

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.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Disambiguation	4
4. Scope and Intent	4
5. Direct Anonymous Attestation	5
5.1. Endorsers	5
5.2. Endorsers for Direct Anonymous Attestation	6
6. Normative Prerequisites	6
7. Generic Information Elements	7
8. Interaction Models	9
8.1. Challenge/Response Remote Attestation	10
8.2. Uni-Directional Remote Attestation	11
8.3. Streaming Remote Attestation	13
9. Additional Application-Specific Requirements	15
9.1. Confidentiality	15
9.2. Mutual Authentication	15
9.3. Hardware-Enforcement/Support	15
10. Implementation Status	15
10.1. Implementer	16
10.2. Implementation Name	16
10.3. Implementation URL	16
10.4. Maturity	16
10.5. Coverage and Version Compatibility	16
10.6. License	16
10.7. Implementation Dependencies	16
10.8. Contact	17
11. Security and Privacy Considerations	17
12. Acknowledgments	17
13. Change Log	17
14. References	19
14.1. Normative References	19
14.2. Informative References	20
Appendix A. CDDL Specification for a simple CoAP Challenge/Response Interaction	20
Authors' Addresses	21

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 document 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:

- o prevent inconsistencies in descriptions of these interaction models in other documents (due to text cloning over time),
- o 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
- o 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:

- o 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.
- o 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.
- o 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 Conceptual 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:

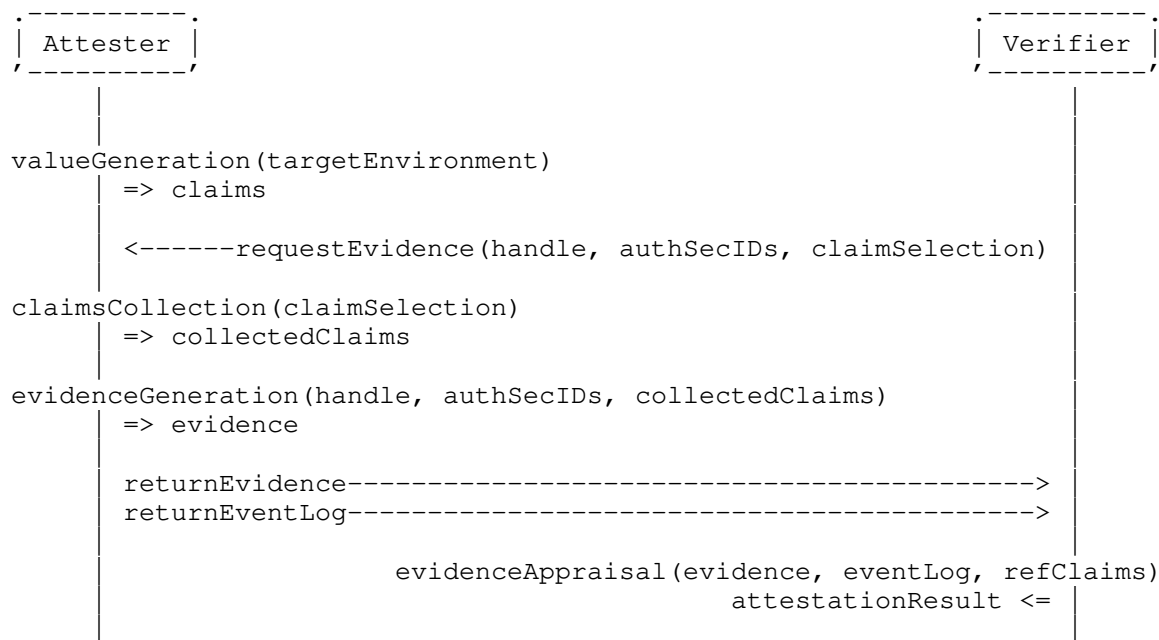
1. Challenge/Response Remote Attestation
2. Uni-Directional Remote Attestation
3. Streaming Remote Attestation

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 way 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.

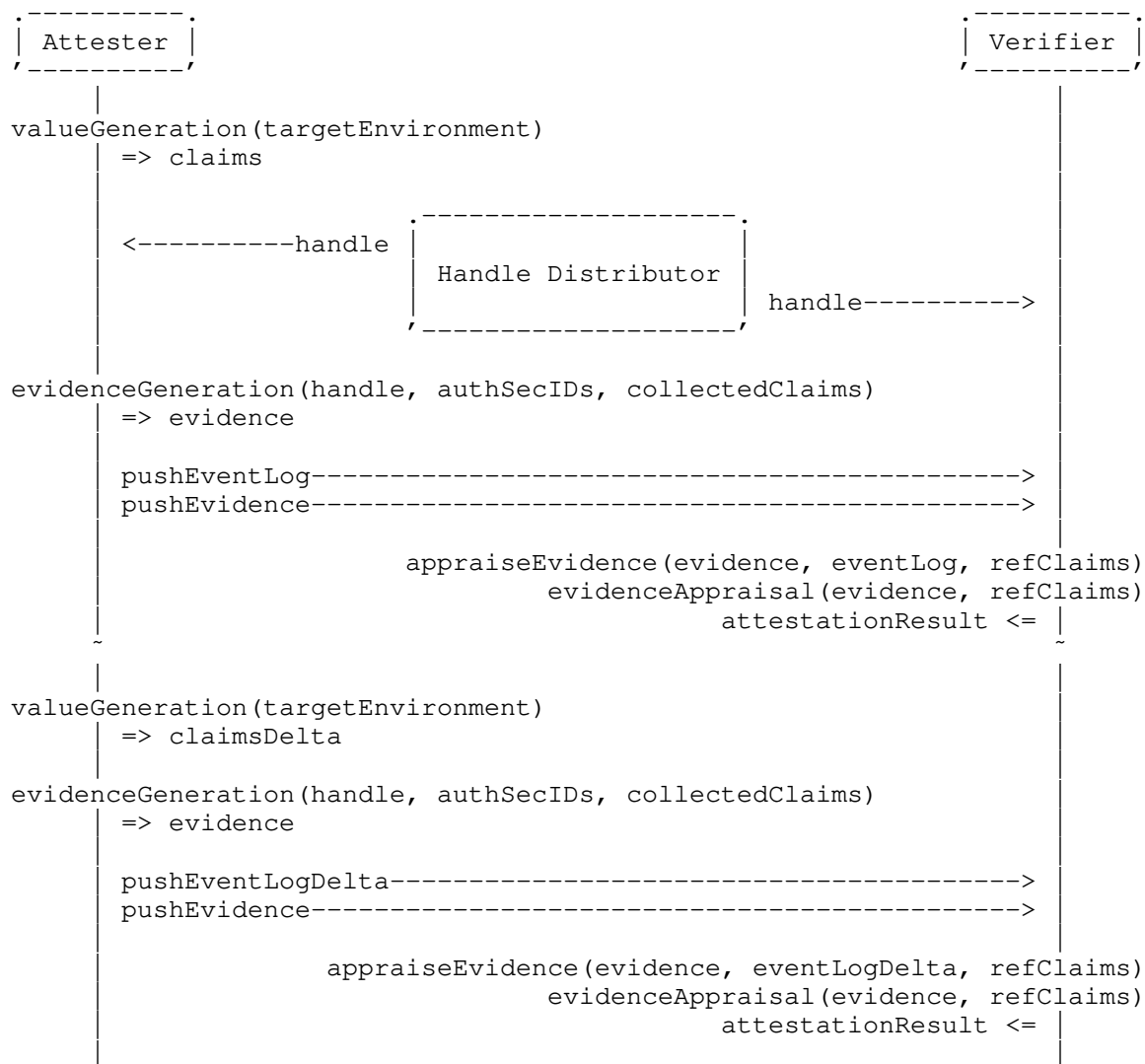
The Claim Selection narrows down the set of Claims collected and used to create Evidence to those that the Verifier requires. If the Claim Selection is omitted, then by default all Claims that are known and available on the Attester MUST be used to create corresponding Evidence. For example when performing 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.

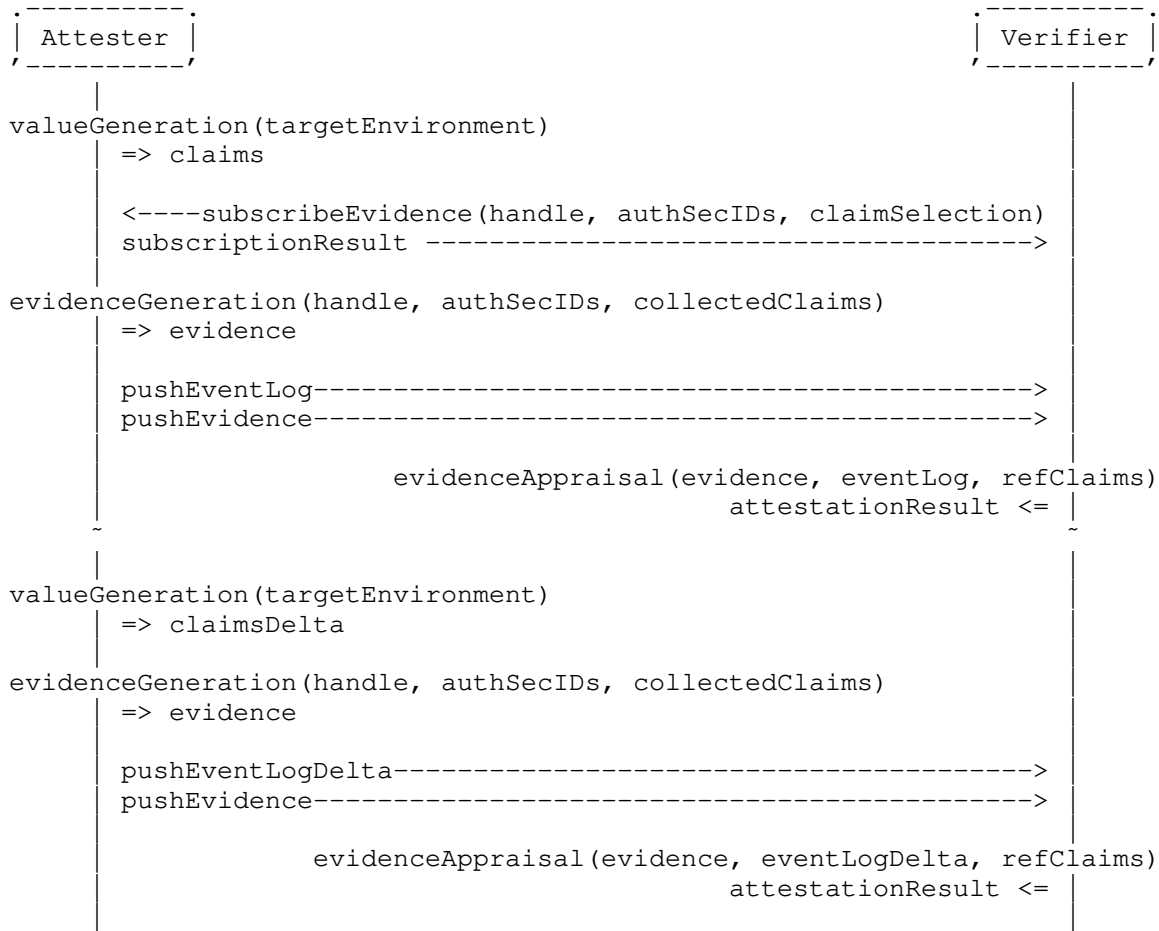
8.2. Uni-Directional Remote Attestation



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]). Timestamps 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, simultaneously. 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 used 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

- o Initial draft -00
- o 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
- o 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, "Turtles 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: Informational
Expires: 19 April 2021

H. Birkholz
Fraunhofer SIT
D. Thaler
Microsoft
M. Richardson
Sandelman Software Works
N. Smith
Intel
W. Pan
Huawei Technologies
16 October 2020

Remote Attestation Procedures Architecture
draft-ietf-rats-architecture-07

Abstract

In network protocol exchanges, it is often the case that one entity (a Relying Party) requires evidence about a remote peer to assess the peer's trustworthiness, and a way to appraise such evidence. The evidence is typically a set of claims about its software and hardware platform. This document describes an architecture for such remote attestation procedures (RATS).

Note to Readers

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

Source for this draft and an issue tracker can be found at <https://github.com/ietf-rats-wg/architecture> (<https://github.com/ietf-rats-wg/architecture>).

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 19 April 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Terminology	4
3.	Reference Use Cases	5
3.1.	Network Endpoint Assessment	6
3.2.	Confidential Machine Learning (ML) Model Protection	6
3.3.	Confidential Data Retrieval	7
3.4.	Critical Infrastructure Control	7
3.5.	Trusted Execution Environment (TEE) Provisioning	7
3.6.	Hardware Watchdog	8
3.7.	FIDO Biometric Authentication	8
4.	Architectural Overview	9
4.1.	Appraisal Policies	10
4.2.	Reference Values	10
4.3.	Two Types of Environments of an Attester	10
4.4.	Layered Attestation Environments	11
4.5.	Composite Device	13
5.	Topological Models	16
5.1.	Passport Model	16
5.2.	Background-Check Model	17
5.3.	Combinations	18
6.	Roles and Entities	19
7.	Trust Model	20
7.1.	Relying Party	20
7.2.	Attester	21
7.3.	Relying Party Owner	21

7.4. Verifier	21
7.5. Endorser, Reference Value Provider, and Verifier Owner	23
8. Conceptual Messages	23
8.1. Evidence	23
8.2. Endorsements	24
8.3. Attestation Results	24
9. Claims Encoding Formats	25
10. Freshness	27
11. Privacy Considerations	29
12. Security Considerations	29
12.1. Attester and Attestation Key Protection	30
12.1.1. On-Device Attester and Key Protection	30
12.1.2. Attestation Key Provisioning Processes	31
12.2. Integrity Protection	31
13. IANA Considerations	32
14. Acknowledgments	32
15. Notable Contributions	33
16. Appendix A: Time Considerations	33
16.1. Example 1: Timestamp-based Passport Model Example	34
16.2. Example 2: Nonce-based Passport Model Example	36
16.3. Example 3: Handle-based Passport Model Example	37
16.4. Example 4: Timestamp-based Background-Check Model Example	39
16.5. Example 5: Nonce-based Background-Check Model Example	39
17. References	40
17.1. Normative References	40
17.2. Informative References	40
Contributors	41
Authors' Addresses	43

1. Introduction

In Remote Attestation Procedures (RATS), one peer (the "Attester") produces believable information about itself - Evidence - to enable a remote peer (the "Relying Party") to decide whether to consider that Attester a trustworthy peer or not. RATS are facilitated by an additional vital party, the Verifier.

The Verifier appraises Evidence via appraisal policies and creates the Attestation Results to support Relying Parties in their decision process. This document defines a flexible architecture consisting of attestation roles and their interactions via conceptual messages. Additionally, this document defines a universal set of terms that can be mapped to various existing and emerging Remote Attestation Procedures. Common topological models and the data flows associated with them, such as the "Passport Model" and the "Background-Check Model" are illustrated. The purpose is to define useful terminology for attestation and enable readers to map their solution architecture

to the canonical attestation architecture provided here. Having a common terminology that provides well-understood meanings for common themes such as roles, device composition, topological models, and appraisal is vital for semantic interoperability across solutions and platforms involving multiple vendors and providers.

Amongst other things, this document is about trust and trustworthiness. Trust is a choice one makes about another system. Trustworthiness is a quality about the other system that can be used in making one's decision to trust it or not. This is subtle difference and being familiar with the difference is crucial for using this document. Additionally, the concepts of freshness and trust relationships with respect to RATS are elaborated on to enable implementers in order to choose appropriate solutions to compose their Remote Attestation Procedures.

2. Terminology

This document uses the following terms.

Appraisal Policy for Evidence: A set of rules that informs how a Verifier evaluates the validity of information about an Attester. Compare /security policy/ in [RFC4949]

Appraisal Policy for Attestation Results: A set of rules that direct how a Relying Party uses the Attestation Results regarding an Attester generated by the Verifiers. Compare /security policy/ in [RFC4949]

Attestation Result: The output generated by a Verifier, typically including information about an Attester, where the Verifier vouches for the validity of the results

Attester: A role performed by an entity (typically a device) whose Evidence must be appraised in order to infer the extent to which the Attester is considered trustworthy, such as when deciding whether it is authorized to perform some operation

Claim: A piece of asserted information, often in the form of a name/value pair. (Compare /claim/ in [RFC7519])

Endorsement: A secure statement that an Endorser vouches for the integrity of an Attester's various capabilities such as Claims collection and Evidence signing

Endorser: An entity (typically a manufacturer) whose Endorsements help Verifiers appraise the authenticity of Evidence

Evidence: A set of information about an Attester that is to be appraised by a Verifier. Evidence may include configuration data, measurements, telemetry, or inferences.

Reference Value Provider: An entity (typically a manufacturer) whose Reference Values help Verifiers appraise the authenticity of Evidence.

Reference Values: A set of values against which values of Claims can be compared as part of applying an Appraisal Policy for Evidence. Reference Values are sometimes referred to in other documents as known-good values, golden measurements, or nominal values, although those terms typically assume comparison for equality, whereas here Reference Values might be more general and be used in any sort of comparison.

Relying Party: A role performed by an entity that depends on the validity of information about an Attester, for purposes of reliably applying application specific actions. Compare /relying party/ in [RFC4949]

Relying Party Owner: An entity (typically an administrator), that is authorized to configure Appraisal Policy for Attestation Results in a Relying Party

Verifier: A role performed by an entity that appraises the validity of Evidence about an Attester and produces Attestation Results to be used by a Relying Party

Verifier Owner: An entity (typically an administrator), that is authorized to configure Appraisal Policy for Evidence in a Verifier

3. Reference Use Cases

This section covers a number of representative use cases for remote attestation, independent of specific solutions. The purpose is to provide motivation for various aspects of the architecture presented in this draft. Many other use cases exist, and this document does not intend to have a complete list, only to have a set of use cases that collectively cover all the functionality required in the architecture.

Each use case includes a description followed by a summary of the Attester and a Relying Party roles.

3.1. Network Endpoint Assessment

Network operators want a trustworthy report that includes identity and version of information of the hardware and software on the machines attached to their network, for purposes such as inventory, audit, anomaly detection, record maintenance and/or trending reports (logging). The network operator may also want a policy by which full access is only granted to devices that meet some definition of hygiene, and so wants to get claims about such information and verify their validity. Remote attestation is desired to prevent vulnerable or compromised devices from getting access to the network and potentially harming others.

Typically, solutions start with a specific component (called a "root of trust") that provides device identity and protected storage for measurements. The system components perform a series of measurements that may be signed by the root of trust, considered as Evidence about the hardware, firmware, BIOS, software, etc. that is running.

Attester: A device desiring access to a network

Relying Party: A network infrastructure device such as a router, switch, or access point

3.2. Confidential Machine Learning (ML) Model Protection

A device manufacturer wants to protect its intellectual property. This is primarily the ML model it developed and runs in the devices purchased by its customers. The goals for the protection include preventing attackers, potentially the customer themselves, from seeing the details of the model.

This typically works by having some protected environment in the device go through a remote attestation with some manufacturer service that can assess its trustworthiness. If remote attestation succeeds, then the manufacturer service releases either the model, or a key to decrypt a model the Attester already has in encrypted form, to the requester.

Attester: A device desiring to run an ML model

Relying Party: A server or service holding ML models it desires to protect

3.3. Confidential Data Retrieval

This is a generalization of the ML model use case above, where the data can be any highly confidential data, such as health data about customers, payroll data about employees, future business plans, etc. An assessment of system state is made against a set of policies to evaluate the state of a system using attestations for the system requesting data. Attestation is desired to prevent leaking data to compromised devices.

Attester: An entity desiring to retrieve confidential data

Relying Party: An entity that holds confidential data for retrieval by other entities

3.4. Critical Infrastructure Control

In this use case, potentially dangerous physical equipment (e.g., power grid, traffic control, hazardous chemical processing, etc.) is connected to a network. The organization managing such infrastructure needs to ensure that only authorized code and users can control such processes, and they are protected from malware or other adversaries. When a protocol operation can affect some critical system, the device attached to the critical equipment thus wants some assurance that the requester has not been compromised. As such, remote attestation can be used to only accept commands from requesters that are within policy.

Attester: A device or application wishing to control physical equipment

Relying Party: A device or application connected to potentially dangerous physical equipment (hazardous chemical processing, traffic control, power grid, etc.)

3.5. Trusted Execution Environment (TEE) Provisioning

A "Trusted Application Manager (TAM)" server is responsible for managing the applications running in the TEE of a client device. To do this, the TAM wants to assess the state of a TEE, or of applications in the TEE, of a client device. The TEE conducts a remote attestation procedure with the TAM, which can then decide whether the TEE is already in compliance with the TAM's latest policy, or if the TAM needs to uninstall, update, or install approved applications in the TEE to bring it back into compliance with the TAM's policy.

Attester: A device with a trusted execution environment capable of

running trusted applications that can be updated

Relying Party: A Trusted Application Manager

3.6. Hardware Watchdog

One significant problem is malware that holds a device hostage and does not allow it to reboot to prevent updates from being applied. This is a significant problem, because it allows a fleet of devices to be held hostage for ransom.

In the case, the Relying Party is the watchdog timer in the TPM/secure enclave itself, as described in [TCGarch] section 43.3. The Attestation Results are returned to the device, and provided to the enclave.

If the watchdog does not receive regular, and fresh, Attestation Results as to the systems' health, then it forces a reboot.

Attester: The device that is desired to keep from being held hostage for a long period of time

Relying Party: A remote server that will securely grant the Attester permission to continue operating (i.e., not reboot) for a period of time

3.7. FIDO Biometric Authentication

In the Fast IDentity Online (FIDO) protocol [WebAuthN], [CTAP], the device in the user's hand authenticates the human user, whether by biometrics (such as fingerprints), or by PIN and password. FIDO authentication puts a large amount of trust in the device compared to typical password authentication because it is the device that verifies the biometric, PIN and password inputs from the user, not the server. For the Relying Party to know that the authentication is trustworthy, the Relying Party needs to know that the Authenticator part of the device is trustworthy. The FIDO protocol employs remote attestation for this.

The FIDO protocol supports several remote attestation protocols and a mechanism by which new ones can be registered and added. Remote attestation defined by RATS is thus a candidate for use in the FIDO protocol.

Other biometric authentication protocols such as the Chinese IFAA standard and WeChat Pay as well as Google Pay make use of attestation in one form or another.

Attester: Every FIDO Authenticator contains an Attester.

Relying Party: Any web site, mobile application back end or service that does biometric authentication.

4. Architectural Overview

Figure 1 depicts the data that flows between different roles, independent of protocol or use case.

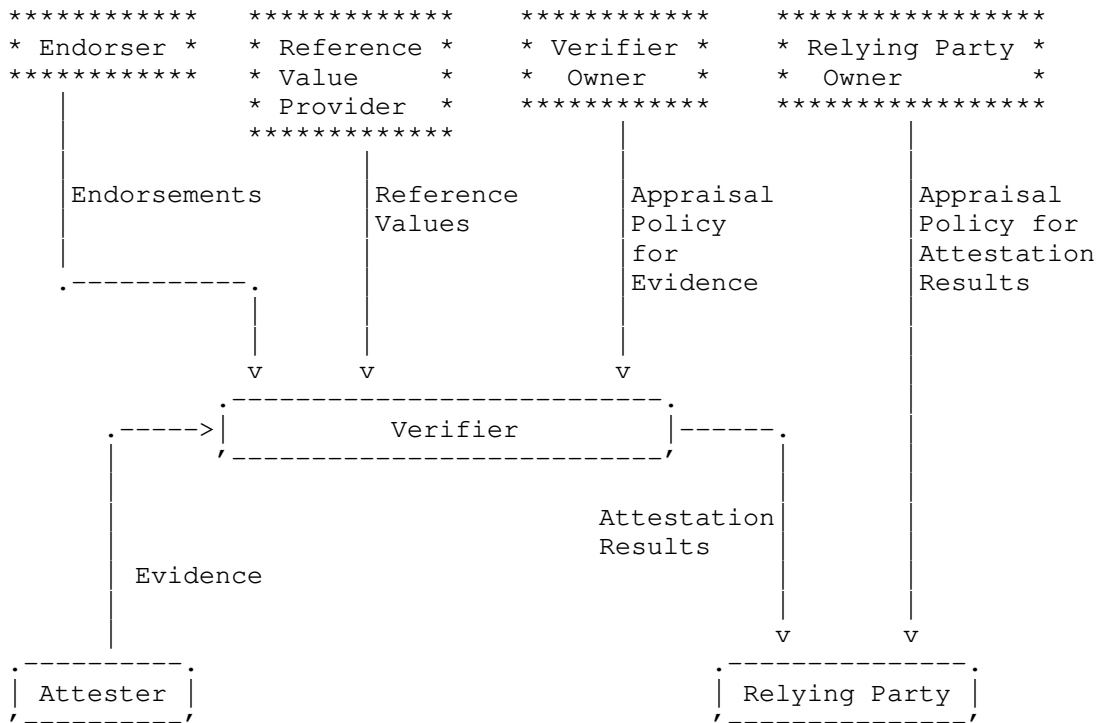


Figure 1: Conceptual Data Flow

An Attester creates Evidence that is conveyed to a Verifier.

The Verifier uses the Evidence, and any Endorsements from Endorsers, by applying an Appraisal Policy for Evidence to assess the trustworthiness of the Attester, and generates Attestation Results for use by Relying Parties. The Appraisal Policy for Evidence might be obtained from an Endorser along with the Endorsements, and/or might be obtained via some other mechanism such as being configured in the Verifier by the Verifier Owner.

The Relying Party uses Attestation Results by applying its own appraisal policy to make application-specific decisions such as authorization decisions. The Appraisal Policy for Attestation Results is configured in the Relying Party by the Relying Party Owner, and/or is programmed into the Relying Party.

4.1. Appraisal Policies

The Verifier, when appraising Evidence, or the Relying Party, when appraising Attestation Results, checks the values of some claims against constraints specified in its appraisal policy. Such constraints might involve a comparison for equality against a Reference Value, or a check for being in a range bounded by Reference Values, or membership in a set of Reference Values, or a check against values in other claims, or any other test.

4.2. Reference Values

Reference Values used in appraisal might be specified as part of the appraisal policy itself, or might be obtained from a separate source, such as an Endorsement, and then used by the appraisal policy.

The actual data format and semantics of any Reference Values are specific to claims and implementations. This architecture document does not define any general purpose format for them or general means for comparison.

4.3. Two Types of Environments of an Attester

An Attester consists of at least one Attesting Environment and at least one Target Environment. In some implementations, the Attesting and Target Environments might be combined. Other implementations might have multiple Attesting and Target Environments, such as in the examples described in more detail in Section 4.4 and Section 4.5. Other examples may exist, and the examples discussed could even be combined into even more complex implementations.

Claims are collected from Target Environments, as shown in Figure 2. That is, Attesting Environments collect the values and the information to be represented in Claims, by reading system registers and variables, calling into subsystems, taking measurements on code or memory and so on of the Target Environment. Attesting Environments then format the claims appropriately, and typically use key material and cryptographic functions, such as signing or cipher algorithms, to create Evidence. There is no limit to or requirement on the places that an Attesting Environment can exist, but they typically are in Trusted Execution Environments (TEE), embedded Secure Elements (eSE), and BIOS firmware. An execution environment

may not, by default, be capable of claims collection for a given Target Environment. Execution environments that are designed to be capable of claims collection are referred to in this document as Attesting Environments.

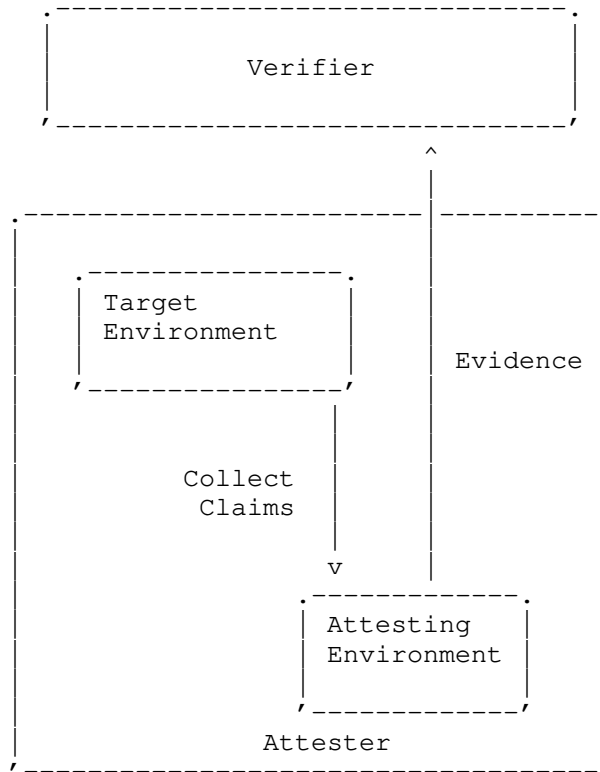


Figure 2: Two Types of Environments

4.4. Layered Attestation Environments

By definition, the Attester role creates Evidence. An Attester may consist of one or more nested or staged environments, adding complexity to the architectural structure. The unifying component is the root of trust and the nested, staged, or chained attestation Evidence produced. The nested or chained structure includes Claims, collected by the Attester to aid in the assurance or believability of the attestation Evidence.

Figure 3 depicts an example of a device that includes (A) a BIOS stored in read-only memory in this example, (B) an updatable bootloader, and (C) an operating system kernel.

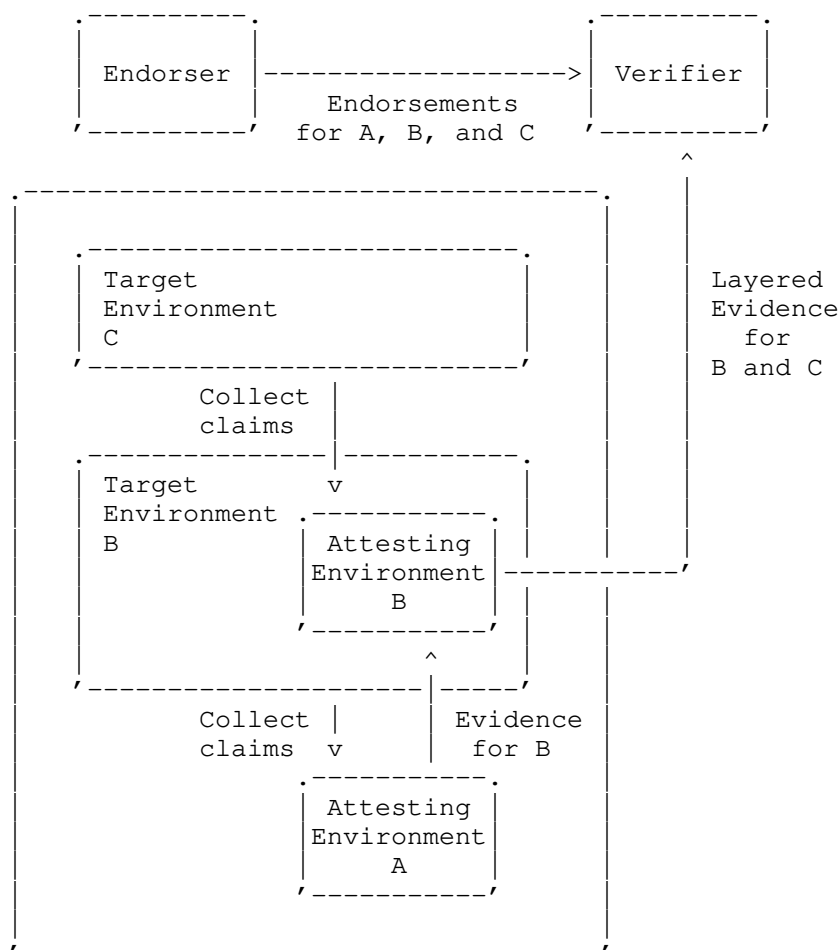


Figure 3: Layered Attester

Attesting Environment A, the read-only BIOS in this example, has to ensure the integrity of the bootloader (Target Environment B). There are potentially multiple kernels to boot, and the decision is up to the bootloader. Only a bootloader with intact integrity will make an appropriate decision. Therefore, these Claims have to be measured securely. At this stage of the boot-cycle of the device, the Claims collected typically cannot be composed into Evidence.

After the boot sequence is started, the BIOS conducts the most important and defining feature of layered attestation, which is that the successfully measured Target Environment B now becomes (or contains) an Attesting Environment for the next layer. This

procedure in Layered Attestation is sometimes called "staging". It is important that the new Attesting Environment B not be able to alter any Claims about its own Target Environment B. This can be ensured having those Claims be either signed by Attesting Environment A or stored in an untamperable manner by Attesting Environment A.

Continuing with this example, the bootloader's Attesting Environment B is now in charge of collecting Claims about Target Environment C, which in this example is the kernel to be booted. The final Evidence thus contains two sets of Claims: one set about the bootloader as measured and signed by the BIOS, plus a set of Claims about the kernel as measured and signed by the bootloader.

This example could be extended further by making the kernel become another Attesting Environment for an application as another Target Environment. This would result in a third set of Claims in the Evidence pertaining to that application.

The essence of this example is a cascade of staged environments. Each environment has the responsibility of measuring the next environment before the next environment is started. In general, the number of layers may vary by device or implementation, and an Attesting Environment might even have multiple Target Environments that it measures, rather than only one as shown in Figure 3.

4.5. Composite Device

A Composite Device is an entity composed of multiple sub-entities such that its trustworthiness has to be determined by the appraisal of all these sub-entities.

Each sub-entity has at least one Attesting Environment collecting the claims from at least one Target Environment, then this sub-entity generates Evidence about its trustworthiness. Therefore each sub-entity can be called an Attester. Among all the Attesters, there may be only some which have the ability to communicate with the Verifier while others do not.

For example, a carrier-grade router consists of a chassis and multiple slots. The trustworthiness of the router depends on all its slots' trustworthiness. Each slot has an Attesting Environment such as a TEE collecting the claims of its boot process, after which it generates Evidence from the claims. Among these slots, only a main slot can communicate with the Verifier while other slots cannot. But other slots can communicate with the main slot by the links between them inside the router. So the main slot collects the Evidence of other slots, produces the final Evidence of the whole router and conveys the final Evidence to the Verifier. Therefore the router is a Composite Device, each slot is an Attester, and the main slot is the lead Attester.

Another example is a multi-chassis router composed of multiple single carrier-grade routers. The multi-chassis router provides higher throughput by interconnecting multiple routers and can be logically treated as one router for simpler management. A multi-chassis router provides a management point that connects to the Verifier. Other routers are only connected to the main router by the network cables, and therefore they are managed and appraised via this main router's help. So, in this case, the multi-chassis router is the Composite Device, each router is an Attester and the main router is the lead Attester.

Figure 4 depicts the conceptual data flow for a Composite Device.

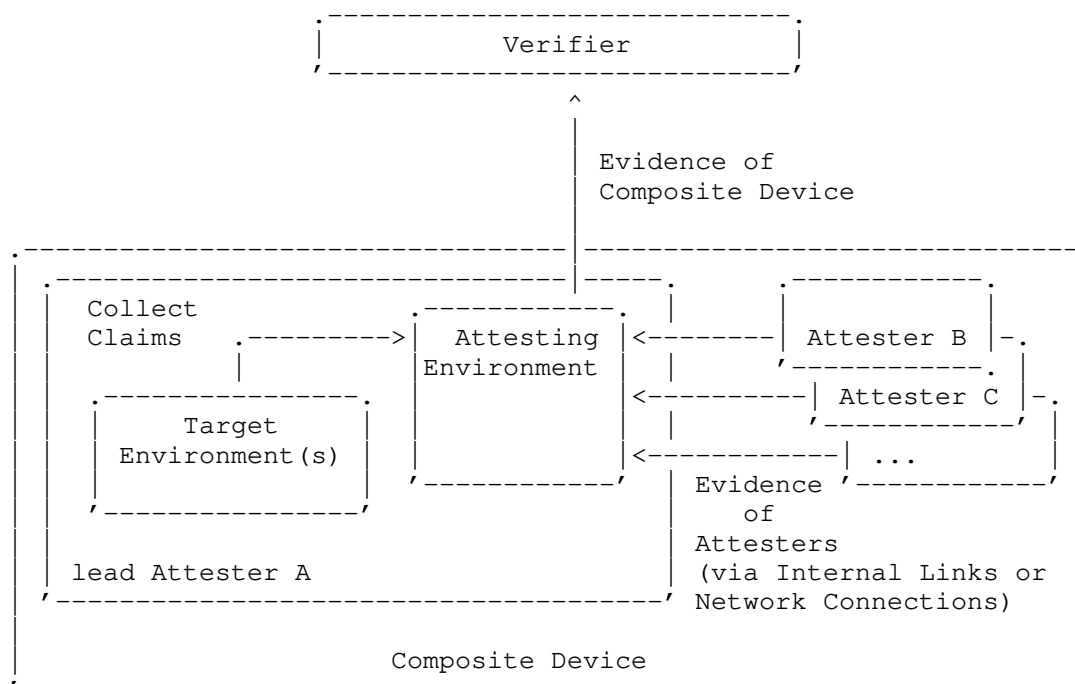


Figure 4: Composite Device

In the Composite Device, each Attester generates its own Evidence by its Attesting Environment(s) collecting the claims from its Target Environment(s). The lead Attester collects the Evidence of all other Attesters and then generates the Evidence of the whole Composite Attester.

An entity can take on multiple RATS roles (e.g., Attester, Verifier, Relying Party, etc.) at the same time. The combination of roles can be arbitrary. For example, in this Composite Device scenario, the entity inside the lead Attester can also take on the role of a Verifier, and the outside entity of Verifier can take on the role of a Relying Party. After collecting the Evidence of other Attesters, this inside Verifier uses Endorsements and appraisal policies (obtained the same way as any other Verifier) in the verification process to generate Attestation Results. The inside Verifier then conveys the Attestation Results of other Attesters, whether in the same conveyance protocol as the Evidence or not, to the outside Verifier.

In this situation, the trust model described in Section 7 is also suitable for this inside Verifier.

5. Topological Models

Figure 1 shows a basic model for communication between an Attester, a Verifier, and a Relying Party. The Attester conveys its Evidence to the Verifier for appraisal, and the Relying Party gets the Attestation Result from the Verifier. There are multiple other possible models. This section includes some reference models. This is not intended to be a restrictive list, and other variations may exist.

5.1. Passport Model

The passport model is so named because of its resemblance to how nations issue passports to their citizens. The nature of the Evidence that an individual needs to provide to its local authority is specific to the country involved. The citizen retains control of the resulting passport document and presents it to other entities when it needs to assert a citizenship or identity claim, such as an airport immigration desk. The passport is considered sufficient because it vouches for the citizenship and identity claims, and it is issued by a trusted authority. Thus, in this immigration desk analogy, the passport issuing agency is a Verifier, the passport is an Attestation Result, and the immigration desk is a Relying Party.

In this model, an Attester conveys Evidence to a Verifier, which compares the Evidence against its appraisal policy. The Verifier then gives back an Attestation Result. If the Attestation Result was a successful one, the Attester can then present the Attestation Result to a Relying Party, which then compares the Attestation Result against its own appraisal policy.

There are three ways in which the process may fail. First, the Verifier may refuse to issue the Attestation Result due to some error in processing, or some missing input to the Verifier. The second way in which the process may fail is when the Attestation Result is examined by the Relying Party, and based upon the appraisal policy, the result does not pass the policy. The third way is when the Verifier is unreachable.

Since the resource access protocol between the Attester and Relying Party includes an Attestation Result, in this model the details of that protocol constrain the serialization format of the Attestation Result. The format of the Evidence on the other hand is only constrained by the Attester-Verifier remote attestation protocol.

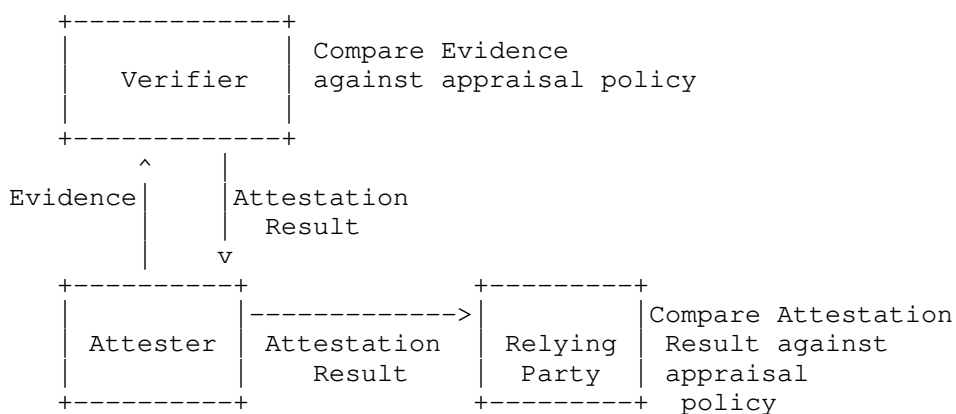


Figure 5: Passport Model

5.2. Background-Check Model

The background-check model is so named because of the resemblance of how employers and volunteer organizations perform background checks. When a prospective employee provides claims about education or previous experience, the employer will contact the respective institutions or former employers to validate the claim. Volunteer organizations often perform police background checks on volunteers in order to determine the volunteer's trustworthiness. Thus, in this analogy, a prospective volunteer is an Attester, the organization is the Relying Party, and a former employer or government agency that issues a report is a Verifier.

In this model, an Attester conveys Evidence to a Relying Party, which simply passes it on to a Verifier. The Verifier then compares the Evidence against its appraisal policy, and returns an Attestation Result to the Relying Party. The Relying Party then compares the Attestation Result against its own appraisal policy.

The resource access protocol between the Attester and Relying Party includes Evidence rather than an Attestation Result, but that Evidence is not processed by the Relying Party. Since the Evidence is merely forwarded on to a trusted Verifier, any serialization format can be used for Evidence because the Relying Party does not need a parser for it. The only requirement is that the Evidence can be encapsulated in the format required by the resource access protocol between the Attester and Relying Party.

However, like in the Passport model, an Attestation Result is still consumed by the Relying Party and so the serialization format of the Attestation Result is still important. If the Relying Party is a

constrained node whose purpose is to serve a given type resource using a standard resource access protocol, it already needs the parser(s) required by that existing protocol. Hence, the ability to let the Relying Party obtain an Attestation Result in the same serialization format allows minimizing the code footprint and attack surface area of the Relying Party, especially if the Relying Party is a constrained node.

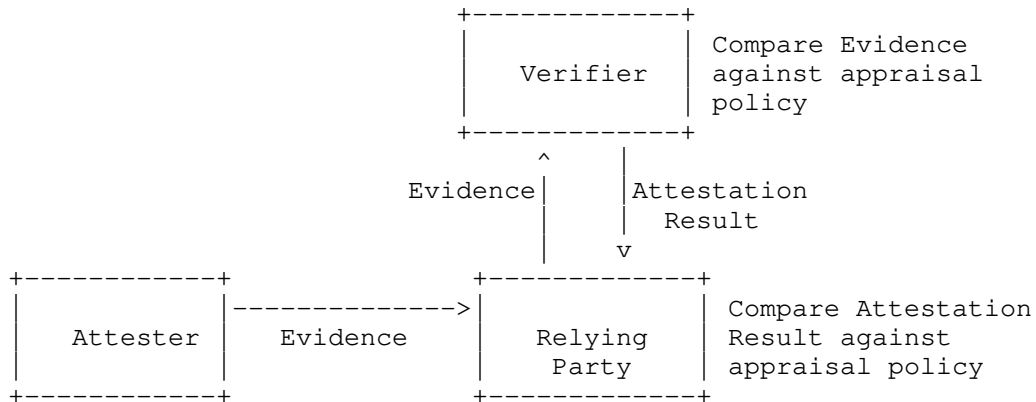


Figure 6: Background-Check Model

5.3. Combinations

One variation of the background-check model is where the Relying Party and the Verifier are on the same machine, performing both functions together. In this case, there is no need for a protocol between the two.

It is also worth pointing out that the choice of model is generally up to the Relying Party. The same device may need to create Evidence for different Relying Parties and/or different use cases. For instance, it would provide Evidence to a network infrastructure device to gain access to the network, and to a server holding confidential data to gain access to that data. As such, both models may simultaneously be in use by the same device.

Figure 7 shows another example of a combination where Relying Party 1 uses the passport model, whereas Relying Party 2 uses an extension of the background-check model. Specifically, in addition to the basic functionality shown in Figure 6, Relying Party 2 actually provides the Attestation Result back to the Attester, allowing the Attester to use it with other Relying Parties. This is the model that the Trusted Application Manager plans to support in the TEEP architecture [I-D.ietf-teep-architecture].

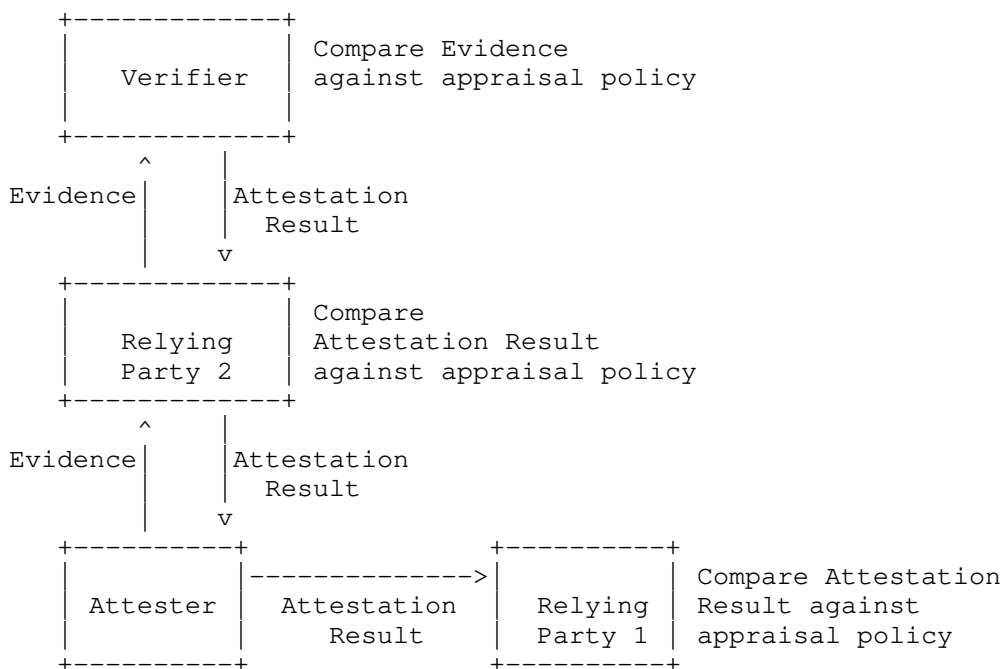


Figure 7: Example Combination

6. Roles and Entities

An entity in the RATS architecture includes at least one of the roles defined in this document. An entity can aggregate more than one role into itself. These collapsed roles combine the duties of multiple roles.

In these cases, interaction between these roles do not necessarily use the Internet Protocol. They can be using a loopback device or other IP-based communication between separate environments, but they do not have to. Alternative channels to convey conceptual messages include function calls, sockets, GPIO interfaces, local busses, or hypervisor calls. This type of conveyance is typically found in Composite Devices. Most importantly, these conveyance methods are out-of-scope of RATS, but they are presumed to exist in order to convey conceptual messages appropriately between roles.

For example, an entity that both connects to a wide-area network and to a system bus is taking on both the Attester and Verifier roles. As a system bus entity, a Verifier consumes Evidence from other devices connected to the system bus that implement Attester roles. As a wide-area network connected entity, it may implement an Attester role. The entity, as a system bus Verifier, may choose to fully isolate its role as a wide-area network Attester.

In essence, an entity that combines more than one role creates and consumes the corresponding conceptual messages as defined in this document.

7. Trust Model

7.1. Relying Party

The scope of this document is scenarios for which a Relying Party trusts a Verifier that can appraise the trustworthiness of information about an Attester. Such trust might come by the Relying Party trusting the Verifier (or its public key) directly, or might come by trusting an entity (e.g., a Certificate Authority) that is in the Verifier's certificate chain.

The Relying Party might implicitly trust a Verifier, such as in a Verifier/Relying Party combination where the Verifier and Relying Party roles are combined. Or, for a stronger level of security, the Relying Party might require that the Verifier first provide information about itself that the Relying Party can use to assess the trustworthiness of the Verifier before accepting its Attestation Results.

For example, one explicit way for a Relying Party "A" to establish such trust in a Verifier "B", would be for B to first act as an Attester where A acts as a combined Verifier/Relying Party. If A then accepts B as trustworthy, it can choose to accept B as a Verifier for other Attesters.

As another example, the Relying Party can establish trust in the Verifier by out of band establishment of key material, combined with a protocol like TLS to communicate. There is an assumption that between the establishment of the trusted key material and the creation of the Evidence, that the Verifier has not been compromised.

Similarly, the Relying Party also needs to trust the Relying Party Owner for providing its Appraisal Policy for Attestation Results, and in some scenarios the Relying Party might even require that the Relying Party Owner go through a remote attestation procedure with it before the Relying Party will accept an updated policy. This can be done similarly to how a Relying Party could establish trust in a Verifier as discussed above.

7.2. Attester

In some scenarios, Evidence might contain sensitive information such as Personally Identifiable Information. Thus, an Attester must trust entities to which it conveys Evidence, to not reveal sensitive data to unauthorized parties. The Verifier might share this information with other authorized parties, according to rules that it controls. In the background-check model, this Evidence may also be revealed to Relying Party(s).

In some cases where Evidence contains sensitive information, an Attester might even require that a Verifier first go through a TLS authentication or a remote attestation procedure with it before the Attester will send the sensitive Evidence. This can be done by having the Attester first act as a Verifier/Relying Party, and the Verifier act as its own Attester, as discussed above.

7.3. Relying Party Owner

The Relying Party Owner might also require that the Relying Party first act as an Attester, providing Evidence that the Owner can appraise, before the Owner would give the Relying Party an updated policy that might contain sensitive information. In such a case, mutual authentication or attestation might be needed, in which case typically one side's Evidence must be considered safe to share with an untrusted entity, in order to bootstrap the sequence.

7.4. Verifier

The Verifier trusts (or more specifically, the Verifier's security policy is written in a way that configures the Verifier to trust) a manufacturer, or the manufacturer's hardware, so as to be able to appraise the trustworthiness of that manufacturer's devices. In a typical solution, a Verifier comes to trust an Attester indirectly by having an Endorser (such as a manufacturer) vouch for the Attester's ability to securely generate Evidence.

In some solutions, a Verifier might be configured to directly trust an Attester by having the Verifier have the Attester's key material (rather than the Endorser's) in its trust anchor store.

Such direct trust must first be established at the time of trust anchor store configuration either by checking with an Endorser at that time, or by conducting a security analysis of the specific device. Having the Attester directly in the trust anchor store narrows the Verifier's trust to only specific devices rather than all devices the Endorser might vouch for, such as all devices manufactured by the same manufacturer in the case that the Endorser is a manufacturer.

Such narrowing is often important since physical possession of a device can also be used to conduct a number of attacks, and so a device in a physically secure environment (such as one's own premises) may be considered trusted whereas devices owned by others would not be. This often results in a desire to either have the owner run their own Endorser that would only endorse devices one owns, or to use Attesters directly in the trust anchor store. When there are many Attesters owned, the use of an Endorser becomes more scalable.

That is, it might appraise the trustworthiness of an application component, operating system component, or service under the assumption that information provided about it by the lower-layer firmware or software is true. A stronger level of assurance of security comes when information can be vouched for by hardware or by ROM code, especially if such hardware is physically resistant to hardware tampering. In most cases, components that have to be vouched for via Endorsements because no Evidence is generated about them are referred to as roots of trust.

The manufacturer of the Attester arranges for its Attesting Environment to be provisioned with key material. The key material is typically in the form of an asymmetric key pair (e.g., an RSA or ECDSA private key and a manufacturer-signed IDevID certificate) secured in the Attester.

The Verifier is provided with an appropriate trust anchor, or provided with a database of public keys (rather than certificates), or even carefully secured lists of symmetric keys. The nature of how the Verifier manages to validate the signatures produced by the Attester is critical to the secure operation an Attestation system, but is not the subject of standardization within this architecture.

A conveyance protocol that provides authentication and integrity protection can be used to convey unprotected Evidence, assuming the following properties exists:

1. The key material used to authenticate and integrity protect the conveyance channel is trusted by the Verifier to speak for the Attesting Environment(s) that collected claims about the Target Environment(s).
2. All unprotected Evidence that is conveyed is supplied exclusively by the Attesting Environment that has the key material that protects the conveyance channel
3. The root of trust protects both the conveyance channel key material and the Attesting Environment with equivalent strength protections.

See Section 12 for discussion on security strength.

7.5. Endorser, Reference Value Provider, and Verifier Owner

In some scenarios, the Endorser, Reference Value Provider, and Verifier Owner may need to trust the Verifier before giving the Endorsement, Reference Values, or appraisal policy to it. This can be done similarly to how a Relying Party might establish trust in a Verifier as discussed above, and in such a case, mutual authentication or attestation might even be needed as discussed in Section 7.3.

8. Conceptual Messages

8.1. Evidence

Evidence is a set of claims about the target environment that reveal operational status, health, configuration or construction that have security relevance. Evidence is evaluated by a Verifier to establish its relevance, compliance, and timeliness. Claims need to be collected in a manner that is reliable. Evidence needs to be securely associated with the target environment so that the Verifier cannot be tricked into accepting claims originating from a different environment (that may be more trustworthy). Evidence also must be protected from man-in-the-middle attackers who may observe, change or misdirect Evidence as it travels from Attester to Verifier. The timeliness of Evidence can be captured using claims that pinpoint the time or interval when changes in operational status, health, and so forth occur.

8.2. Endorsements

An Endorsement is a secure statement that some entity (e.g., a manufacturer) vouches for the integrity of the device's signing capability. For example, if the signing capability is in hardware, then an Endorsement might be a manufacturer certificate that signs a public key whose corresponding private key is only known inside the device's hardware. Thus, when Evidence and such an Endorsement are used together, an appraisal procedure can be conducted based on appraisal policies that may not be specific to the device instance, but merely specific to the manufacturer providing the Endorsement. For example, an appraisal policy might simply check that devices from a given manufacturer have information matching a set of Reference Values, or an appraisal policy might have a set of more complex logic on how to appraise the validity of information.

However, while an appraisal policy that treats all devices from a given manufacturer the same may be appropriate for some use cases, it would be inappropriate to use such an appraisal policy as the sole means of authorization for use cases that wish to constrain which compliant devices are considered authorized for some purpose. For example, an enterprise using remote attestation for Network Endpoint Assessment may not wish to let every healthy laptop from the same manufacturer onto the network, but instead only want to let devices that it legally owns onto the network. Thus, an Endorsement may be helpful information in authenticating information about a device, but is not necessarily sufficient to authorize access to resources which may need device-specific information such as a public key for the device or component or user on the device.

8.3. Attestation Results

Attestation Results are the input used by the Relying Party to decide the extent to which it will trust a particular Attester, and allow it to access some data or perform some operation. Attestation Results may be a Boolean simply indicating compliance or non-compliance with a Verifier's appraisal policy, or a rich set of Claims about the Attester, against which the Relying Party applies its Appraisal Policy for Attestation Results.

A result that indicates non-compliance can be used by an Attester (in the passport model) or a Relying Party (in the background-check model) to indicate that the Attester should not be treated as authorized and may be in need of remediation. In some cases, it may even indicate that the Evidence itself cannot be authenticated as being correct.

An Attestation Result that indicates compliance can be used by a Relying Party to make authorization decisions based on the Relying Party's appraisal policy. The simplest such policy might be to simply authorize any party supplying a compliant Attestation Result signed by a trusted Verifier. A more complex policy might also entail comparing information provided in the result against Reference Values, or applying more complex logic on such information.

Thus, Attestation Results often need to include detailed information about the Attester, for use by Relying Parties, much like physical passports and drivers licenses include personal information such as name and date of birth. Unlike Evidence, which is often very device- and vendor-specific, Attestation Results can be vendor-neutral if the Verifier has a way to generate vendor-agnostic information based on the appraisal of vendor-specific information in Evidence. This allows a Relying Party's appraisal policy to be simpler, potentially based on standard ways of expressing the information, while still allowing interoperability with heterogeneous devices.

Finally, whereas Evidence is signed by the device (or indirectly by a manufacturer, if Endorsements are used), Attestation Results are signed by a Verifier, allowing a Relying Party to only need a trust relationship with one entity, rather than a larger set of entities, for purposes of its appraisal policy.

9. Claims Encoding Formats

The following diagram illustrates a relationship to which remote attestation is desired to be added:

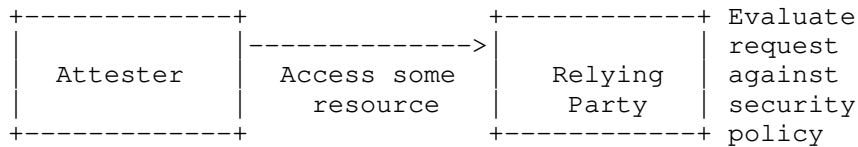


Figure 8: Typical Resource Access

In this diagram, the protocol between Attester and a Relying Party can be any new or existing protocol (e.g., HTTP(S), COAP(S), ROLIE [RFC8322], 802.1x, OPC UA, etc.), depending on the use case. Such protocols typically already have mechanisms for passing security information for purposes of authentication and authorization. Common formats include JWTs [RFC7519], CWTs [RFC8392], and X.509 certificates.

To enable remote attestation to be added to existing protocols, enabling a higher level of assurance against malware for example, it is important that information needed for appraising the Attester be usable with existing protocols that have constraints around what formats they can transport. For example, OPC UA [OPCUA] (probably the most common protocol in industrial IoT environments) is defined to carry X.509 certificates and so security information must be embedded into an X.509 certificate to be passed in the protocol. Thus, remote attestation related information could be natively encoded in X.509 certificate extensions, or could be natively encoded in some other format (e.g., a CWT) which in turn is then encoded in an X.509 certificate extension.

Especially for constrained nodes, however, there is a desire to minimize the amount of parsing code needed in a Relying Party, in order to both minimize footprint and to minimize the attack surface area. So while it would be possible to embed a CWT inside a JWT, or a JWT inside an X.509 extension, etc., there is a desire to encode the information natively in the format that is natural for the Relying Party.

This motivates having a common "information model" that describes the set of remote attestation related information in an encoding-agnostic way, and allowing multiple encoding formats (CWT, JWT, X.509, etc.) that encode the same information into the claims format needed by the Relying Party.

The following diagram illustrates that Evidence and Attestation Results might each have multiple possible encoding formats, so that they can be conveyed by various existing protocols. It also motivates why the Verifier might also be responsible for accepting Evidence that encodes claims in one format, while issuing Attestation Results that encode claims in a different format.

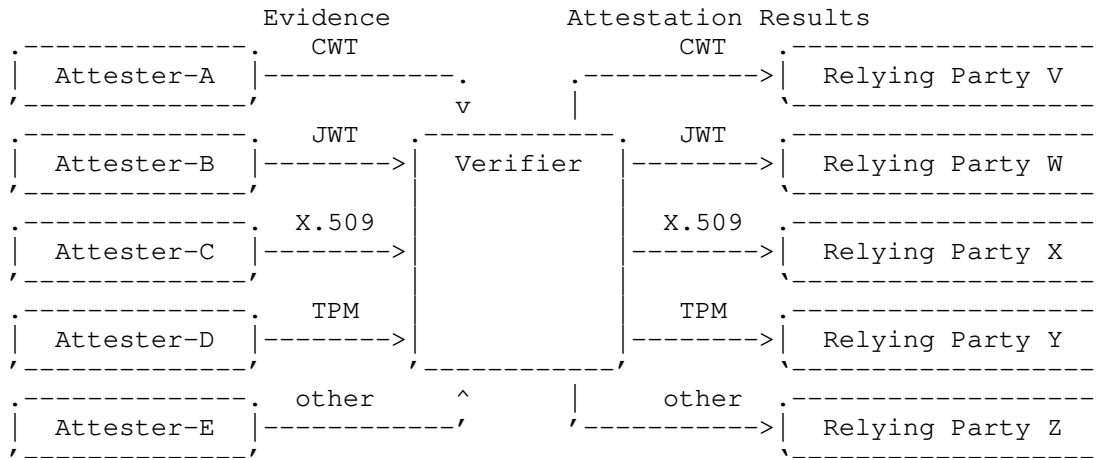


Figure 9: Multiple Attesters and Relying Parties with Different Formats

10. Freshness

A Verifier or Relying Party may need to learn the point in time (i.e., the "epoch") an Evidence or Attestation Result has been produced. This is essential in deciding whether the included Claims and their values can be considered fresh, meaning they still reflect the latest state of the Attester, and that any Attestation Result was generated using the latest Appraisal Policy for Evidence.

Freshness is assessed based on the Appraisal Policy for Evidence or Attestation Results, that compares the estimated epoch against an "expiry" threshold defined locally to that policy. There is, however, always a race condition possible in that the state of the Attester, and the appraisal policies might change immediately after the Evidence or Attestation Result was generated. The goal is merely to narrow their recentness to something the Verifier (for Evidence) or Relying Party (for Attestation Result) is willing to accept. Freshness is a key component for enabling caching and reuse of both Evidence and Attestation Results, which is especially valuable in cases where their computation uses a substantial part of the resource budget (e.g., energy in constrained devices).

There are two common approaches for determining the epoch of an Evidence or Attestation Result.

The first approach is to rely on synchronized and trustworthy clocks, and include a signed timestamp (see [I-D.birkholz-rats-tuda]) along with the Claims in the Evidence or Attestation Result. Timestamps

can be added on a per-Claim basis, to distinguish the time of creation of Evidence or Attestation Result from the time that a specific Claim was generated. The clock's trustworthiness typically requires additional Claims about the signer's time synchronization mechanism.

A second approach places the onus of timekeeping solely on the Verifier (for Evidence), or the Relying Party (for Attestation Results), and might be suitable, for example, in case the Attester does not have a reliable clock or time synchronisation is otherwise impaired. In this approach, a non-predictable nonce is sent by the appraising entity, and the nonce is then signed and included along with the Claims in the Evidence or Attestation Result. After checking that the sent and received nonces are the same, the appraising entity knows that the Claims were signed after the nonce was generated. This allows associating a "rough" epoch to the Evidence or Attestation Result. In this case the epoch is said to be rough because:

- * The epoch applies to the entire claim set instead of a more granular association, and
- * The time between the creation of Claims and the collection of Claims is indistinguishable.

Implicit and explicit timekeeping can be combined into hybrid mechanisms. For example, if clocks exist and are considered trustworthy but are not synchronized, a nonce-based exchange may be used to determine the (relative) time offset between the involved peers, followed by any number of timestamp based exchanges. In another setup where all Roles (Attesters, Verifiers and Relying Parties) share the same broadcast channel, the nonce-based approach may be used to anchor all parties to the same (relative) timeline, without requiring synchronized clocks, by having a central entity emit nonces at regular intervals and have the "current" nonce included in the produced Evidence or Attestation Result.

It is important to note that the actual values in Claims might have been generated long before the Claims are signed. If so, it is the signer's responsibility to ensure that the values are still correct when they are signed. For example, values generated at boot time might have been saved to secure storage until network connectivity is established to the remote Verifier and a nonce is obtained.

A more detailed discussion with examples appears in Section 16.

11. Privacy Considerations

The conveyance of Evidence and the resulting Attestation Results reveal a great deal of information about the internal state of a device as well as any users the device is associated with. In many cases, the whole point of the Attestation process is to provide reliable information about the type of the device and the firmware/software that the device is running. This information might be particularly interesting to many attackers. For example, knowing that a device is running a weak version of firmware provides a way to aim attacks better.

Many claims in Attestation Evidence and Attestation Results are potentially PII (Personally Identifying Information) depending on the end-to-end use case of the attestation. Attestation that goes up to include containers and applications may further reveal details about a specific system or user.

In some cases, an attacker may be able to make inferences about attestations from the results or timing of the processing. For example, an attacker might be able to infer the value of specific claims if it knew that only certain values were accepted by the Relying Party.

Evidence and Attestation Results data structures are expected to support integrity protection encoding (e.g., COSE, JOSE, X.509) and optionally might support confidentiality protection (e.g., COSE, JOSE). Therefore, if confidentiality protection is omitted or unavailable, the protocols that convey Evidence or Attestation Results are responsible for detailing what kinds of information are disclosed, and to whom they are exposed.

Furthermore, because Evidence might contain sensitive information, Attesters are responsible for only sending such Evidence to trusted Verifiers. Some Attesters might want a stronger level of assurance of the trustworthiness of a Verifier before sending Evidence to it. In such cases, an Attester can first act as a Relying Party and ask for the Verifier's own Attestation Result, and appraising it just as a Relying Party would appraise an Attestation Result for any other purpose.

12. Security Considerations

12.1. Attester and Attestation Key Protection

Implementers need to pay close attention to the isolation and protection of the Attester and the factory processes for provisioning the Attestation Key Material. When either of these are compromised, the remote attestation becomes worthless because the attacker can forge Evidence.

Remote attestation applies to use cases with a range of security requirements, so the protections discussed here range from low to high security where low security may be only application or process isolation by the device's operating system and high security involves specialized hardware to defend against physical attacks on a chip.

12.1.1. On-Device Attester and Key Protection

It is assumed that the Attester is located in an isolated environment of a device like a process, a dedicated chip a TEE or such that collects the Claims, formats them and signs them with an Attestation Key. The Attester must be protected from unauthorized modification to ensure it behaves correctly. There must also be confidentiality so that the signing key is not captured and used elsewhere to forge evidence.

In many cases the user or owner of the device must not be able to modify or exfiltrate keys from the Attesting Environment of the Attester. For example the owner or user of a mobile phone or FIDO authenticator is not trusted. The point of remote attestation is for the Relying Party to be able to trust the Attester even though they don't trust the user or owner.

Some of the measures for low level security include process or application isolation by a high-level operating system, and perhaps restricting access to root or system privilege. For extremely simple single-use devices that don't use a protected mode operating system, like a Bluetooth speaker, the isolation might only be the plastic housing for the device.

At medium level security, a special restricted operating environment like a Trusted Execution Environment (TEE) might be used. In this case, only security-oriented software has access to the Attester and key material.

For high level security, specialized hardware will likely be used providing protection against chip decapping attacks, power supply and clock glitching, faulting injection and RF and power side channel attacks.

12.1.2. Attestation Key Provisioning Processes

Attestation key provisioning is the process that occurs in the factory or elsewhere that establishes the signing key material on the device and the verification key material off the device. Sometimes this is referred to as "personalization".

One way to provision a key is to first generate it external to the device and then copy the key onto the device. In this case, confidentiality of the generator, as well as the path over which the key is provisioned, is necessary. This can be achieved in a number of ways.

Confidentiality can be achieved entirely with physical provisioning facility security involving no encryption at all. For low-security use cases, this might be simply locking doors and limiting personnel that can enter the facility. For high-security use cases, this might involve a special area of the facility accessible only to select security-trained personnel.

Cryptography can also be used to support confidentiality, but keys that are used to then provision attestation keys must somehow have been provisioned securely beforehand (a recursive problem).

In many cases both some physical security and some cryptography will be necessary and useful to establish confidentiality.

Another way to provision the key material is to generate it on the device and export the verification key. If public key cryptography is being used, then only integrity is necessary. Confidentiality is not necessary.

In all cases, the Attestation Key provisioning process must ensure that only attestation key material that is generated by a valid Endorser is established in Attesters and then configured correctly. For many use cases, this will involve physical security at the facility, to prevent unauthorized devices from being manufactured that may be counterfeit or incorrectly configured.

12.2. Integrity Protection

Any solution that conveys information used for security purposes, whether such information is in the form of Evidence, Attestation Results, Endorsements, or appraisal policy must support end-to-end integrity protection and replay attack prevention, and often also needs to support additional security properties, including:

- * end-to-end encryption,

- * denial of service protection,
- * authentication,
- * auditing,
- * fine grained access controls, and
- * logging.

Section 10 discusses ways in which freshness can be used in this architecture to protect against replay attacks.

To assess the security provided by a particular appraisal policy, it is important to understand the strength of the root of trust, e.g., whether it is mutable software, or firmware that is read-only after boot, or immutable hardware/ROM.

It is also important that the appraisal policy was itself obtained securely. As such, if appraisal policies for a Relying Party or for a Verifier can be configured via a network protocol, the ability to create Evidence about the integrity of the entity providing the appraisal policy needs to be considered.

The security of conveyed information may be applied at different layers, whether by a conveyance protocol, or an information encoding format. This architecture expects attestation messages (i.e., Evidence, Attestation Results, Endorsements and Policies) are end-to-end protected based on the role interaction context. For example, if an Attester produces Evidence that is relayed through some other entity that doesn't implement the Attester or the intended Verifier roles, then the relaying entity should not expect to have access to the Evidence.

13. IANA Considerations

This document does not require any actions by IANA.

14. Acknowledgments

Special thanks go to Joerg Borchert, Nancy Cam-Winget, Jessica Fitzgerald-McKay, Thomas Fossati, Diego Lopez, Laurence Lundblade, Paul Rowe, Hannes Tschofenig, Frank Xia, and David Wooten.

15. Notable Contributions

Thomas Hardjono created older versions of the terminology section in collaboration with Ned Smith. Eric Voit provided the conceptual separation between Attestation Provision Flows and Attestation Evidence Flows. Monty Wisemen created the content structure of the first three architecture drafts. Carsten Bormann provided many of the motivational building blocks with respect to the Internet Threat Model.

16. Appendix A: Time Considerations

The table below defines a number of relevant events, with an ID that is used in subsequent diagrams. The times of said events might be defined in terms of an absolute clock time such as Coordinated Universal Time, or might be defined relative to some other timestamp or timeticks counter.

ID	Event	Explanation of event
VG	Value generated	A value to appear in a Claim was created. In some cases, a value may have technically existed before an Attester became aware of it but the Attester might have no idea how long it has had that value. In such a case, the Value created time is the time at which the Claim containing the copy of the value was created.
HD	Handle distribution	A centrally generated identifier for time-bound recentness across a domain of devices is successfully distributed to Attesters.
NS	Nonce sent	A nonce not predictable to an Attester (recentness & uniqueness) is sent to an Attester.
NR	Nonce relayed	A nonce is relayed to an Attester by another entity.
HR	Handle received	A handle distributed by a Handle Distributor was received.
EG	Evidence generation	An Attester creates Evidence from collected Claims.
ER	Evidence	A Relying Party relays Evidence to a

	relayed	Verifier.
RG	Result generation	A Verifier appraises Evidence and generates an Attestation Result.
RR	Result relayed	A Relying Party relays an Attestation Result to a Relying Party.
RA	Result appraised	The Relying Party appraises Attestation Results.
OP	Operation performed	The Relying Party performs some operation requested by the Attester. For example, acting upon some message just received across a session created earlier at time(RA).
RX	Result expiry	An Attestation Result should no longer be accepted, according to the Verifier that generated it.

Table 1

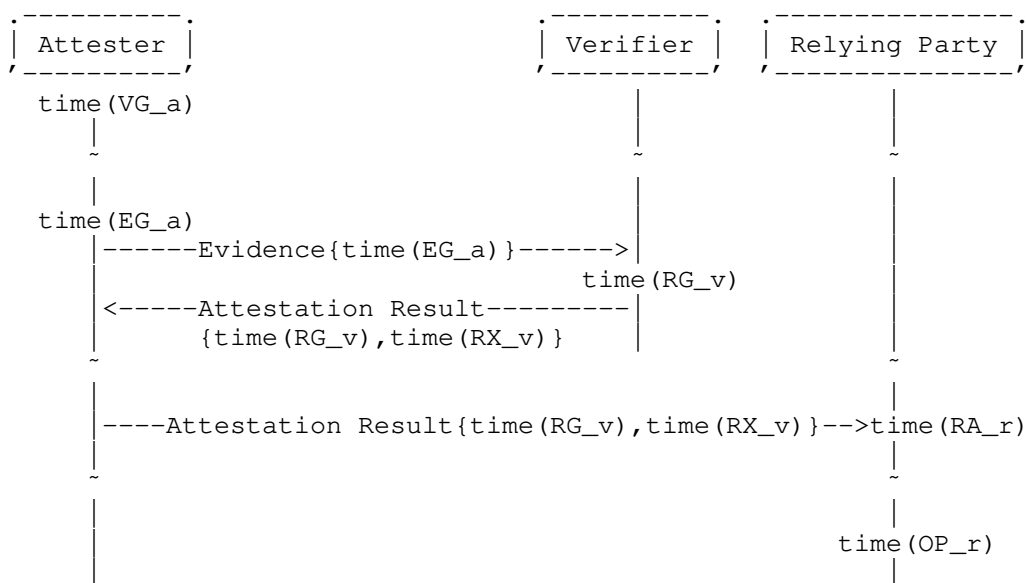
Using the table above, a number of hypothetical examples of how a solution might be built are illustrated below. a solution might be built. This list is not intended to be complete, but is just representative enough to highlight various timing considerations.

All times are relative to the local clocks, indicated by an "a" (Attester), "v" (Verifier), or "r" (Relying Party) suffix.

How and if clocks are synchronized depends upon the model.

16.1. Example 1: Timestamp-based Passport Model Example

The following example illustrates a hypothetical Passport Model solution that uses timestamps and requires roughly synchronized clocks between the Attester, Verifier, and Relying Party, which depends on using a secure clock synchronization mechanism. As a result, the receiver of a conceptual message containing a timestamp can directly compare it to its own clock and timestamps.



The Verifier can check whether the Evidence is fresh when appraising it at time(RG_v) by checking "time(RG_v) - time(EG_a) < Threshold", where the Verifier's threshold is large enough to account for the maximum permitted clock skew between the Verifier and the Attester.

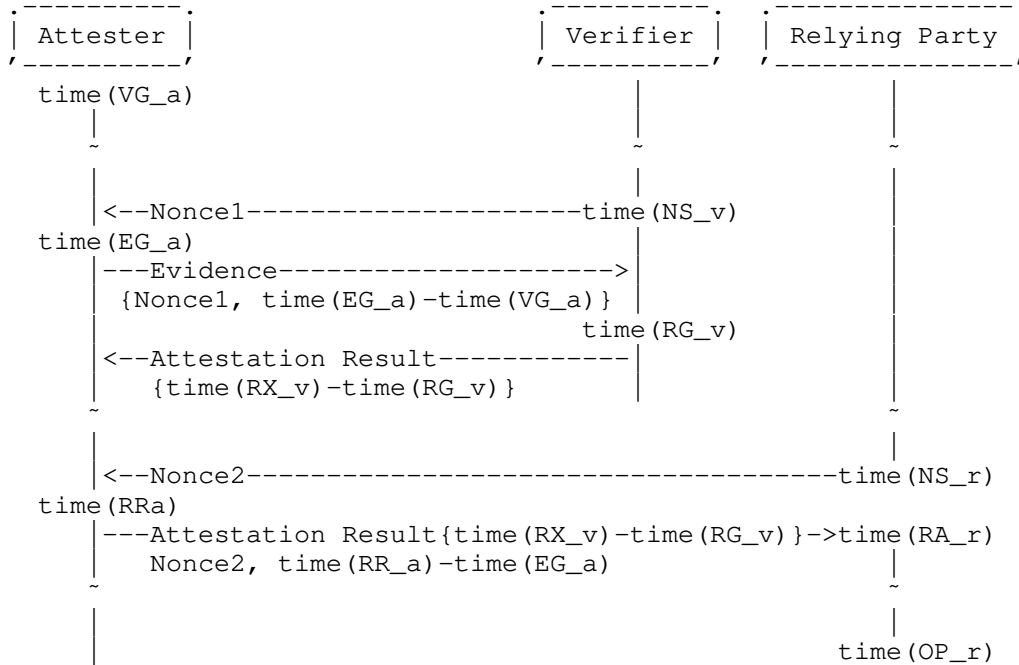
If time(VG_a) is also included in the Evidence along with the claim value generated at that time, and the Verifier decides that it can trust the time(VG_a) value, the Verifier can also determine whether the claim value is recent by checking "time(RG_v) - time(VG_a) < Threshold", again where the threshold is large enough to account for the maximum permitted clock skew between the Verifier and the Attester.

The Relying Party can check whether the Attestation Result is fresh when appraising it at time(RA_r) by checking "time(RA_r) - time(RG_v) < Threshold", where the Relying Party's threshold is large enough to account for the maximum permitted clock skew between the Relying Party and the Verifier. The result might then be used for some time (e.g., throughout the lifetime of a connection established at time(RA_r)). The Relying Party must be careful, however, to not allow continued use beyond the period for which it deems the Attestation Result to remain fresh enough. Thus, it might allow use (at time(OP_r)) as long as "time(OP_r) - time(RG_v) < Threshold". However, if the Attestation Result contains an expiry time time(RX_v) then it could explicitly check "time(OP_r) < time(RX_v)".

16.2. Example 2: Nonce-based Passport Model Example

The following example illustrates a hypothetical Passport Model solution that uses nonces and thus does not require that any clocks are synchronized.

As a result, the receiver of a conceptual message containing a timestamp cannot directly compare it to its own clock or timestamps. Thus we use a suffix ("a" for Attester, "v" for Verifier, and "r" for Relying Party) on the IDs below indicating which clock generated them, since times from different clocks cannot be compared. Only the delta between two events from the sender can be used by the receiver.



In this example solution, the Verifier can check whether the Evidence is fresh at "time(RG_v)" by verifying that "time(RG_v) - time(NS_v) < Threshold".

The Verifier cannot, however, simply rely on a Nonce to determine whether the value of a claim is recent, since the claim value might have been generated long before the nonce was sent by the Verifier. However, if the Verifier decides that the Attester can be trusted to correctly provide the delta "time(EG_a) - time(VG_a)", then it can determine recency by checking "time(RG_v) - time(NS_v) + time(EG_a) - time(VG_a) < Threshold".

Similarly if, based on an Attestation Result from a Verifier it trusts, the Relying Party decides that the Attester can be trusted to correctly provide time deltas, then it can determine whether the Attestation Result is fresh by checking $\text{time(OP_r)} - \text{time(NS_r)} + \text{time(RR_a)} - \text{time(EG_a)} < \text{Threshold}$. Although the Nonce2 and $\text{time(RR_a)} - \text{time(EG_a)}$ values cannot be inside the Attestation Result, they might be signed by the Attester such that the Attestation Result vouches for the Attester's signing capability.

The Relying Party must still be careful, however, to not allow continued use beyond the period for which it deems the Attestation Result to remain valid. Thus, if the Attestation Result sends a validity lifetime in terms of $\text{time(RX_v)} - \text{time(RG_v)}$, then the Relying Party can check $\text{time(OP_r)} - \text{time(NS_r)} < \text{time(RX_v)} - \text{time(RG_v)}$.

16.3. Example 3: Handle-based Passport Model Example

Handles are a third option to establish time-keeping next to nonces or timestamps. Handles are opaque data intended to be available to all RATS roles that interact with each other, such as the Attester or Verifier, in specified intervals. To enable this availability, handles are distributed centrally by the Handle Distributor role over the network. As any other role, the Handle Distributor role can be taken on by a dedicated entity or collapsed with other roles, such as a Verifier. The use of handles can compensate for a lack of clocks or other sources of time on entities taking on RATS roles. The only entity that requires access to a source of time is the entity taking on the role of Handle Distributor.

Handles are different from nonces as they can be used more than once and can be used by more than one entity at the same time. Handles are different from timestamps as they do not have to convey information about a point in time, but their reception creates that information. The reception of a handle is similar to the event that increments a relative tickcounter. Receipt of a new handle invalidates a previously received handle.

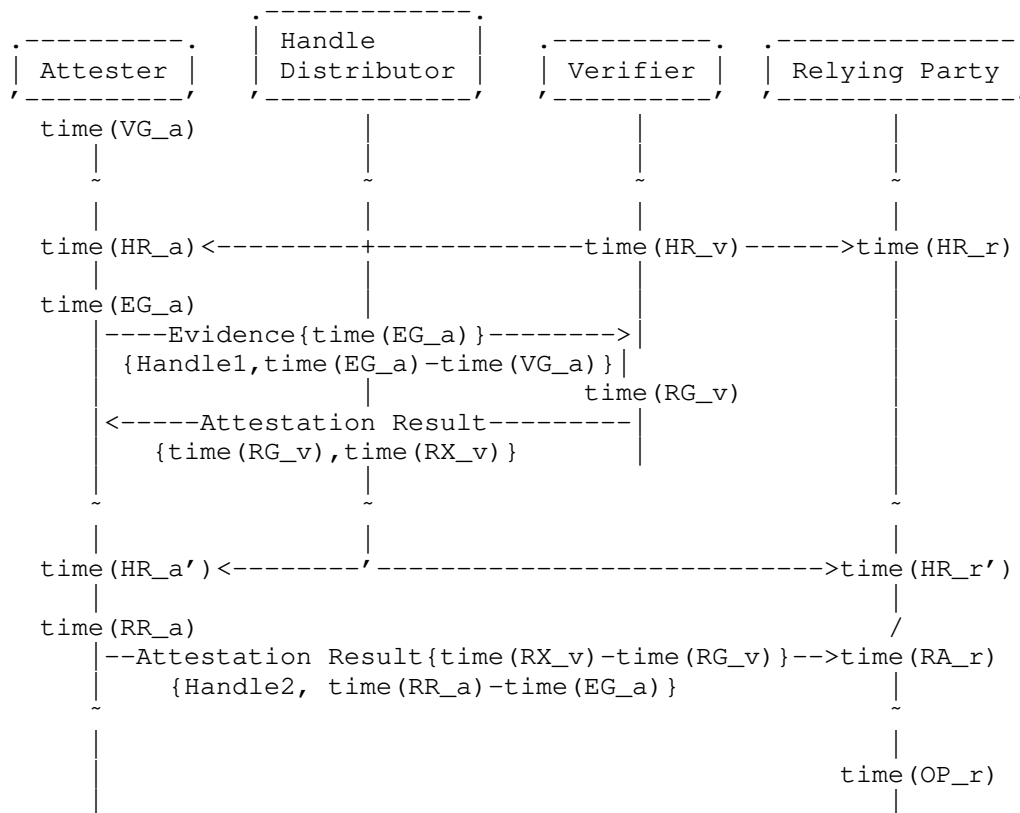
In this example, Evidence generation based on received handles always uses the current (most recent) handle. As handles are distributed over the network, all involved entities receive a fresh handle at roughly the same time. Due to distribution over the network, there is some jitter with respect to the time the Handle is received, time(HR) , for each involved entity. To compensate for this jitter, there is a small period of overlap (a specified offset) in which both a current handle and corresponding former handle are valid in Evidence appraisal: $\text{validity-duration} = \text{time(HR_v)} + \text{offset} - \text{time(HR_v)}$. The offset is typically based on a network's round trip

time. Analogously, the generation of valid Evidence is only possible, if the age of the handle used is lower than the validity-duration: "time(HR_v) - time(EG_a) < validity-duration".

From the point of view of a Verifier, the generation of valid Evidence is only possible, if the age of the handle used in the Evidence generation is younger than the duration of the distribution interval - "(time(HR'_v)-time(HR_v)) - (time(HR_a)-time(EG_a)) < validity-duration".

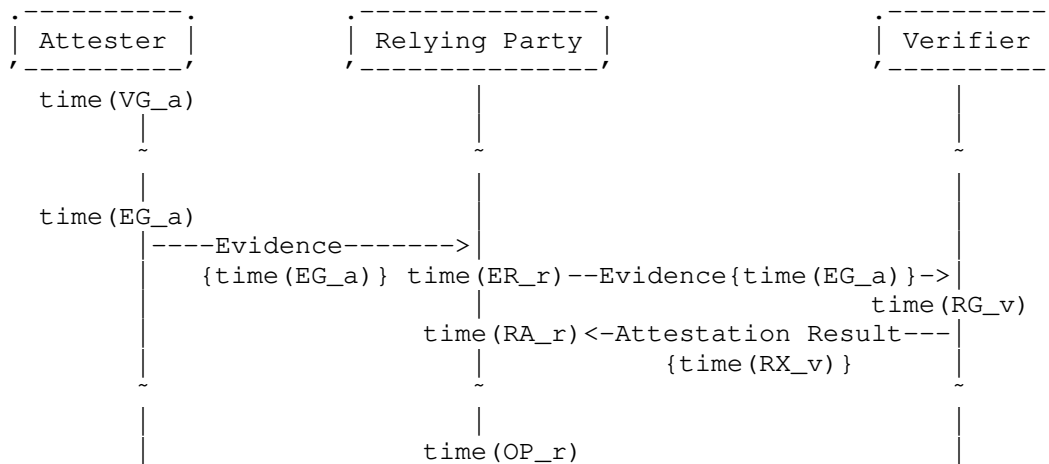
Due to the validity-duration of handles, multiple different pieces of Evidence can be generated based on the same handle. The resulting granularity (time resolution) of Evidence freshness is typically lower than the resolution of clock-based tickcounters.

The following example illustrates a hypothetical Background-Check Model solution that uses handles and requires a trustworthy time source available to the Handle Distributor role.



16.4. Example 4: Timestamp-based Background-Check Model Example

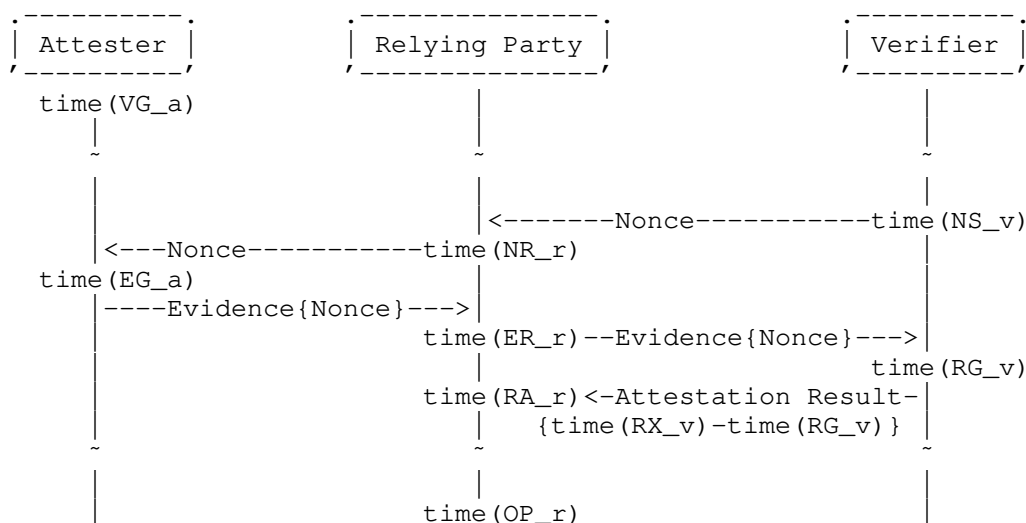
The following example illustrates a hypothetical Background-Check Model solution that uses timestamps and requires roughly synchronized clocks between the Attester, Verifier, and Relying Party.



The time considerations in this example are equivalent to those discussed under Example 1 above.

16.5. Example 5: Nonce-based Background-Check Model Example

The following example illustrates a hypothetical Background-Check Model solution that uses nonces and thus does not require that any clocks are synchronized. In this example solution, a nonce is generated by a Verifier at the request of a Relying Party, when the Relying Party needs to send one to an Attester.



The Verifier can check whether the Evidence is fresh, and whether a claim value is recent, the same as in Example 2 above.

However, unlike in Example 2, the Relying Party can use the Nonce to determine whether the Attestation Result is fresh, by verifying that "time(OP_r)-time(NR_r) < Threshold".

The Relying Party must still be careful, however, to not allow continued use beyond the period for which it deems the Attestation Result to remain valid. Thus, if the Attestation Result sends a validity lifetime in terms of "time(RX_v)-time(RG_v)", then the Relying Party can check "time(OP_r)-time(ER_r) < time(RX_v)-time(RG_v)".

17. References

17.1. Normative References

- [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>>.
- [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>>.

17.2. Informative References

- [CTAP] FIDO Alliance, "Client to Authenticator Protocol", n.d., <<https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-client-to-authenticator-protocol-v2.0-id-20180227.html>>.
- [I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", Work in Progress, Internet-Draft, draft-birkholz-rats-tuda-03, 13 July 2020, <<http://www.ietf.org/internet-drafts/draft-birkholz-rats-tuda-03.txt>>.
- [I-D.ietf-teep-architecture] Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", Work in Progress, Internet-Draft, draft-ietf-teep-architecture-12, 13 July 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-teep-architecture-12.txt>>.
- [OPCUA] OPC Foundation, "OPC Unified Architecture Specification, Part 2: Security Model, Release 1.03", OPC 10000-2 , 25 November 2015, <<https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/>>.
- [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>>.
- [RFC8322] Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange (ROLIE)", RFC 8322, DOI 10.17487/RFC8322, February 2018, <<https://www.rfc-editor.org/info/rfc8322>>.
- [TCGarch] Trusted Computing Group, "Trusted Platform Module Library - Part 1: Architecture", n.d., <https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p62_Part1_Architecture_7july2020.pdf>.
- [WebAuthN] W3C, "Web Authentication: An API for accessing Public Key Credentials", n.d., <<https://www.w3.org/TR/webauthn-1/>>.

Contributors

Monty Wiseman

Email: montywiseman32@gmail.com

Liang Xia

Email: frank.xialiang@huawei.com

Laurence Lundblade

Email: lgl@island-resort.com

Eliot Lear

Email: elear@cisco.com

Jessica Fitzgerald-McKay

Sarah C. Helbe

Andrew Guinn

Peter Lostcco

Email: pete.loscocco@gmail.com

Eric Voit

Thomas Fossati

Email: thomas.fossati@arm.com

Paul Rowe

Carsten Bormann

Email: cabo@tzi.org

Giri Mandyam

Email: mandyam@qti.qualcomm.com

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany

Email: henk.birkholz@sit.fraunhofer.de

Dave Thaler
Microsoft
United States of America

Email: dthaler@microsoft.com

Michael Richardson
Sandelman Software Works
Canada

Email: mcr+ietf@sandelman.ca

Ned Smith
Intel Corporation
United States of America

Email: ned.smith@intel.com

Wei Pan
Huawei Technologies

Email: william.panwei@huawei.com

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

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 March 4, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	4
1.1.	CDDL, CWT and JWT	4
1.2.	Entity Overview	5
1.3.	EAT Operating Models	5
1.4.	What is Not Standardized	6
1.4.1.	Transmission Protocol	6
1.4.2.	Signing Scheme	7
2.	Terminology	7
3.	The Claims	8
3.1.	Token ID Claim (cti and jti)	8
3.2.	Timestamp claim (iat)	9
3.3.	Nonce Claim (nonce)	9
3.3.1.	nonce CDDL	9
3.4.	Universal Entity ID Claim (ueid)	9
3.4.1.	ueid CDDL	12
3.5.	Origination Claim (origination)	12
3.5.1.	origination CDDL	12
3.6.	OEM Identification by IEEE (oemid)	12
3.6.1.	oemid CDDL	13
3.7.	The Security Level Claim (security-level)	13
3.7.1.	security-level CDDL	14
3.8.	Secure Boot and Debug Enable State Claims (boot-state)	14
3.8.1.	Secure Boot Enabled	14
3.8.2.	Debug Disabled	15
3.8.3.	Debug Disabled Since Boot	15
3.8.4.	Debug Permanent Disable	15
3.8.5.	Debug Full Permanent Disable	15
3.8.6.	boot-state CDDL	15
3.9.	The Location Claim (location)	15
3.9.1.	location CDDL	16
3.10.	The Age Claim (age)	16

3.10.1. age CDDL	16
3.11. The Uptime Claim (uptime)	16
3.11.1. uptime CDDL	16
3.12. The Submods Part of a Token (submods)	17
3.12.1. Two Types of Submodules	17
3.12.1.1. Non-token Submodules	17
3.12.1.2. Nested EATs	17
3.12.2. No Inheritance	18
3.12.3. Security Levels	18
3.12.4. Submodule Names	18
3.12.5. submods CDDL	18
4. Encoding	18
4.1. Common CDDL Types	19
4.2. CDDL for CWT-defined Claims	19
4.3. JSON	19
4.3.1. JSON Labels	19
4.3.2. JSON Interoperability	20
4.4. CBOR	20
4.4.1. CBOR Labels	20
4.4.2. CBOR Interoperability	21
4.5. Collected CDDL	22
5. IANA Considerations	23
5.1. Reuse of CBOR Web Token (CWT) Claims Registry	23
5.1.1. Claims Registered by This Document	23
6. Privacy Considerations	24
6.1. UEID Privacy Considerations	24
7. Security Considerations	25
7.1. Key Provisioning	25
7.1.1. Transmission of Key Material	25
7.2. Transport Security	25
7.3. Multiple EAT Consumers	26
8. References	26
8.1. Normative References	26
8.2. Informative References	28
Appendix A. Examples	30
A.1. Very Simple EAT	30
A.2. Example with Submodules, Nesting and Security Levels	30
Appendix B. UEID Design Rationale	30
B.1. Collision Probability	30
B.2. No Use of UUID	33
Appendix C. Changes from Previous Drafts	34
C.1. From draft-rats-eat-01	34
C.2. From draft-mandyam-rats-eat-00	34
C.3. From draft-ietf-rats-eat-01	34
C.4. From draft-ietf-rats-eat-02	34
Authors' Addresses	35

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:

- o Proof of the make and model of the device hardware (HW)
- o Proof of the make and model of the device processor, particularly for security-oriented chips
- o Measurement of the software (SW) running on the device
- o Configuration and state of the device
- o 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.

- o 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.
- o 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.
- o 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 facilitate 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 ECDAAs is used, (e.g., as used by Enhanced Privacy ID, i.e. EPID) then it is the key material needed for ECDAAs.

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.

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

CDDL along with text descriptions is used to define each claim independent 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:

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

- o 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 used 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 enable 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:

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

- o `bstr` - must be base64url encoded
- o `time` - must be encoded as `NumericDate` as described section 2 of [RFC7519].
- o `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.

- o 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.
- o 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.

- o Major type 2, `bstr`, SHOULD be have tag 21 to indicate conversion to `base64url` in case that conversion is performed.
- o 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.
- o 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.
- o URIs - URIs should be encoded as text strings and marked with tag 32.
- o Floating Point - The entity may use any floating-point encoding. The relying party must support decoding of all types of floating-point.
- o 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

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

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

- o 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.

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

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

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

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

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

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements notation	3
2. The YANG Module for Basic Remote Attestation Procedures . . .	3
2.1. Tree Diagram	3
2.2. YANG Modules	6
2.2.1. ietf-tpm-remote-attestation	6
2.2.2. ietf-tcg-algs	35
3. IANA considerations	51
4. Security Considerations	51
5. Acknowledgements	52
6. Change Log	52
7. References	53
7.1. Normative References	53
7.2. Informative References	54
Authors' Addresses	55

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 or 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}?
    +---rw 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
+---:(netequi_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 <netequi_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 attester.";
  }
}

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 tpml2-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 tpml2-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 tpml2-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 tpml2-pcr-info-short;
}

grouping tpml2-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 tpml2-pcr-selection;
    leaf value-size {
      type uint32;
      description
        "This SHALL be the size of the 'tpml2-pcr-value' field
        (not the number of PCRs).";
    }
    leaf-list tpml2-pcr-value {
      type binary;
      description
        "The list of TPM_PCRVALUES from each PCR selected in sequence
        of tpml2-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 tpml2-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 tpml2-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 tpml2-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-version).";
        case tpm12-quotel {
            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 across
        reboots.";
}
leaf compute-node {
    when "../.../compute-nodes";
    type compute-node-ref;
    config false;
    mandatory true;
    description
        "When there is more than 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

```



```
    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 collapsing/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_rlp32_pub Table 2";
}

identity anonymous_signing {
  description
    "A TCG recognized anonymous signing algorithm.";
  reference
    "TCG_Algorithm_Registry_rlp32_pub Table 2";
}

identity encryption_mode {
  description
    "A TCG recognized encryption mode.";
  reference
    "TCG_Algorithm_Registry_rlp32_pub Table 2";
}

identity method {
  description
    "A TCG recognized method such as a mask generation function.";
  reference
    "TCG_Algorithm_Registry_rlp32_pub Table 2";
}

identity object_type {
  description
    "A TCG recognized object type.";
  reference
    "TCG_Algorithm_Registry_rlp32_pub Table 2";
}

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

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_ECDA {
  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_rlp32_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_rlp32_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_rlp32_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_rlp32_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>`

- o `<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>`

- o `<tpm-name>` - Although shown as 'rw', it is system generated
- o `<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.
- o `<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:

- o moved to `tcg-algs`
- o cleaned up model to eliminate sources of errors
- o removed key establishment RPC
- o added lots of XPATH which must all be scrubbed still

- o Descriptive text added on model contents.

Changes from version 01 to version 02:

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

Changes from version 00 to version 01:

- o Addressed author's comments
- o Extended complementary details about attestation-certificates
- o Relabeled chunk-size to log-entry-quantity
- o Relabeled location with compute-node or tpm-name where appropriate
- o Added a valid entity-mib physical-index to compute-node and tpm-name to map it back to hardware inventory
- o Relabeled name to tpm_name
- o 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., <<http://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