MA-S, formerly known as OUI-36, a 36-bit value. Many companies already have purchased one of these. A CID is also a 24-bit value from the same space as an MA-L, but not for use as a MAC address. IEEE has published Guidelines for Use of EUI, OUI, and CID [OUI.Guide] and provides a lookup services [OUI.Lookup]

Companies that have more than one of these IDs or MAC address blocks should pick one and prefer that for all their devices.

Commonly, these are expressed in Hexadecimal Representation [IEEE.802-2001] also called the Canonical format. When this claim is encoded the order of bytes in the bstr are the same as the order in the Hexadecimal Representation. For example, an MA-L like "AC-DE-48" would be encoded in 3 bytes with values 0xAC, 0xDE, 0x48. For JSON encoded tokens, this is further base64url encoded.

3.6.1. oemid CDDL

```
oemid-claim = (
    oemid => bstr
)
```

3.7. The Security Level Claim (security-level)

EATs have a claim that roughly characterizes the device / entities ability to defend against attacks aimed at capturing the signing key, forging claims and at forging EATs. This is done by roughly defining four security levels as described below. This is similar to the security levels defined in the Metadata Service defined by the Fast Identity Online (FIDO) Alliance (TODO: reference).

These claims describe security environment and countermeasures available on the end-entity / client device where the attestation key reside and the claims originate.

1 - Unrestricted There is some expectation that implementor will protect the attestation signing keys at this level. Otherwise the EAT provides no meaningful security assurances.

2- Restricted Entities at this level should not be general-purpose operating environments that host features such as app download systems, web browsers and complex productivity applications. It is akin to the Secure Restricted level (see below) without the security orientation. Examples include a Wi-Fi subsystem, an IoT camera, or sensor device.

3 - Secure Restricted Entities at this level must meet the criteria defined by FIDO Allowed Restricted Operating Environments (TODO: reference). Examples include TEE's and schemes using virtualization-based security. Like the FIDO security goal, security at this level is aimed at defending well against large-scale network / remote attacks against the device.

4 - Hardware Entities at this level must include substantial defense against physical or electrical attacks against the device itself. It is assumed any potential attacker has captured the device and can disassemble it. Example include TPMs and Secure Elements.

This claim is not intended as a replacement for a proper end-device security certification schemes such as those based on FIPS (TODO: reference) or those based on Common Criteria (TODO: reference). The claim made here is solely a self-claim made by the Entity Originator.

3.7.1. security-level CDDL

```
security-level-type = &(
    unrestricted: 1,
    restricted: 2,
    secure-restricted: 3,
    hardware: 4
)

security-level-claim = (
    security-level => security-level-type
)
```

3.8. Secure Boot and Debug Enable State Claims (boot-state)

This claim is an array of five Boolean values indicating the boot and debug state of the entity.

3.8.1. Secure Boot Enabled

This indicates whether secure boot is enabled either for an entire device or an individual submodule. If it appears at the device level, then this means that secure boot is enabled for all submodules. Secure boot enablement allows a secure boot loader to authenticate software running either in a device or a submodule prior allowing execution.

3.8.2. Debug Disabled

This indicates whether debug capabilities are disabled for an entity (i.e. value of 'true'). Debug disablement is considered a prerequisite before an entity is considered operational.

3.8.3. Debug Disabled Since Boot

This claim indicates whether debug capabilities for the entity were not disabled in any way since boot (i.e. value of 'true').

3.8.4. Debug Permanent Disable

This claim indicates whether debug capabilities for the entity are permanently disabled (i.e. value of 'true'). This value can be set to 'true' also if only the manufacturer is allowed to enabled debug, but the end user is not.

3.8.5. Debug Full Permanent Disable

This claim indicates whether debug capabilities for the entity are permanently disabled (i.e. value of 'true'). This value can only be set to 'true' if no party can enable debug capabilities for the entity. Often this is implemented by blowing a fuse on a chip as fuses cannot be restored once blown.

3.8.6. boot-state CDDL

```
boot-state-type = [
    secure-boot-enabled => bool,
    debug-disabled => bool,
    debug-disabled-since-boot => bool,
    debug-permanent-disable => bool,
    debug-full-permanent-disable => bool
]

boot-state-claim = (
    boot-state => boot-state-type
)
```

3.9. The Location Claim (location)

The location claim is a CBOR-formatted object that describes the location of the device entity from which the attestation originates. It is comprised of a map of additional sub claims that represent the actual location coordinates (latitude, longitude and altitude). The location coordinate claims are consistent with the WGS84 coordinate

system [WGS84]. In addition, a sub claim providing the estimated accuracy of the location measurement is defined.

3.9.1. location CDDL

```
location-type = {
    latitude => number,
    longitude => number,
    ? altitude => number,
    ? accuracy => number,
    ? altitude-accuracy => number,
    ? heading => number,
    ? speed => number
}

location-claim = (
    location => location-type
)
```

3.10. The Age Claim (age)

The "age" claim contains a value that represents the number of seconds that have elapsed since the token was created, measurement was made, or location was obtained. Typical attestable values are sent as soon as they are obtained. However, in the case that such a value is buffered and sent at a later time and a sufficiently accurate time reference is unavailable for creation of a timestamp, then the age claim is provided.

3.10.1. age CDDL

```
age-claim = (
    age => uint
)
```

3.11. The Uptime Claim (uptime)

The "uptime" claim contains a value that represents the number of seconds that have elapsed since the entity or submod was last booted.

3.11.1. uptime CDDL

```
uptime-claim = (
    uptime => uint
)
```

3.12. The Submods Part of a Token (submods)

Some devices are complex, having many subsystems or submodules. A mobile phone is a good example. It may have several connectivity submodules for communications (e.g., Wi-Fi and cellular). It may have subsystems for low-power audio and video playback. It may have one or more security-oriented subsystems like a TEE or a Secure Element.

The claims for each these can be grouped together in a submodule.

The submods part of a token a single map/object with many entries, one per submodule. There is only one submods map in a token. It is identified by its specific label. It is a peer to other claims, but it is not called a claim because it is a container for a claim set rather than an individual claim. This submods part of a token allows what might be called recursion. It allows claim sets inside of claim sets inside of claims sets...

3.12.1. Two Types of Submodules

Each entry in the submod map one of two types:

- A non-token submodule that is a map or object directly containing claims for the submodule.

- A nested EAT that is a fully-formed, independently signed EAT token

3.12.1.1. Non-token Submodules

Essentially this type of submodule, is just a sub-map or sub-object containing claims. It is recognized from the other type by being a data item of type map in CBOR or by being an object in JSON.

The contents are claims about the submodule of types defined in this document or anywhere else claims types are defined.

3.12.1.2. Nested EATs

This type of submodule is a fully formed EAT as described here. In this case the submodule has key material distinct from the containing EAT token that allows it to sign on its own.

When an EAT is nested in another EAT as a submodule the nested EAT MUST use the CBOR CWT tag. This clearly distinguishes it from the non-token submodules.

3.12.2. No Inheritance

The subordinate modules do not inherit anything from the containing token. The subordinate modules must explicitly include all of their claims. This is the case even for claims like the nonce and age.

This rule is in place for simplicity. It avoids complex inheritance rules that might vary from one type of claim to another. (TODO: fix the boot claim which does have inheritance as currently described).

3.12.3. Security Levels

The security level of the non-token subordinate modules should always be less than or equal to that of the containing modules in the case of non-token submodules. It makes no sense for a module of lesser security to be signing claims of a module of higher security. An example of this is a TEE signing claims made by the non-TEE parts (e.g. the high-level OS) of the device.

The opposite may be true for the nested tokens. They usually have their own more secure key material. An example of this is an embedded secure element.

3.12.4. Submodule Names

The label or name for each submodule in the submods map is a text string naming the submodule. No submodules may have the same name.

3.12.5. submods CDDL

```
submods-type = { + submodule }

submodule = (
    submod_name => eat-claims / eat-token
)

submod_name = tstr / int

submods-part = (
    submods => submod-type
)
```

4. Encoding

This makes use of the types defined in CDDL Appendix D, Standard Prelude.

4.1. Common CDDL Types

```
string-or-uri = uri / tstr; See JSON section below for JSON encoding of string-or
-uri
```

4.2. CDDL for CWT-defined Claims

This section provides CDDL for the claims defined in CWT. It is non-normative as [RFC8392] is the authoritative definition of these claims.

```
rfc8392-claim //= ( issuer => text )
rfc8392-claim //= ( subject => text )
rfc8392-claim //= ( audience => text )
rfc8392-claim //= ( expiration => time )
rfc8392-claim //= ( not-before => time )
rfc8392-claim //= ( issued-at => time )
rfc8392-claim //= ( cwt-id => bytes )

issuer = 1
subject = 2
audience = 3
expiration = 4
not-before = 5
issued-at = 6
cwt-id = 7

cwt-claim = rfc8392-claim
```

4.3. JSON

4.3.1. JSON Labels

```
ueid = "ueid"
origination = "origination"
oemid = "oemid"
security-level = "security-level"
boot-state = "boot-state"
location = "location"
age = "age"
uptime = "uptime"
nested-eat = "nested-eat"
submods = "submods"

latitude = "lat"
longitude = "long""
altitude = "alt"
accuracy = "accry"
altitude-accuracy = "alt-accry"
heading = "heading"
speed = "speed"
```

4.3.2. JSON Interoperability

JSON should be encoded per RFC 8610 Appendix E. In addition, the following CDDL types are encoded in JSON as follows:

- bstr - must be base64url encoded
- time - must be encoded as NumericDate as described section 2 of [RFC7519].
- string-or-uri - must be encoded as StringOrURI as described section 2 of [RFC7519].

4.4. CBOR

4.4.1. CBOR Labels

```
ueid = To_be_assigned
origination = To_be_assigned
oemid = To_be_assigned
security-level = To_be_assigned
boot-state = To_be_assigned
location = To_be_assigned
age = To_be_assigned
uptime = To_be_assigned
submods = To_be_assigned
nonce = To_be_assigned

latitude = 1
longitude = 2
altitude = 3
accuracy = 4
altitude-accuracy = 5
heading = 6
speed = 7
```

4.4.2. CBOR Interoperability

Variations in the CBOR serializations supported in CBOR encoding and decoding are allowed and suggests that CBOR-based protocols specify how this variation is handled. This section specifies what formats MUST be supported in order to achieve interoperability.

The assumption is that the entity is likely to be a constrained device and relying party is likely to be a very capable server. The approach taken is that the entity generating the token can use whatever encoding it wants, specifically encodings that are easier to implement such as indefinite lengths. The relying party receiving the token must support decoding all encodings.

These rules cover all types used in the claims in this document. They also are recommendations for additional claims.

Canonical CBOR encoding, Preferred Serialization and Deterministically Encoded CBOR are explicitly NOT required as they would place an unnecessary burden on the entity implementation, particularly if the entity implementation is implemented in hardware.

- Integer Encoding (major type 0, 1) - The entity may use any integer encoding allowed by CBOR. The server MUST accept all integer encodings allowed by CBOR.

- String Encoding (major type 2 and 3) - The entity can use any string encoding allowed by CBOR including indefinite lengths. It

may also encode the lengths of strings in any way allowed by CBOR. The server must accept all string encodings.

- Major type 2, bstr, SHOULD be have tag 21 to indicate conversion to base64url in case that conversion is performed.

- Map and Array Encoding (major type 4 and 5) - The entity can use any array or map encoding allowed by CBOR including indefinite lengths. Sorting of map keys is not required. Duplicate map keys are not allowed. The server must accept all array and map encodings. The server may reject maps with duplicate map keys.

- Date and Time - The entity should send dates as tag 1 encoded as 64-bit or 32-bit integers. The entity may not send floating-point dates. The server must support tag 1 epoch-based dates encoded as 64-bit or 32-bit integers. The entity may send tag 0 dates, however tag 1 is preferred. The server must support tag 0 UTC dates.

- URIs - URIs should be encoded as text strings and marked with tag 32.

- Floating Point - The entity may use any floating-point encoding. The relying party must support decoding of all types of floating-point.

- Other types - Use of Other types like bignums, regular expressions and such, SHOULD NOT be used. The server MAY support them but is not required to so interoperability is not guaranteed.

4.5. Collected CDDL

A generic-claim is any CBOR map entry or JSON name/value pair.

```
eat-claims = { ; the top-level payload that is signed using COSE or JOSE
    * claim
}

claim = (
    ueid-claim //
    origination-claim //
    oemid-claim //
    security-level-claim //
    boot-state-claim //
    location-claim //
    age-claim //
    uptime-claim //
    submods-part //
    cwt-claim //
    generic-claim-type //
)

eat-token ; This is a set of eat-claims signed using COSE
```

TODO: copy the rest of the CDDL here (wait until the CDDL is more settled so as to avoid copying multiple times)

5. IANA Considerations

5.1. Reuse of CBOR Web Token (CWT) Claims Registry

Claims defined for EAT are compatible with those of CWT so the CWT Claims Registry is re used. No new IANA registry is created. All EAT claims should be registered in the CWT and JWT Claims Registries.

5.1.1. Claims Registered by This Document

- Claim Name: UEID
- Claim Description: The Universal Entity ID
- JWT Claim Name: N/A
- Claim Key: 8
- Claim Value Type(s): byte string
- Change Controller: IESG
- Specification Document(s): *this document*

TODO: add the rest of the claims in here

6. Privacy Considerations

Certain EAT claims can be used to track the owner of an entity and therefore, implementations should consider providing privacy-preserving options dependent on the intended usage of the EAT. Examples would include suppression of location claims for EAT's provided to unauthenticated consumers.

6.1. UEID Privacy Considerations

A UEID is usually not privacy-preserving. Any set of relying parties that receives tokens that happen to be from a single device will be able to know the tokens are all from the same device and be able to track the device. Thus, in many usage situations ueid violates governmental privacy regulation. In other usage situations UEID will not be allowed for certain products like browsers that give privacy for the end user. It will often be the case that tokens will not have a UEID for these reasons.

There are several strategies that can be used to still be able to put UEID's in tokens:

- The device obtains explicit permission from the user of the device to use the UEID. This may be through a prompt. It may also be through a license agreement. For example, agreements for some online banking and brokerage services might already cover use of a UEID.

- The UEID is used only in a particular context or particular use case. It is used only by one relying party.

- The device authenticates the relying party and generates a derived UEID just for that particular relying party. For example, the relying party could prove their identity cryptographically to the device, then the device generates a UEID just for that relying party by hashing a proofed relying party ID with the main device UEID.

Note that some of these privacy preservation strategies result in multiple UEIDs per device. Each UEID is used in a different context, use case or system on the device. However, from the view of the relying party, there is just one UEID and it is still globally universal across manufacturers.

7. Security Considerations

The security considerations provided in Section 8 of [RFC8392] and Section 11 of [RFC7519] apply to EAT in its CWT and JWT form, respectively. In addition, implementors should consider the following.

7.1. Key Provisioning

Private key material can be used to sign and/or encrypt the EAT, or can be used to derive the keys used for signing and/or encryption. In some instances, the manufacturer of the entity may create the key material separately and provision the key material in the entity itself. The manfuacturer of any entity that is capable of producing an EAT should take care to ensure that any private key material be suitably protected prior to provisioning the key material in the entity itself. This can require creation of key material in an enclave (see [RFC4949] for definition of "enclave"), secure transmission of the key material from the enclave to the entity using an appropriate protocol, and persistence of the private key material in some form of secure storage to which (preferably) only the entity has access.

7.1.1. Transmission of Key Material

Regarding transmission of key material from the enclave to the entity, the key material may pass through one or more intermediaries. Therefore some form of protection ("key wrapping") may be necessary. The transmission itself may be performed electronically, but can also be done by human courier. In the latter case, there should be minimal to no exposure of the key material to the human (e.g. encrypted portable memory). Moreover, the human should transport the key material directly from the secure enclave where it was created to a destination secure enclave where it can be provisioned.

7.2. Transport Security

As stated in Section 8 of [RFC8392], "The security of the CWT relies upon on the protections offered by COSE". Similar considerations apply to EAT when sent as a CWT. However, EAT introduces the concept of a nonce to protect against replay. Since an EAT may be created by an entity that may not support the same type of transport security as the consumer of the EAT, intermediaries may be required to bridge communications between the entity and consumer. As a result, it is RECOMMENDED that both the consumer create a nonce, and the entity leverage the nonce along with COSE mechanisms for encryption and/or signing to create the EAT.

Similar considerations apply to the use of EAT as a JWT. Although the security of a JWT leverages the JSON Web Encryption (JWE) and JSON Web Signature (JWS) specifications, it is still recommended to make use of the EAT nonce.

7.3. Multiple EAT Consumers

In many cases, more than one EAT consumer may be required to fully verify the entity attestation. Examples include individual consumers for nested EATs, or consumers for individual claims with an EAT. When multiple consumers are required for verification of an EAT, it is important to minimize information exposure to each consumer. In addition, the communication between multiple consumers should be secure.

For instance, consider the example of an encrypted and signed EAT with multiple claims. A consumer may receive the EAT (denoted as the "receiving consumer"), decrypt its payload, verify its signature, but then pass specific subsets of claims to other consumers for evaluation ("downstream consumers"). Since any COSE encryption will be removed by the receiving consumer, the communication of claim subsets to any downstream consumer should leverage a secure protocol (e.g.one that uses transport-layer security, i.e. TLS),

However, assume the EAT of the previous example is hierarchical and each claim subset for a downstream consumer is created in the form of a nested EAT. Then transport security between the receiving and downstream consumers is not strictly required. Nevertheless, downstream consumers of a nested EAT should provide a nonce unique to the EAT they are consuming.

8. References

8.1. Normative References

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

[IANA.JWT.Claims]
IANA, "JSON Web Token (JWT) Claims",
<https://www.iana.org/assignments/jwt>.

[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/info/rfc2119>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[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/info/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/info/rfc8392>.

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

[ThreeGPP.IMEI] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification", 2019, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=729>.

[TIME_T] The Open Group Base Specifications, "Vol. 1: Base Definitions, Issue 7", Section 4.15 'Seconds Since the Epoch', IEEE Std 1003.1, 2013 Edition, 2013, <http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15>.

[WGS84] National Imagery and Mapping Agency, "National Imagery and Mapping Agency Technical Report 8350.2, Third Edition", 2000, <http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>.

8.2. Informative References

[ASN.1] International Telecommunication Union, "Information Technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.

[BirthdayAttack] "Birthday attack", <https://en.wikipedia.org/wiki/Birthday_attack.>.

[ECMAScript] "Ecma International, "ECMAScript Language Specification, 5.1 Edition", ECMA Standard 262", June 2011, <http://www.ecma-international.org/ecma-262/5.1/ECMA-262.pdf>.

[IDevID] "IEEE Standard, "IEEE 802.1AR Secure Device Identifier"", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[IEEE.802-2001] "IEEE Standard For Local And Metropolitan Area Networks Overview And Architecture", 2007, <https://webstore.ansi.org/standards/ieee/ieee8022001r2007>.

[IEEE.RA] "IEEE Registration Authority", <https://standards.ieee.org/products-services/regauth/index.html>.

[OUI.Guide] "Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)", August 2017, <https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/eui.pdf>.

[OUI.Lookup] "IEEE Registration Authority Assignments", <https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[Webauthn] Worldwide Web Consortium, "Web Authentication: A Web API for accessing scoped credentials", 2016.

Appendix A. Examples

A.1. Very Simple EAT

This is shown in CBOR diagnostic form. Only the payload signed by COSE is shown.

```
{
   / nonce /                  9:h'948f8860d13a463e8e',
   / UEID /                  10:h'0198f50a4ff6c05861c8860d13a638ea4fe2f',
   / boot-state /            12:{true, true, true, true, false}
   / time stamp (iat) /       6:1526542894,
}
```

A.2. Example with Submodules, Nesting and Security Levels

```
{
   / nonce /                  9:h'948f8860d13a463e8e',
   / UEID /                  10:h'0198f50a4ff6c05861c8860d13a638ea4fe2f',
   / boot-state /            12:{true, true, true, true, false}
   / time stamp (iat) /       6:1526542894,
   / seclevel /              11:3, / secure restricted OS /

   / submods / 17:
      {
         / first submod, an Android Application / "Android App Foo" :  {
            / seclevel /      11:1, / unrestricted /
            / app data /  -70000:'text string'
         },
         / 2nd submod, A nested EAT from a secure element / "Secure Element Eat"
:
            / eat /         61( 18(
                                / an embedded EAT, bytes of which are not shown /
                            ))
         / 3rd submod, information about Linux Android / "Linux Android": {
            / seclevel /              11:1, / unrestricted /
            / custom - release /  -80000:'8.0.0',
            / custom - version /  -80001:'4.9.51+'
         }
      }
}
```

Appendix B. UEID Design Rationale

B.1. Collision Probability

This calculation is to determine the probability of a collision of UEIDs given the total possible entity population and the number of entities in a particular entity management database.

Three different sized databases are considered. The number of devices per person roughly models non-personal devices such as traffic lights, devices in stores they shop in, facilities they work in and so on, even considering individual light bulbs. A device may have individually attested subsystems, for example parts of a car or a mobile phone. It is assumed that the largest database will have at most 10% of the world's population of devices. Note that databases that handle more than a trillion records exist today.

The trillion-record database size models an easy-to-imagine reality over the next decades. The quadrillion-record database is roughly at the limit of what is imaginable and should probably be accommodated. The 100 quadrillion datadbase is highly speculative perhaps involving nanorobots for every person, livestock animal and domesticated bird. It is included to round out the analysis.

Note that the items counted here certainly do not have IP address and are not individually connected to the network. They may be connected to internal buses, via serial links, Bluetooth and so on. This is not the same problem as sizing IP addresses.

People	Devices / Person	Subsystems / Device	Database Portion	Database Size
10 billion	100	10	10%	trillion (10^12)
10 billion	100,000	10	10%	quadrillion (10^15)
100 billion	1,000,000	10	10%	100 quadrillion (10^17)

This is conceptually similar to the Birthday Problem where m is the number of possible birthdays, always 365, and k is the number of people. It is also conceptually similar to the Birthday Attack where collisions of the output of hash functions are considered.

The proper formula for the collision calculation is

$$p = 1 - e^{-k^2/(2n)}$$

p Collision Probability
n Total possible population
k Actual population

However, for the very large values involved here, this formula requires floating point precision higher than commonly available in calculators and SW so this simple approximation is used. See [BirthdayAttack].

```
p = k^2 / 2n
```

For this calculation:

```
p  Collision Probability
n  Total population based on number of bits in UEID
k  Population in a database
```

Database Size	128-bit UEID	192-bit UEID	256-bit UEID
trillion (10^12)	2 * 10^-15	8 * 10^-35	5 * 10^-55
quadrillion (10^15)	2 * 10^-09	8 * 10^-29	5 * 10^-49
100 quadrillion (10^17)	2 * 10^-05	8 * 10^-25	5 * 10^-45

Next, to calculate the probability of a collision occurring in one year's operation of a database, it is assumed that the database size is in a steady state and that 10% of the database changes per year. For example, a trillion record database would have 100 billion states per year. Each of those states has the above calculated probability of a collision.

This assumption is a worst-case since it assumes that each state of the database is completely independent from the previous state. In reality this is unlikely as state changes will be the addition or deletion of a few records.

The following tables gives the time interval until there is a probability of a collision based on there being one tenth the number of states per year as the number of records in the database.

```
t = 1 / ((k / 10) * p)

t  Time until a collision
p  Collision probability for UEID size
k  Database size
```

Database Size	128-bit UEID	192-bit UEID	256-bit UEID
trillion (10^12)	60,000 years	10^24 years	10^44 years
quadrillion (10^15)	8 seconds	10^14 years	10^34 years
100 quadrillion (10^17)	8 microseconds	10^11 years	10^31 years

Clearly, 128 bits is enough for the near future thus the requirement that UEIDs be a minimum of 128 bits.

There is no requirement for 256 bits today as quadrillion-record databases are not expected in the near future and because this time-to-collision calculation is a very worst case. A future update of the standard may increase the requirement to 256 bits, so there is a requirement that implementations be able to receive 256-bit UEIDs.

B.2. No Use of UUID

A UEID is not a UUID [RFC4122] by conscious choice for the following reasons.

UUIDs are limited to 128 bits which may not be enough for some future use cases.

Today, cryptographic-quality random numbers are available from common CPUs and hardware. This hardware was introduced between 2010 and 2015. Operating systems and cryptographic libraries give access to this hardware. Consequently, there is little need for implementations to construct such random values from multiple sources on their own.

Version 4 UUIDs do allow for use of such cryptographic-quality random numbers, but do so by mapping into the overall UUID structure of time and clock values. This structure is of no value here yet adds complexity. It also slightly reduces the number of actual bits with entropy.

UUIDs seem to have been designed for scenarios where the implementor does not have full control over the environment and uniqueness has to be constructed from identifiers at hand. UEID takes the view that hardware, software and/or manufacturing process directly implement UEID in a simple and direct way. It takes the view that cryptographic quality random number generators are readily available as they are implemented in commonly used CPU hardware.

Appendix C. Changes from Previous Drafts

The following is a list of known changes from the previous drafts. This list is non-authoritative. It is meant to help reviewers see the significant differences.

C.1. From draft-rats-eat-01

- Added UEID design rationale appendix

C.2. From draft-mandyam-rats-eat-00

This is a fairly large change in the orientation of the document, but not new claims have been added.

- Separate information and data model using CDDL.
- Say an EAT is a CWT or JWT
- Use a map to structure the boot_state and location claims

C.3. From draft-ietf-rats-eat-01

- Clarifications and corrections for OEMID claim
- Minor spelling and other fixes
- Add the nonce claim, clarify jti claim

C.4. From draft-ietf-rats-eat-02

- Roll all EUIs back into one UEID type
- UEIDs can be one of three lengths, 128, 192 and 256.
- Added appendix justifying UEID design and size.
- Submods part now includes nested eat tokens so they can be named and there can be more tha one of them
- Lots of fixes to the CDDL
- Added security considerations

Authors' Addresses

Giridhar Mandyam
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, California
USA

Phone: +1 858 651 7200
EMail: mandyam@qti.qualcomm.com

Laurence Lundblade
Security Theory LLC

EMail: lgl@island-resort.com

Miguel Ballesteros
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, California
USA

Phone: +1 858 651 4299
EMail: mballest@qti.qualcomm.com

Jeremy O'Donoghue
Qualcomm Technologies Inc.
279 Farnborough Road
Farnborough GU14 7LS
United Kingdom

Phone: +44 1252 363189
EMail: jodonogh@qti.qualcomm.com

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 3, 2021

H. Birkholz
M. Eckel
Fraunhofer SIT
E. Voit
S. Bhandari
B. Sulzen
Cisco
L. Xia
Huawei
T. Laffey
HPE
G. Fedorkow
Juniper

September 30, 2020

A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs
draft-ietf-rats-yang-tpm-charra-03

Abstract

This document defines a YANG RPC and a minimal datastore required to retrieve attestation evidence about integrity measurements from a device following the operational context defined in [I-D.ietf-rats-tpm-based-network-device-attest]. Complementary measurement logs are also provided by the YANG RPC originating from one or more roots of trust of measurement. The module defined requires at least one TPM 1.2 or TPM 2.0 and corresponding Trusted Software Stack included in the device components of the composite device the YANG server is running on.

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 April 3, 2021.

Table of Contents

1. Introduction

This document is based on the terminology defined in the [I-D.ietf-rats-architecture] and uses the operational context defined in [I-D.ietf-rats-tpm-based-network-device-attest] as well as the interaction model and information elements defined in [I-D.birkholz-rats-reference-interaction-model]. The currently supported hardware security modules (HWM) are the Trusted Platform Module (TPM) [TPM1.2] and [TPM2.0] specified by the Trusted Computing Group (TCG). One ore more TPMs embedded in the components of a composite device - sometimes also referred to as an aggregate device - are required in order to use the YANG module defined in this document. A TPM is used as a root of trust for reporting (RTR) in order to retrieve attestation evidence from a composite device (quote primitive operation). Additionally, it is used as a root of trust

for storage (RTS) in order to retain shielded secrets and store system measurements using a folding hash function (extend primitive operation).

1.1. Requirements notation

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

2. The YANG Module for Basic Remote Attestation Procedures

One or more TPMs MUST be embedded in the composite device that is providing attestation evidence via the YANG module defined in this document. The ietf-basic-remote-attestation YANG module enables a composite device to take on the role of Claimant and Attester in accordance with the Remote Attestation Procedures (RATS) architecture [I-D.ietf-rats-architecture] and the corresponding challenge-response interaction model defined in the [I-D.birkholz-rats-reference-interaction-model] document. A fresh nonce with an appropriate amount of entropy MUST be supplied by the YANG client in order to enable a proof-of-freshness with respect to the attestation evidence provided by the attester running the YANG datastore. The functions of this YANG module are restricted to 0-1 TPMs per hardware component.

2.1. Tree Diagram

```
  module: ietf-tpm-remote-attestation
    +--rw rats-support-structures
       +--rw compute-nodes!
       |  +--ro compute-node* [node-id]
       |     +--ro node-id                string
       |     +--ro node-physical-index?   int32 {ietfhw:entity-mib}?
       |     +--ro node-name?             string
       |     +--ro node-location?         string
       +--rw tpms
       |  +--rw tpm* [tpm-name]
       |     +--rw tpm-name                string
       |     +--ro hardware-based?         boolean
       |     +--ro tpm-physical-index?     int32 {ietfhw:entity-mib}?
       |     +--ro tpm-path?               string
       |     +--ro compute-node            compute-node-ref
       |     +--ro tpm-manufacturer?       string
       |     +--rw tpm-firmware-version    identityref
       |     +--rw TPM12-hash-algo?        identityref
```

```
      |     +--rw TPM12-pcrs*             pcr
      |     +--rw tpm20-pcr-bank* [TPM20-hash-algo]
      |     |  +--rw TPM20-hash-algo    identityref
      |     |  +--rw pcr-index*         tpm:pcr
      |     +--ro tpm-status              enumeration
      |     +--rw certificates
      |        +--rw certificate* [certificate-name]
      |           +--rw certificate-name           string
      |           +--rw certificate-keystore-ref?   leafref
      |           +--rw certificate-type?           enumeration
      +--rw attester-supported-algos
         +--rw tpm12-asymmetric-signing*   identityref {taa:TPM12}?
         +--rw tpm12-hash*                 identityref {taa:TPM12}?
         +--rw tpm20-asymmetric-signing*   identityref {taa:TPM20}?
         +--rw tpm20-hash*                 identityref {taa:TPM20}?

   rpcs:
     +---x tpm12-challenge-response-attestation {taa:TPM12}?
     |  +---w input
     |  |  +---w tpm12-attestation-challenge
     |  |     +---w pcr-index*          pcr
     |  |     +---w nonce-value         binary
     |  |     +---w add-version?        boolean
     |  |     +---w certificate-name*   certificate-name-ref
     |  +--ro output
     |     +--ro tpm12-attestation-response* []
     |        +--ro certificate-name?            certificate-name-ref
     |        +--ro up-time?                     uint32
     |        +--ro node-id?                     string
     |        +--ro node-physical-index?         int32
     |        |       {ietfhw:entity-mib}?
     |        +--ro fixed?                       binary
     |        +--ro external-data?               binary
     |        +--ro signature-size?              uint32
     |        +--ro signature?                   binary
     |        +--ro (tpm12-quote)
     |           +--:(tpm12-quote1)
     |           |  +--ro version* []
     |           |  |  +--ro major?       uint8
     |           |  |  +--ro minor?       uint8
     |           |  |  +--ro rev-Major?   uint8
     |           |  |  +--ro rev-Minor?   uint8
     |           |  +--ro digest-value?          binary
     |           |  +--ro TPM_PCR_COMPOSITE* []
     |           |     +--ro pcr-index*         pcr
     |           |     +--ro value-size?        uint32
     |           |     +--ro tpm12-pcr-value*   binary
     |           +--:(tpm12-quote2)
```

```
|              +--ro tag?                    uint8
|              +--ro pcr-index*              pcr
|              +--ro locality-at-release?    uint8
|              +--ro digest-at-release?      binary
+---x tpm20-challenge-response-attestation {taa:TPM20}?
|  +---w input
|  |  +---w tpm20-attestation-challenge
|  |     +---w nonce-value            binary
|  |     +---w tpm20-pcr-selection* []
|  |     |  +---w TPM20-hash-algo?   identityref
|  |     |  +---w pcr-index*         tpm:pcr
|  |     +---w certificate-name*      certificate-name-ref
|  +--ro output
|     +--ro tpm20-attestation-response* []
|        +--ro certificate-name?      certificate-name-ref
|        +--ro TPMS_QUOTE_INFO        binary
|        +--ro quote-signature?       binary
|        +--ro up-time?               uint32
|        +--ro node-id?               string
|        +--ro node-physical-index?   int32 {ietfhw:entity-mib}?
|        +--ro unsigned-pcr-values* []
|           +--ro TPM20-hash-algo?   identityref
|           +--ro pcr-values* [pcr-index]
|              +--ro pcr-index    pcr
|              +--ro pcr-value?   binary
+---x log-retrieval
   +---w input
   |  +---w log-selector* []
   |  |  +---w tpm-name*                  string
   |  |  +---w (index-type)?
   |  |  |  +--:(last-entry)
   |  |  |  |  +---w last-entry-value?    binary
   |  |  |  +--:(index)
   |  |  |  |  +---w last-index-number?   uint64
   |  |  |  +--:(timestamp)
   |  |  |     +---w timestamp?           yang:date-and-time
   |  |  +---w log-entry-quantity?        uint16
   |  +---w log-type        identityref
   +--ro output
      +--ro system-event-logs
         +--ro node-data* []
            +--ro tpm-name?     string
            +--ro up-time?      uint32
            +--ro log-result
               +--ro (attested_event_log_type)
                  +--:(bios)
                  |  +--ro bios-event-logs
                  |     +--ro bios-event-entry* [event-number]
```

```
                        |        +--ro event-number    uint32
                        |        +--ro event-type?     uint32
                        |        +--ro pcr-index?      pcr
                        |        +--ro digest-list* []
                        |        |  +--ro hash-algo?   identityref
                        |        |  +--ro digest*      binary
                        |        +--ro event-size?     uint32
                        |        +--ro event-data*     uint8
                        +--:(ima)
                        |  +--ro ima-event-logs
                        |     +--ro ima-event-entry* [event-number]
                        |        +--ro event-number                uint64
                        |        +--ro ima-template?               string
                        |        +--ro filename-hint?              string
                        |        +--ro filedata-hash?              binary
                        |        +--ro filedata-hash-algorithm?    string
                        |        +--ro template-hash-algorithm?    string
                        |        +--ro template-hash?              binary
                        |        +--ro pcr-index?                  pcr
                        |        +--ro signature?                  binary
                        +--:(netequip_boot)
                           +--ro boot-event-logs
                              +--ro boot-event-entry* [event-number]
                                 +--ro event-number                uint64
                                 +--ro filename-hint?              string
                                 +--ro filedata-hash?              binary
                                 +--ro filedata-hash-algorithm?    string
                                 +--ro file-version?               string
                                 +--ro file-type?                  string
                                 +--ro pcr-index?                  pcr
```

2.2. YANG Modules

2.2.1. ietf-tpm-remote-attestation

This YANG module imports modules from [RFC6991], [RFC8348], [I-D.ietf-netconf-keystore], ietf-tcg-algs.yang.

2.2.1.1. Identities

This module supports the following types of attestation event logs: <ima>, <bios>, and <netequip_boot>.

2.2.1.2. RPCs

<tpm12-challenge-response-attestation> - Allows a Verifier to request a quote of PCRs from a TPM1.2 compliant cryptoprocessor. When one or

more <certificate-name> is not provided, all TPM1.2 compliant cryptoprocessors will respond.

<tpm20-challenge-response-attestation> - Allows a Verifier to request a quote of PCRs from a TPM2.0 compliant cryptoprocessor. When one or more <certificate-name> is not provided, all TPM2.0 compliant cryptoprocessors will respond.

<log-retrieval> - Allows a Verifier to acquire the evidence which was extended into specific PCRs.

2.2.1.3. Data Nodes

container <rats-support-structures> - This exists when there are more than one TPM for a particular Attester. This allows each specific TPM to identify on which <compute-node> it belongs.

container <tpms> - Provides configuration and operational details for each supported TPM, including the tpm-firmware-version, PCRs which may be quoted, certificates which are associated with that TPM, and the current operational status. Of note is the certificates which are associated with that TPM. As a certificate is associated with a single Attestation key, knowledge of the certificate allows a specific TPM to be identified.

container <attester-supported-algos> - Identifies which TCG algorithms are available for use the Attesting platform. This allows an operator to limit algorithms available for use by RPCs to just a desired set from the universe of all allowed by TCG.

2.2.1.4. YANG Module

```
<CODE BEGINS> file ietf-tpm-remote-attestation@2020-09-18.yang
module ietf-tpm-remote-attestation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-tpm-remote-attestation";
  prefix "tpm";

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-hardware {
    prefix ietfhw;
  }
  import ietf-keystore {
    prefix ks;
  }
  import ietf-tcg-algs {
    prefix taa;
```

```
  }

  organization
   "IETF RATS (Remote ATtestation procedureS) Working Group";

  contact
   "WG Web  : <http://datatracker.ietf.org/wg/rats/>
    WG List : <mailto:rats@ietf.org>
    Author  : Eric Voit <evoit@cisco.com>
    Author  : Henk Birkholz <henk.birkholz@sit.fraunhofer.de>
    Author  : Michael Eckel <michael.eckel@sit.fraunhofer.de>
    Author  : Shwetha Bhandari <shwethab@cisco.com>
    Author  : Bill Sulzen <bsulzen@cisco.com>
    Author  : Liang Xia (Frank) <frank.xialiang@huawei.com>
    Author  : Tom Laffey <tom.laffey@hpe.com>
    Author  : Guy Fedorkow <gfedorkow@juniper.net>";

  description
    "A YANG module to enable a TPM 1.2 and TPM 2.0 based
     remote attestation procedure using a challenge-response
     interaction model and the TPM 1.2 and TPM 2.0 Quote
     primitive operations.

     Copyright (c) 2020 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
     itself for full legal notices.

     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 (RFC 2119)
```

```
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

  revision "2020-09-18" {
    description
      "Initial version";
    reference
      "draft-ietf-rats-yang-tpm-charra";
  }


  /*****************/
  /*   Typedefs    */
  /*****************/

  typedef pcr {
    type uint8 {
      range "0..31";
    }
    description
      "Valid index number for a PCR.  At this point 0-31 is viable.";
  }

  typedef compute-node-ref {
    type leafref {
      path "/tpm:rats-support-structures/tpm:compute-nodes" +
           "/tpm:compute-node/tpm:node-name";
    }
    description
      "This type is used to reference a hardware node.  It is quite
      possible this leafref will eventually point to another YANG
      module's node.";
  }

  typedef certificate-name-ref {
    type leafref {
      path "/tpm:rats-support-structures/tpm:tpms/tpm:tpm" +
           "/tpm:certificates/tpm:certificate/tpm:certificate-name";
    }
    description
      "A type which allows identification of a TPM based certificate.";
  }


  /******************/
  /*   Identities   */
  /******************/
```

```
  identity attested_event_log_type {
    description
      "Base identity allowing categorization of the reasons why and
      attested measurement has been taken on an Attester.";
  }

  identity ima {
    base attested_event_log_type;
    description
      "An event type recorded in IMA.";
  }

  identity bios {
    base attested_event_log_type;
    description
      "An event type associated with BIOS/UEFI.";
  }

  identity netequip_boot {
    base attested_event_log_type;
    description
      "An event type associated with Network Equipment Boot.";
  }

  /*****************/
  /*   Groupings   */
  /*****************/

  grouping TPM20-asymmetric-signing-algo {
    description
      "The signature scheme that is used to sign the TPM2 Quote
      information response.";
    leaf TPM20-asymmetric-signing-algo {
      must "/tpm:rats-support-structures/tpm:attester-supported-algos"
         + "/tpm:tpm20-asymmetric-signing" {
         error-message "Not a platform supported " +
                       "TPM20-asymmetric-signing-algo";
      }
      type identityref {
        base taa:asymmetric;
      }
      description
        "The signature scheme that is used to sign the TPM2.0
         Quote information response. This must be one of those
         supported by a platform.";
    default taa:TPM_ALG_RSA;
    }
  }
```

```
  grouping TPM12-asymmetric-signing-algo {
    description
      "The signature scheme that is used to sign the TPM12 Quote
      information response.";
    leaf TPM12-asymmetric-signing-algo {
      must "/tpm:rats-support-structures/tpm:attester-supported-algos"
         + "/tpm:tpm12-asymmetric-signing" {
         error-message "Not a platform supported " +
                       "TPM12-asymmetric-signing-algo";
      }
      type identityref {
        base taa:asymmetric;
      }
      description
        "The signature scheme that is used to sign the TPM1.2
         Quote information response. This must be one of those
         supported by a platform.";
    default taa:TPM_ALG_RSA;
    }
  }

  grouping TPM20-hash-algo {
    description
      "The cryptographic algorithm used to hash the TPM2 PCRs.  This
      must be from the list of platform supported options.";
    leaf TPM20-hash-algo {
      must "/tpm:rats-support-structures/tpm:attester-supported-algos"
         + "/tpm:tpm20-hash" {
         error-message "Not a platform supported TPM20-hash-algo";
      }
      type identityref {
        base taa:hash;
      }
      description
        "The hash scheme that is used to hash a TPM1.2 PCR. This
        must be one of those supported by a platform.";
    default taa:TPM_ALG_SHA256;
    }
  }

  grouping TPM12-hash-algo {
    description
      "The cryptographic algorithm used to hash the TPM1.2 PCRs.";
    leaf TPM12-hash-algo {
      must "/tpm:rats-support-structures/tpm:attester-supported-algos"
         + "/tpm:tpm12-hash" {
         error-message "Not a platform supported TPM12-hash-algo";
      }
```

```
      type identityref {
        base taa:hash;
      }
      description
        "The hash scheme that is used to hash a TPM1.2 PCR. This
        must be one of those supported by a platform.  This assumes
        that an algorithm other than SHA1 can be supported on some
        TPM1.2 cryptoprocessor variant.";
      default taa:TPM_ALG_SHA1;
    }
  }

  grouping nonce {
    description
      "A nonce to show freshness and to allow the detection
      of replay attacks.";
    leaf nonce-value {
      type binary;
      mandatory true;
      description
        "This nonce SHOULD be generated via a registered
         cryptographic-strength algorithm. In consequence,
         the length of the nonce depends on the hash algorithm
         used. The algorithm used in this case is independent
         from the hash algorithm used to create the hash-value
         in the response of the attestor.";
    }
  }

  grouping tpm12-pcr-selection {
    description
      "A Verifier can request one or more PCR values using its
       individually created Attestation Key Certificate (AC).
       The corresponding selection filter is represented in this
       grouping.
       Requesting a PCR value that is not in scope of the AC used,
       detailed exposure via error msg should be avoided.";
    leaf-list pcr-index {
      /*  the following XPATH must be updated to ensure that only
            selectable PCRs are allowed in the RPC
      must "/tpm:rats-support-structures/tpm:tpms" +
           "/tpm:tpm[tpm-name = current()]" +
           "/tpm:tpm[TPM12-pcrs = current()]" {
        error-message "Acquiring this PCR index is not supported";
      }
      */
      type pcr;
      description
```

```
        "The numbers/indexes of the PCRs. At the moment this is limited
        to 32.";
    }
  }

  grouping tpm20-pcr-selection {
    description
      "A Verifier can acquire one or more PCR values, which are hashed
       together in a TPM2B_DIGEST coming from the TPM2.  The selection
       list of desired PCRs and the Hash Algorithm is represented in
       this grouping.";
    list tpm20-pcr-selection {
      unique "TPM20-hash-algo";
      description
        "Specifies the list of PCRs and Hash Algorithms that can be
        returned within a TPM2B_DIGEST.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
         TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.9.7";
      uses TPM20-hash-algo;
      leaf-list pcr-index {
        /*  the following XPATH must be updated to ensure that only
            selectable PCRs are allowed in the RPC
        must "/tpm:rats-support-structures/tpm:tpms" +
             "/tpm:tpm[tpm-name = current()]" +
             "/tpm:tpm20-pcr-bank[pcr-index = current()]" {
          error-message "Acquiring this PCR index is not supported";
        }
        */
        type tpm:pcr;
        description
          "The numbers of the PCRs that which are being tracked
          with a hash based on the TPM20-hash-algo.";
      }
    }
  }

  grouping certificate-name-ref {
    description
      "Identifies a certificate in a keystore.";
    leaf certificate-name {
      type certificate-name-ref;
        description
          "Identifies a certificate in a keystore.";
    }
  }

  grouping tpm-name {
```

```
    description
      "A unique TPM on a device.";
    leaf tpm-name {
      type string;
      description
        "Unique system generated name for a TPM on a device.";
    }
  }

  grouping tpm-name-selector {
    description
      "One or more TPM on a device.";
    leaf-list tpm-name {
      type string;
      config false;
      description
        "Name of one or more unique TPMs on a device.  If this object
        exists, a selection should pull only the objects related to
        these TPM(s).  If it does not exist, all qualifying TPMs that
        are 'hardware-based' equals true on the device are selected.";
    }
  }

  grouping compute-node-identifier {
    description
      "In a distributed system with multiple compute nodes
       this is the node identified by name and physical-index.";
    leaf node-id {
       type string;
       description
         "ID of the compute node, such as Board Serial Number.";
    }
    leaf node-physical-index {
      if-feature ietfhw:entity-mib;
      type int32 {
        range "1..2147483647";
      }
      config false;
       description
         "The entPhysicalIndex for the compute node.";
       reference
         "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
    }
  }

  grouping tpm12-pcr-info-short {
    description
      "This structure is for defining a digest at release when the only
```

```
       information that is necessary is the release configuration.";
    uses tpm12-pcr-selection;
    leaf locality-at-release {
      type uint8;
      description
        "This SHALL be the locality modifier required to release the
         information (TPM 1.2 type TPM_LOCALITY_SELECTION)";
      reference
        "TPM Main Part 2 TPM Structures v1.2 July 2007
        Section 8.6";
    }
    leaf digest-at-release {
      type binary;
      description
        "This SHALL be the digest of the PCR indices and PCR values
         to verify when revealing auth data (TPM 1.2 type
         TPM_COMPOSITE_HASH).";
      reference
        "TPM Main Part 2 TPM Structures v1.2 July 2007
        Section 5.4.1.";
    }
  }

  grouping tpm12-version {
    description
      "This structure provides information relative the version of
      the TPM.";
    list version {
      description
        "This indicates the version of the structure
         (TPM 1.2 type TPM_STRUCT_VER). This MUST be 1.1.0.0.";
      reference
        "TPM Main Part 2 TPM Structures v1.2 July 2007
        Section 5.1.";
      leaf major {
        type uint8;
        description
          "Indicates the major version of the structure.
           MUST be 0x01.";
      }
      leaf minor {
        type uint8;
        description
          "Indicates the minor version of the structure.
           MUST be 0x01.";
      }
      leaf rev-Major {
        type uint8;
```

```
        description
          "Indicates the rev major version of the structure.
           MUST be 0x00.";
      }
      leaf rev-Minor {
        type uint8;
        description
          "Indicates the rev minor version of the structure.
           MUST be 0x00.";
      }
    }
  }

  grouping tpm12-quote-info-common {
    description
      "These statements are within both quote variants of the TPM 1.2";
    reference
      "TPM Main Part 2 TPM Structures v1.2 July 2007,
      Section 11.3 & 11.4.";
    leaf fixed {
      type binary;
      description
        "This SHALL always be the string 'QUOT' or 'QUO2'
         (length is 4 bytes).";
    }
    leaf external-data {
      type binary;
      description
        "160 bits of externally supplied data, typically a nonce.";
    }
    leaf signature-size {
      type uint32;
      description
       "The size of TPM 1.2 'signature' value.";
    }
    leaf signature {
      type binary;
      description
        "Signature over hash of tpm12-quote-info2'.";
    }
  }

  grouping tpm12-quote-info {
    description
      "This structure provides the mechanism for the TPM to quote the
       current values of a list of PCRs (as used by the TPM_Quote2
       command).";
    uses tpm12-version;
```

```
    leaf digest-value {
      type binary;
      description
        "This SHALL be the result of the composite hash algorithm using
         the current values of the requested PCR indices
         (TPM 1.2 type TPM_COMPOSITE_HASH.)";
    }
  }

  grouping tpm12-quote-info2 {
    description
      "This structure provides the mechanism for the TPM to quote the
       current values of a list of PCRs
       (as used by the TPM_Quote2 command).";
    leaf tag {
      type uint8;
      description
        "This SHALL be TPM_TAG_QUOTE_INFO2.";
    }
    uses tpm12-pcr-info-short;
  }

  grouping tpm12-cap-version-info {
    description
      "TPM returns the current version and revision of the TPM 1.2 .";
    list TPM_PCR_COMPOSITE {
      description
        "The TPM 1.2 TPM_PCRVALUEs for the pcr-indices.";
      reference
        "TPM Main Part 2 TPM Structures v1.2 July 2007, Section 8.2";
      uses tpm12-pcr-selection;
      leaf value-size {
        type uint32;
        description
          "This SHALL be the size of the 'tpm12-pcr-value' field
           (not the number of PCRs).";
      }
      leaf-list tpm12-pcr-value {
        type binary;
        description
          "The list of TPM_PCRVALUEs from each PCR selected in sequence
           of tpm12-pcr-selection.";
      }
      list version-info {
        description
          "An optional output parameter from a TPM 1.2 TPM_Quote2.";
        leaf tag {
          type uint16;  /* This should be converted into an ENUM */
```

```
          description
            "The TPM 1.2 version and revision
             (TPM 1.2 type TPM_STRUCTURE_TAG).
             This MUST be TPM_CAP_VERSION_INFO (0x0030)";
        }
        uses tpm12-version;
        leaf spec-level {
          type uint16;
          description
            "A number indicating the level of ordinals supported.";
        }
        leaf errata-rev {
          type uint8;
          description
            "A number indicating the errata version of the
             specification.";
        }
        leaf tpm-vendor-id {
          type binary;
          description
            "The vendor ID unique to each TPM manufacturer.";
        }
        leaf vendor-specific-size {
          type uint16;
          description
            "The size of the vendor-specific area.";
        }
        leaf vendor-specific {
          type binary;
          description
            "Vendor specific information.";
        }
      }
    }
  }

  grouping tpm12-pcr-composite {
    description
      "The actual values of the selected PCRs (a list of TPM_PCRVALUEs
       (binary) and associated metadata for TPM 1.2.";
    list TPM_PCR_COMPOSITE {
      description
        "The TPM 1.2 TPM_PCRVALUEs for the pcr-indices.";
      reference
        "TPM Main Part 2 TPM Structures v1.2 July 2007, Section 8.2";
      uses tpm12-pcr-selection;
      leaf value-size {
        type uint32;
```

```
        description
          "This SHALL be the size of the 'tpm12-pcr-value' field
           (not the number of PCRs).";
      }
      leaf-list tpm12-pcr-value {
        type binary;
        description
          "The list of TPM_PCRVALUEs from each PCR selected in sequence
           of tpm12-pcr-selection.";
      }
    }
  }

  grouping node-uptime {
    description
      "Uptime in seconds of the node.";
    leaf up-time {
      type uint32;
      description
        "Uptime in seconds of this node reporting its data";
    }
  }

  grouping tpm12-attestation {
    description
      "Contains an instance of TPM1.2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester
      information.";
    uses node-uptime;
    uses compute-node-identifier;
    uses tpm12-quote-info-common;
    choice tpm12-quote {
      mandatory true;
      description
        "Either a tpm12-quote-info or tpm12-quote-info2, depending
         on whether TPM_Quote or TPM_Quote2 was used
         (cf. input field add-verson).";
      case tpm12-quote1 {
        description
          "BIOS/UEFI event logs";
        uses tpm12-quote-info;
        uses tpm12-pcr-composite;
      }
      case tpm12-quote2 {
        description
          "BIOS/UEFI event logs";
        uses tpm12-quote-info2;
      }
```

```
    }
  }

  grouping tpm20-attestation {
    description
      "Contains an instance of TPM2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester
      information.";
    leaf TPMS_QUOTE_INFO {
      mandatory true;
      type binary;
      description
        "A hash of the latest PCR values (and the hash algorithm used)
        which have been returned from a Verifier for the selected PCRs
        and Hash Algorithms.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
        TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.12.1";
    }
    leaf quote-signature {
      type binary;
      description
        "Quote signature returned by TPM Quote.  The signature was
        generated using the key associated with the
        certificate-name.";
    }
    uses node-uptime;
    uses compute-node-identifier;
    list unsigned-pcr-values {
      description
        "PCR values in each PCR bank. This often should not be
         necessary for TPM2, as the raw information needing
         signature and hash validation will be coming from
         the 'quote' leaf";
      uses TPM20-hash-algo;
      list pcr-values {
        key pcr-index;
        description
          "List of one PCR bank.";
        leaf pcr-index {
          type pcr;
          description
            "PCR index number.";
        }
        leaf pcr-value {
          type binary;
          description
            "PCR value.";
```

```
        }
      }
    }
  }


  grouping log-identifier {
    description
      "Identifier for type of log to be retrieved.";
    leaf log-type {
      type identityref {
        base attested_event_log_type;
      }
      mandatory true;
      description
        "The corresponding measurement log type identity.";
    }
  }

  grouping boot-event-log {
    description
      "Defines an event log corresponding to the event that extended
      the PCR";
    leaf event-number {
      type uint32;
      description
        "Unique event number of this event";
    }
    leaf event-type {
        type uint32;
        description
          "log event type";
    }
    leaf pcr-index {
      type pcr;
      description
        "Defines the PCR index that this event extended";
    }
    list digest-list {
      description
        "Hash of event data";
      leaf hash-algo {
        type identityref {
          base taa:hash;
        }
        description
          "The hash scheme that is used to compress the event data in
          each of the leaf-list digest items.";
```

```
      }
      leaf-list digest {
        type binary;
        description
          "The hash of the event data";
      }
    }
    leaf event-size {
      type uint32;
      description
        "Size of the event data";
    }
    leaf-list event-data {
      type uint8;
      description
        "The event data size determined by event-size";
    }
  }

  grouping bios-event-log {
    description
      "Measurement log created by the BIOS/UEFI.";
    list bios-event-entry {
      key event-number;
        description
          "Ordered list of TCG described event log
           that extended the PCRs in the order they
           were logged";
      uses boot-event-log;
    }
  }

  grouping ima-event {
    description
      "Defines an hash log extend event for IMA measurements";
    leaf event-number {
      type uint64;
      description
        "Unique number for this event for sequencing";
    }
    leaf ima-template {
      type string;
      description
        "Name of the template used for event logs
         for e.g. ima, ima-ng, ima-sig";
    }
    leaf filename-hint {
      type string;
```

```
      description
        "File that was measured";
    }
    leaf filedata-hash {
      type binary;
      description
        "Hash of filedata";
    }
    leaf filedata-hash-algorithm {
      type string;
      description
        "Algorithm used for filedata-hash";
    }
    leaf template-hash-algorithm {
      type string;
      description
        "Algorithm used for template-hash";
    }
    leaf template-hash {
      type binary;
      description
        "hash(filedata-hash, filename-hint)";
    }
    leaf pcr-index {
      type pcr;
      description
        "Defines the PCR index that this event extended";
    }
    leaf signature {
      type binary;
      description
        "The file signature";
    }
  }

  grouping ima-event-log {
    description
      "Measurement log created by IMA.";
    list ima-event-entry {
      key event-number;
      description
      "Ordered list of ima event logs by event-number";
       uses ima-event;
    }
  }

  grouping netequip-boot-event {
    description
```

```
      "Defines an hash log extend event for Network Equipment Boot.";
    leaf event-number {
      type uint64;
      description
        "Unique number for this event for sequencing";
    }
    leaf filename-hint {
      type string;
      description
        "File that was measured";
    }
    leaf filedata-hash {
      type binary;
      description
        "Hash of filedata";
    }
    leaf filedata-hash-algorithm {
      type string;
      description
        "Algorithm used for filedata-hash.";
      }
    leaf file-version {
      type string;
      description
        "File version information.";
    }
    leaf file-type {
      type string;
      description
        "Indicating at which boot stage the file is loaded,
         such as BIOS, BootLoader, etc.";
    }
    leaf pcr-index {
      type pcr;
      description
        "Defines the PCR index that this event extended";
    }
  }

  grouping network-equipment-boot-event-log {
    description
      "Measurement log created by Network Equipment Boot.";
    list boot-event-entry {
      key event-number;
      description
        "Ordered list of Network Equipment Boot event logs
         by event-number.";
      uses netequip-boot-event;
```

```
    }
  }

  grouping event-logs {
    description
      "A selector for the log and its type.";
    choice attested_event_log_type {
      mandatory true;
      description
        "Event log type determines the event logs content.";
      case bios {
        description
          "BIOS/UEFI event logs";
        container bios-event-logs {
          description
            "BIOS/UEFI event logs";
          uses bios-event-log;
        }
      }
      case ima {
        description
          "IMA event logs.";
        container ima-event-logs {
          description
            "IMA event logs.";
          uses ima-event-log;
        }
      }
      case netequip_boot {
        description
          "Network Equipment Boot event logs";
        container boot-event-logs {
          description
            "Network equipment boot event logs.";
          uses network-equipment-boot-event-log;
        }
      }
    }
  }

  /**********************/
  /*   RPC operations   */
  /**********************/

  rpc tpm12-challenge-response-attestation {
    if-feature "taa:TPM12";
    description
      "This RPC accepts the input for TSS TPM 1.2 commands made to the
```

```
       attesting device.";
    input {
      container tpm12-attestation-challenge {
        description
          "This container includes every information element defined
           in the reference challenge-response interaction model for
           remote attestation. Corresponding values are based on
           TPM 1.2 structure definitions";
        uses tpm12-pcr-selection;
        uses nonce;
        leaf add-version {
          type boolean;
          description
            "Whether or not to include TPM_CAP_VERSION_INFO; if true,
             then TPM_Quote2 must be used to create the response.";
          reference
            "TPM Main Part 2 TPM Structures v1.2 July 2007,
            Section 21.6";
        }
        leaf-list certificate-name {
          must "/tpm:rats-support-structures/tpm:tpms" +
               "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm12']" +
               "/tpm:certificates/" +
               "/tpm:certificate[certificate-name-ref=current()]" {
            error-message "Not an available TPM1.2 AIK certificate.";
          }
          type certificate-name-ref;
          description
            "When populated, the RPC will only get a Quote for the
            TPMs associated with these certificate(s).";
        }
      }
    }
    output {
      list tpm12-attestation-response {
        unique "certificate-name";
        description
          "The binary output of TPM 1.2 TPM_Quote/TPM_Quote2, including
           the PCR selection and other associated attestation evidence
           metadata";
        uses certificate-name-ref {
          description
            "Certificate associated with this tpm12-attestation.";
        }
        uses tpm12-attestation;
      }
    }
  }
```

```
  rpc tpm20-challenge-response-attestation {
    if-feature "taa:TPM20";
    description
      "This RPC accepts the input for TSS TPM 2.0 commands of the
       managed device. ComponentIndex from the hardware manager YANG
       module to refer to dedicated TPM in composite devices,
       e.g. smart NICs, is still a TODO.";
    input {
      container tpm20-attestation-challenge {
        description
          "This container includes every information element defined
           in the reference challenge-response interaction model for
           remote attestation. Corresponding values are based on
           TPM 2.0 structure definitions";
        uses nonce;
        uses tpm20-pcr-selection;
        leaf-list certificate-name {
          must "/tpm:rats-support-structures/tpm:tpms" +
               "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm20']" +
               "/tpm:certificates/" +
               "/tpm:certificate[certificate-name-ref=current()]" {
            error-message "Not an available TPM2.0 AIK certificate.";
          }
          type certificate-name-ref;
            description
              "When populated, the RPC will only get a Quote for the
              TPMs associated with the certificates.";
        }
      }
    }
    output {
      list tpm20-attestation-response {
        unique "certificate-name";
        description
          "The binary output of TPM2b_Quote in one TPM chip of the
           node which identified by node-id. An TPMS_ATTEST structure
           including a length, encapsulated in a signature";
        uses certificate-name-ref {
          description
            "Certificate associated with this tpm20-attestation.";
        }
        uses tpm20-attestation;
      }
    }
  }

  rpc log-retrieval {
    description
```

```
      "Logs Entries are either identified via indices or via providing
       the last line received. The number of lines returned can be
       limited. The type of log is a choice that can be augmented.";
    input {
      list log-selector {
        description
          "Selection of log entries to be reported.";
        uses tpm-name-selector;
        choice index-type {
          description
            "Last log entry received, log index number, or timestamp.";
          case last-entry {
            description
              "The last entry of the log already retrieved.";
            leaf last-entry-value {
              type binary;
              description
                "Content of an log event which matches 1:1 with a
                 unique event record contained within the log.  Log
                 entries subsequent to this will be passed to the
                 requester.  Note: if log entry values are not unique,
                 this MUST return an error.";
            }
          }
          case index {
            description
              "Numeric index of the last log entry retrieved, or
               zero.";
            leaf last-index-number {
              type uint64;
              description
                "The last numeric index number of a log entry.
                 Zero means to start at the beginning of the log.
                 Entries subsequent to this will be passed to the
                 requester.";
            }
          }
          case timestamp {
            leaf timestamp {
              type yang:date-and-time;
              description
                "Timestamp from which to start the extraction.  The
                 next log entry subsequent to this timestamp is to
                 be sent.";
            }
            description
              "Timestamp from which to start the extraction.";
          }
```

```
        }
        leaf log-entry-quantity {
          type uint16;
          description
            "The number of log entries to be returned. If omitted, it
             means all of them.";
        }
      }
      uses log-identifier;
    }

    output {
      container system-event-logs {
        description
          "The requested data of the measurement event logs";
        list node-data {
          unique "tpm-name";
          description
            "Event logs of a node in a distributed system
             identified by the node name";
          uses tpm-name;
          uses node-uptime;
          container log-result {
            description
              "The requested entries of the corresponding log.";
            uses event-logs;
          }
        }
      }
    }
  }

  /*************************************/
  /*   Config & Oper accessible nodes   */
  /*************************************/

  container rats-support-structures {
    description
      "The datastore definition enabling verifiers or relying
       parties to discover the information necessary to use the
       remote attestation RPCs appropriately.";
    container compute-nodes {
      presence
        "Indicates that more than one TPM exists on a device.";
      description
        "Holds the set device subsystems/components in this composite
         device that support TPM operations.";
      list compute-node {
```

```
        key node-id;
        config false;
        min-elements 2;
        uses compute-node-identifier;
        description
          "A components in this composite device that RATS which
          supports TPM operations.";
        leaf node-name {
          type string;
          description
            "Name of the compute node.";
        }
        leaf node-location {
          type string;
          description
            "Location of the compute node, such as slot number.";
        }
      }
    }
    container tpms {
      description
        "Holds the set of TPMs within an Attester.";
      list tpm {
        key tpm-name;
        unique "tpm-path";
          description
           "A list of TPMs in this composite device that RATS
           can be conducted with.";
        uses tpm-name;
        leaf hardware-based {
          type boolean;
          config false;
          description
            "Answers the question: is this TPM is a hardware based
            TPM?";
        }
        leaf tpm-physical-index {
          if-feature ietfhw:entity-mib;
          type int32 {
            range "1..2147483647";
          }
          config false;
          description
            "The entPhysicalIndex for the TPM.";
          reference
            "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
        }
        leaf tpm-path {
```

```
          type string;
          config false;
          description
            "Path to a unique TPM on a device.  This can change agross
            reboots.";
        }
        leaf compute-node {
          when "../../../compute-nodes";
          type compute-node-ref;
          config false;
          mandatory true;
          description
            "When there is more that one TPM, this indicates for which
            compute node this TPM services.";
        }
        leaf tpm-manufacturer {
          type string;
          config false;
          description
            "TPM manufacturer name.";
        }
        leaf tpm-firmware-version {
          type identityref {
            base taa:cryptoprocessor;
          }
          mandatory true;
          description
            "Identifies the cryptoprocessor API set supported.  This
            cannot be configured.  However it is referenced via XPATH
            as part of configuration, so is shown as 'rw'
            to eliminate YANG warnings related NMDA.";
        }
        uses TPM12-hash-algo {
          when "tpm-firmware-version = 'taa:tpm12'";
          refine TPM12-hash-algo {
            description
              "The hash algorithm overwrites the default used for PCRs
              on this TPM1.2 compliant cryptoprocessor.";
          }
        }
        leaf-list TPM12-pcrs {
          when "../tpm-firmware-version = 'taa:tpm12'";
          type pcr;
          description
            "The PCRs which may be extracted from this TPM1.2
            compliant cryptoprocessor.";
        }
        list tpm20-pcr-bank {
```

```
          when "../tpm-firmware-version = 'taa:tpm20'";
          key "TPM20-hash-algo";
          description
            "Specifies the list of PCRs that may be extracted for
            a specific Hash Algorithm on this TPM2 compliant
            cryptoprocessor.  A bank is a set of PCRs which are
            extended using a particular hash algorithm.";
          reference
            "https://www.trustedcomputinggroup.org/wp-content/uploads/
             TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.9.7";
          leaf TPM20-hash-algo {
            must "/tpm:rats-support-structures"
               + "/tpm:attester-supported-algos"
               + "/tpm:tpm20-hash" {
              error-message "Not a platform supported TPM20-hash-algo";
            }
            type identityref {
              base taa:hash;
            }
            description
              "The hash scheme actively being used to hash a
              one or more TPM2.0 PCRs.";
          }
          leaf-list pcr-index {
            type tpm:pcr;
            description
              "Defines what TPM2 PCRs are available to be extracted.";
          }
        }
        leaf tpm-status {
          type enumeration {
            enum operational {
              value 0;
              description
                "The TPM currently is currently running normally and
                is ready to accept and process TPM quotes.";
              reference
                "TPM-Rev-2.0-Part-1-Architecture-01.07-2014-03-13.pdf
                Section 12";
            }
            enum non-operational {
              value 1;
              description
                "TPM is in a state such as startup or shutdown which
                precludes the processing of TPM quotes.";
            }
          }
          config false;
```

```
          mandatory true;
          description
            "TPM chip self-test status.";
        }
        container certificates {
          description
            "The TPM's certificates, including EK certificates
             and AK certificates.";
          list certificate {
            key "certificate-name";
            description
              "Three types of certificates can be accessed via
               this statement, including Initial Attestation
               Key Cert, Local Attestation Key Cert or
               Endorsement Key Cert.";
            leaf certificate-name {
              type string;
              description
                "An arbitrary name uniquely identifying a certificate
                 associated within key within a TPM.";
            }
            leaf certificate-keystore-ref {
              type leafref {
                path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
                   + "/ks:certificates/ks:certificate/ks:name";
              }
              description
                "A reference to a specific certificate of an
                 asymmetric key in the Keystore.";
                 /* Note: It is also possible to import a grouping which
                    allows local definition via an imported keystore
                    schema. */
            }
            leaf certificate-type {
              type enumeration {
                enum endorsement-cert {
                  value 0;
                  description
                    "Endorsement Key (EK) Certificate type.";
                }
                enum initial-attestation-cert {
                  value 1;
                  description
                    "Initial Attestation key (IAK) Certificate type.";
                }
                enum local-attestation-cert {
                  value 2;
                  description
```

```
                    "Local Attestation Key (LAK) Certificate type.";
                }
              }
              description
                "Type of this certificate";
            }
          }
        }
      }
    }
    container attester-supported-algos {
      description
        "Identifies which TPM algorithms are available for use on an
        attesting platform.";
      leaf-list tpm12-asymmetric-signing {
        if-feature "taa:TPM12";
        when "../../tpm:tpms" +
             "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm12']";
        type identityref {
          base taa:asymmetric;
        }
        description
          "Platform Supported TPM12 asymmetric algorithms.";
      }
      leaf-list tpm12-hash {
        if-feature "taa:TPM12";
        when "../../tpm:tpms" +
             "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm12']";
        type identityref {
          base taa:hash;
        }
        description
          "Platform supported TPM12 hash algorithms.";
      }
      leaf-list tpm20-asymmetric-signing {
        if-feature "taa:TPM20";
        when "../../tpm:tpms" +
             "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm20']";
        type identityref {
          base taa:asymmetric;
        }
        description
          "Platform Supported TPM20 asymmetric algorithms.";
      }
      leaf-list tpm20-hash {
        if-feature "taa:TPM20";
        when "../../tpm:tpms" +
             "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm20']";
```

```
        type identityref {
          base taa:hash;
        }
        description
          "Platform supported TPM20 hash algorithms.";
      }
    }
  }
}
<CODE ENDS>
```

2.2.2. ietf-tcg-algs

Cryptographic algorithm types were initially included within -v14 NETCONF's iana-crypto-types.yang. Unfortunately all this content including the algorithms needed here failed to make the -v15 used WGLC. As a result this document has encoded the TCG Algorithm definitions of [TCG-Algos], revision 1.32. By including this full table as a separate YANG file within this document, it is possible for other YANG models to leverage the contents of this model.

2.2.2.1. Features

There are two types of features supported <TPM12> and <TPM20>. Support for either of these features indicates that a cryptoprocessor supporting the corresponding type of TCG API is present on an Attester. Most commonly, only one type of cryptoprocessor will be available on an Attester.

2.2.2.2. Identities

There are three types of identities in this model.

The first are the cryptographic functions supportable by a TPM algorithm, these include: <asymmetric>, <symmetric>, <hash>, <signing>, <anonymous_signing>, <encryption_mode>, <method>, and <object_type>. The definitions of each of these are in Table 2 of [TCG-Algos].

The second are API specifications for tpms: <tpm12> and <tpm2>.

The third are specific algorithm types. Each algorithm type defines what cryptographic functions may be supported, and on which type of API specification. It is not required that an implementation of a specific TPM will support all algorithm types. The contents of each specific algorithm mirrors what is in Table 3 of [TCG-Algos].

2.2.2.3. YANG Module

```
<CODE BEGINS> ietf-tcg-algs@2020-09-18.yang
module ietf-tcg-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcg-algs";
  prefix taa;

  organization
    "IETF RATS Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/rats/>
     WG List:  <mailto:rats@ietf.org>
     Author:   Eric Voit <mailto:evoit@cisco.com>";

  description
    "This module defines a identities for asymmetric algorithms.

     Copyright (c) 2020 IETF Trust and the persons identified
     as authors of the code. All rights reserved.
     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
     itself for full legal notices.
     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 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

  revision 2020-09-18 {
    description
      "Initial version";
    reference
      "RFC XXXX: tbd";
  }

  /*****************/
  /*   Features    */
  /*****************/
```

```
  feature TPM12 {
    description
      "This feature indicates algorithm support for the TPM 1.2 API
      as per TPM-main-1.2-Rev94-part-2, Section 4.8.";
  }

  feature TPM20 {
    description
      "This feature indicates algorithm support for the TPM 2.0 API
      as per TPM-Rev-2.0-Part-1-Architecture-01.38 Section 11.4.";
  }


  /*****************/
  /*  Identities   */
  /*****************/

  /*  There needs to be collasping/verification of some of the identity
      types between the various algorithm types listed below */

  identity asymmetric {
    description
      "A TCG recognized asymmetric algorithm with a public and
      private key.";
    reference
      "http://trustedcomputinggroup.org/resource/tcg-algorithm-registry/
      TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity symmetric {
    description
      "A TCG recognized symmetric algorithm with only a private key.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity hash {
    description
      "A TCG recognized hash algorithm that compresses input data to
      a digest value or indicates a method that uses a hash.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity signing {
    description
      "A TCG recognized signing algorithm";
    reference
```

```
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity anonymous_signing {
    description
      "A TCG recognized anonymous signing algorithm.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity encryption_mode {
    description
      "A TCG recognized encryption mode.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity method {
    description
      "A TCG recognized method such as a mask generation function.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity object_type {
    description
      "A TCG recognized object type.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

  identity cryptoprocessor {
    description
      "Base identity identifying a crytoprocessor.";
  }

  identity tpm12 {
    if-feature "TPM12";
    base cryptoprocessor;
    description
      "Supportable by a TPM1.2.";
    reference
      "TPM-Main-Part-2-TPM-Structures_v1.2_rev116_01032011.pdf
      TPM_ALGORITHM_ID values, page 18";
  }

  identity tpm20 {
    if-feature "TPM12";
```

```
    base cryptoprocessor;
    description
      "Supportable by a TPM2.";
    reference
      "TPM-Rev-2.0-Part-2-Structures-01.38.pdf
      The TCG Algorithm Registry. Table 9";
  }

  identity TPM_ALG_RSA {
    if-feature "TPM12 or TPM20";
    base tpm12;
    base tpm20;
    base asymmetric;
    base object_type;
    description
      "RSA algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      RFC 8017. ALG_ID: 0x0001";

  }

  identity TPM_ALG_TDES {
    if-feature "TPM12";
    base tpm12;
    base symmetric;
    description
      "Block cipher with various key sizes (Triple Data Encryption
       Algorithm, commonly called Triple Data Encryption Standard)
       Note: was banned in TPM1.2 v94";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 18033-3. ALG_ID: 0x0003";

  }

  identity TPM_ALG_SHA1 {
    if-feature "TPM12 or TPM20";
    base hash;
    base tpm12;
    base tpm20;
    description
      "SHA1 algorithm - Deprecated due to insufficient cryptographic
      protection.  However it is still useful for hash algorithms
      where protection is not required.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10118-3. ALG_ID: 0x0004";
```

```
  }

  identity TPM_ALG_HMAC  {
    if-feature "TPM12 or TPM20";
    base tpm12;
    base tpm20;
    base hash;
    base signing;
    description
      "Hash Message Authentication Code (HMAC) algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3,
      ISO/IEC 9797-2 and RFC2014. ALG_ID: 0x0005";

  }

  identity TPM_ALG_AES  {
    if-feature "TPM12";
    base tpm12;
    base symmetric;
    description
      "The AES algorithm with various key sizes";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 18033-3. ALG_ID: 0x0006";

  }

  identity TPM_ALG_MGF1 {
    if-feature "TPM20";
    base tpm20;
    base hash;
    base method;
    description
      "hash-based mask-generation function";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3,
      IEEE Std 1363-2000 and IEEE Std 1363a -2004.
      ALG_ID: 0x0007";

  }

  identity TPM_ALG_KEYEDHASH {
    if-feature "TPM20";
    base tpm20;
    base hash;
    base object_type;
    description
```

```
      "An encryption or signing algorithm using a keyed hash.  These
      may use XOR for encryption or an HMAC for signing and may
      also refer to a data object that is neither signing nor
      encrypting.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      TCG TPM 2.0 library specification. . ALG_ID: 0x0008";

    }

  identity TPM_ALG_XOR {
    if-feature "TPM12 or TPM20";
    base tpm12;
    base tpm20;
    base hash;
    base symmetric;
    description
      "The XOR encryption algorithm.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      TCG TPM 2.0 library specification. ALG_ID: 0x000A";

  }

  identity TPM_ALG_SHA256 {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "The SHA 256 algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10118-3. ALG_ID: 0x000B";

  }

  identity TPM_ALG_SHA384  {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "The SHA 384 algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10118-3. ALG_ID: 0x000C";

  }
```

```
  identity TPM_ALG_SHA512  {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "The SHA 512 algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10118-3. ALG_ID: 0x000D";

  }

  identity TPM_ALG_NULL {
    if-feature "TPM20";
    base tpm20;
    description
      "NULL algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      TCG TPM 2.0 library specification. ALG_ID: 0x0010";

  }

  identity TPM_ALG_SM3_256 {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "The SM3 hash algorithm.";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      GM/T 0004-2012 - SM3_256. ALG_ID: 0x0012";

  }

  identity TPM_ALG_SM4 {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    description
      "SM4 symmetric block cipher";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      GB/T 32907-2016. ALG_ID: 0x0013";

  }

  identity TPM_ALG_RSASSA {
```

```
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    description
      "Signature algorithm defined in section 8.2 (RSASSAPKCS1-v1_5)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and RFC 8017.
      ALG_ID: 0x0014";

  }

  identity TPM_ALG_RSAES {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base encryption_mode;
    description
      "Signature algorithm defined in section 7.2 (RSAES-PKCS1-v1_5)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and RFC 8017
      ALG_ID: 0x0015";

  }

  identity TPM_ALG_RSAPSS {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    description
      "Padding algorithm defined in section 8.1 (RSASSA PSS)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and RFC 8017.
      ALG_ID: 0x0016";

  }

  identity TPM_ALG_OAEP {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base encryption_mode;
    description
      "Padding algorithm defined in section 7.1 (RSASSA OAEP)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and RFC 8017.
      ALG_ID: 0x0017";
```

```
  }

  identity TPM_ALG_ECDSA {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    description
      "Signature algorithm using elliptic curve cryptography (ECC)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 14888-3. ALG_ID: 0x0018";

  }

  identity TPM_ALG_ECDH {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base method;
    description
      "Secret sharing using ECC";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-56A and RFC 7748. ALG_ID: 0x0019";

  }

  identity TPM_ALG_ECDAA {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    base anonymous_signing;
    description
      "Elliptic-curve based anonymous signing scheme";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      TCG TPM 2.0 library specification. ALG_ID: 0x001A";

  }

  identity TPM_ALG_SM2 {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    base encryption_mode;
```

```
    base method;
    description
      "SM2 - depending on context, either an elliptic-curve based,
      signature algorithm, an encryption scheme, or a key exchange
      protocol";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      A GM/T 0003.1-2012, GM/T 0003.2-2012, GM/T 0003.3-2012,
      GM/T 0003.5-2012 SM2. ALG_ID: 0x001B";

  }

  identity TPM_ALG_ECSCHNORR {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    description
      "Elliptic-curve based Schnorr signature";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      TCG TPM 2.0 library specification. ALG_ID: 0x001C";

  }

  identity TPM_ALG_ECMQV {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base method;
    description
      "Two-phase elliptic-curve key";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-56A. ALG_ID: 0x001D";

  }

  identity TPM_ALG_KDF1_SP800_56A {
    if-feature "TPM20";
    base tpm20;
    base hash;
    base method;
    description
      "Concatenation key derivation function";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-56A  (approved alternative1) section 5.8.1.
```

```
      ALG_ID: 0x0020";

  }

  identity TPM_ALG_KDF2  {
    if-feature "TPM20";
    base tpm20;
    base hash;
    base method;
    description
      "Key derivation function";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      IEEE 1363a-2004 KDF2 section 13.2. ALG_ID: 0x0021";

  }

  identity TPM_ALG_KDF1_SP800_108 {
    base TPM_ALG_KDF2;
    description
      "A key derivation method";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-108 - Section 5.1 KDF. ALG_ID: 0x0022";

  }

  identity TPM_ALG_ECC {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base object_type;
    description
      "Prime field ECC";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 15946-1. ALG_ID: 0x0023";

  }

  identity TPM_ALG_SYMCIPHER {
    if-feature "TPM20";
    base tpm20;
    description
      "Object type for a symmetric block cipher";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      TCG TPM 2.0 library specification. ALG_ID: 0x0025";
```

```
  }

  identity TPM_ALG_CAMELLIA {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    description
      "The Camellia algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 18033-3. ALG_ID: 0x0026";

  }

  identity TPM_ALG_SHA3_256  {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "ISO/IEC 10118-3 - the SHA 256 algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST PUB FIPS 202. ALG_ID: 0x0027";

  }

  identity TPM_ALG_SHA3_384 {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "The SHA 384 algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST PUB FIPS 202. ALG_ID: 0x0028";

  }

  identity TPM_ALG_SHA3_512 {
    if-feature "TPM20";
    base tpm20;
    base hash;
    description
      "The SHA 512 algorithm";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST PUB FIPS 202. ALG_ID: 0x0029";
```

```
  }

  identity TPM_ALG_CMAC {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base signing;
    description
      "block Cipher-based Message Authentication Code (CMAC)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 9797-1:2011 Algorithm 5. ALG_ID: 0x003F";

  }

  identity TPM_ALG_CTR {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base encryption_mode;
    description
      "Counter mode";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10116. ALG_ID: 0x0040";

  }

  identity TPM_ALG_OFB {
    base tpm20;
    base symmetric;
    base encryption_mode;
    description
      "Output Feedback mode";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10116. ALG_ID: 0x0041";

  }

  identity TPM_ALG_CBC {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base encryption_mode;
    description
      "Cipher Block Chaining mode";
    reference
```

```
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10116. ALG_ID: 0x0042";

  }

  identity TPM_ALG_CFB {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base encryption_mode;
    description
      "Cipher Feedback mode";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10116. ALG_ID: 0x0043";

  }

  identity TPM_ALG_ECB  {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base encryption_mode;
    description
      "Electronic Codebook mode";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      ISO/IEC 10116. ALG_ID: 0x0044";

  }

  identity TPM_ALG_CCM  {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base signing;
    base encryption_mode;
    description
      "Counter with Cipher Block Chaining-Message Authentication
      Code (CCM)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-38C. ALG_ID: 0x0050";

  }

  identity TPM_ALG_GCM {
    if-feature "TPM20";
```

```
    base tpm20;
    base symmetric;
    base signing;
    base encryption_mode;
    description
      "Galois/Counter Mode (GCM)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-38D. ALG_ID: 0x0051";

  }

  identity TPM_ALG_KW {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base signing;
    base encryption_mode;
    description
      "AES Key Wrap (KW)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-38F. ALG_ID: 0x0052";

  }

  identity TPM_ALG_KWP {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base signing;
    base encryption_mode;
    description
      "AES Key Wrap with Padding (KWP)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-38F. ALG_ID: 0x0053";

  }

  identity TPM_ALG_EAX  {
    if-feature "TPM20";
    base tpm20;
    base symmetric;
    base signing;
    base encryption_mode;
    description
      "Authenticated-Encryption Mode";
```

```
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      NIST SP800-38F. ALG_ID: 0x0054";

  }

  identity TPM_ALG_EDDSA {
    if-feature "TPM20";
    base tpm20;
    base asymmetric;
    base signing;
    description
      "Edwards-curve Digital Signature Algorithm (PureEdDSA)";
    reference
      "TCG_Algorithm_Registry_r1p32_pub Table 3 and
      RFC 8032. ALG_ID: 0x0060";

  }

}
<CODE ENDS>
```

Note that not all cryptographic functions are required for use by ietf-tpm-remote-attestation.yang. However the full definition of Table 3 of [TCG-Algos] will allow use by additional YANG specifications.

3. IANA considerations

This document will include requests to IANA:

To be defined yet. But keeping up with changes to ietf-tcg-algs.yang will be necessary.

4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config)

to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

Container: </rats-support-structures/attester-supported-algos>

- <tpm12-asymmetric-signing>, <tpm12-hash>, <tpm20-asymmetric-signing>, and <tpm20-hash> all could be populated with algorithms which are not supported by the underlying physical TPM installed by the equipment vendor.

Container: </rats-support-structures/tpms>

- <tpm-name> - Although shown as 'rw', it is system generated

- <tpm20-pcr-bank> - It is possible to configure PCRs for extraction which are not being extended by system software. This could unnecessarily use TPM resources.

- <certificates> - It is possible to provision a certificate which does not correspond to a Attestation Identity Key (AIK) within the TPM.

RPC: <tpm12-challenge-response-attestation> - Need to verify that the certificate is for an active AIK.

RPC: <tpm20-challenge-response-attestation> - Need to verify that the certificate is for an active AIK.

RPC: <log-retrieval> - Pulling lots of logs can chew up system resources.

5. Acknowledgements

Not yet.

6. Change Log

Changes from version 02 to version 03:

- moved to tcg-algs

- cleaned up model to eliminate sources of errors

- removed key establishment RPC

- added lots of XPATH which must all be scrubbed still

- Descriptive text added on model contents.

Changes from version 01 to version 02:

- Extracted Crypto-types into a separate YANG file
- Mades the algorithms explicit, not strings
- Hash Algo as key the selected TPM2 PCRs
- PCR numbers are their own type
- Eliminated nested keys for node-id plus tpm-name
- Eliminated TPM-Name of "ALL"
- Added TPM-Path

Changes from version 00 to version 01:

- Addressed author's comments
- Extended complementary details about attestation-certificates
- Relabeled chunk-size to log-entry-quantity
- Relabeled location with compute-node or tpm-name where appropriate
- Added a valid entity-mib physical-index to compute-node and tpm-name to map it back to hardware inventory
- Relabeled name to tpm_name
- Removed event-string in last-entry

7. References

7.1. Normative References

[I-D.birkholz-rats-reference-interaction-model]
Birkholz, H., Eckel, M., Newton, C., and L. Chen, "Reference Interaction Models for Remote Attestation Procedures", draft-birkholz-rats-reference-interaction-model-03 (work in progress), July 2020.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", draft-ietf-netconf-keystore-20 (work in progress), August 2020.

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-06 (work in progress), September 2020.

[I-D.ietf-rats-tpm-based-network-device-attest]
Fedorkow, G., Voit, E., and J. Fitzgerald-McKay, "TPM-based Network Device Remote Integrity Verification", draft-ietf-rats-tpm-based-network-device-attest-04 (work in progress), September 2020.

[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/info/rfc2119>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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/info/rfc8174>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

[TCG-Algos]
"TCG_Algorithm_Registry_r1p32_pub", n.d., <hhttp://trustedcomputinggroup.org/resource/tcg-algorithm-registry/>.

[TPM1.2] TCG, ., "TPM 1.2 Main Specification", October 2003, <https://trustedcomputinggroup.org/resource/tpm-main-specification/>.

[TPM2.0] TCG, ., "TPM 2.0 Library Specification", March 2013, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

7.2. Informative References

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Shwetha Bhandari
Cisco Systems

Email: shwethab@cisco.com

Bill Sulzen
Cisco Systems

Email: bsulzen@cisco.com

Liang Xia (Frank)
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

Tom Laffey
Hewlett Packard Enterprise

Email: tom.laffey@hpe.com

Guy C. Fedorkow
Juniper Networks
10 Technology Park Drive
Westford, Massachusetts 01886

Email: gfedorkow@juniper.net