RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2020

H. Birkholz
Fraunhofer SIT
E. Voit
Cisco
W. Pan
Huawei

June 24, 2020

Attestation Event Stream Subscription
draft-birkholz-rats-network-device-subscription-00

Abstract

This document defines how to subscribe to a stream of attestation related Evidence on TPM-based network devices.

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 December 26, 2020.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
2.1. Requirements Notation 3
3. Operational Model 3
3.1. Sequence Diagram 3
3.2. Continuously Verifying Freshness 5
3.2.1. TPM 1.2 Quote 5
3.2.2. TPM 2 Quote 5
4. Remote Attestation Event Stream 6
4.1. Subscription to the <attestation> Event Stream 7
4.2. Replaying a history of previous TPM extend operations . . 7
4.2.1. TPM2 Heartbeat 8
4.3. YANG notifications placed on the <attestation> Event Stream 8
4.3.1. tpm-extend 8
4.3.2. tpm12-attestation 9
4.3.3. tpm20-attestation 10
4.4. Filtering Evidence at the Attester 11
4.5. Replaying previous PCR Extend events 11
4.6. Configuring the <attestation> Event Stream 12
5. YANG Module 12
6. Security Considerations 18
7. IANA Considerations 19
8. References 19
8.1. Normative References 19
8.2. Informative References 19
Appendix A. Acknowledgements 20
A.1. Open Questions 20
Authors' Addresses 20

1. Introduction

[RATS-Device] and [RATS-YANG] define the operational prerequisites and a YANG Model for the acquisition of Evidence from a TPM-based network device. However, there is a limitation inherent in the challenge-response interaction models upon which these documents are based. This limitation is that it is up to the Verifier to request Evidence. The result is that the interval between the occurrence of a security event, and the event's visibility within the Relying Party can be unacceptably long.

This limitation results in two adverse effects:

1. Evidence is not streamed to an interested Verifier as soon as it is generated.

2. If it were to be streamed, the Evidence is not appraisable for freshness.

This specification addresses the first adverse effect by enabling a Verifier to subscribe via [RFC8639] to a YANG <attestation> Event Stream which exists upon the Attester. When subscribed, the Attester will continuously stream a subscribed set of Evidence to the Verifier.

The second adverse effect results from the nonce based challenge-response of [RATS-YANG]. In that document, an Attester must wait for a new nonce from a Verifier before it generates a new TPM Quote. In this case, the nonce acts as an implicit timestamp that a windows of freshness is tied to. To address delays resulting from such a synchronous wait for nonce based Evidence generation, this specification enables freshness to be asserted in an asynchronous manner.

By removing these two adverse effects, it becomes possible for a Verifier to continuously maintain an appraisal of the Attested device without relying on continuous polling.

2. Terminology

The following terms are imported from [I-D.ietf-rats-architecture]: Attester, Evidence, Relying Party, and Verifier. Also imported are the time definitions time(vg), time(ns), time(eg), time(rg), and time(ra) from that document's appendices. The following terms at imported from [RFC8639]: Event Stream, Subscription, Event Stream Filter, Dynamic Subscription.

2.1. 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.

3. Operational Model

3.1. Sequence Diagram

Figure 1 below is a sequence diagram which updates Figure 5 of [RATS-Device]. This sequence diagram replaces the [RATS-Device] challenge-response interaction model with an [RFC8639] Dynamic Subscription to an <attestation> Event Stream. The contents of the <attestation> Event Stream are defined below within Section 4.

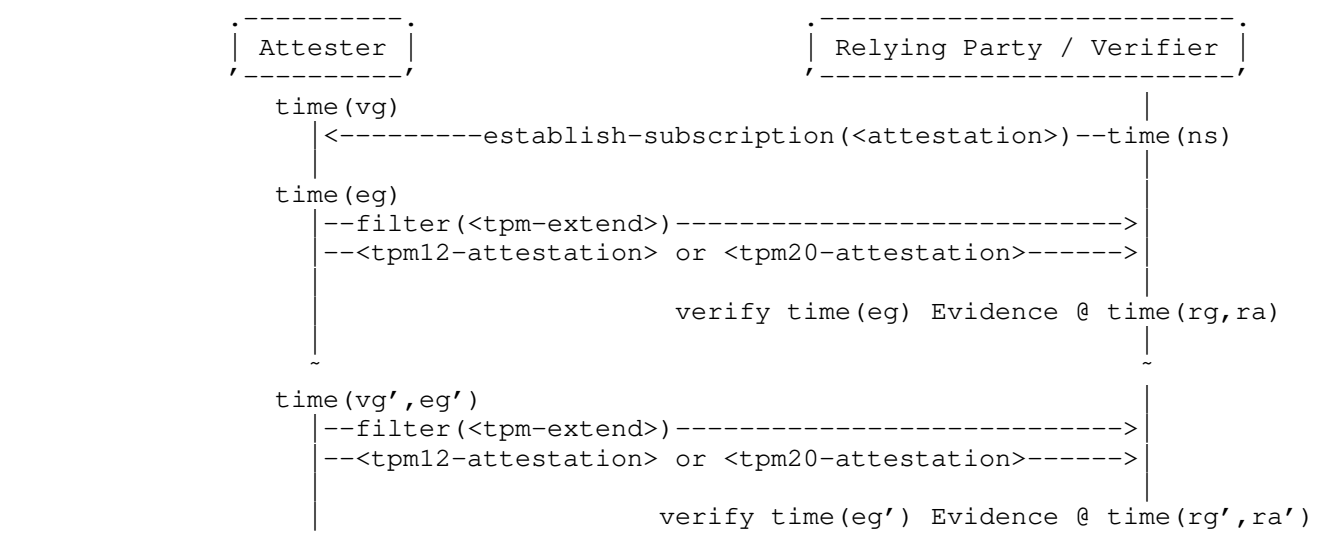


Figure 1: YANG Subscription Model for Remote Attestation

- time(vg,rg,ra) are identical to the corresponding times from Figure 5 of [RATS-Device].

- time(rg',ra') are subsequent instances of the corresponding times from Figure 5 of [RATS-Device].

- time(ns): The Verifier generates a nonce and makes an [RFC8639] <establish-subscription> request. This request also includes the augmentations defined in this document's YANG model. Key subscription RPC parameters include:

  * the nonce,

  * a set of PCRs of interest which the Verifier wants to appraise, and

  * an optional filter which can reduce the logged events on the <attestation> stream pushed to the Verifier.

- time(eg) - An initial response of Evidence is returned to the Verifier. This includes:

  * a replay of filtered log entries which have extended into a PCR of interest since boot are sent in the <tpm-extend> notification, and a

* a signed TPM quote that contains at least the PCRs from the <establish-subscription> RPC are included in a <tpm12-attestation> or <tpm20-attestation>). This quote must have included the nonce provided at time(ns).

o time(vg',eg') - This occurs when a PCR is extended subsequent to time(eg). Immediately after the extension, the following information needs to be pushed to the Verifier:

* Any values extended into a PCR of interest, and

* a signed TPM Quote showing the result the PCR extension.

3.2. Continuously Verifying Freshness

As there is no new Verifier nonce provided at time(eg'), it is important to validate the freshness of TPM Quotes which are delivered at that time. The method of doing this verification will vary based on the capabilities of the TPM cryptoprocessor used.

3.2.1. TPM 1.2 Quote

The [RFC8639] notification format includes the <eventTime> object. This can be used to determine the amount of time subsequent to the initial subscription each notification was sent. However, this time is not part of the signed results which are returned from the Quote, and therefore is not trustworthy as objects returned in the Quote. Therefore, a Verifier MUST periodically issue a new nonce, and receive this nonce within a TPM quote response in order to ensure the freshness of the results. This can be done using the <tpm12-challenge-response-attestation> RPC from [RATS-YANG].

3.2.2. TPM 2 Quote

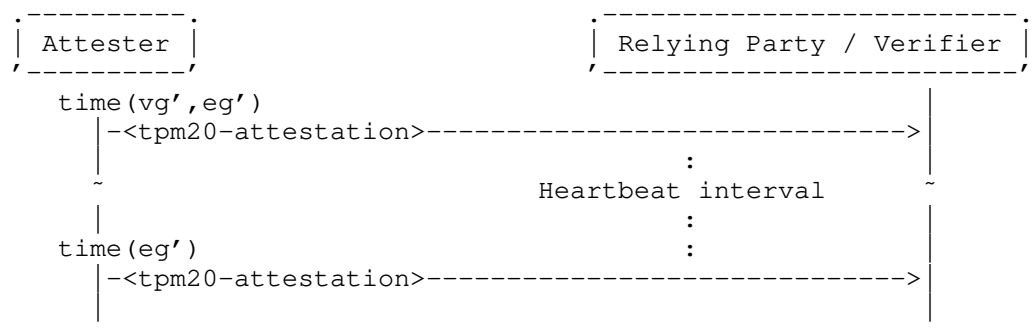
When the Attester includes a TPM2 compliant cryptoprocessor, internal time-related counters are included within the signed TPM Quote. By including an initial nonce in the [RFC8639] subscription request, fresh values for these counters are pushed as part of the first TPM Quote returned to the Verifier. Then, as shown by [I-D.birkholz-rats-tuda], subsequent TPM Quotes delivered to the Verifier can the be appraised for freshness based on the predictable incrementing of these time-related counters.

The relevant internal time-related counters defined within [TPM2.0] can be seen within <tpms-clock-info>. These counters include the <clock>, <reset-counter>, and <restart-counter> objects. Normative rules for appraising these objects are as follows:

- If the <clock> has incremented for no more than the same duration as both the <eventTime> and the Verifier's internal time since the initial time(eg) and any previous time(eg'), then the TPM Quote MAY be considered fresh. Note that [TPM2.0] allows for +/- 15% clock drift. However, many chips significantly improve on this maximum drift. If available, chip specific maximum drifts SHOULD be considered during the appraisal process.

- If the <reset-counter>, <restart-counter> has incremented. The existing subscription MUST be terminated, and a new <establish-subscription> SHOULD be generated.

- If a TPM Quote on any subscribed PCR has not been pushed to the Verifier for a duration of an Attester defined heartbeat interval, then a new TPM Quote notification SHOULD be sent to the Verifier. This may often be the case, as certain PCRs might be infrequently updated.



4. Remote Attestation Event Stream

The <remote-attestation> Event Stream is an [RFC8639] complaint Event Stream which is defined within this section and within the YANG Module of [RATS-YANG]. This Event Stream contains YANG notifications which carry Evidence assisting a Verifier in the appraisal of an Attester. Data Nodes within Section 4.6 allow the configuration of this Event Stream's contents on an Attester.

This <remote-attestation> Event Stream may only be exposed on Attesters supporting [RATS-Device]. As with [RATS-Device], it is up to the Verifier to understand which types of cryptoprocessors and keys are acceptable.

4.1. Subscription to the <attestation> Event Stream

To establish a subscription to an Attester in a way which provides provably fresh Evidence, initial randomness must be provided to the Attester. This is done via the augmentation of a <nonce-value> into [RFC8639] the <establish-subscription> RPC. Additionally, a Verifier must ask for PCRs of interest from a platform.

```
augment /sn:establish-subscription/sn:input:
  +---w nonce-value    binary
  +---w pcr-index*     tpm:pcr
```

The result of the subscription will be that passing of the following information:

1. <tpm12-attestation> and <tpm20-attestation> notifications which include the provided <nonce-value>. These attestation notifications MUST at least include all the <pcr-indicies> requested in the RPC.

2. a series of <tpm-extend> notifications which reference the requested PCRs on all TPM based cryptoprocessors on the Attester.

3. <tpm12-attestation> and <tpm20-attestation> notifications generated within a few seconds of the <tpm-extend> notifications. These attestation notifications MUST at least include any PCRs extended.

If the Verifier does not want to see the logged extend operations for all PCRs available from an Attester, an Event Stream Filter should be applied. This filter will remove Evidence from any PCRs which are not interesting to the Verifier.

4.2. Replaying a history of previous TPM extend operations

Unless it is relying on Known Good Values, a Verifier will need to acquire a history of PCR extensions since the Attester has been booted. This history may be requested from the Attester as part of the <establish-subscription> RPC. This request is accomplished by placing a very old <replay-start-time> within the original RPC request. As the very old <replay-start-time> will pre-date the time of Attester boot, a <replay-start-time-revision> will be returned in the <establish-subscription> RPC response, indicating when the Attester booted. Immediately following the response (and before the notifications above) one or more <tpm-extend> notifications which document all extend operations which have occurred for the requested PCRs since boot will be sent. Many extend operations to a single PCR

index on a single TPM SHOULD be included within a single notification.

Note that if a Verifier has a partial history of extensions, the <replay-start-time> can be adjusted so that known extensions are not forwarded.

The end of this history replay will be indicated with the [RFC8639] <replay-completed> notification. For more on this sequence, see Section 2.4.2.1 of [RFC8639].

After the <replay-complete> notification is provided, a TPM Quote will be requested and the result passed to the Verifier via a <tpm12-attestation> and <tpm20-attestation> notification. If there have been any additional extend operations which have changed a subscribed PCR value in this quote, these MUST be pushed to the Verifier before the <tpm12-attestation> and <tpm20-attestation> notification.

At this point, the Verifier has sufficient Evidence to appraise the reported extend operations for each PCR, as well as compare the expected value of the PCR value against that signed by the TPM.

4.2.1. TPM2 Heartbeat

For TPM2, every requested PCR MUST at least be sent once within a <tpm20-attestation> heartbeat interval. This MAY be done with a single <tpm20-attestation> notification that includes all requested PCRs every heartbeat interval. Alternatively, this MAY be done with several <tpm20-attestation> notifications at different times during that heartbeat interval.

4.3. YANG notifications placed on the <attestation> Event Stream

4.3.1. tpm-extend

This notification documents when a single subscribed PCR is extended within a single TPM cryptoprocessor. Corresponding notifications SHOULD be emitted no less than a <marshalling-period> after the PCR is first extended (the reason for the marshalling is that it is quite possible that multiple extensions to the same PCR have been made in quick succession). A notification MUST be emitted prior to a <tpm12-attestation> or <tpm20-attestation> notification which has included and signed the results of any specific PCR extension.

```
      +---n tpm-extend
         +--ro certificate-name?    string
         +--ro pcr-index-changed    tpm:pcr
         +--ro attested-event* []
            +--ro attested-event
               +--ro extended-with    binary
               +--ro event-type?      identityref
               +--ro event-details?   <anydata>
```

Each <tpm-extend> MUST include one or more values being extended into the PCR. These are conveyed within the <extended-with> object. For each extension, details of the event MAY be provided within the <event-details> object.

The format of any included <event-details> is identified by the <event-type>. This document includes two YANG structures which may be inserted into the <event-details>. These two structures are: <ima-event-log> and <bios-event-log>. Implementations wanting to provide additional documentation of a type of PCR extension may choose to define additional YANG structures which can be placed into <event-details>.

4.3.2. tpm12-attestation

This notification type contains an instance of a TPM1.2 style signed cryptoprocessor measurement. This notification is generated at two points in time:

1. Upon initial subscription

2. Every time at least one subscribed PCR has changed from the directly previous <tpm12-attestation>. In this case, the notification SHOULD be emitted within a <marshalling-period> since a the first subscribed PCR changed.

This notification MUST NOT include the returned quote digest the results from any PCR extensions not previously reportable by a <tpm-extend>.

```
      +---n tpm12-attestation {tpm:TPM12}?
         +--ro certificate-name?            string
         +--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 revMajor?   uint8
            |  |  +--ro revMinor?   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
```

All YANG objects above are defined within [RATS-YANG]. The objects MAY include Attester information such as <tpm12-pcr-value> which are not signed. The <tpm12-attestation> is not replayable.

4.3.3. tpm20-attestation

This notification contains an instance of TPM2 style signed cryptoprocessor measurements. This notification is generated at three points in time:

1. Upon initial subscription

2. Every time at least one subscribed PCR has changed from the directly previous <tpm20-attestation>. In this case, the notification SHOULD be emitted within a <marshalling-period> since a the first subscribed PCR changed.

3. After a minimum heartbeat interval since a previous <tpm20-attestation> was sent.

This notification MUST NOT include the returned <quote> the results from any PCR extensions not previously reportable by a <tpm-extend>.

```
    +---n tpm20-attestation {tpm:TPM20}?
       +--ro certificate-name?           string
       +--ro up-time?                    uint32
       +--ro node-id?                    string
       +--ro node-physical-index?        int32 {ietfhw:entity-mib}?
       +--ro quote?                      binary
       +--ro quote-signature?            binary
       +--ro pcr-bank-values* []
       |  +--ro TPM2_Algo?    identityref
       |  +--ro pcr-values* [pcr-index]
       |     +--ro pcr-index    pcr
       |     +--ro pcr-value?   binary
       +--ro pcr-digest-algo-in-quote
          +--ro TPM2_Algo?   identityref
```

All YANG objects above are defined within [RATS-YANG]. The objects MAY include Attester information such as <pcr-bank-values> which are not signed. The <tpm20-attestation> is not replayable.

4.4. Filtering Evidence at the Attester

It can be useful NOT to receive all Evidence related to a PCR. An example of this is would be a when a Verifier maintains Known Good Values of a PCR. In this case, it is not necessary to send each extend operation.

To accomplish this reduction, when an RFC8639 <establish-subscription> RPC is sent, a <stream-filter> as per RFC8639, Section 2.2 can be set to discard a <tpm-extend> notification when the <pcr-index-changed> is uninteresting to the verifier.

4.5. Replaying previous PCR Extend events

To verify the value of a PCR, a Verifier must either know that the value is a Known Good Value [KGV] or be able to reconstruct the hash value by viewing all the PCR-Extends since the Attester rebooted. Wherever a hash reconstruction might be needed, the <remote-attestation> Event Stream MUST support the RFC8639 <replay> feature. Through the <replay> feature, it is possible for a Verifier to retrieve and sequentially hash all of the PCR extending events since an Attester booted. And thus, the Verifier has access to all the evidence needed to verify a PCR’s current value.

4.6. Configuring the <attestation> Event Stream

Figure 2 is tree diagram which exposes the operator configurable elements of the <remote-attestation> Event Stream. This allows an Attester to select what information should be available on the stream. A fetch operation also allows an external device such as a Verifier to understand the current configuration of stream.

Almost all YANG objects below are defined via reference from [RATS-YANG]. There is one object which is new with this model however. <tpm2-heartbeat> defines the maximum amount of time which should pass before a subscriber to the Event Stream should get a <tpm20-attestation> notification from devices which contain a TPM2.

```
  +--rw rats-support-structures
    +--rw rats-support-structures
     +--rw supported-algos*                          identityref
     +--rw tpms* [tpm-name]
     |  +--rw tpm-name                       string
     |  +--rw tras:leafref-to-keystore?      string
     |  +--rw (tras:subscribable)?
     |     +--:(tras:tpm12-stream) {tpm:TPM12}?
     |     |  +--rw tras:tpm12-pcr-index*   tpm:pcr
     |     +--:(tras:tpm20-stream) {tpm:TPM20}?
     |        +--rw tras:tpm20-pcr-index*   tpm:pcr
     +--rw tras:marshalling-period?                  uint8
     +--rw tras:tpm12-subscribed-signature-scheme?
     |       -> ../tpm:supported-algos {tpm:TPM12}?
     +--rw tras:tpm20-subscribed-signature-scheme?
     |       -> ../tpm:supported-algos {tpm:TPM20}?
     +--rw tras:tpm20-subscription-heartbeat?        uint16
             {tpm:TPM20}?
```

Figure 2: Configuring the Attestation Stream

5. YANG Module

This YANG module imports modules from [RATS-YANG] and [RFC8639]. It is also work-in-progress.

```
<CODE BEGINS> ietf-tpm-remote-attestation-stream@2020-06-10.yang
module ietf-tpm-remote-attestation-stream {
  yang-version 1.1;
  namespace
     "urn:ietf:params:xml:ns:yang:ietf-tpm-remote-attestation-stream";
  prefix tras;
```

```
  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }
  import ietf-tpm-remote-attestation {
    prefix tpm;
    reference
      "draft-ietf-rats-yang-tpm-charra";
  }
  import ietf-yang-structure-ext {
    prefix sx;
    reference
      "draft-ietf-netmod-yang-data-ext";
  }

  organization "IETF";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rats/>
     WG List:  <mailto:rats@ietf.org>

     Editor:   Eric Voit
               <mailto:evoit@cisco.com>";

  description
    "This module contains conceptual YANG specifications for
     subscribing to attestation streams being generated from TPM chips.

     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 2020-06-10 {
    description
      "Initial version.";
    reference
      "draft-birkholz-rats-network-device-subscription";
  }

  /*
   * IDENTITIES
   */

  identity pcr-unsubscribable {
    base sn:establish-subscription-error;
    description
      "Requested PCR is subscribable by the Attester.";
  }


  /*
   * Groupings
   */

  grouping heartbeat {
    description
      "Allows an Attester to push verifiable, current TPM PCR values even
      when there have been no recent changes to PCRs.";
    leaf tpm20-subscription-heartbeat {
      type uint16;
      description
        "Number of seconds before the Attestation stream should send a new
        notification with a fresh quote.  This allows confirmation
        that the PCR values haven't changed since the last
        tpm20-attestation.";
    }
  }


  /*
   * RPCs
   */
```

```
  augment "/sn:establish-subscription/sn:input" {
    when 'derived-from-or-self(sn:stream, "attestation")';
    description
      "This augmentation adds a nonce to as a subscription parameters
       that apply specifically to datastore updates to RPC input.";
    uses tpm:nonce;
    leaf-list pcr-index {
      type tpm:pcr;
      min-elements 1;
      description
        "The numbers/indexes of the PCRs. This will act as a filter for the
        subscription so that 'tpm-extend' notifications related to
        non-requested PCRs will not be sent to a subscriber.";
    }
  }

  /*
   * NOTIFICATIONS
   */

  notification tpm-extend {
    description
      "This notification indicates that a PCR has extended within a TPM based
      cryptoprocessor.  In less that 10 seconds, it should be followed with
      either a tpm12-attestation or tpm20-attestation notification.";
    uses tpm:certificate-name;
    leaf pcr-index-changed {
      type tpm:pcr;
      mandatory true;
      description
        "The number of the PCR extended.";
    }
    list attested-event {
      description
        "A set of events which extended an Attester PCR.  The sequence of
        elements represented in list must match the sequence of events
        placed into the TPM.";
      container attested-event {
        description
          "An instance of an event which extended an Attester PCR";
        leaf extended-with {
          type binary;
          mandatory true;
          description
            "Information extending the PCR.";
        }
        leaf event-type {
          type identityref {
```

```
            base tpm:attested-event-log-type;
          }
          description
            "Indicates what kind of event happened the Attester thought was
            worthy of recording in a PCR.";
        }
        anydata event-details {
          description
            "Any structure reference 'event-type' contains supporting
            information which allows an Attester to evaluate the trust
            implications.

            Event details may be populated with YANG log structures defined
            at the bottom of this module.";
        }
      }
    }
  }

  notification tpm12-attestation {
    if-feature "tpm:TPM12";
    description
      "Contains an instance of TPM1.2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester information.";
    uses tpm:tpm12-attestation;
  }

  notification tpm20-attestation {
    if-feature "tpm:TPM20";
    description
      "Contains an instance of TPM2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester information.";
    uses tpm:tpm20-attestation;
  }


  /*
   * DATA NODES
   */

  augment "/tpm:rats-support-structures" {
    description
      "Defines platform wide 'attestation' stream subscription parameters.";
    leaf marshalling-period {
      config true;
      type uint8;
      default 5;
      description
```

```
        "The maximum number of seconds between the time an event extends a PCR,
        and the 'tpm-extend' notification which reports it to a subscribed
        Verifier.  This period allows multiple extend operations bundled
        together and handled as a group.";
    }
    leaf tpm12-subscribed-signature-scheme {
      if-feature "tpm:TPM12";
      type leafref {
        path "../tpm:supported-algos";
          /* a specific algorithm, need to check syntax */
      }
      description
        "A single signature-scheme which will be used to sign the evidence
        from a TPM 1.2. which is then placed onto the 'attestation' event
        stream.";
    }
    leaf tpm20-subscribed-signature-scheme {
      if-feature "tpm:TPM20";
      type leafref {
        path "../tpm:supported-algos";
          /* a specific algorithm, need to check syntax */
      }
      description
        "A single signature-scheme which will be used to sign the evidence
        from a TPM 2.0. which is then placed onto the 'attestation' event
        stream.";
    }
    uses heartbeat{
      if-feature "tpm:TPM20";
    }
  }

  augment "/tpm:rats-support-structures/tpm:tpms" {
    description
      "Allows the configuration 'attestation' stream parameters for a TPM.";
    leaf leafref-to-keystore {
      config true;
      type string;
      description
        "needs to be replaced with Reference to keystore draft.";
    }
    choice subscribable {
      config true;
      description
        "Indicates that the set of notifications which comprise the attestation
        stream can be modified or tuned by a network administrator.";
      case tpm12-stream {
        if-feature "tpm:TPM12";
```

```
        description
          "Configuration elements for a TPM1.2 event stream.";
        leaf-list tpm12-pcr-index {
          type tpm:pcr;
          description
            "The numbers/indexes of the PCRs which can be subscribed.";
        }
      }
      case tpm20-stream {
        if-feature "tpm:TPM20";
        description
          "Configuration elements for a TPM2.0 event stream.";
        leaf-list tpm20-pcr-index {
          type tpm:pcr;
          description
            "The numbers/indexes of the PCRs which can be subscribed.";
        }
        /* We need to decide if more than one hash-algo is subscribable */
      }
    }
  }


  /*
   * STRUCTURES - these contain the schema of reportable event types
   */

   sx:structure bios-event-log {
     when 'derived-from(../event-type, "bios-event-log")';
     description
       "BIOS/UEFI event log format";
     uses tpm:bios-event-log;
   }

   sx:structure ima-event-log {
     when 'derived-from(../event-type, "ima-event-log")';
     description
       "IMA event log format";
     uses tpm:ima-event-log;
   }
}
<CODE ENDS>
```

6. Security Considerations

To be written.

7. IANA Considerations

To be written.

8. References

8.1. Normative References

[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.

[RATS-Device]
"Network Device Remote Integrity Verification", n.d., <https://tools.ietf.org/html/draft-ietf-rats-tpm-based-network-device-attest-00>.

[RATS-YANG]
"A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs", n.d., <https://datatracker.ietf.org/doc/draft-ietf-rats-yang-tpm-charra/>.

[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>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

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

8.2. Informative References

[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.

[KGV] TCG, "KGV", October 2003, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-NetEq-Attestation-Workflow-Outline_v1r9b_pubrev.pdf>.

Appendix A. Acknowledgements

Tim Jenkins, Paul Merlo, Wayne Mills

A.1. Open Questions

- Do we need a notification correlator object to easily allow correlation on which extensions have been embodied within a specific attestation?

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Eric Voit
Cisco Systems, Inc.

Email: evoit@cisco.com

Wei Pan
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: william.panwei@huawei.com

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

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

July 08, 2020

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2021.

Table of Contents

1. Introduction 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 documents captures interaction models that can be used in specific RATS-related solution documents. The primary focus of this document is the conveyance of attestation Evidence. Specific goals of this document are to:

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

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

2. Terminology

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

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

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

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

3. Disambiguation

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

4. Scope and Intent

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

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

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

5. Direct Anonymous Attestation

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

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

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

5.1. Endorsers

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

5.2. Endorsers for Direct Anonymous Attestation

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

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

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

6. Normative Prerequisites

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

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

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

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

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

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

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

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

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

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

7. Generic Information Elements

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

Attester Identity ('attesterIdentity'): _mandatory_

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

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

Authentication Secret IDs ('authSecID'): _mandatory_

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

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

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

Handle ('handle'): _mandatory_

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

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

Claims ('claims'): _mandatory_

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

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

Reference Claims ('refClaims') _mandatory_

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

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

Claim Selection ('claimSelection'): _optional_

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

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

Evidence ('signedAttestationEvidence'): _mandatory_

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

Attestation Result ('attestationResult'): _mandatory_

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

8. Interaction Models

The following subsections introduce and illustrate the interaction models:

1. Challenge/Response Remote Attestation

2. Uni-Directional Remote Attestation

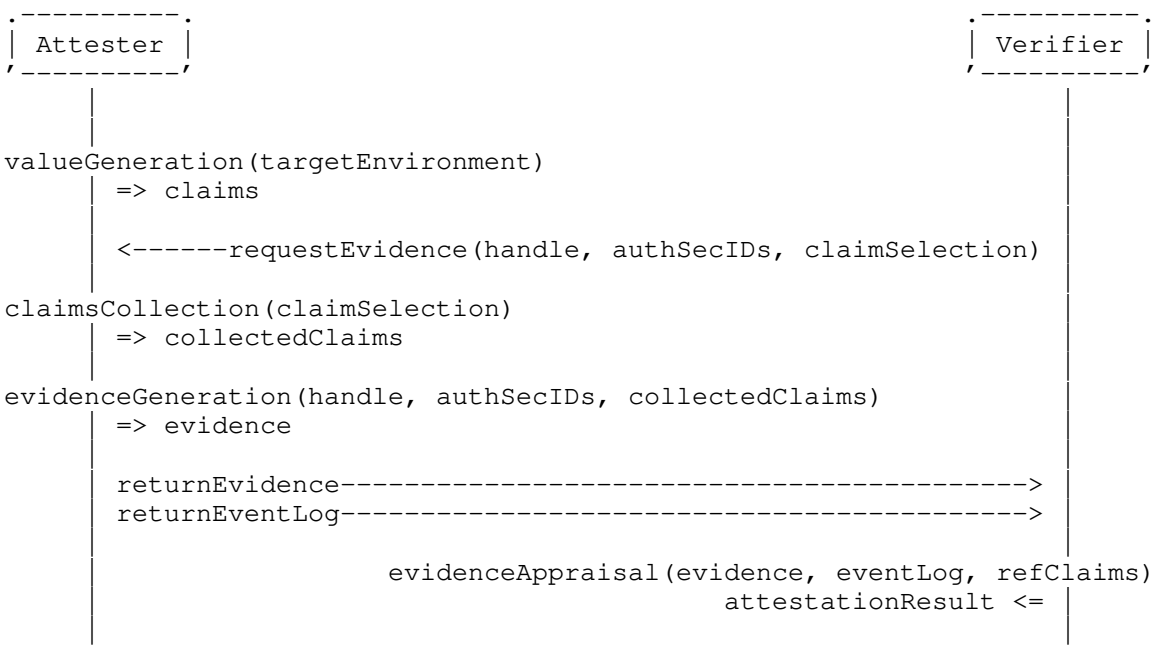
3. Streaming Remote Attestation

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

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

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

8.1. Challenge/Response Remote Attestation



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

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

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

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

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

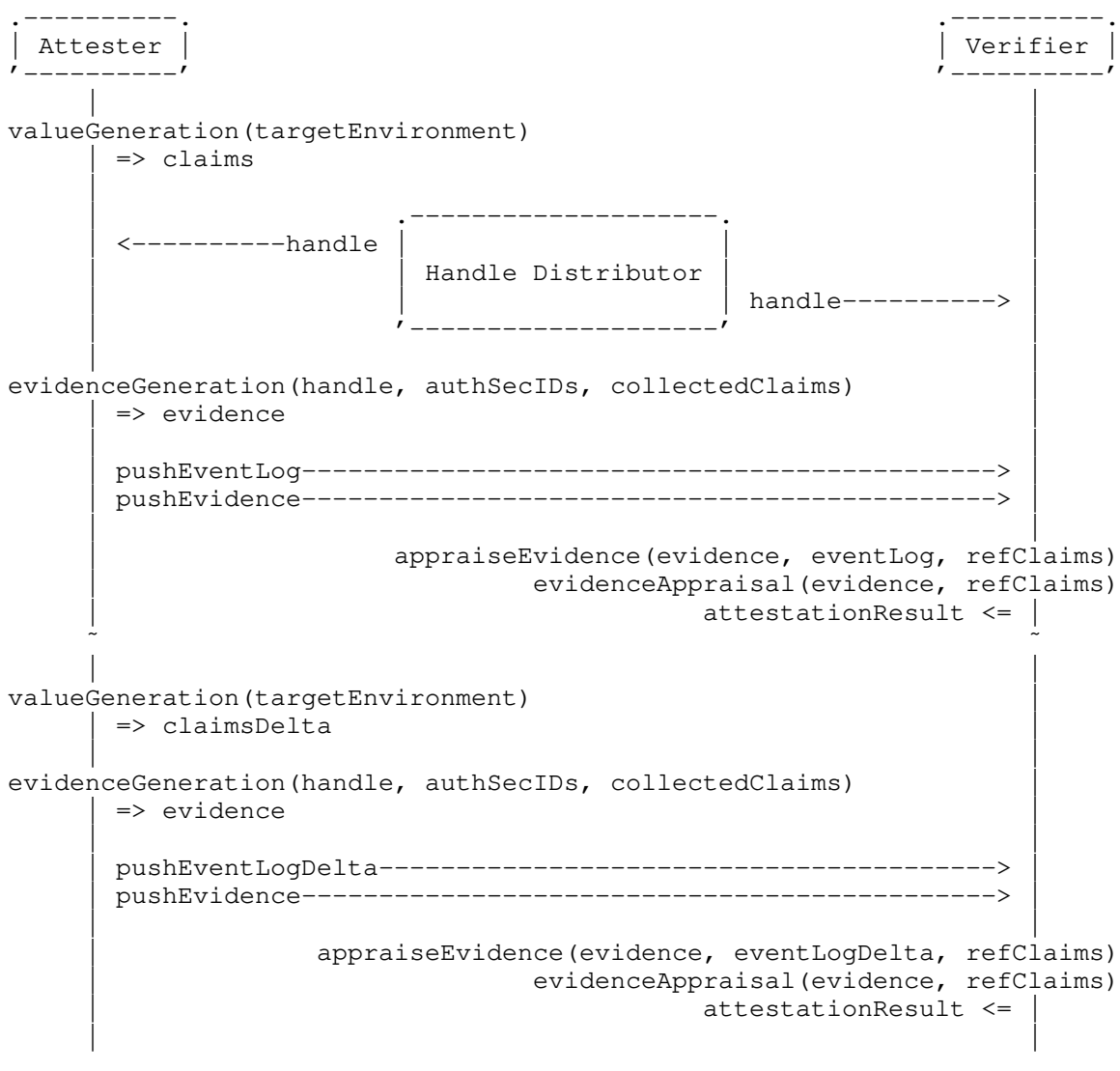
The Claim Selection narrows down the set of Claims collected and used to create Evidence to those that the Verifier requires. If the Claim Selection is omitted, then by default all Claims that are known and available on the Attester MUST be used to create corresponding Evidence. For example when performing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

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

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

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

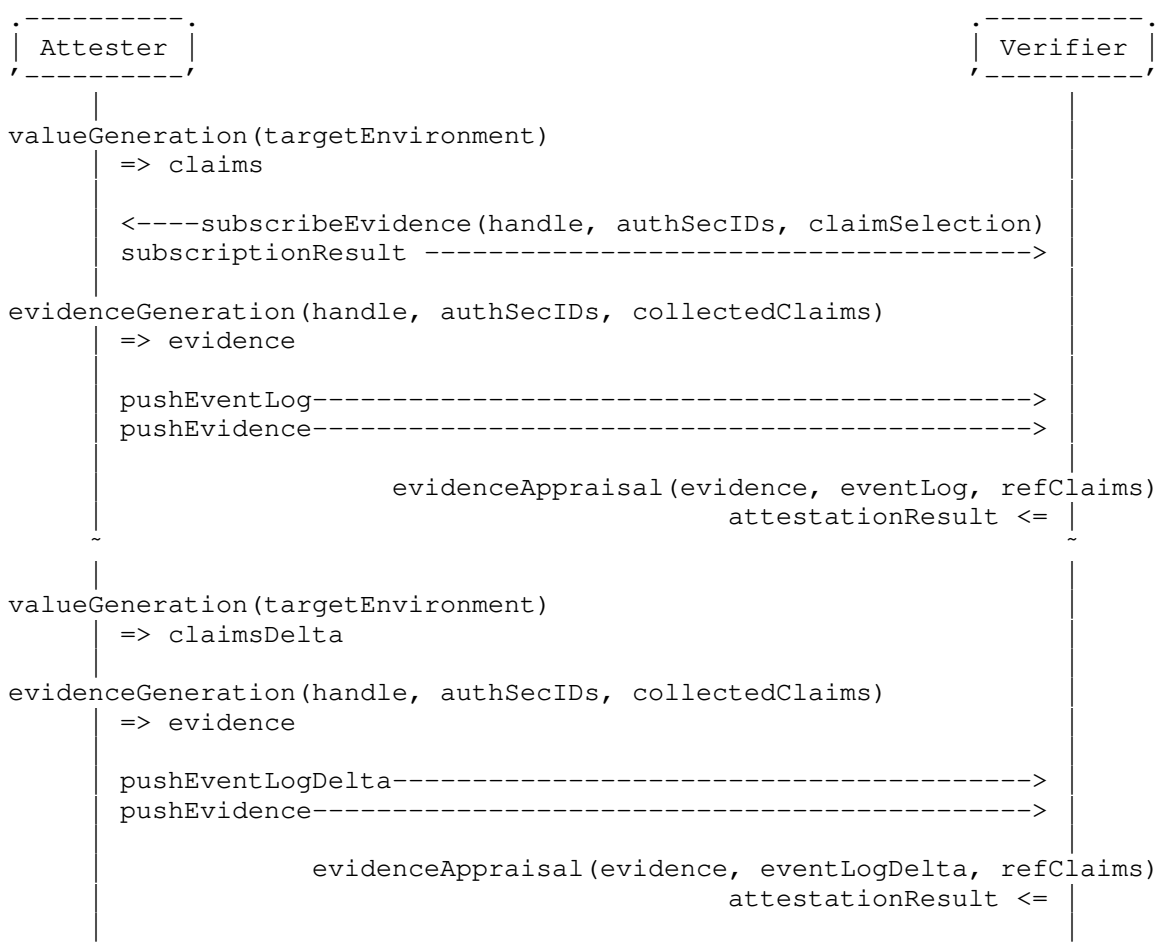
8.2. Uni-Directional Remote Attestation



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

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

8.3. Streaming Remote Attestation



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

9. Additional Application-Specific Requirements

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

9.1. Confidentiality

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

9.2. Mutual Authentication

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

9.3. Hardware-Enforcement/Support

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

10. Implementation Status

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

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

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

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

10.1. Implementer

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

10.2. Implementation Name

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

10.3. Implementation URL

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

10.4. Maturity

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

10.5. Coverage and Version Compatibility

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

10.6. License

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

10.7. Implementation Dependencies

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

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

10.8. Contact

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

11. Security and Privacy Considerations

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

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

12. Acknowledgments

Olaf Bergmann, Michael Richardson, and Ned Smith

13. Change Log

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

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

    * Added Implementation Status section

    * Updated interaction model

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

  o Changes from version 02 to version 00 RATS related document

    * update of the challenge/response diagram

    * minor rephrasing of Prerequisites section

    * rephrasing to information elements and interaction model section

  o Changes from version 00 to version 01

    * added Attestation Authenticity, updated Identity and Secret

    * relabeled Secret ID to Authentication Secret ID + rephrasing

    * relabeled Claim Selection to Assertion Selection + rephrasing

    * relabeled Evidence to (Signed) Attestation Evidence

    * Added Attestation Result and Reference Assertions

    * update of the challenge/response diagram and expositional text

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

  o Changes from version 01 to version 02

    * prepared the inclusion of additional reference models

    * update to Introduction and Scope section

    * major update to (Normative) Prerequisites

    * relabeled Attestation Authenticity to Att. Evidence Authenticity

    * relabeled Assertion term back to Claim terms

    * added BCP205 Implementation Status section related to Appendix CDDL

  o Changes from version 02 to version 03

    * major refactoring to now accommodate three interaction models

    * updated existing and added two new diagrams for models

    * major refactoring of existing and adding of new diagram description

    * incorporated content about Direct Anonymous Attestation

    * integrated comments from Michael Richardson

    * updated roster

14. References

14.1. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

14.2. Informative References

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

Christopher Newton
University of Surrey

Email: cn0016@surrey.ac.uk

Liqun Chen
University of Surrey

Email: liqun.chen@surrey.ac.uk

RATS
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

H. Birkholz
Fraunhofer SIT
B. Moran
Arm Limited

July 13, 2020

Trustworthiness Vectors for the Software Updates of Internet of Things (SUIT) Workflow Model
draft-birkholz-rats-suit-claims-00

Abstract

The IETF Remote Attestation Procedures (RATS) architecture defines Conceptual Messages as input and output of the appraisal process that assesses the trustworthiness of remote peers: Evidence and Attestation Results. Based on the Trustworthiness Vectors defined in Trusted Path Routing, this document defines a core set of Claims to be used in Evidence and Attestation Results for the Software Update for the Internet of Things (SUIT) Workflow Model. Consecutively, this document is in support of the Trusted Execution Environment Provisioning (TEEP) architecture, which defines the assessment of remote peers via RATS and uses SUIT for evidence generation as well as a remediation measure to improve trustworthiness of given remote peers.

Table of Contents

1. Introduction 2
1.1. SUIT Workflow Model and Procedures 3
1.2. Terminology 4
2. Trustworthiness Vectors 4
3. SUIT Claims 5
3.1. System Properties Claims 5
3.1.1. vendor-identifier 6
3.1.2. class-identifier 6
3.1.3. device-identifier 6
3.1.4. component-identifier 6
3.1.5. image-digest 6
3.1.6. image-size 6
3.1.7. minimum-battery 7
3.1.8. version 7
3.2. Interpreter Record Claims 7
3.2.1. record-success 7
3.2.2. component-index 7
3.2.3. dependency-index 7
3.2.4. command-index 7
3.2.5. nominal-parameters 8
3.2.6. nominal-parameters 8
3.3. Generic Record Conditions (TBD) 8
4. List of Commands (TBD) 8
5. References 9
5.1. Normative References 9
5.2. Informative References 10
Authors' Addresses 10

1. Introduction

Attestation Results are an essential output of Verifiers as defined in the Remote ATtestation procedureS (RATS) architecture [I-D.ietf-rats-architecture]. They are consumed by Relying Parties: the entities that intend to build future decisions on trustworthiness assessments of remote peers. Attestation Results must be easily

digestable by Relying Parties - in contrast to the rather complex or domain-specific Evidence digested by Verifiers.

In order to create Attestation Results, a Verifier must consume Evidence generated by a given Attester (amongst other Conceptual Messages, such as Endorsements and Attestation Policies). Both Evidence and Attestation Results are composed of Claims. This document highlights and defines a set of Claims to be used in Evidence and Attestation Results that are based on the SUIT Workflow Model [I-D.ietf-suit-manifest]. In the scope of this document, an Attester takes on the role of a SUIT Recipient: the system that receives a SUIT Manifest.

1.1. SUIT Workflow Model and Procedures

This document focuses on Evidence and Attestation Results that can be generated based on the output of SUIT Procedures. The SUIT Workflow Model allows for two types of SUIT Procedures generating Reports on the Attester as defined in the SUIT Manifest specification:

Update Procedures: A procedure that updates a device by fetching dependencies, software images, and installing them

An Update Procedure creates a Report on mutable software components that are installed or updated on hardware components.

Boot Procedures: A procedure that boots a device by checking dependencies and images, loading images, and invoking one or more image

A Boot Procedure creates a Report on measured boot events (e.g. during Secure Boot).

The Records contained in each type of Report can be used as Claims in Evidence generation on the Attester and in Attestation as described in this document. Analogously, a corresponding Verifier appraising that Evidence can create Attestation Results using the Claims defined in this document.

Both types of SUIT Procedures pass several stages (e.g. dependency-checking is one stage). The type and sequence of stages are defined by the Command Sequences included in a SUIT Manifest. For each stage in which a Command from the Command Sequence is executed a Record is created. All Records of a SUIT procedure contain binary results limited to "fail" or "pass". The aggregated sequence of all Records is composed into a Report.

This document specifies new Claims derived from Command Sequence Reports and highlights existing Claims as defined in Trusted Path Routing [I-D.voit-rats-trusted-path-routing] that are applicable to the operational state of installed and updated software.

The Claims defined in this document are in support of the Trusted Execution Environment Provisioning (TEEP) architecture. During TEEP, the current operational state of an Attester is assessed via RATS. If the corresponding Attestation Results - as covered in this document - indicate insufficient Trustworthiness Levels with respect to installed software, the SUIT Workflow Model is used for remediation.

1.2. Terminology

This document uses the terms and concepts defined in [I-D.ietf-rats-architecture], [I-D.ietf-suit-manifest], and [I-D.ietf-teep-architecture].

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. Trustworthiness Vectors

While there are usage scenarios where Attestation Results can be binary decisions, more often than not the assessment of trustworthiness is represented by a more fine-grained spectrum or based on multiple factors. These shades of Attestation Results are captured by the definition of Trustworthiness Vectors in Trusted Path Routing [I-D.voit-rats-trusted-path-routing]. Trustworthiness Vectors are sets of Claims representing appraisal outputs created by a Verifier. Each of these Claims is called a Trustworthiness Level. Multiple Trustworthiness Levels are composed into a vector.

An Attester processing SUIT Manifests can create three types of Claims about its Target Environments. This includes Claims about:

- installed manifests including initial state (e.g. factory default),
- hardware component identifiers that represent the targets of updates, and
- SUIT Interpreter results (e.g. test-failed) created during updates.

Every SUIT Manifest maps to a certain intended state of a device. Every intended device composition of software components associated with hardware components can therefore be expressed based on a SUIT Manifest. The current operational state of a device can be represented in the same form, including the initial state.

As a result, the Claims defined in this document are bundled by the scope of the information represented in SUIT Manifests, i.e., dedicated blobs of software that are the payload of a SUIT Manifest. All Claims associated with an identifiable SUIT Manifest must always be bundled together in a Claims set that is limited to the Claims defined in this document.

3. SUIT Claims

The Claim description in this document uses CDDL as the formal modeling language for Claims. This approach is derived from [I-D.ietf-rats-eat]. All Claims are based on information elements as used in the SUIT Manifest specification [I-D.ietf-suit-manifest]. For instance, a SUIT Vendor ID is represented as an UUID. Analogously, the corresponding vendor-identifier Claim found below is based on a UUID. SUIT Claims are differentiated in:

- software and hardware characteristics (System Properties), and
- reports about updates their SUIT Commands (SUIT Records).

Both types of Claims are always bundled in dedicated Claim Sets. Implementations can encode this information in various different ways (data models), e.g., sets, sequences, or nested structures. The following subsections define the SUIT Report Claims for RATS and are structured according to the following CDDL expression.

```
suit-report = {
  suit-system-properties => [ + system-property-claims ],
  suit-records => [ + interpreter-record-claims ],
}

system-property-claims => { + $$system-property-claim }
interpreter-record-claims => { + $$interpreter-record-claim }
```

3.1. System Properties Claims

System Properties Claims are composed of:

- Hardware Component Claims and
- Software Component Claims.

Correspondingly, the Claim definitions below highlight if a Claim is generic or hw/sw-component specific.

3.1.1. vendor-identifier

A RFC 4122 UUID representing the vendor of the Attester or one of its hardware and/or software components.

```
$$system-property-claim //= ( vendor-identifier => RFC4122_UUID )
```

3.1.2. class-identifier

A RFC 4122 UUID representing the class of the Attester or one of its hardware and/or software components.

```
$$system-property-claim //= ( class-identifier => RFC4122_UUID )
```

3.1.3. device-identifier

A RFC 4122 UUID representing the Attester.

```
$$system-property-claim //= ( device-identifier => RFC4122_UUID )
```

3.1.4. component-identifier

A sequence of binary identifiers that is intended to identify a software-component of an Attester uniquely. A binary identifier can represent a CoSWID [I-D.ietf-sacm-coswid] tag-id.

```
$$system-property-claim //=  ( class-identifier => [ + identifier ] )
```

3.1.5. image-digest

A fingerprint computed over a software component image on the Attester. This Claim is always bundled with a component-identifier or component-index.

```
$$system-property-claim //= ( image-digest => digest )
```

3.1.6. image-size

The size of a firmware image on the Attester.

```
$$system-property-claim //= ( image-size => size )
```

3.1.7. minimum-battery

The configured minimum battery level of the Attester in mWh.

```
$$system-property-claim //= ( minimum-battery => charge )
```

3.1.8. version

The Version of a hardware or software component of the Attester.

```
$$system-property-claim //= ( version => version-value )
```

3.2. Interpreter Record Claims

This class of Claims represents the content of SUIT Records generated by Interpreters running on Recipients. They are always bundled into Claim Sets representing SUIT Reports and are intended to be included in Evidence generated by an Attester. The Interpreter Record Claims appraised by a Verifier can steer a corresponding a Firmware Appraisal procedures that consumes this Evidence. Analogously, these Claims can be re-used in generated Attestation Results as Trustworthiness Vectors [I-D.voit-rats-trusted-path-routing].

3.2.1. record-success

The result of a Command that was executed by the Interpreter on an Attester.

```
$$interpreter-record-claim //= ( record-success => bool )
```

3.2.2. component-index

A positive integer representing an entry in a flat list of indices mapped to software component identifiers to be updated.

```
$$system-property-claim //= ( component-index => uint )
```

3.2.3. dependency-index

A thumbprint of a software component that an update depends on.

```
$$interpreter-record-claim //= ( dependency-index => digest )
```

3.2.4. command-index

A positive integer representing an entry in a SUIT_Command_Sequence identifying a Command encoded as a SUIT Manifest Directive or SUIT Manifest Condition.

```
$$interpreter-record-claim //= ( command-index => uint )
```

3.2.5. nominal-parameters

A list of SUIT_Parameters associated with a specific Command encoded as a SUIT Manifest Directive.

```
$$interpreter-record-claim //= ( nominal-parameters => parameter-list )
```

3.2.6. nominal-parameters

A list of SUIT_Parameters associated with a specific Command that was executed by the Interpreter on an Attester.

```
$$interpreter-record-claim //= ( actual-parameters => parameter-list )
```

3.3. Generic Record Conditions (TBD)

- test-failed
- unsupported-command
- unsupported-parameter
- unsupported-component-id
- payload-unavailable
- dependency-unavailable
- critical-application-failure
- watchdog-timeout

4. List of Commands (TBD)

- Check Vendor Identifier
- Check Class Identifier
- Verify Image
- Set Component Index
- Override Parameters
- Set Dependency Index

- Set Parameters
- Process Dependency
- Run
- Fetch
- Use Before
- Check Component Offset
- Check Device Identifier
- Check Image Not Match
- Check Minimum Battery
- Check Update Authorized
- Check Version
- Abort
- Try Each
- Copy
- Swap
- Wait For Event
- Run Sequence
- Run with Arguments

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

5.2. Informative References

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-05 (work in progress), July 2020.

[I-D.ietf-rats-eat]
Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", draft-ietf-rats-eat-03 (work in progress), February 2020.

[I-D.ietf-sacm-coswid]
Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", draft-ietf-sacm-coswid-15 (work in progress), May 2020.

[I-D.ietf-suit-manifest]
Moran, B., Tschofenig, H., Birkholz, H., and K. Zandberg, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", draft-ietf-suit-manifest-08 (work in progress), July 2020.

[I-D.ietf-teep-architecture]
Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", draft-ietf-teep-architecture-11 (work in progress), July 2020.

[I-D.voit-rats-trusted-path-routing]
Voit, E., "Trusted Path Routing", draft-voit-rats-trusted-path-routing-02 (work in progress), June 2020.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT

EMail: henk.birkholz@sit.fraunhofer.de

Brendan Moran
Arm Limited

EMail: Brendan.Moran@arm.com

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 3, 2020

H. Birkholz
Fraunhofer SIT
J. "O'Donoghue"
Qualcomm Technologies Inc.
N. Cam-Winget
Cisco Systems
C. Bormann
Universitaet Bremen TZI
June 01, 2020

A CBOR Tag for Unprotected CWT Claims Sets
draft-birkholz-rats-uccs-01

Abstract

CBOR Web Token (CWT, RFC 8392) Claims Sets sometimes do not need the protection afforded by wrapping them into COSE, as is required for a true CWT. This specification defines a CBOR tag for such unprotected CWT Claims Sets (UCCS) and discusses conditions for its proper use.

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 December 3, 2020.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Motivation and Requirements 3
3. Characteristics of a Secure Channel 3
3.1. UCCS and Remote ATtestation procedureS (RATS) 4
3.2. Privacy Preserving Channels 5
4. IANA Considerations 5
5. Security Considerations 5
6. References 6
6.1. Normative References 6
6.2. Informative References 7
Appendix A. Example 7
Authors' Addresses 7

1. Introduction

A CBOR Web Token (CWT) as specified by [RFC8392] is always wrapped in a CBOR Object Signing and Encryption (COSE, [RFC8152]) envelope. COSE provides - amongst other things - the integrity protection mandated by RFC 8392 and optional encryption for CWTs. Under the right circumstances, though, a signature providing proof for authenticity and integrity can be provided through the transfer protocol and thus omitted from the information in a CWT without compromising the intended goal of authenticity and integrity. If a mutually Secured Channel is established between two remote peers, and if that Secure Channel provides the correct properties, it is possible to omit the protection provided by COSE, creating a use case for unprotected CWT Claims Sets. Similarly, if there is one-way authentication, the party that did not authenticate may be in a position to send authentication information through this channel that allows the already authenticated party to authenticate the other party.

This specification allocates a CBOR tag to mark Unprotected CWT Claims Sets (UCCS) as such and discusses conditions for its proper use in the scope of Remote ATtestation procedureS (RATS).

This specification does not change [RFC8392]: A true CWT does not make use of the tag allocated here; the UCCS tag is an alternative to using COSE protection and a CWT tag. Consequently, in a well-defined scope, it might be acceptable to strip a CWT of its COSE container an

replace the CWT Claims Set's CWT CBOR tag with a UCCS CBOR tag for further processing - or vice versa.

1.1. Terminology

The term Claim is used as in [RFC8725].

The terms Claim Key, Claim Value, and CWT Claims Set are used as in [RFC8392].

UCCS: Unprotected CWT Claims Set; a CBOR map of Claims as defined by the CWT Claims Registry that are composed of pairs of Claim Keys and Claim Values.

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. Motivation and Requirements

Use cases involving the conveyance of claims, in particular, remote attestations [I-D.ietf-rats-architecture] require a standardized data schema and format that can be trasferred and transported using different communication channels. As these are Claims, [RFC8392] are a suitable format but how these Claims are secured depends on the deployment, the security capabilities of the device, as well as their software stack. For example, a Claim may be securely stored and conveyed using the device's trusted execution environment or especially in some resource constrained environments the same process that provides the secure communication transport is also the delegate to compose the Claim to be conveyed. Whether it is a transfer or transport, a Secure Channel is presumed to be used for conveying such UCCS. The following section further describes the requirements and scenarios in which UCCS can be used.

3. Characteristics of a Secure Channel

A Secure Channel for the conveyance of UCCS needs to provide the security properties that would otherwise be provided by COSE for a CWT. In this regard, UCCS is similar in security considerations to JWTs [RFC8725] using the algorithm "none". RFC 8725 states: "if a JWT is cryptographically protected end-to-end by a transport layer, such as TLS using cryptographically current algorithms, there may be no need to apply another layer of cryptographic protections to the JWT. In such cases, the use of the "none" algorithm can be perfectly acceptable.". Analogously, the considerations discussed in Sections

2.1, 3.1, and 3.2 of RFC 8725 apply to the use of UCCS as elaborated on in this document.

Secure Channels are often set up in a handshake protocol that mutually derives a session key, where the handshake protocol establishes the authenticity of one of both ends of the communication. The session key can then be used to provide confidentiality and integrity of the transfer of information inside the Secure Channel. A well-known example of a such a Secure Channel setup protocol is the TLS [RFC8446] handshake; the TLS record protocol can then be used for secure conveyance.

As UCCS were initially created for use in Remote ATtestation procedureS (RATS) Secure Channels, the following subsection provides a discussion of their use in these channels. Where other environments are intended to be used to convey UCCS, similar considerations need to be documented before UCCS can be used.

3.1. UCCS and Remote ATtestation procedureS (RATS)

Secure Channels can be transient in nature. For the purposes of this specification, the mechanisms used to establish a Secure Channel are out of scope. As a minimum requirement in the scope of RATS Claims, however, the Verifier must authenticate the Attester as part of the Secure Channel establishment.

If only authenticity/integrity for a Claim is required, a Secure Channel MUST be established to, at minimum, provide integrity of the communication. Further, the provider of the UCCS SHOULD be authenticated by the reciever to ensure the channel is truly secured and the sender is validated. If confidentiality is also required, the receiving side SHOULD also be authenticated.

The extent to which a Secure Channel can provide assurances that UCCS originate from a trustworthy attesting environment depends on the characteristics of both the cryptographic mechanisms used to establish the channel and the characteristics of the attesting environment itself. A Secure Channel established or maintained using weak cryptography may not provide the assurance required by a relying party of the authenticity and integrity of the UCCS.

Where the security assurance required of an attesting environment by a relying party requires it, the attesting environment may be implemented using techniques designed to provide enhanced protection from an attacker wishing to tamper with or forge UCCS. A possible approach might be to implement the attesting environment in a hardened environment such as a TEE [I-D.ietf-teep-architecture] or a TPM [TPM2].

As with EATs nested in other EATs (Section 3.12.1.2 of [I-D.ietf-rats-eat]), the Secure Channel does not endorse fully formed CWTs transferred through it. Effectively, the COSE envelope of a CWT shields the CWT Claims Set from the endorsement of the Secure Channel. (Note that EAT might add a nested UCCS Claim, and this statement does not apply to UCCS nested into UCCS, only to fully formed CWTs)

3.2. Privacy Preserving Channels

A Secure Channel which preserves the privacy of the Attester may provide security properties equivalent to COSE, but only inside the life-span of the session established. In general, a Verifier cannot correlate UCCS received in different sessions from the same attesting environment based on the cryptographic mechanisms used when a privacy preserving Secure Channel is employed.

In the case of a Remote Attestation, the attester must consider whether any UCCS it returns over a privacy preserving Secure Channel compromises the privacy in unacceptable ways. As an example, the use of the EAT UEID [I-D.ietf-rats-eat] Claim in UCCS over a privacy preserving Secure Channel allows a verifier to correlate UCCS from a single attesting environment across many Secure Channel sessions. This may be acceptable in some use-cases (e.g. if the attesting environment is a physical sensor in a factory) and unacceptable in others (e.g. if the attesting environment is a device belonging to a child).

4. IANA Considerations

In the registry [IANA.cbor-tags], IANA is requested to allocate the tag in Table 1 from the FCFS space, with the present document as the specification reference.

Tag	Data Item	Semantics
TBD601	map	Unprotected CWT Claims Set [RFCthis]

Table 1: Values for Tags

5. Security Considerations

The security considerations of [RFC7049] and [RFC8392] apply.

{#secchan} discusses security considerations for Secure Channels, in which UCCS might be used. This documents provides the CBOR tag

definition for UCCS and a discussion on security consideration for the use of UCCS in Remote ATtestation procedureS (RATS). Uses of UCCS outside the scope of RATS are not covered by this document. The UCCS specification - and the use of the UCCS CBOR tag, correspondingly - is not intended for use in a scope where a scope-specific security consideration discussion has not been conducted, vetted and approved for that use.

6. References

6.1. Normative References

[IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <http://www.iana.org/assignments/cbor-tags>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

[TPM2] "Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 01.59 ed., Trusted Computing Group", 2019.

6.2. Informative References

[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.

[I-D.ietf-rats-eat]
Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", draft-ietf-rats-eat-03 (work in progress), February 2020.

[I-D.ietf-teep-architecture]
Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", draft-ietf-teep-architecture-08 (work in progress), April 2020.

Appendix A. Example

The example CWT Claims Set from Appendix A.1 of [RFC8392] can be turned into an UCCS by enclosing it with a tag number TBD601:

```
 <TBD601>(
   {
     / iss / 1: "coap://as.example.com",
     / sub / 2: "erikw",
     / aud / 3: "coap://light.example.com",
     / exp / 4: 1444064944,
     / nbf / 5: 1443944944,
     / iat / 6: 1443944944,
     / cti / 7: h'0b71'
   }
 )
```

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

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

Email: jodonogh@qti.qualcomm.com

Nancy Cam-Winget
Cisco Systems
3550 Cisco Way
San Jose, CA 95134
USA

Email: ncamwing@cisco.com

Carsten Bormann
Universitaet Bremen TZI
Bibliothekstrasse 1
Bremen 28369
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.de

RATS Working Group
Internet-Draft
Intended status: Informational
Expires: 5 March 2021

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

Remote Attestation Procedures Architecture
draft-ietf-rats-architecture-06

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/.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Reference Use Cases 5
3.1. Network Endpoint Assessment 5
3.2. Confidential Machine Learning (ML) Model Protection . . . 6
3.3. Confidential Data Retrieval 6
3.4. Critical Infrastructure Control 7
3.5. Trusted Execution Environment (TEE) Provisioning 7
3.6. Hardware Watchdog 7
3.7. FIDO Biometric Authentication 8
4. Architectural Overview 8
4.1. Appraisal Policies 10
4.2. Two Types of Environments of an Attester 10
4.3. Layered Attestation Environments 11
4.4. 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 and Verifier Owner 22
8. Conceptual Messages 22
8.1. Evidence 22
8.2. Endorsements 22
8.3. Attestation Results 23
9. Claims Encoding Formats 24
10. Freshness 26
11. Privacy Considerations 28
12. Security Considerations 28
12.1. Attester and Attestation Key Protection 29
12.1.1. On-Device Attester and Key Protection 29
12.1.2. Attestation Key Provisioning Processes 30
12.2. Integrity Protection 30
13. IANA Considerations 31
14. Acknowledgments 31
15. Contributors 32
16. Appendix A: Time Considerations 32
16.1. Example 1: Timestamp-based Passport Model Example . . . 33
16.2. Example 2: Nonce-based Passport Model Example 35
16.3. Example 3: Handle-based Passport Model Example 36
16.4. Example 4: Timestamp-based Background-Check Model Example 38
16.5. Example 5: Nonce-based Background-Check Model Example . 38
17. References 39
17.1. Normative References 39
17.2. Informative References 39
Authors' Addresses 40

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 documents 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 some entity (typically a manufacturer) vouches for the integrity of an Attester's signing capability

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.

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 to do inferencing

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.

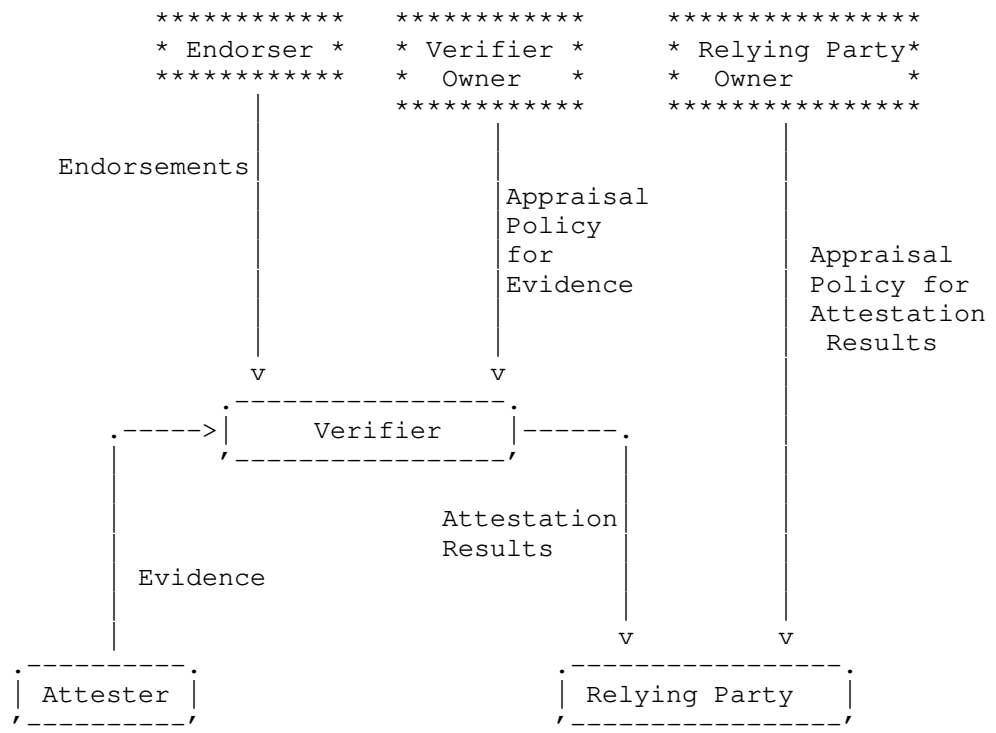


Figure 1: Conceptual Data Flow

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.

Such reference values 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.

4.2. 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 4.3 and Section 4.4. 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.

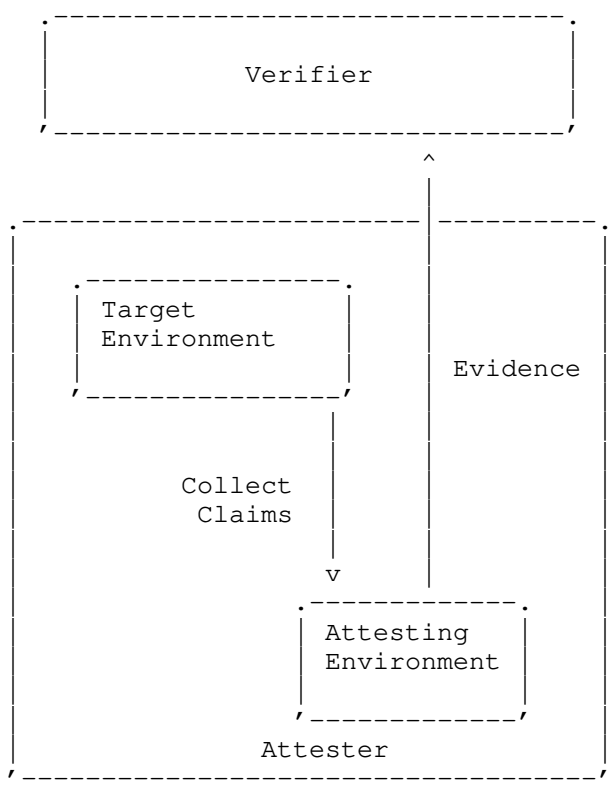


Figure 2: Two Types of Environments

4.3. 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.

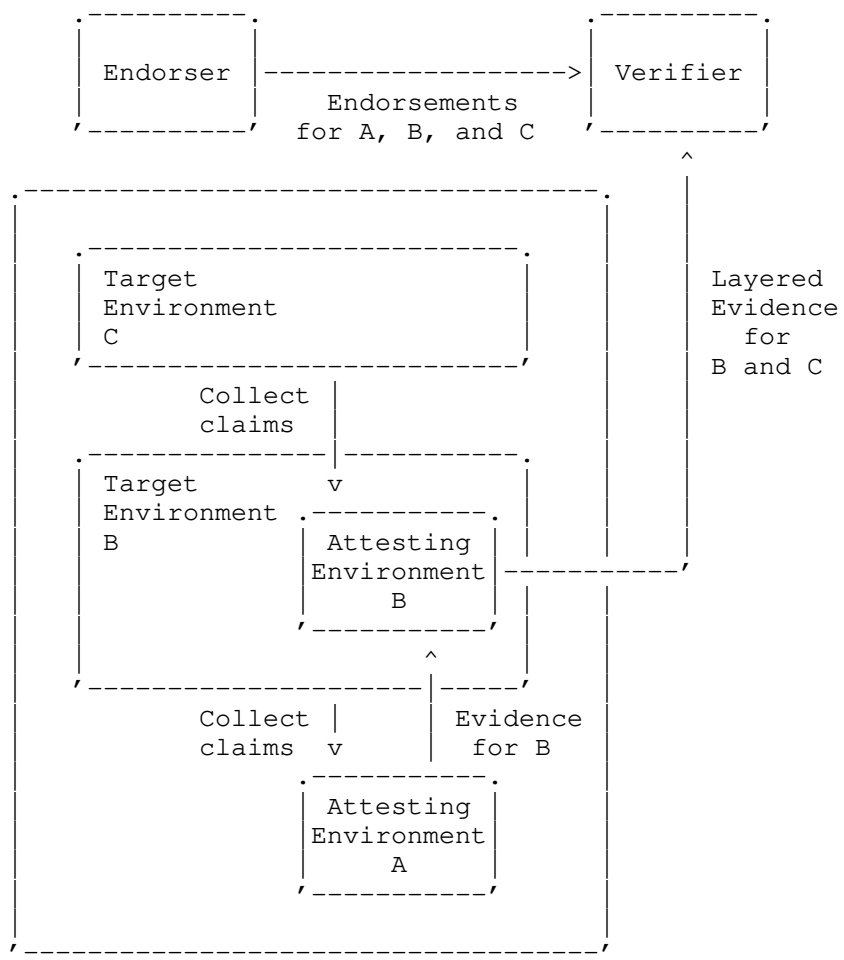


Figure 3: Layered Attester

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.

4.4. 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.

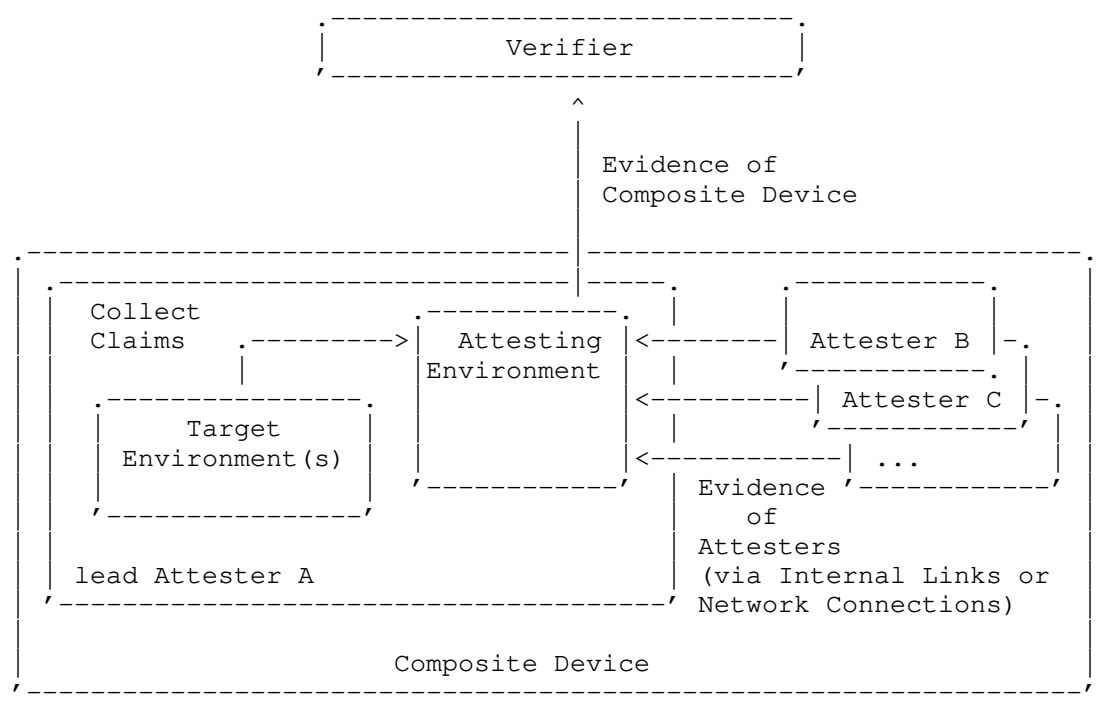


Figure 4: Conceptual Data Flow for a Composite Device

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.

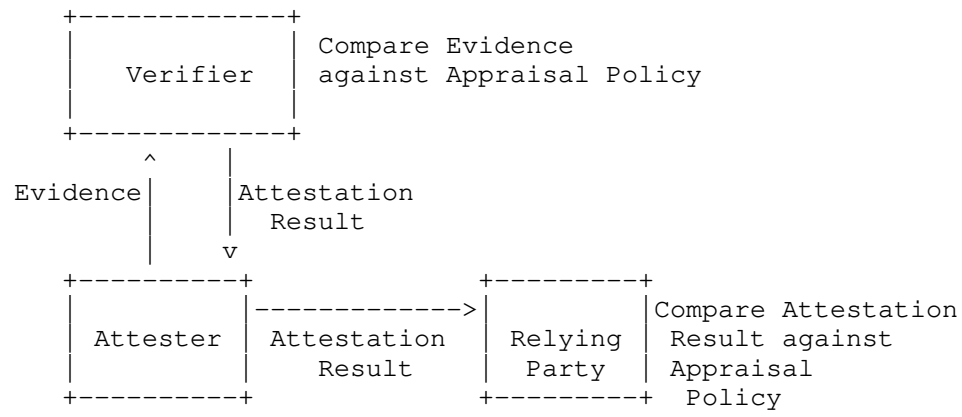


Figure 5: Passport Model

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.

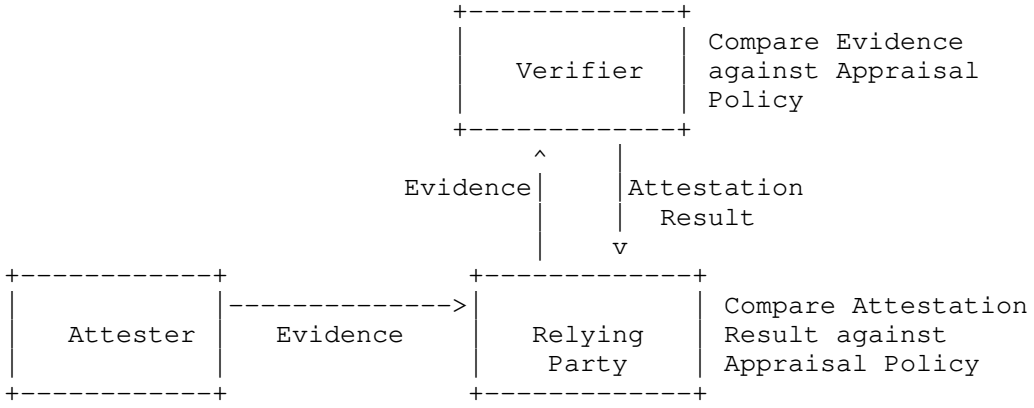


Figure 6: Background-Check Model

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].

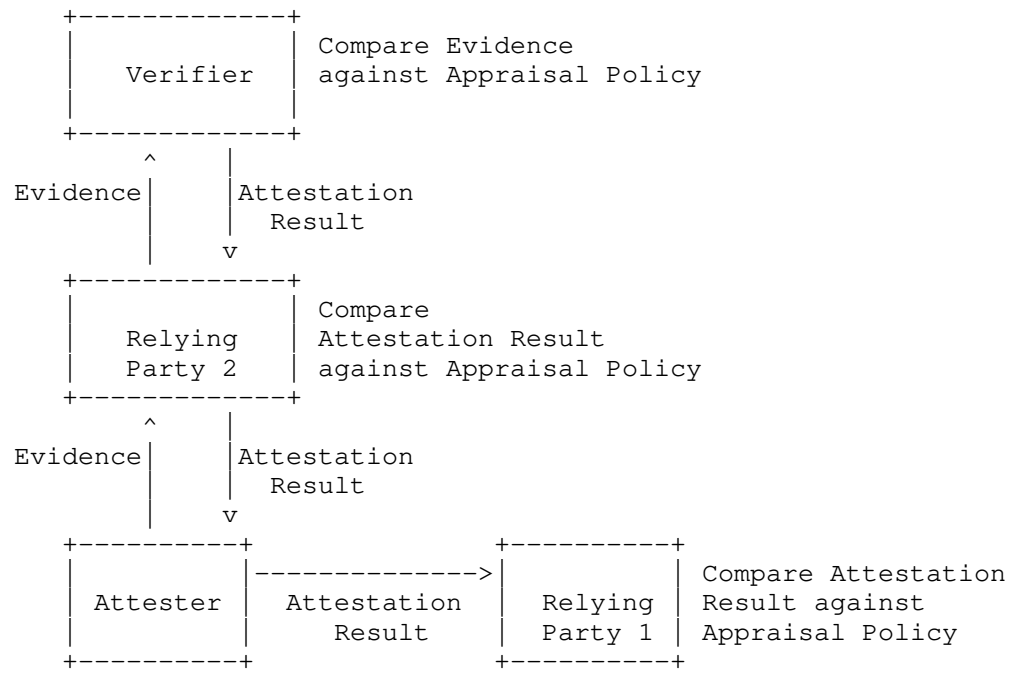


Figure 7: Example Combination

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.

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 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 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 solutions with weaker security, a Verifier might be configured to implicitly trust firmware or even software (e.g., a hypervisor). 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 hypervisor or firmware 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. The component that is implicitly trusted is often referred to as a Root of Trust.

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.

7.5. Endorser and Verifier Owner

In some scenarios, the Endorser and Verifier Owner may need to trust the Verifier before giving the Endorsement and 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 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 known-good 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 known-good 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:

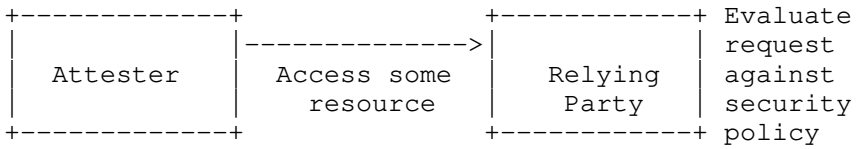


Figure 8: Typical Resource Access

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.

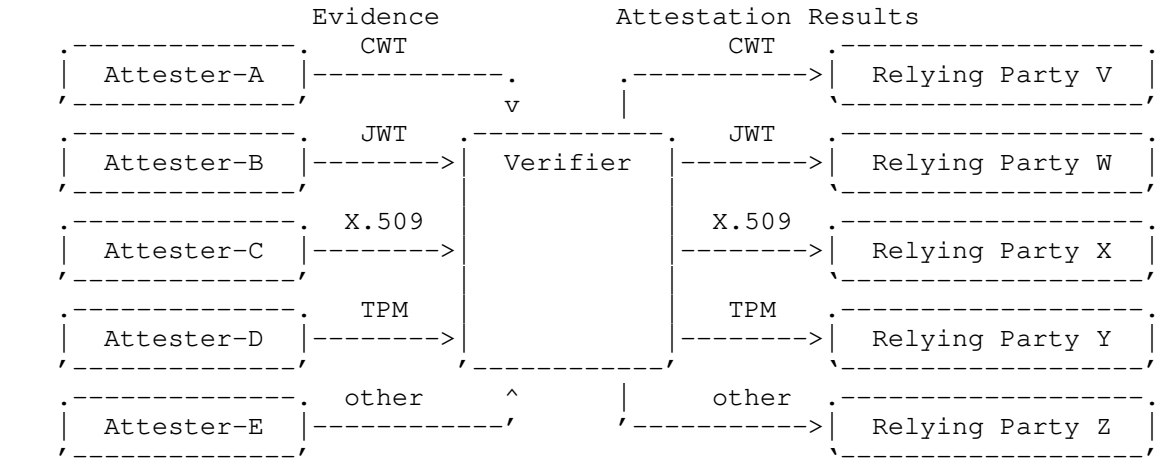


Figure 9: Multiple Attesters and Relying Parties with Different Formats

10. Freshness

A remote entity (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 a policy defined by the consuming entity, Verifier or Relying Party, 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 Policy for Evidence, 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 appraising entity, i.e., 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. Contributors

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

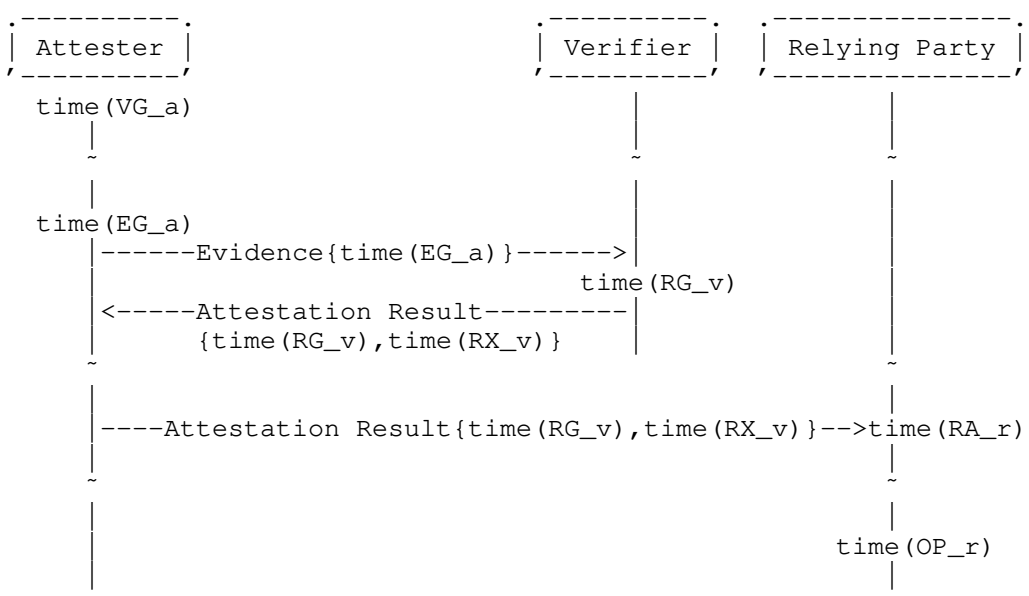
Using the table above, a number of hypothetical examples of how a solution might be built are illustrated below. a solution might be built. This list is not intended to be complete, but is just representative enough to highlight various timing considerations.

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.

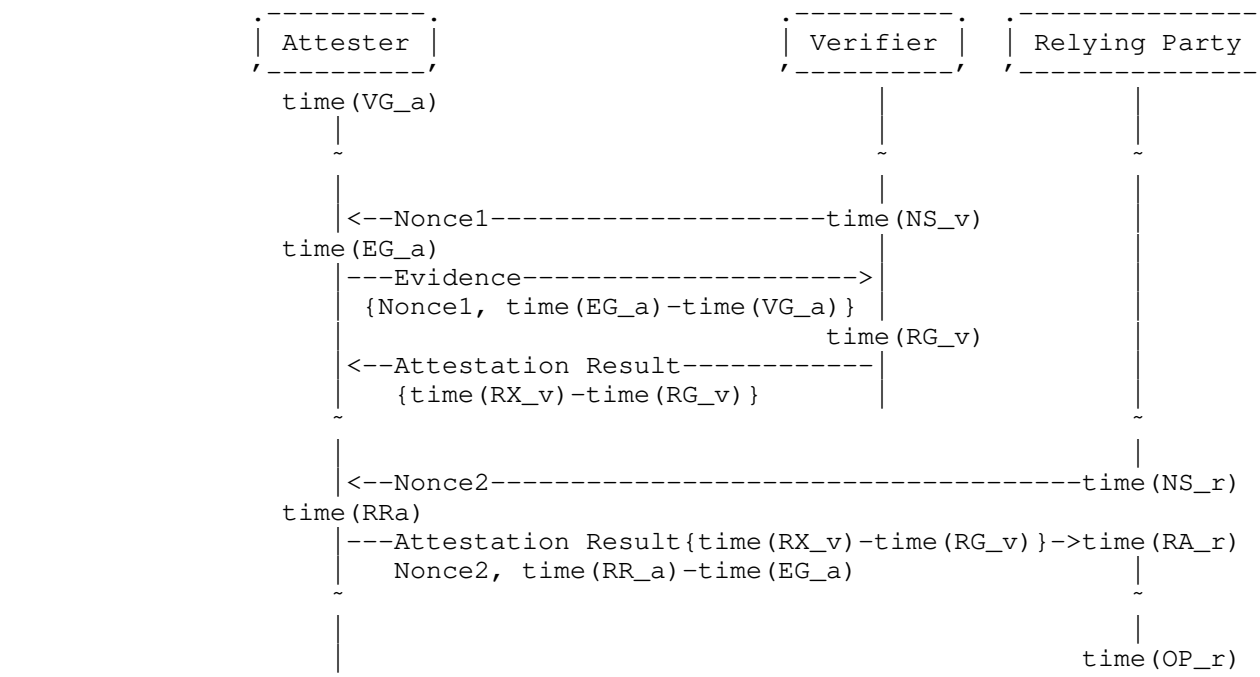
If time(VG_a) is also included in the Evidence along with the claim value generated at that time, and the Verifier decides that it can trust the time(VG_a) value, the Verifier can also determine whether the claim value is recent by checking "time(RG_v) - time(VG_a) < Threshold", again where the threshold 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 "time(OP_r)-time(NS_r) + time(RR_a)-time(EG_a) < Threshold". Although the Nonce2 and "time(RR_a)-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 "time(RX_v)-time(RG_v)", then the Relying Party can check "time(OP_r)-time(NS_r) < time(RX_v)-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.

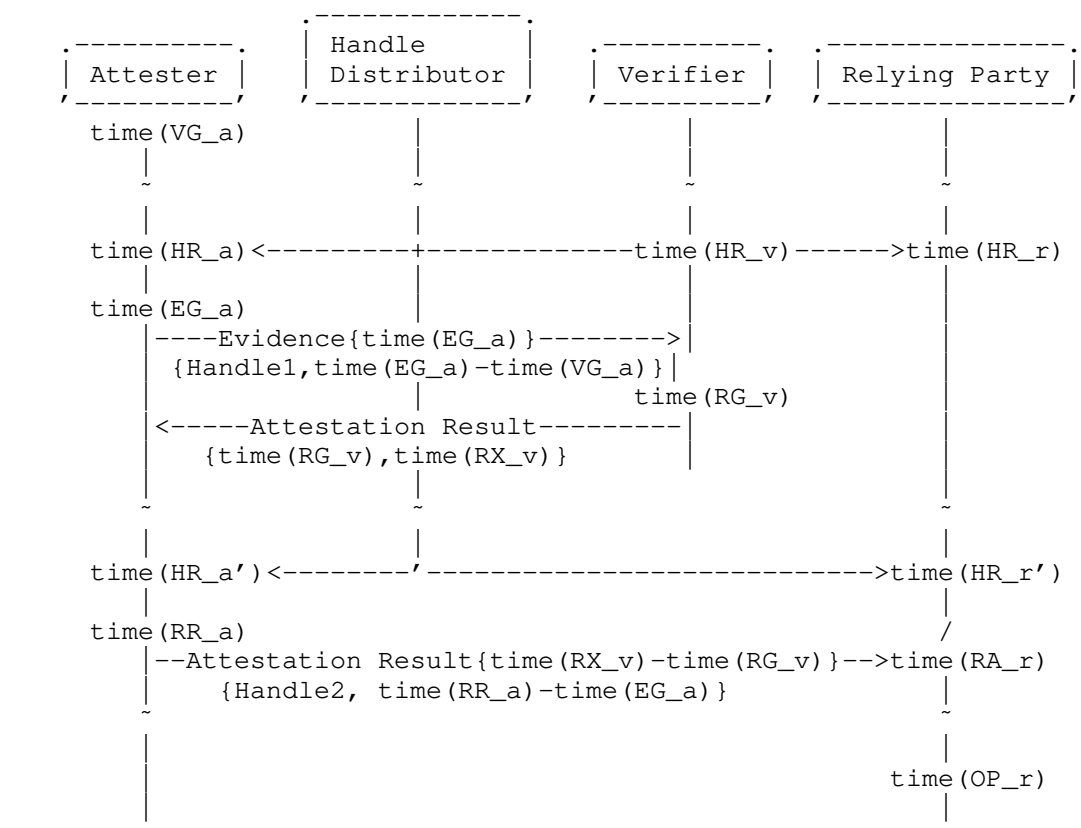
In this example, Evidence generation based on received handles always uses the current (most recent) handle. As handles are distributed over the network, all involved entities receive a fresh handle at roughly the same time. Due to distribution over the network, there is some jitter with respect to the 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: "validity-duration = time(HR'_v) + offset - 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".

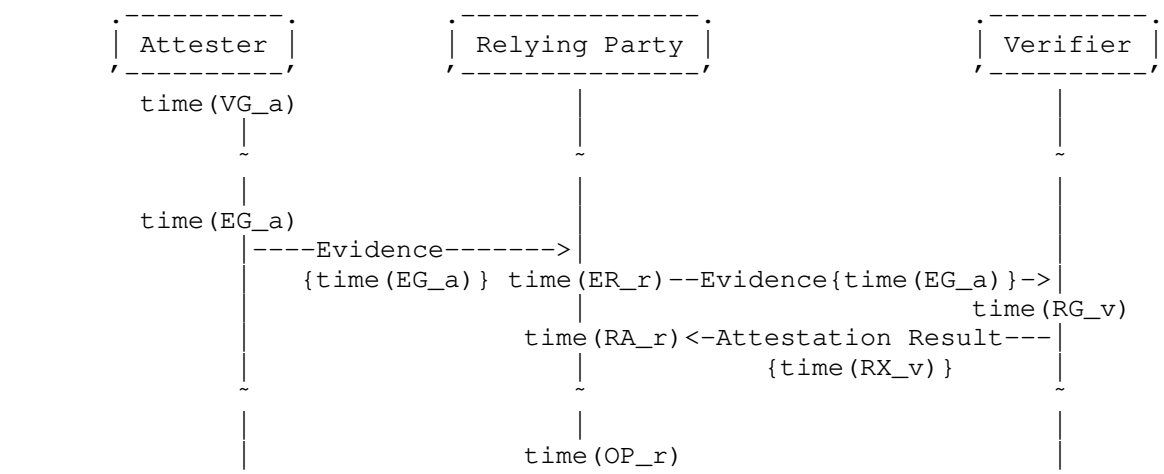
Due to the validity-duration of handles, multiple different pieces of Evidence can be generated based on the same handle. The resulting granularity (time resolution) of Evidence freshness is typically lower than the resolution of clock-based tickcounters.

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

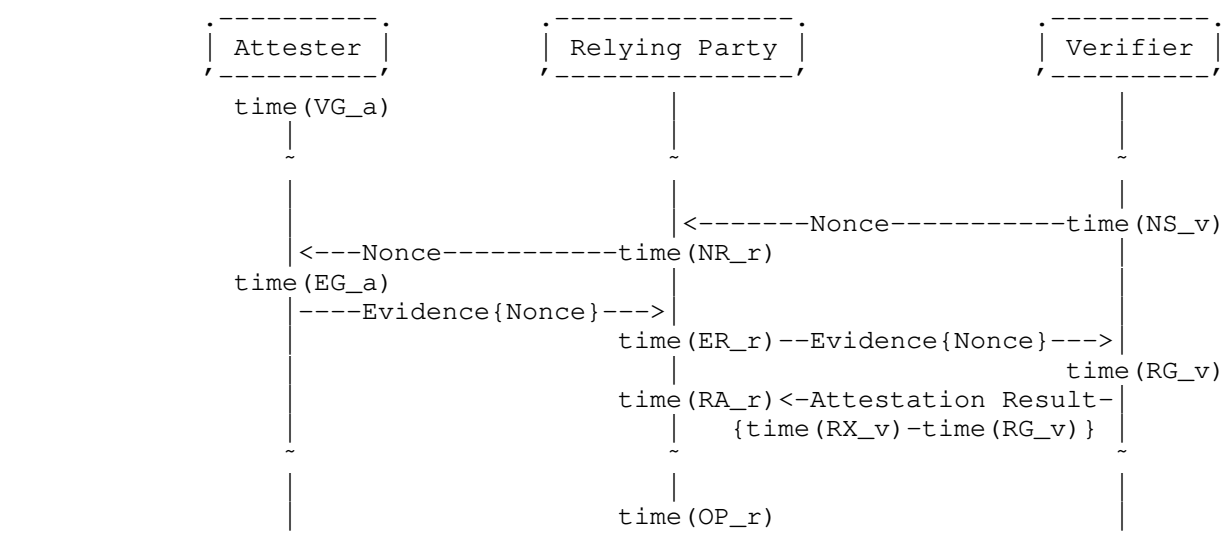
The following example illustrates a hypothetical Background-Check Model solution that uses timestamps and requires roughly synchronized clocks between the Attester, Verifier, and Relying Party.



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/>.

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

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:

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

1.1. CDDL, CWT and JWT

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

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

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

1.2. Entity Overview

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

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

1.3. EAT Operating Models

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

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

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

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

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

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

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

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

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

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

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

1.4. What is Not Standardized

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

1.4.1. Transmission Protocol

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

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

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

1.4.2. Signing Scheme

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

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

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

2. Terminology

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

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

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

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

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

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

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

3. The Claims

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

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

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

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

3.1. Token ID Claim (cti and jti)

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

3.2. Timestamp claim (iat)

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

3.3. Nonce Claim (nonce)

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

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

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

Multiple nonces are allowed to accommodate multistage verification and consumption.

3.3.1. nonce CDDL

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

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

3.4. Universal Entity ID Claim (ueid)

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

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

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

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

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

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

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

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

Table 1: UEID Composition Types

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

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

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

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

3.4.1. ueid CDDL

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

3.5. Origination Claim (origination)

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

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

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

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

3.5.1. origination CDDL

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

3.6. OEM Identification by IEEE (oemid)

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

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

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

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

3.6.1. oemid CDDL

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

3.7. The Security Level Claim (security-level)

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

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

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

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

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

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

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

3.7.1. security-level CDDL

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

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

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

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

3.8.1. Secure Boot Enabled

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

3.8.2. Debug Disabled

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

3.8.3. Debug Disabled Since Boot

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

3.8.4. Debug Permanent Disable

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

3.8.5. Debug Full Permanent Disable

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

3.8.6. boot-state CDDL

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

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

3.9. The Location Claim (location)

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

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

3.9.1. location CDDL

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

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

3.10. The Age Claim (age)

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

3.10.1. age CDDL

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

3.11. The Uptime Claim (uptime)

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

3.11.1. uptime CDDL

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

3.12. The Submods Part of a Token (submods)

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

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

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

3.12.1. Two Types of Submodules

Each entry in the submod map one of two types:

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

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

3.12.1.1. Non-token Submodules

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

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

3.12.1.2. Nested EATs

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

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

3.12.2. No Inheritance

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

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

3.12.3. Security Levels

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

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

3.12.4. Submodule Names

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

3.12.5. submods CDDL

```
submods-type = { + submodule }

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

submod_name = tstr / int

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

4. Encoding

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

4.1. Common CDDL Types

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

4.2. CDDL for CWT-defined Claims

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

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

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

cwt-claim = rfc8392-claim
```

4.3. JSON

4.3.1. JSON Labels

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

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

4.3.2. JSON Interoperability

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

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

4.4. CBOR

4.4.1. CBOR Labels

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

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

4.4.2. CBOR Interoperability

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

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

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

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

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

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

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

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

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

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

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

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

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

4.5. Collected CDDL

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

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

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

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

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

5. IANA Considerations

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

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

5.1.1. Claims Registered by This Document

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

TODO: add the rest of the claims in here

6. Privacy Considerations

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

6.1. UEID Privacy Considerations

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

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

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

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

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

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

7. Security Considerations

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

7.1. Key Provisioning

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

7.1.1. Transmission of Key Material

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

7.2. Transport Security

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

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

7.3. Multiple EAT Consumers

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

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

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

8. References

8.1. Normative References

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

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

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

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

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

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

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

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

8.2. Informative References

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

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

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

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

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

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

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

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

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

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

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

Appendix A. Examples

A.1. Very Simple EAT

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

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

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

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

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

Appendix B. UEID Design Rationale

B.1. Collision Probability

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

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

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

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

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

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

The proper formula for the collision calculation is

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

p Collision Probability
n Total possible population
k Actual population

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

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

For this calculation:

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

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

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

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

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

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

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

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

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

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

B.2. No Use of UUID

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

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

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

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

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

Appendix C. Changes from Previous Drafts

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

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

- Added UEID design rationale appendix

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

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

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

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

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

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

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

Authors' Addresses

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

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

Laurence Lundblade
Security Theory LLC

EMail: lgl@island-resort.com

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

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

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

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

RATS Working Group
Internet-Draft
Intended status: Informational
Expires: March 22, 2021

G. Fedorkow, Ed.
Juniper Networks, Inc.
E. Voit
Cisco Systems, Inc.
J. Fitzgerald-McKay
National Security Agency

September 18, 2020

TPM-based Network Device Remote Integrity Verification
draft-ietf-rats-tpm-based-network-device-attest-04

Abstract

This document describes a workflow for remote attestation of the integrity of firmware and software installed on network devices that contain Trusted Platform Modules [TPM1.2], [TPM2.0].

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 22, 2021.

Table of Contents

1. Introduction 3
1.1. Terminology 3
1.2. Document Organization 4
1.3. Goals 5
1.4. Description of Remote Integrity Verification (RIV) . . . 5
1.5. Solution Requirements 7
1.6. Scope 8
1.6.1. Out of Scope 8
2. Solution Overview 9
2.1. RIV Software Configuration Attestation using TPM 9
2.1.1. What Does RIV Attest? 10
2.1.2. Notes on PCR Allocations 12
2.2. RIV Keying 13
2.3. RIV Information Flow 14
2.4. RIV Simplifying Assumptions 16
2.4.1. Reference Integrity Manifests (RIMs) 17
2.4.2. Attestation Logs 18
3. Standards Components 19
3.1. Prerequisites for RIV 19
3.1.1. Unique Device Identity 19
3.1.2. Keys 19
3.1.3. Appraisal Policy for Evidence 19
3.2. Reference Model for Challenge-Response 20
3.2.1. Transport and Encoding 22
3.3. Centralized vs Peer-to-Peer 23
4. Privacy Considerations 24
5. Security Considerations 25
5.1. Keys Used in RIV 25
5.2. Prevention of Spoofing and Man-in-the-Middle Attacks . . 27
5.3. Replay Attacks 28
5.4. Owner-Signed Keys 28
5.5. Other Trust Anchors 29
6. Conclusion 30
7. IANA Considerations 30
8. Appendix 30
8.1. Using a TPM for Attestation 30
8.2. Root of Trust for Measurement 32
8.3. Layering Model for Network Equipment Attester and Verifier 32
8.3.1. Why is OS Attestation Different? 34
8.4. Implementation Notes 34
9. References 36
9.1. Normative References 36

9.2. Informative References 38
Authors' Addresses 41

1. Introduction

There are many aspects to consider in fielding a trusted computing device, from operating systems to applications. Mechanisms to prove that a device installed at a customer's site is authentic (i.e., not counterfeit) and has been configured with authorized software, all as part of a trusted supply chain, are just a few of the many aspects which need to be considered concurrently to have confidence that a device is truly trustworthy.

A generic architecture for remote attestation has been defined in [I-D.ietf-rats-architecture]. Additionally, the use cases for remotely attesting networking devices are discussed within Section 6 of [I-D.richardson-rats-usecases]. However, these documents do not provide sufficient guidance for network equipment vendors and operators to design, build, and deploy interoperable devices.

The intent of this document is to provide such guidance. It does this by outlining the Remote Integrity Verification (RIV) problem, and then identifies elements that are necessary to get the complete, scalable attestation procedure working with commercial networking products such as routers, switches and firewalls. An underlying assumption will be the availability within the device of a Trusted Platform Module [TPM1.2], [TPM2.0] compliant cryptoprocessor to enable the trustworthy remote assessment of the device's software and hardware.

1.1. Terminology

A number of terms are reused from [I-D.ietf-rats-architecture]. These include: Appraisal Policy for Attestation Results, Attestation Result, Attester, Evidence, Relying Party, Verifier, and Verifier Owner.

Additionally, this document defines the following terms:

Remote Attestation: the process of creating, conveying and appraising claims about device trustworthiness characteristics, including supply chain trust, identity, device provenance, software configuration, hardware configuration, device composition, compliance to test suites, functional and assurance evaluations, etc.

This document uses the term Endorser to refer to the trusted authority for any signed object relating to the device, such as

certificates or reference measurement. Typically, the manufacturer of an embedded device would be accepted as an Endorser.

The goal of attestation is simply to assure an administrator that the software that was launched when the device was last started is an authentic and untampered-with copy of the software that the device vendor shipped.

Within the Trusted Computing Group context, attestation is the process by which an independent Verifier can obtain cryptographic proof as to the identity of the device in question, and evidence of the integrity of software loaded on that device when it started up, and then verify that what's there is what's supposed to be there. For networking equipment, a Verifier capability can be embedded in a Network Management Station (NMS), a posture collection server, or other network analytics tool (such as a software asset management solution, or a threat detection and mitigation tool, etc.). While informally referred to as attestation, this document focuses on a subset defined here as Remote Integrity Verification (RIV). RIV takes a network equipment centric perspective that includes a set of protocols and procedures for determining whether a particular device was launched with authentic software, starting from Roots of Trust. While there are many ways to accomplish attestation, RIV sets out a specific set of protocols and tools that work in environments commonly found in Networking Equipment. RIV does not cover other device characteristics that could be attested (e.g., geographic location, connectivity; see [I-D.richardson-rats-usecases]), although it does provide evidence of a secure infrastructure to increase the level of trust in other device characteristics attested by other means (e.g., by Entity Attestation Tokens [I-D.ietf-rats-eat]).

1.2. Document Organization

The remainder of this document is organized into several sections:

- The remainder of this section covers goals and requirements, plus a top-level description of RIV.
- The Solution Overview section outlines how Remote Integrity Verification works.
- The Standards Components section links components of RIV to normative standards.
- Privacy and Security shows how specific features of RIV contribute to the trustworthiness of the Attestation Result.
- Supporting material is in an appendix at the end.

1.3. Goals

Network operators benefit from a trustworthy attestation mechanism that provides assurance that their network comprises authentic equipment, and has loaded software free of known vulnerabilities and unauthorized tampering. In line with the overall goal of assuring integrity, attestation can be used to assist in asset management, vulnerability and compliance assessment, plus configuration management.

The RIV attestation workflow outlined in this document is intended to meet the following high-level goals:

- Provable Device Identity - This specification requires that an attesting device includes a cryptographic identifier unique to each device. Effectively this means that the TPM must be so provisioned during the manufacturing cycle.

- Software Inventory - A key goal is to identify the software release(s) installed on the attesting device, and to provide evidence that the software stored within hasn't been altered without authorization.

- Verifiability - Verification of software and configuration of the device shows that the software that was authorized for installation by the administrator has actually been launched.

In addition, RIV is designed to operate either in a centralized environment, such as with a central authority that manages and configures a number of network devices, or 'peer-to-peer', where network devices independently verify one another to establish a trust relationship. (See Section 3.3 below, and also [I-D.voit-rats-trusted-path-routing])

1.4. Description of Remote Integrity Verification (RIV)

Attestation requires two interlocking services between the Attester network device and the Verifier:

- Device Identity, the mechanism providing trusted identity, can reassure network managers that the specific devices they ordered from authorized manufacturers for attachment to their network are those that were installed, and that they continue to be present in their network. As part of the mechanism for Device Identity, cryptographic proof of the identity of the manufacturer is also provided.

- Software Measurement is the mechanism that reports the state of mutable software components on the device, and can assure network managers that they have known, authentic software configured to run in their network.

Using these two interlocking services, RIV is a component in a chain of procedures that can assure a network operator that the equipment in their network can be reliably identified, and that authentic software of a known version is installed on each device. Equipment in the network includes devices that make up the network itself, such as routers, switches and firewalls.

RIV includes several major processes:

1. Creation of Evidence is the process whereby an Attester generates cryptographic proof (Evidence) of claims about device properties. In particular, the device identity and its software configuration are both of critical importance.

2. Device Identification refers to the mechanism assuring the Relying Party (ultimately, a network administrator) of the identity of devices that make up their network, and that their manufacturers are known.

3. Software used to boot a device can be described as a chain of measurements, anchored at the start by a Root of Trust for Measurement, that normally ends when the system software is loaded. A measurement signifies the identity, integrity and version of each software component registered with an attesting device's TPM [TPM1.2], [TPM2.0], so that the subsequent appraisal stage can determine if the software installed is authentic, up-to-date, and free of tampering.

4. Conveyance of Evidence reliably transports at least the minimum amount of Evidence from Attester to a Verifier to allow a management station to perform a meaningful appraisal in Step 5. The transport is typically carried out via a management network. The channel must provide integrity and authenticity, and, in some use cases, may also require confidentiality.

5. Finally, Appraisal of Evidence occurs. As the Verifier and Relying Party roles are often combined within RIV, this is the process of verifying the Evidence received by a Verifier from the Attesting device, and using an Appraisal Policy to develop an Attestation Result, used to inform decision making. In practice, this means comparing the device measurements reported as Evidence with the Attester configuration expected by the Verifier. Subsequently the Appraisal Policy for Attestation Results might

match what was found against Reference Integrity Measurements (aka Golden Measurements) which represent the intended configured state of the connected device.

All implementations supporting this RIV specification require the support of the following three technologies:

1. Identity: Device identity MUST be based on IEEE 802.1AR Device Identity (DevID) [IEEE-802-1AR], coupled with careful supply-chain management by the manufacturer. The DevID certificate contains a statement by the manufacturer that establishes the identity of the device as it left the factory. Some applications with a more-complex post-manufacture supply chain (e.g., Value Added Resellers), or with different privacy concerns, may want to use alternative mechanisms for platform authentication (for example, TCG Platform Certificates [Platform-Certificates]).

2. Platform Attestation provides evidence of configuration of software elements present in the device. This form of attestation can be implemented with TPM Platform Configuration Registers (PCRs), Quote and Log mechanisms, which provide cryptographically authenticated evidence to report what software was started on the device through the boot cycle. Successful attestation requires an unbroken chain from a boot-time root of trust through all layers of software needed to bring the device to an operational state, in which each stage measures components of the next stage, updates the attestation log, and extends hashes into a PCR. The TPM can then report the hashes of all the measured hashes as signed evidence called a Quote (see Section 8.1 for an overview of TPM operation, or [TPM1.2] and [TPM2.0] for many more details).

3. Reference Integrity Measurements must be conveyed from the Endorser (the entity accepted as the software authority, often the manufacturer for embedded systems) to the system in which verification will take place.

1.5. Solution Requirements

Remote Integrity Verification must address the "Lying Endpoint" problem, in which malicious software on an endpoint may subvert the intended function, and also prevent the endpoint from reporting its compromised status. (See Section 5 for further Security Considerations)

RIV attestation is designed to be simple to deploy at scale. RIV should work "out of the box" as far as possible, that is, with the fewest possible provisioning steps or configuration databases needed

at the end-user's site, as network equipment is often required to "self-configure", to reliably reach out without manual intervention to prove its identity and operating posture, then download its own configuration. See [RFC8572] for an example of Secure Zero Touch Provisioning.

1.6. Scope

Remote Attestation is a very general problem that could apply to most network-connected computing devices. However, this document includes several assumptions that limit the scope to Network Equipment (e.g., routers, switches and firewalls):

- This solution is for use in non-privacy-preserving applications (for example, networking, Industrial IoT), avoiding the need for a Privacy Certificate Authority for attestation keys [AIK-Enrollment] or TCG Platform Certificates [Platform-Certificates]

- This document assumes network protocols that are common in networking equipment such as YANG [RFC7950] and NETCONF [RFC6241], but not generally used in other applications.

- The approach outlined in this document mandates the use of a compliant TPM [TPM1.2], [TPM2.0].

1.6.1. Out of Scope

- Run-Time Attestation: Run-time attestation of Linux or other multi-threaded operating system processes considerably expands the scope of the problem. Many researchers are working on that problem, but this document defers the run-time attestation problem.

- Multi-Vendor Embedded Systems: Additional coordination would be needed for devices that themselves comprise hardware and software from multiple vendors, integrated by the end user.

- Processor Sleep Modes: Network equipment typically does not "sleep", so sleep and hibernate modes are not considered. Although out of scope for RIV, Trusted Computing Group specifications do encompass sleep and hibernate states.

- Virtualization and Containerization: In a non-virtualized system, the host OS is responsible for measuring each User Space file or process, but that't the end of the boot process. For virtualized systems, the host OS must verify the hypervisor, which then manages its own chain of trust through the virtual machine.

Virtualization and containerization technologies are increasingly used in Network equipment, but are not considered in this revision of the document.

2. Solution Overview

2.1. RIV Software Configuration Attestation using TPM

RIV Attestation is a process which can be used to determine the identity of software running on a specifically-identified device. Remote Attestation is broken into two phases, shown in Figure 1:

- During system startup, each distinct software object is "measured". Its identity, hash (i.e., cryptographic digest) and version information are recorded in a log. Hashes are also extended, or cryptographically folded, into the TPM, in a way that can be used to validate the log entries. The measurement process generally follows the Chain of Trust model used in Measured Boot, where each stage of the system measures the next one, and extends its measurement into the TPM, before launching it.

- Once the device is running and has operational network connectivity, a separate, trusted Verifier will interrogate the network device to retrieve the logs and a copy of the digests collected by hashing each software object, signed by an attestation private key known only to the TPM.

The result is that the Verifier can verify the device's identity by checking the Subject Field and signature of certificate containing the TPM's attestation public key, and can validate the software that was launched by verifying the correctness of the logs by comparing with the signed digests from the TPM, and comparing digests in the log with known-good values.

It should be noted that attestation and identity are inextricably linked; signed Evidence that a particular version of software was loaded is of little value without cryptographic proof of the identity of the Attester producing the Evidence.

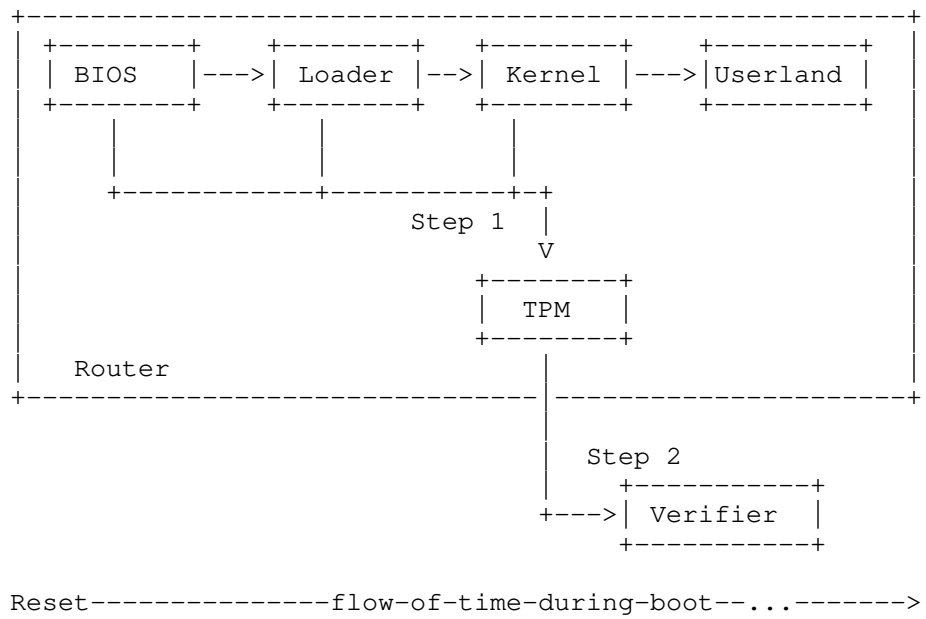


Figure 1: RIV Attestation Model

In Step 1, measurements are "extended", or hashed, into the TPM as processes start, with the result that the PCR ends up containing a hash of all the measured hashes. In Step 2, signed PCR digests are retrieved from the TPM for off-box analysis after the system is operational.

2.1.1. What Does RIV Attest?

TPM attestation is strongly focused on Platform Configuration Registers (PCRs), but those registers are only vehicles for certifying accompanying Evidence, conveyed in log entries. It is the hashes in log entries that are extended into PCRs, where the final PCR values can be retrieved in the form of a structured called a Quote, signed by an Attestation key known only to the TPM. The use of multiple PCRs serves only to provide some independence between different classes of object, so that one class of objects can be updated without changing the extended hash for other classes. Although PCRs can be used for any purpose, this section outlines the objects within the scope of this document which may be extended into the TPM.

In general, assignment of measurements to PCRs is a policy choice made by the device manufacturer, selected to independently attest three classes of object:

- Code, (i.e., instructions) to be executed by a CPU.

- Configuration – Many devices offer numerous options controlled by non-volatile configuration variables which can impact the device's security posture. These settings may have vendor defaults, but often can be changed by administrators, who may want to verify via attestation that the settings they intend are still in place.

- Credentials – Administrators may wish to verify via attestation that keys (and other credentials) outside the Root of Trust have not been subject to unauthorized tampering. (By definition, keys inside the root of trust can't be verified independently).

The TCG PC Client Platform Firmware Profile Specification [PC-Client-BIOS-TPM-2.0] gives considerable detail on what is to be measured during the boot phase of platform startup using a UEFI BOIS (www.uefi.org), but the goal is simply to measure every bit of code executed in the process of starting the device, along with any configuration information related to security posture, leaving no gap for unmeasured code to remain undetected and subvert the chain.

For devices using a UEFI BIOS, [PC-Client-BIOS-TPM-2.0] gives detailed normative requirements for PCR usage. But for other platform architectures, the table in Figure 2 gives guidance for PCR assignment that generalizes the specific details of [PC-Client-BIOS-TPM-2.0].

By convention, most PCRs are assigned in pairs, which the even-numbered PCR used to measure executable code, and the odd-numbered PCR used to measure whatever data and configuration are associated with that code. It is important to note that each PCR may contain results from dozens (or even thousands) of individual measurements.

Function	Assigned PCR # Code	Configuration
Firmware Static Root of Trust, (i.e., initial boot firmware and drivers)	0	1
Drivers and initialization for optional or add-in devices	2	3
OS Loader code and configuration, (i.e., the code launched by firmware) to load an operating system kernel. These PCRs record each boot attempt, and an identifier for where the loader was found	4	5
Vendor Specific Measurements during boot	6	6
Secure Boot Policy. This PCR records keys and configuration used to validate the OS loader		7
Measurements made by the OS Loader (e.g GRUB2 for Linux)	8	9
Measurements made by OS (e.g., Linux IMA)	10	10

Figure 2: Attested Objects

2.1.2. Notes on PCR Allocations

It is important to recognize that PCR[0] is critical. The first measurement into PCR[0] taken by the Root of Trust for Measurement, is critical to establishing the chain of trust for all subsequent measurements. If the PCR[0] measurement cannot be trusted, the validity of the entire chain is put into question.

Distinctions Between PCR[0], PCR[2], PCR[4] and PCR[8] are summarized below:

- PCR[0] typically represents a consistent view of the Host Platform between boot cycles, allowing Attestation and Sealed Storage policies to be defined using the less changeable components of the transitive trust chain. This PCR typically provides a consistent view of the platform regardless of user selected options.

- PCR[2] is intended to represent a "user configurable" environment where the user has the ability to alter the components that are measured into PCR[2]. This is typically done by adding adapter cards, etc., into user-accessible PCI or other slots. In UEFI systems these devices may be configured by Option ROMs measured into PCR[2] and executed by the BIOS.

- PCR[4] is intended to represent the software that manages the transition between the platform's Pre-Operating System Start and the state of a system with the Operating System present. This PCR, along with PCR[5], identifies the initial operating system loader (e.g., GRUB for Linux).

- PCR[8] is used by the OS loader to record measurements of the various components of the operating system.

Although the TCG PC Client document specifies the use of the first eight PCRs very carefully to ensure interoperability among multiple UEFI BIOS vendors, it should be noted that embedded software vendors may have considerably more flexibility. Verifiers typically need to know which log entries are consequential and which are not (possibly controlled by local policies) but the Verifier may not need to know what each log entry means or why it was assigned to a particular PCR. Designers must recognize that some PCRs may cover log entries that a particular Verifier considers critical and other log entries that are not considered important, so differing PCR values may not on their own constitute a check for authenticity.

Designers may allocate particular events to specific PCRs in order to achieve a particular objective with Local Attestation, (e.g., allowing a procedure to execute only if a given PCR is in a given state). It may also be important to designers to consider whether streaming notification of PCR updates is required (see [I-D.birkholz-rats-network-device-subscription]). Specific log entries can only be validated if the Verifier receives every log entry affecting the relevant PCR, so (for example) a designer might want to separate rare, high-value events such as configuration changes, from high-volume, routine measurements such as IMA [IMA] logs.

2.2. RIV Keying

RIV attestation relies on two keys:

- An identity key is required to certify the identity of the Attester itself. RIV specifies the use of an IEEE 802.1AR Device Identity (DevID) [IEEE-802-1AR], signed by the device manufacturer, containing the device serial number.

- An Attestation Key is required to sign the Quote generated by the TPM to report evidence of software configuration.

In TPM application, the Attestation key MUST be protected by the TPM, and the DevID SHOULD be as well. Depending on other TPM configuration procedures, the two keys are likely be different; some of the considerations are outlined in TCG Guidance for Securing Network Equipment [NetEq].

TCG Guidance for Securing Network Equipment specifies further conventions for these keys:

- When separate Identity and Attestation keys are used, the Attestation Key (AK) and its X.509 certificate should parallel the DevID, with the same device ID information as the DevID certificate (i.e., the same Subject Name and Subject Alt Name, even though the key pairs are different). This allows a quote from the device, signed by an AK, to be linked directly to the device that provided it, by examining the corresponding AK certificate.

- Network devices that are expected to use secure zero touch provisioning as specified in [RFC8572]) MUST be shipped by the manufacturer with pre-provisioned keys (Initial DevID and AK, called IDevID and IAK). Inclusion of an IDevID and IAK by a vendor does not preclude a mechanism whereby an Administrator can define Local Identity and Attestation Keys (LDevID and LAK) if desired. IDevID and IAK certificates MUST both be signed by the Endorser (typically the device manufacturer).

2.3. RIV Information Flow

RIV workflow for networking equipment is organized around a simple use case where a network operator wishes to verify the integrity of software installed in specific, fielded devices. This use case implies several components:

1. The Attesting Device, which the network operator wants to examine.

2. A Verifier (which might be a network management station) somewhere separate from the Device that will retrieve the information and analyze it to pass judgment on the security posture of the device.

3. A Relying Party, which can act on Attestation Results. Interaction between the Relying Party and the Verifier is considered out of scope for RIV.

4. Signed Reference Integrity Manifests (RIMs), containing Reference Integrity Measurements, can either be created by the device manufacturer and shipped along with the device as part of its software image, or alternatively, could be obtained several other ways (direct to the Verifier from the manufacturer, from a third party, from the owner's observation of what's thought to be a "known good system", etc.). Retrieving RIMs from the device itself allows attestation to be done in systems that may not have access to the public internet, or by other devices that are not management stations per se (e.g., a peer device; see Section 3.1.3). If Reference Integrity Measurements are obtained from multiple sources, the Verifier may need to evaluate the relative level of trust to be placed in each source in case of a discrepancy.

These components are illustrated in Figure 3.

A more-detailed taxonomy of terms is given in [I-D.ietf-rats-architecture]

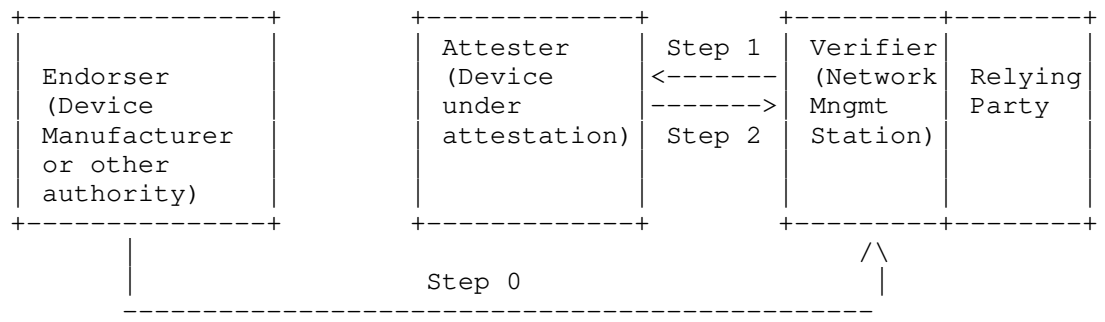


Figure 3: RIV Reference Configuration for Network Equipment

In Step 0, The Endorser (the device manufacturer or other authority) provides a software image to the Attester (the device under attestation), and makes one or more Reference Integrity Manifests (RIMs) signed by the Endorser, available to the Verifier (see Section 3.1.3 for "in-band" and "out of band" ways to make this happen). In Step 1, the Verifier (Network Management Station), on behalf of a Relying Party, requests Identity, Measurement Values, and possibly RIMs, from the Attester. In Step 2, the Attester responds to the request by providing a DevID, quotes (measured values, signed by the Attester), and optionally RIMs.

To achieve interoperability, the following standards components SHOULD be used:

1. TPM Keys MUST be configured according to [Platform-DevID-TPM-2.0], [PC-Client-BIOS-TPM-1.2], or [Platform-ID-TPM-1.2].

2. For devices using UEFI and Linux, measurements of firmware and bootable modules SHOULD be taken according to TCG PC Client [PC-Client-BIOS-TPM-2.0] and Linux IMA [IMA]

3. Device Identity MUST be managed as specified in IEEE 802.1AR Device Identity certificates [IEEE-802-1AR], with keys protected by TPMs.

4. Attestation logs SHOULD be formatted according to the Canonical Event Log format [Canonical-Event-Log], although other specialized formats may be used. UEFI-based systems MAY use the TCG UEFI BIOS event log [EFI-TPM]).

5. Quotes are retrieved from the TPM according to the TCG TAP Information Model [TAP]. While the TAP IM gives a protocol-independent description of the data elements involved, it's important to note that quotes from the TPM are signed inside the TPM, so MUST be retrieved in a way that does not invalidate the signature, as specified in [I-D.ietf-rats-yang-tpm-charra], to preserve the trust model. (See Section 5 Security Considerations).

6. Reference Integrity Measurements SHOULD be encoded as CoSWID tags, as defined in the TCG RIM document [RIM], compatible with NIST IR 8060 [NIST-IR-8060] and the IETF CoSWID draft [I-D.ietf-sacm-coswid]. See Section 2.4.1.

2.4. RIV Simplifying Assumptions

This document makes the following simplifying assumptions to reduce complexity:

- The product to be attested MUST be shipped with an IEEE 802.1AR Device Identity and an Initial Attestation Key (IAK) with certificate. The IAK cert contains the same identity information as the DevID (specifically, the same Subject Name and Subject Alt Name, signed by the manufacturer), but it's a type of key that can be used to sign a TPM Quote. This convention is described in TCG Guidance for Securing Network Equipment [NetEq]. For network equipment, which is generally non-privacy-sensitive, shipping a device with both an IDevID and an IAK already provisioned substantially simplifies initial startup. Privacy-sensitive applications may use the TCG Platform Certificate and additional

procedures to install identity credentials into the device after manufacture.

- The product MUST be equipped with a Root of Trust for Measurement, Root of Trust for Storage and Root of Trust for Reporting (as defined in [SP800-155]) that are capable of conforming to the TCG Trusted Attestation Protocol (TAP) Information Model [TAP].

- The authorized software supplier MUST make available Reference Integrity Measurements (i.e., known-good measurements) in the form of signed CoSWID tags [I-D.ietf-sacm-coswid], [SWID], as described in TCG Reference Integrity Measurement Manifest Information Model [RIM].

2.4.1. Reference Integrity Manifests (RIMs)

[I-D.ietf-rats-yang-tpm-charra] focuses on collecting and transmitting evidence in the form of PCR measurements and attestation logs. But the critical part of the process is enabling the Verifier to decide whether the measurements are "the right ones" or not.

While it must be up to network administrators to decide what they want on their networks, the software supplier should supply the Reference Integrity Measurements that may be used by a Verifier to determine if evidence shows known good, known bad or unknown software configurations.

In general, there are two kinds of reference measurements:

1. Measurements of early system startup (e.g., BIOS, boot loader, OS kernel) are essentially single-threaded, and executed exactly once, in a known sequence, before any results could be reported. In this case, while the method for computing the hash and extending relevant PCRs may be complicated, the net result is that the software (more likely, firmware) vendor will have one known good PCR value that "should" be present in the relevant PCRs after the box has booted. In this case, the signed reference measurement could simply list the expected hashes for the given version. However, a RIM that contains the intermediate hashes can be useful in debugging cases where the expected final hash is not the one reported.

2. Measurements taken later in operation of the system, once an OS has started (for example, Linux IMA[IMA]), may be more complex, with unpredictable "final" PCR values. In this case, the Verifier must have enough information to reconstruct the expected PCR values from logs and signed reference measurements from a trusted authority.

In both cases, the expected values can be expressed as signed SWID or CoSWID tags, but the SWID structure in the second case is somewhat more complex, as reconstruction of the extended hash in a PCR may involve thousands of files and other objects.

The TCG has published an information model defining elements of reference integrity manifests under the title TCG Reference Integrity Manifest Information Model [RIM]. This information model outlines how SWID tags should be structured to allow attestation, and defines "bundles" of SWID tags that may be needed to describe a complete software release. The RIM contains metadata relating to the software release it belongs to, plus hashes for each individual file or other object that could be attested.

TCG has also published the PC Client Reference Integrity Measurement specification [PC-Client-RIM], which focuses on a SWID-compatible format suitable for expressing expected measurement values in the specific case of a UEFI-compatible BIOS, where the SWID focus on files and file systems is not a direct fit. While the PC Client RIM is not directly applicable to network equipment, many vendors do use a conventional UEFI BIOS to launch their network OS.

2.4.2. Attestation Logs

Quotes from a TPM can provide evidence of the state of a device up to the time the evidence was recorded, but to make sense of the quote in most cases an event log that identifies which software modules contributed which values to the quote during startup MUST also be provided. The log MUST contain enough information to demonstrate its integrity by allowing exact reconstruction of the digest conveyed in the signed quote (i.e., PCR values).

There are multiple event log formats which may be supported as viable formats of Evidence between the Attester and Verifier:

- Event log exports from [I-D.ietf-rats-yang-tpm-charra]
- IMA Event log file exports [IMA]
- TCG UEFI BIOS event log (TCG EFI Platform Specification for TPM Family 1.1 or 1.2, Section 7) [EFI-TPM])
- TCG Canonical Event Log [Canonical-Event-Log]

Devices which use UEFI BIOS and Linux SHOULD use TCG Canonical Event Log [Canonical-Event-Log] and TCG UEFI BIOS event log [EFI-TPM])

3. Standards Components

3.1. Prerequisites for RIV

The Reference Interaction Model for Challenge-Response-based Remote Attestation is based on the standard roles defined in [I-D.ietf-rats-architecture]. However additional prerequisites have been established to allow for interoperable RIV use case implementations. These prerequisites are intended to provide sufficient context information so that the Verifier can acquire and evaluate Attester measurements.

3.1.1. Unique Device Identity

A secure Device Identity (DevID) in the form of an IEEE 802.1AR DevID certificate [IEEE-802-1AR] MUST be provisioned in the Attester's TPMs.

3.1.2. Keys

The Attestation Identity Key (AIK) and certificate MUST also be provisioned on the Attester according to [Platform-DevID-TPM-2.0], [PC-Client-BIOS-TPM-1.2], or [Platform-ID-TPM-1.2].

The Attester's TPM Keys MUST be associated with the DevID on the Verifier (see [Platform-DevID-TPM-2.0] and Section 5 Security Considerations, below).

3.1.3. Appraisal Policy for Evidence

The Verifier MUST obtain trustworthy Endorsements in the form of reference measurements (e.g., Known Good Values, encoded as CoSWID tags [I-D.birkholz-yang-swid]). These reference measurements will eventually be compared to signed PCR Evidence acquired from an Attester's TPM using Attestation Policies chosen by the administrator or owner of the device.

This document does not specify the format or contents for the Appraisal Policy for Evidence, but Endorsements may be acquired in one of two ways:

1. a Verifier may obtain reference measurements directly from an Endorser chosen by the Verifier administrator (e.g., through a web site).

2. Signed reference measurements may be distributed by the Endorser to the Attester, as part of a software update. From there, the reference measurement may be acquired by the Verifier.

In either case, the Verifier Owner MUST select the source of trusted endorsements through the Appraisal Policy for Evidence.

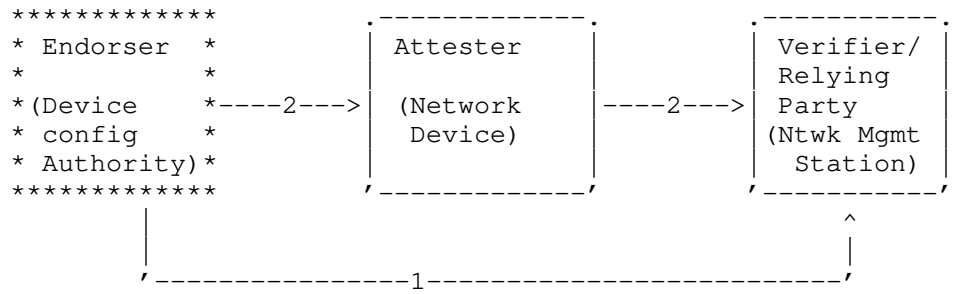


Figure 4: Appraisal Policy for Evidence Prerequisites

In either case the Endorsements must be generated, acquired and delivered in a secure way, including reference measurements of firmware and bootable modules taken according to TCG PC Client [PC-Client-BIOS-TPM-2.0] and Linux IMA [IMA]. Endorsementa MUST be encoded as SWID or CoSWID tags, signed by the device manufacturer, as defined in the TCG RIM document [RIM], compatible with NIST IR 8060 [NIST-IR-8060] or the IETF CoSWID draft [I-D.ietf-sacm-coswid].

3.2. Reference Model for Challenge-Response

Once the prerequisites for RIV are met, a Verifier is able to acquire Evidence from an Attester. The following diagram illustrates a RIV information flow between a Verifier and an Attester, derived from Section 8.1 of [I-D.birkholz-rats-reference-interaction-model]. Event times shown correspond to the time types described within Appendix A of [I-D.ietf-rats-architecture]:

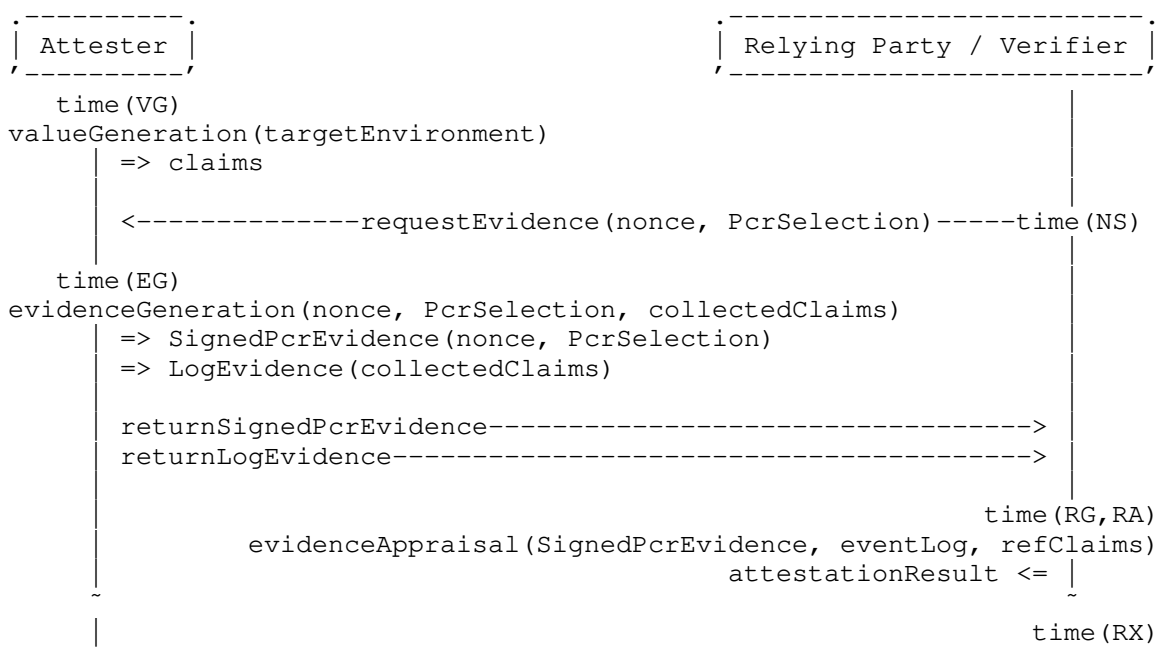


Figure 5: IETF Attestation Information Flow

- Step 1 (time(VG)): One or more Attesting Network Device PCRs are extended with measurements. RIV provides no direct link between the time at which the event takes place and the time that it's attested, although streaming attestation as in [I-D.birkholz-rats-network-device-subscription] could.

- Step 2 (time(NS)): The Verifier generates a unique nonce ("number used once"), and makes a request attestation data for one or more PCRs from an Attester. For interoperability, this MUST be accomplished via a YANG [RFC7950] interface that implements the TCG TAP model (e.g., YANG Module for Basic Challenge-Response-based Remote Attestation Procedures [I-D.ietf-rats-yang-tpm-charra]).

- Step 3 (time(EG)): On the Attester, measured values are retrieved from the Attester's TPM. This requested PCR evidence is signed by the Attestation Identity Key (AIK) associated with the DevID. Quotes are retrieved according to TCG TAP Information Model [TAP]. While the TAP IM gives a protocol-independent description of the data elements involved, it's important to note that quotes from the TPM are signed inside the TPM, so must be retrieved in a way that does not invalidate the signature, as specified in [I-D.ietf-rats-yang-tpm-charra], to preserve the trust model.

(See Section 5 Security Considerations). At the same time, the Attester collects log evidence showing what values have been extended into that PCR.

- Step 4: Collected Evidence is passed from the Attester to the Verifier

- Step 5 (time(RG,RA)): The Verifier reviews the Evidence and takes action as needed. As the interaction between Relying Party and Verifier is out of scope for RIV, this can happen in one step.

  * If the signed PCR values do not match the set of log entries which have extended a particular PCR, the device SHOULD NOT be trusted.

  * If the log entries that the Verifier considers important do not match known good values, the device SHOULD NOT be trusted. We note that the process of collecting and analyzing the log can be omitted if the value in the relevant PCR is already a known-good value.

  * If the set of log entries are not seen as acceptable by the Appraisal Policy for Evidence, the device SHOULD NOT be trusted.

  * If the AIK signature is not correct, or freshness such as that provided by the nonce is not included in the response, the device SHOULD NOT be trusted.

  * If time(RG)-time(NS) is greater than the threshold in the Appraisal Policy for Evidence, the Evidence is considered stale and SHOULD NOT be trusted.

3.2.1. Transport and Encoding

Network Management systems may retrieve signed PCR based Evidence as shown in Figure 5, and can be accomplished via NETCONF or RESTCONF, with XML, JSON, or CBOR encoded Evidence.

Implementations that use NETCONF MUST do so over a TLS or SSH secure tunnel. Implementations that use RESTCONF transport MAY do so over a TLS or SSH secure tunnel.

Retrieval of Log Evidence SHOULD be via log interfaces on the network device. (For example, see [I-D.ietf-rats-yang-tpm-charra]).

3.3. Centralized vs Peer-to-Peer

Figure 5 above assumes that the Verifier is implicitly trusted, while the Attesting device is not. In a Peer-to-Peer application such as two routers negotiating a trust relationship [I-D.voit-rats-trusted-path-routing], the two peers can each ask the other to prove software integrity. In this application, the information flow is the same, but each side plays a role both as an Attester and a Verifier. Each device issues a challenge, and each device responds to the other's challenge, as shown in Figure 6. Peer-to-peer challenges, particularly if used to establish a trust relationship between routers, require devices to carry their own signed reference measurements (RIMs). Devices may also have to carry Appraisal Policy for Evidence for each possible peer device so that each device has everything needed for attestation, without having to resort to a central authority.

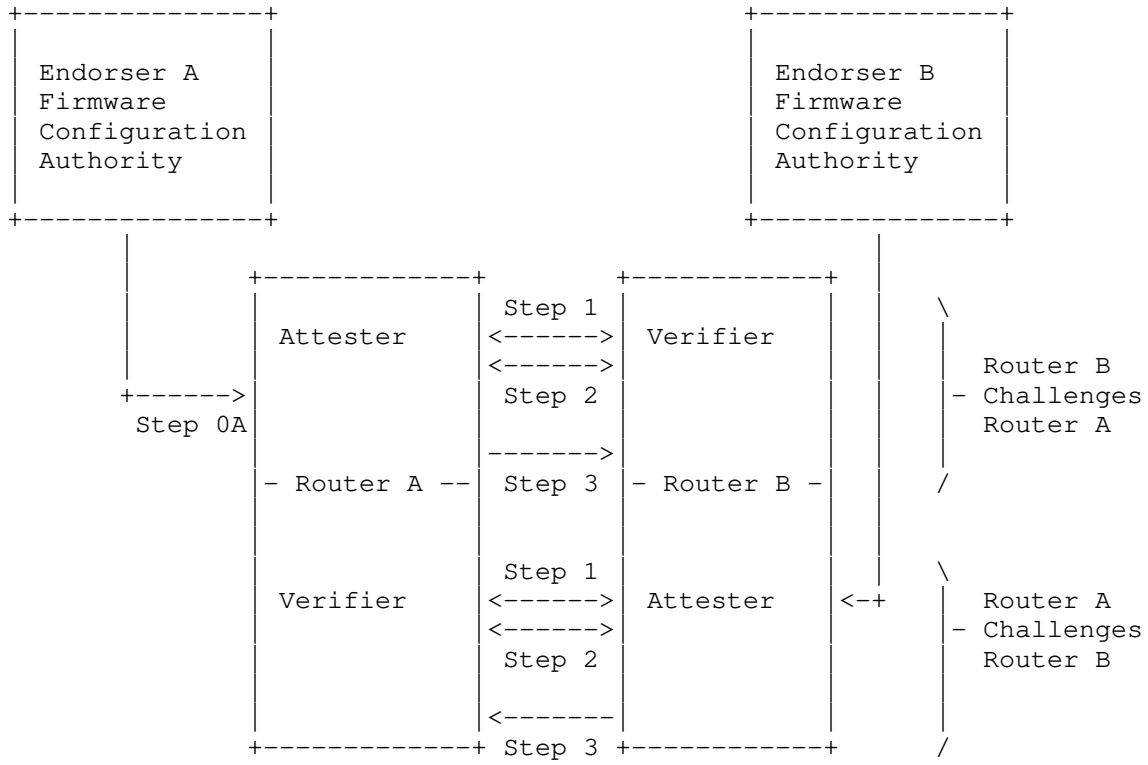


Figure 6: Peer-to-Peer Attestation Information Flow

In this application, each device may need to be equipped with signed RIMs to act as an Attester, and also an Appraisal Policy for Evidence and a selection of trusted X.509 root certificates, to allow the device to act as a Verifier. An existing link layer protocol such as 802.1x [IEEE-802.1x] or 802.1AE [IEEE-802.1ae], with Evidence being enclosed over a variant of EAP [RFC3748] or LLDP [LLDP] are suitable methods for such an exchange.

4. Privacy Considerations

Networking Equipment, such as routers, switches and firewalls, has a key role to play in guarding the privacy of individuals using the network:

- Packets passing through the device must not be sent to unauthorized destinations. For example:
  - Routers often act as Policy Enforcement Points, where individual subscribers may be checked for authorization to access a network. Subscriber login information must not be released to unauthorized parties.
  - Networking Equipment is often called upon to block access to protected resources from unauthorized users.
- Routing information, such as the identity of a router's peers, must not be leaked to unauthorized neighbors.
- If configured, encryption and decryption of traffic must be carried out reliably, while protecting keys and credentials.

Functions that protect privacy are implemented as part of each layer of hardware and software that makes up the networking device. In light of these requirements for protecting the privacy of users of the network, the Network Equipment must identify itself, and its boot configuration and measured device state (for example, PCR values), to the Equipment's Administrator, so there's no uncertainty as to what function each device and configuration is configured to carry out. This allows the administrator to ensure that the network provides individual and peer privacy guarantees.

RIV specifically addresses the collection of information from enterprise network devices by authorized agents of the enterprise. As such, privacy is a fundamental concern for those deploying this solution, given EU GDPR, California CCPA, and many other privacy regulations. The enterprise SHOULD implement and enforce their duty of care.

See [NetEq] for more context on privacy in networking devices.

5. Security Considerations

Attestation Results from the RIV procedure are subject to a number of attacks:

- Keys may be compromised.
- A counterfeit device may attempt to impersonate (spoof) a known authentic device.
- Man-in-the-middle attacks may be used by a counterfeit device to attempt to deliver responses that originate in an actual authentic device.
- Replay attacks may be attempted by a compromised device.

5.1. Keys Used in RIV

Trustworthiness of RIV attestation depends strongly on the validity of keys used for identity and attestation reports. RIV takes full advantage of TPM capabilities to ensure that results can be trusted.

Two sets of keys are relevant to RIV attestation:

- A DevID key is used to certify the identity of the device in which the TPM is installed.
- An Attestation Key (AK) key signs attestation reports (called 'quotes' in TCG documents), used to provide evidence for integrity of the software on the device.

TPM practices usually require that these keys be different, as a way of ensuring that a general-purpose signing key cannot be used to spoof an attestation quote.

In each case, the private half of the key is known only to the TPM, and cannot be retrieved externally, even by a trusted party. To ensure that's the case, specification-compliant private/public key-pairs are generated inside the TPM, where they're never exposed, and cannot be extracted (See [Platform-DevID-TPM-2.0]).

Keeping keys safe is just part of attestation security; knowing which keys are bound to the device in question is just as important.

While there are many ways to manage keys in a TPM (see [Platform-DevID-TPM-2.0]), RIV includes support for "zero touch"

provisioning (also known as zero-touch onboarding) of fielded devices (e.g., Secure ZTP, [RFC8572]), where keys which have predictable trust properties are provisioned by the device vendor.

Device identity in RIV is based on IEEE 802.1AR Device Identity (DevID). This specification provides several elements:

- A DevID requires a unique key pair for each device, accompanied by an X.509 certificate,
- The private portion of the DevID key is to be stored in the device, in a manner that provides confidentiality (Section 6.2.5 [IEEE-802-1AR])

The X.509 certificate contains several components:

- The public part of the unique DevID key assigned to that device allows a challenge of identity.
- An identifying string that's unique to the manufacturer of the device. This is normally the serial number of the unit, which might also be printed on a label on the device.
- The certificate must be signed by a key traceable to the manufacturer's root key.

With these elements, the device's manufacturer and serial number can be identified by analyzing the DevID certificate plus the chain of intermediate certificates leading back to the manufacturer's root certificate. As is conventional in TLS or SSH connections, a nonce must be signed by the device in response to a challenge, proving possession of its DevID private key.

RIV uses the DevID to validate a TLS or SSH connection to the device as the attestation session begins. Security of this process derives from TLS or SSH security, with the DevID providing proof that the session terminates on the intended device. See [RFC8446], [RFC4253].

Evidence of software integrity is delivered in the form of a quote signed by the TPM itself. Because the contents of the quote are signed inside the TPM, any external modification (including reformatting to a different data format) after measurements have been taken will be detected as tampering. An unbroken chain of trust is essential to ensuring that blocks of code that are taking measurements have been verified before execution (see Figure 1.

Requiring results of attestation of the operating software to be signed by a key known only to the TPM also removes the need to trust

the device's operating software (beyond the first measurement; see below); any changes to the quote, generated and signed by the TPM itself, made by malicious device software, or in the path back to the Verifier, will invalidate the signature on the quote.

A critical feature of the YANG model described in [I-D.ietf-rats-yang-tpm-charra] is the ability to carry TPM data structures in their native format, without requiring any changes to the structures as they were signed and delivered by the TPM. While alternate methods of conveying TPM quotes could compress out redundant information, or add an additional layer of signing using external keys, the implementation MUST preserve the TPM signing, so that tampering anywhere in the path between the TPM itself and the Verifier can be detected.

5.2. Prevention of Spoofing and Man-in-the-Middle Attacks

Prevention of spoofing attacks against attestation systems is also important. There are two cases to consider:

- The entire device could be spoofed, that is, when the Verifier goes to verify a specific device, it might be redirected to a different device. Use of the 802.1AR Device Identity (DevID) in the TPM ensures that the Verifier's TLS or SSH session is in fact terminating on the right device.

- A compromised device could respond with a spoofed Attestation Result, that is, a compromised OS could return a fabricated quote.

Protection against spoofed quotes from a device with valid identity is a bit more complex. An identity key must be available to sign any kind of nonce or hash offered by the Verifier, and consequently, could be used to sign a fabricated quote. To block a spoofed Attestation Result, the quote generated inside the TPM must be signed by a key that's different from the DevID, called an Attestation Key (AK).

Given separate Attestation and DevID keys, the binding between the AK and the same device must also be proven to prevent a man-in-the-middle attack (e.g., the 'Asokan Attack' [RFC6813]).

This is accomplished in RIV through use of an AK certificate with the same elements as the DevID (i.e., same manufacturer's serial number, signed by the same manufacturer's key), but containing the device's unique AK public key instead of the DevID public key.

[Editor's Note: does this require an OID that says the key is known by the CA to be an Attestation key?]

These two keys and certificates are used together:

- The DevID is used to validate a TLS connection terminating on the device with a known serial number.

- The AK is used to sign attestation quotes, providing proof that the attestation evidence comes from the same device.

5.3. Replay Attacks

Replay attacks, where results of a previous attestation are submitted in response to subsequent requests, are usually prevented by inclusion of a nonce in the request to the TPM for a quote. Each request from the Verifier includes a new random number (a nonce). The resulting quote signed by the TPM contains the same nonce, allowing the Verifier to determine freshness, (i.e., that the resulting quote was generated in response to the Verifier's specific request). Time-Based Uni-directional Attestation [I-D.birkholz-rats-tuda] provides an alternate mechanism to verify freshness without requiring a request/response cycle.

5.4. Owner-Signed Keys

Although device manufacturers MUST pre-provision devices with easily verified DevID and AK certificates, use of those credentials is not mandatory. IEEE 802.1AR incorporates the idea of an Initial Device ID (IDevID), provisioned by the manufacturer, and a Local Device ID (LDevID) provisioned by the owner of the device. RIV and [Platform-DevID-TPM-2.0] extends that concept by defining an Initial Attestation Key (IAK) and Local Attestation Key (LAK) with the same properties.

Device owners can use any method to provision the Local credentials.

- The TCG document [Platform-DevID-TPM-2.0] shows how the initial Attestation keys can be used to certify LDevID and LAK keys. Use of the LDevID and LAK allows the device owner to use a uniform identity structure across device types from multiple manufacturers (in the same way that an "Asset Tag" is used by many enterprises to identify devices they own). The TCG document [Provisioning-TPM-2.0] also contains guidance on provisioning identity keys in TPM 2.0.

- Device owners, however, can use any other mechanism they want to assure themselves that Local identity certificates are inserted into the intended device, including physical inspection and programming in a secure location, if they prefer to avoid placing trust in the manufacturer-provided keys.

Clearly, Local keys can't be used for secure Zero Touch provisioning; installation of the Local keys can only be done by some process that runs before the device is installed for network operation.

On the other end of the device life cycle, provision should be made to wipe Local keys when a device is decommissioned, to indicate that the device is no longer owned by the enterprise. The manufacturer's Initial identity keys must be preserved, as they contain no information that's not already printed on the device's serial number plate.

5.5. Other Trust Anchors

In addition to trustworthy provisioning of keys, RIV depends on other trust anchors. (See [SP800-155] for definitions of Roots of Trust.)

- Secure identity depends on mechanisms to prevent per-device secret keys from being compromised. The TPM provides this capability as a Root of Trust for Storage.

- Attestation depends on an unbroken chain of measurements, starting from the very first measurement. That first measurement is made by code called the Root of Trust for Measurement, typically done by trusted firmware stored in boot flash. Mechanisms for maintaining the trustworthiness of the RTM are out of scope for RIV, but could include immutable firmware, signed updates, or a vendor-specific hardware verification technique. See Section 8.1 for background on TPM practices.

- The device owner SHOULD provide some level of physical defense for the device. If a TPM that has already been programmed with an authentic DevID is stolen and inserted into a counterfeit device, attestation of that counterfeit device may become indistinguishable from an authentic device.

RIV also depends on reliable reference measurements, as expressed by the RIM [RIM]. The definition of trust procedures for RIMs is out of scope for RIV, and the device owner is free to use any policy to validate a set of reference measurements. RIMs may be conveyed out-of-band or in-band, as part of the attestation process (see Section 3.1.3). But for embedded devices, where software is usually shipped as a self-contained package, RIMs signed by the manufacturer and delivered in-band may be more convenient for the device owner.

The validity of RIV attestation results is also influenced by procedures used to create reference measurements:

- While the RIM itself is signed, supply-chains SHOULD be carefully scrutinized to ensure that the values are not subject to unexpected manipulation prior to signing. Insider-attacks against code bases and build chains are particularly hard to spot.

- Designers SHOULD guard against hash collision attacks. Reference Integrity Measurements often give hashes for large objects of indeterminate size; if one of the measured objects can be replaced with an implant engineered to produce the same hash, RIV will be unable to detect the substitution. TPM1.2 uses SHA-1 hashes only, which have been shown to be susceptible to collision attack. TPM2.0 will produce quotes with SHA-256, which so far has resisted such attacks. Consequently RIV implementations SHOULD use TPM2.0.

6. Conclusion

TCG technologies can play an important part in the implementation of Remote Integrity Verification. Standards for many of the components needed for implementation of RIV already exist:

- Platform identity can be based on IEEE 802.1AR Device Identity, coupled with careful supply-chain management by the manufacturer.

- Complex supply chains can be certified using TCG Platform Certificates [Platform-Certificates].

- The TCG TAP mechanism can be used to retrieve attestation evidence. Work is needed on a YANG model for this protocol.

- Reference Integrity Measurements must be conveyed from the software authority (e.g., the manufacturer) to the system in which verification will take place. IETF CoSWID work forms the basis for this, but new work is needed to create an information model and YANG implementation.

7. IANA Considerations

This memo includes no request to IANA.

8. Appendix

8.1. Using a TPM for Attestation

The Trusted Platform Module and surrounding ecosystem provide three interlocking capabilities to enable secure collection of evidence from a remote device, Platform Configuration Registers (PCRs), a Quote mechanism, and a standardized Event Log.

Each TPM has at least between eight and twenty-four PCRs (depending on the profile and vendor choices), each one large enough to hold one hash value (SHA-1, SHA-256, and other hash algorithms can be used, depending on TPM version). PCRs can't be accessed directly from outside the chip, but the TPM interface provides a way to "extend" a new security measurement hash into any PCR, a process by which the existing value in the PCR is hashed with the new security measurement hash, and the result placed back into the same PCR. The result is a composite fingerprint of all the security measurements extended into each PCR since the system was reset.

Every time a PCR is extended, an entry should be added to the corresponding Event Log. Logs contain the security measurement hash plus informative fields offering hints as to what event it was that generated the security measurement.
The Event Log itself is protected against accidental manipulation, but it is implicitly tamper-evident - any verification process can read the security measurement hash from the log events, compute the composite value and compare that to what ended up in the PCR. If there's a discrepancy, the logs do not provide an accurate view of what was placed into the PCR.

In a conventional TPM Attestation environment, the first measurement must be made and extended into the TPM by trusted device code (called the Root of Trust for Measurement, RTM). That first measurement should cover the segment of code that is run immediately after the RTM, which then measures the next code segment before running it, and so on, forming an unbroken chain of trust. See [TCGRoT] for more on Mutable vs Immutable roots of trust.

The TPM provides another mechanism called a Quote that can read the current value of the PCRs and package them into a data structure signed by an Attestation Key (which is private key that is known only to the TPM).

The Verifier uses the Quote and Log together. The Quote, containing the composite hash of the complete sequence of security measurement hashes, is used to verify the integrity of the Event Log. Each hash in the validated Quote can then be compared to corresponding expected values in the set of Reference Integrity Measurements to validate overall system integrity.

A summary of information exchanged in obtaining quotes from TPM1.2 and TPM2.0 can be found in [TAP], Section 4. Detailed information about PCRs and Quote data structures can be found in [TPM1.2], [TPM2.0]. Recommended log formats include [PC-Client-BIOS-TPM-2.0] and [Canonical-Event-Log].

8.2. Root of Trust for Measurement

The measurements needed for attestation require that the device being attested is equipped with a Root of Trust for Measurement, that is, some trustworthy mechanism that can compute the first measurement in the chain of trust required to attest that each stage of system startup is verified, a Root of Trust for Storage (i.e., the TPM PCRs) to record the results, and a Root of Trust for Reporting to report the results [TCGRoT], [SP800-155].

While there are many complex aspects of a Root of Trust, two aspects that are important in the case of attestation are:

- The first measurement computed by the Root of Trust for Measurement, and stored in the TPM's Root of Trust for Storage, is presumed to be correct.

- There must not be a way to reset the Root of Trust for Storage without re-entering the Root of Trust for Measurement code.

The first measurement must be computed by code that is implicitly trusted; if that first measurement can be subverted, none of the remaining measurements can be trusted. (See [NIST-SP-800-155])

8.3. Layering Model for Network Equipment Attester and Verifier

Retrieval of identity and attestation state uses one protocol stack, while retrieval of Reference Measurements uses a different set of protocols. Figure 5 shows the components involved.

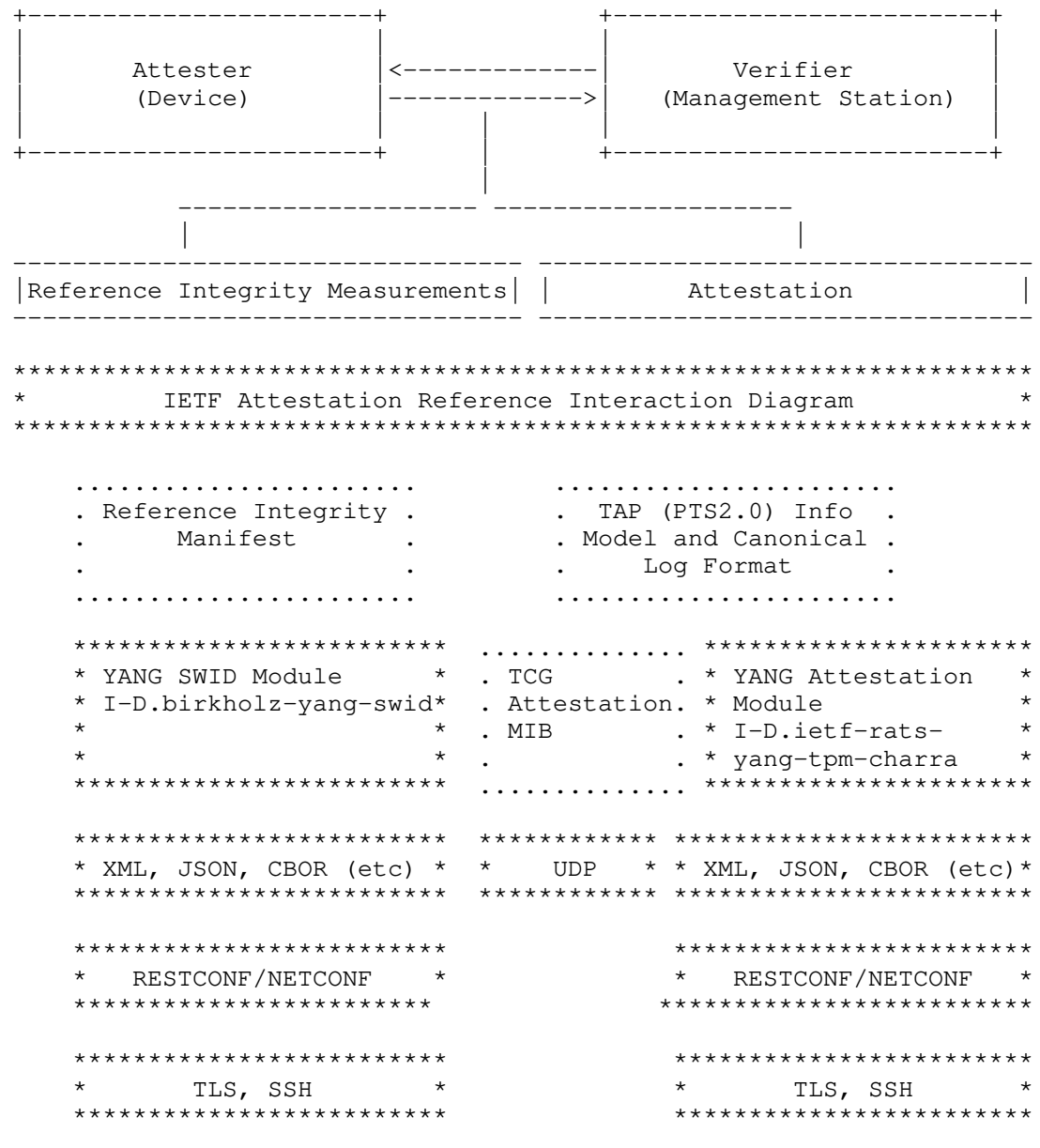


Figure 7: RIV Protocol Stacks

IETF documents are captured in boxes surrounded by asterisks. TCG documents are shown in boxes surrounded by dots.

8.3.1. Why is OS Attestation Different?

Even in embedded systems, adding Attestation at the OS level (e.g., Linux IMA, Integrity Measurement Architecture [IMA]) increases the number of objects to be attested by one or two orders of magnitude, involves software that's updated and changed frequently, and introduces processes that begin in an unpredictable order.

TCG and others (including the Linux community) are working on methods and procedures for attesting the operating system and application software, but standardization is still in process.

8.4. Implementation Notes

Figure 8 summarizes many of the actions needed to complete an Attestation system, with links to relevant documents. While documents are controlled by several standards organizations, the implied actions required for implementation are all the responsibility of the manufacturer of the device, unless otherwise noted. It should be noted that, while the YANG model is RECOMMENDED for attestation, this table identifies an optional SNMP MIB as well, [Attest-MIB].

Component	Controlling Specification
Make a Secure execution environment o Attestation depends on a secure root of trust for measurement outside the TPM, as well as roots for storage and reporting inside the TPM. o Refer to TCG Root of Trust for Measurement. o NIST SP 800-193 also provides guidelines on Roots of Trust	TCG RoT UEFI.org
Provision the TPM as described in TCG documents.	TCG TPM DevID TCG Platform Certificate
Put a DevID or Platform Cert in the TPM o Install an Initial Attestation Key at the same time so that Attestation can work out of the box o Equipment suppliers and owners may want to implement Local Device ID as well as Initial Device ID	TCG TPM DevID TCG Platform Certificate IEEE 802.1AR

Connect the TPM to the TLS stack o Use the DevID in the TPM to authenticate TAP connections, identifying the device	Vendor TLS stack (This action is simply configuring TLS to use the DevID as its trust anchor.)
Make CoSWID tags for BIOS/LoaderLKernel objects o Add reference measurements into SWID tags o Manufacturer should sign the SWID tags o The TCG RIM-IM identifies further procedures to create signed RIM documents that provide the necessary reference information	IETF CoSWID ISO/IEC 19770-2 NIST IR 8060
Package the SWID tags with a vendor software release o A tag-generator plugin such as [SWID-Gen] can be used	Retrieve tags with draft-birkholz-yang-swid TCG PC Client RIM
Use PC Client measurement definitions to define the use of PCRs (although Windows OS is rare on Networking Equipment, UEFI BIOS is not)	TCG PC Client BIOS
Use TAP to retrieve measurements o Map TAP to SNMP o Map to YANG Use Canonical Log Format	TCG SNMP MIB YANG Module for Basic Attestation TCG Canonical Log Format
Posture Collection Server (as described in IETF SACMs ECP) should request the attestation and analyze the result The Management application might be broken down to several more components: o A Posture Manager Server which collects reports and stores them in a database o One or more Analyzers that can look at the results and figure out what it means.	

Figure 8: Component Status

9. References

9.1. Normative References

[Canonical-Event-Log]
Trusted Computing Group, "DRAFT Canonical Event Log Format Version: 1.0, Revision: .12", October 2018.

[I-D.birkholz-yang-swid]
Birkholz, H., "Software Inventory YANG module based on Software Identifiers", draft-birkholz-yang-swid-02 (work in progress), October 2018.

[I-D.ietf-rats-yang-tpm-charra]
Birkholz, H., Eckel, M., Bhandari, S., Sulzen, B., Voit, E., Xia, L., Laffey, T., and G. Fedorkow, "A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs", draft-ietf-rats-yang-tpm-charra-02 (work in progress), June 2020.

[I-D.ietf-sacm-coswid]
Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", draft-ietf-sacm-coswid-15 (work in progress), May 2020.

[IEEE-802-1AR]
Seaman, M., "802.1AR-2018 - IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity, IEEE Computer Society", August 2018.

[PC-Client-BIOS-TPM-1.2]
Trusted Computing Group, "TCG PC Client Specific Implementation Specification for Conventional BIOS, Specification Version 1.21 Errata, Revision 1.00", February 2012, <https://trustedcomputinggroup.org/resource/pc-client-work-group-specific-implementation-specification-for-conventional-bios/>.

[PC-Client-BIOS-TPM-2.0]
Trusted Computing Group, "PC Client Specific Platform Firmware Profile Specification Family "2.0", Level 00 Revision 1.04", June 2019, <https://trustedcomputinggroup.org/pc-client-specific-platform-firmware-profile-specification>.

[PC-Client-RIM]
Trusted Computing Group, "DRAFT: TCG PC Client Reference Integrity Manifest Specification, v.09", December 2019, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_PC_Client_RIM_r0p15_15june2020.pdf>.

[Platform-DevID-TPM-2.0]
Trusted Computing Group, "DRAFT: TPM Keys for Platform DevID for TPM2, Specification Version 0.7, Revision 0", October 2018.

[Platform-ID-TPM-1.2]
Trusted Computing Group, "TPM Keys for Platform Identity for TPM 1.2, Specification Version 1.0, Revision 3", August 2015, <https://trustedcomputinggroup.org/resource/tpm-keys-for-platform-identity-for-tpm-1-2-2/>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[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>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

[RIM] Trusted Computing Group, "DRAFT: TCG Reference Integrity Manifest Information Model", June 2019, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIM_Model_v1-r13_2feb20.pdf>.

[SWID] The International Organization for Standardization/ International Electrotechnical Commission, "Information Technology Software Asset Management Part 2: Software Identification Tag, ISO/IEC 19770-2", October 2015, <https://www.iso.org/standard/65666.html>.

[TAP] Trusted Computing Group, "TCG Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0, Version 1.0, Revision 0.36", October 2018, <https://trustedcomputinggroup.org/resource/tcg-tap-information-model/>.

9.2. Informative References

[AIK-Enrollment]
Trusted Computing Group, "TCG Infrastructure Working Group - A CMC Profile for AIK Certificate Enrollment Version 1.0, Revision 7", March 2011, <https://trustedcomputinggroup.org/resource/tcg-infrastructure-working-group-a-cmc-profile-for-aik-certificate-enrollment/>.

[Attest-MIB]
Trusted Computing Group, "SNMP MIB for TPM-Based Attestation, Version 0.8Revision 0.02", May 2018, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_SNMP_MIB_for_TPM-Based_Attestation_v0.8r2_PUBLIC_REVIEW.pdf>.

[EFI-TPM] Trusted Computing Group, "TCG EFI Platform Specification for TPM Family 1.1 or 1.2, Specification Version 1.22, Revision 15", January 2014, <https://trustedcomputinggroup.org/resource/tcg-efi-platform-specification/>.

[I-D.birkholz-rats-network-device-subscription]
Birkholz, H., Voit, E., and W. Pan, "Attestation Event Stream Subscription", draft-birkholz-rats-network-device-subscription-00 (work in progress), June 2020.

[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.birkholz-rats-tuda]
Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", draft-birkholz-rats-tuda-03 (work in progress), July 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-eat]
Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", draft-ietf-rats-eat-04 (work in progress), August 2020.

[I-D.richardson-rats-usecases]
Richardson, M., Wallace, C., and W. Pan, "Use cases for Remote Attestation common encodings", draft-richardson-rats-usecases-07 (work in progress), March 2020.

[I-D.voit-rats-trusted-path-routing]
Voit, E., "Trusted Path Routing", draft-voit-rats-trusted-path-routing-02 (work in progress), June 2020.

[IEEE-802.1ae]
Seaman, M., "802.1AE MAC Security (MACsec)", 2018, <https://1.ieee802.org/security/802-1ae/>.

[IEEE-802.1x]
IEEE Computer Society, "802.1X-2020 - IEEE Standard for Local and Metropolitan Area Networks--Port-Based Network Access Control", February 2020, <https://standards.ieee.org/standard/802_1X-2020.html>.

[IMA] and , "Integrity Measurement Architecture", June 2019, <https://sourceforge.net/p/linux-ima/wiki/Home/>.

[LLDP] IEEE Computer Society, "802.1AB-2016 - IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery", March 2016, <https://standards.ieee.org/standard/802_1AB-2016.html>.

[NetEq] Trusted Computing Group, "TCG Guidance for Securing Network Equipment, Version 1.0, Revision 29", January 2018, <https://trustedcomputinggroup.org/resource/tcg-guidance-securing-network-equipment/>.

[NIST-IR-8060]
National Institute for Standards and Technology, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", April 2016, <https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8060.pdf>.

[NIST-SP-800-155]
National Institute for Standards and Technology, "BIOS Integrity Measurement Guidelines (Draft)", December 2011, <https://csrc.nist.gov/csrc/media/publications/sp/800-155/draft/documents/draft-sp800-155_dec2011.pdf>.

[Platform-Certificates]
Trusted Computing Group, "TCG Platform Attribute Credential Profile, Specification Version 1.0, Revision 16", January 2018, <https://trustedcomputinggroup.org/resource/tcg-platform-attribute-credential-profile/>.

[Provisioning-TPM-2.0]
Trusted Computing Group, "TCG TPM v2.0 Provisioning Guidance, Version 1.0, Revision 1.0", March 2015, <https://trustedcomputinggroup.org/wp-content/uploads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC6813] Salowey, J. and S. Hanna, "The Network Endpoint Assessment (NEA) Asokan Attack Analysis", RFC 6813, DOI 10.17487/RFC6813, December 2012, <https://www.rfc-editor.org/info/rfc6813>.

[SP800-155]
National Institute of Standards and Technology, "BIOS Integrity Measurement Guidelines (Draft)", December 2011, <https://csrc.nist.gov/csrc/media/publications/sp/800-155/draft/documents/draft-sp800-155_dec2011.pdf>.

[SWID-Gen]
Labs64, Munich, Germany, "SoftWare IDentification (SWID) Tags Generator (Maven Plugin)", n.d., <https://github.com/Labs64/swid-maven-plugin>.

[TCGRoT] Trusted Computing Group, "DRAFT: TCG Roots of Trust Specification", October 2018, <https://trustedcomputinggroup.org/wp-content/uploads/TCG_Roots_of_Trust_Specification_v0p20_PUBLIC_REVIEW.pdf>.

[TPM1.2] Trusted Computing Group, "TPM Main Specification Level 2 Version 1.2, Revision 116", March 2011, <https://trustedcomputinggroup.org/resource/tpm-main-specification/>.

[TPM2.0] Trusted Computing Group, "Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 01.59", November 2019, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

Authors' Addresses

Guy Fedorkow (editor)
Juniper Networks, Inc.
US

Email: gfedorkow@juniper.net

Eric Voit
Cisco Systems, Inc.
US

Email: evoit@cisco.com

Jessica Fitzgerald-McKay
National Security Agency
US

Email: jmfitz2@nsa.gov

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2020

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

June 24, 2020

A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs
draft-ietf-rats-yang-tpm-charra-02

Abstract

This document defines a YANG RPC and a minimal datastore tree required to retrieve attestation evidence about integrity measurements from a composite device with one or more roots of trust for reporting. 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 December 26, 2020.

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 7
2.2.1. ietf-tpm-remote-attestation 7
2.3. ietf-asymmetric-algs 32
3. IANA considerations 42
4. Security Considerations 42
5. Acknowledgements 42
6. Change Log 43
7. References 43
7.1. Normative References 43
7.2. Informative References 44
Authors' Addresses 44

1. Introduction

This document is based on the terminology defined in the [I-D.ietf-rats-architecture] and uses the interaction model and information elements defined in the [I-D.birkholz-rats-reference-interaction-model] document. The currently supported hardware security modules (HWM) - sometimes also referred to as an embedded secure element (eSE) - is the Trusted Platform Module (TPM) version 1.2 and 2.0 specified by the Trusted Computing Group (TCG). One ore more TPMs embedded in the components of a composite device - sometimes also referred to as an aggregate device - are required in order to use the YANG module defined in this document. A TPM is used as a root of trust for reporting (RTR) in order to retrieve attestation evidence from a composite device (quote primitive operation). Additionally, it is used as a root of trust

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

1.1. Requirements notation

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

2. The YANG Module for Basic Remote Attestation Procedures

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

2.1. Tree Diagram

```
  module: ietf-tpm-remote-attestation
    +--rw rats-support-structures
       +--rw supported-algos*   identityref
       +--ro compute-nodes* [node-id]
       |  +--ro node-id                 string
       |  +--ro node-physical-index?   int32 {ietfhw:entity-mib}?
       |  +--ro node-name?             string
       |  +--ro node-location?         string
       +--rw tpms* [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
          +--ro tpm-firmware-version?         string
          +--ro tpm-specification-version     identityref
          +--ro tpm-status?                   string
```

```
          +--rw certificates
             +--rw certificate* [certificate-name]
                +--rw certificate-name    string
                +--rw certificate-ref?    leafref
                +--rw certificate-type?   enumeration

    rpcs:
      +---x tpm12-challenge-response-attestation {TPM12}?
      |  +---w input
      |  |  +---w tpm1-attestation-challenge
      |  |     +---w pcr-index*              pcr
      |  |     +---w nonce-value             binary
      |  |     +---w TPM12_Algo?             identityref
      |  |     +---w (key-identifier)?
      |  |     |  +--:(public-key)
      |  |     |  |  +---w pub-key-id?       binary
      |  |     |  +--:(TSS_UUID)
      |  |     |     +---w TSS_UUID-value
      |  |     |        +---w ulTimeLow?       uint32
      |  |     |        +---w usTimeMid?       uint16
      |  |     |        +---w usTimeHigh?      uint16
      |  |     |        +---w bClockSeqHigh?   uint8
      |  |     |        +---w bClockSeqLow?    uint8
      |  |     |        +---w rgbNode*         uint8
      |  |     +---w add-version?            boolean
      |  |     +---w tpm-name*               string
      |  +--ro output
      |     +--ro tpm12-attestation-response* []
      |        +--ro certificate-name?            string
      |        +--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 revMajor?   uint8
      |           |  |  +--ro revMinor?   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 {TPM20}?
|  +---w input
|  |  +---w tpm20-attestation-challenge
|  |     +---w nonce-value          binary
|  |     +---w challenge-objects* []
|  |        +---w pcr-list* [TPM2_Algo]
|  |        |  +---w TPM2_Algo    identityref
|  |        |  +---w pcr-index*   tpm:pcr
|  |        +---w TPM2_Algo?          identityref
|  |        +---w (key-identifier)?
|  |        |  +--:(public-key)
|  |        |  |  +---w pub-key-id?   binary
|  |        |  +--:(uuid)
|  |        |     +---w uuid-value?   binary
|  |        +---w tpm-name*           string
|  +--ro output
|     +--ro tpm20-attestation-response* []
|        +--ro certificate-name?           string
|        +--ro up-time?                    uint32
|        +--ro node-id?                    string
|        +--ro node-physical-index?        int32
|        |       {ietfhw:entity-mib}?
|        +--ro quote?                      binary
|        +--ro quote-signature?            binary
|        +--ro pcr-bank-values* []
|        |  +--ro TPM2_Algo?    identityref
|        |  +--ro pcr-values* [pcr-index]
|        |     +--ro pcr-index    pcr
|        |     +--ro pcr-value?   binary
|        +--ro pcr-digest-algo-in-quote
|           +--ro TPM2_Algo?   identityref
+---x basic-trust-establishment
|  +---w input
|  |  +---w nonce-value         binary
|  |  +---w TPM2_Algo?          identityref
|  |  +---w tpm-name*           string
|  |  +---w certificate-name?   string
|  +--ro output
|     +--ro attestation-certificates* []
|        +--ro attestation-certificate?   ct:end-entity-cert-cms
|        +--ro (key-identifier)?
|           +--:(public-key)
```

```
|           |  +--ro pub-key-id?          binary
|           +--:(uuid)
|              +--ro uuid-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 up-time?            uint32
            +--ro certificate-name?   string
            +--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
```

2.2. YANG Modules

2.2.1. ietf-tpm-remote-attestation

This YANG module imports modules from [RFC6991], [RFC8348], [I-D.ietf-netconf-crypto-types], ietf-asymmetric-algs.yang.

```
<CODE BEGINS> file ietf-tpm-remote-attestation@2020-06-23.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-crypto-types {
    prefix ct;
  }
  import ietf-keystore {
    prefix ks;
  }
  import ietf-asymmetric-algs {
    prefix aa;
  }

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

  contact
   "WG Web  : <http://datatracker.ietf.org/wg/rats/>
    WG List : <mailto:rats@ietf.org>
    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  : Eric Voit <evoit@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 "2020-06-23" {
    description
      "Initial version";
    reference
      "draft-ietf-rats-yang-tpm-charra";
  }

  /*****************/
  /*   Features    */
  /*****************/

  feature TPM12 {
    description
      "This feature indicates that an Attester includes cryptoprocessors
       capable of supporting the TPM 1.2 API.";
  }

  feature TPM20 {
    description
      "This feature indicates that an Attester includes cryptoprocessors
       capable of supporting the TPM 2 API.";
```

```
  }

  /*****************/
  /*   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: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.";
  }


  /******************/
  /*   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 cryptoprocessor {
    description
```

```
      "Base identity identifying a crytoprocessor.";
  }

  identity tpm12 {
    base cryptoprocessor;
    description
      "A cryptoprocessor capable of supporting the TPM 1.2 API.";
  }

  identity tpm20 {
    base cryptoprocessor;
    description
      "A cryptoprocessor capable of supporting the TPM 2.0 API.";
  }


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

  grouping TPM2_Algo {
    description
      "The signature scheme that is used to sign the TPM2 Quote
       information response.";
    leaf TPM2_Algo {
      type identityref {
        base aa:tpm2-asymmetric-algorithm;
      }
      description
        "The signature scheme that is used to sign the TPM
         Quote information response.";
    }
  }

  grouping TPM12_Algo {
    description
      "The signature scheme that is used to sign the TPM2 Quote
      information response.";
    leaf TPM12_Algo {
      type identityref {
        base aa:tpm12-asymmetric-algorithm;
      }
      description
        "The signature scheme that is used to sign the TPM1.2
         Quote information response.";
    }
  }
```

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

  grouping tpm12-pcr-selection {
    description
      "A Verifier can request one or more PCR values using its
       individually created Attestation Key Certificate (AC).
       The corresponding selection filter is represented in this
       grouping.
       Requesting a PCR value that is not in scope of the AC used,
       detailed exposure via error msg should be avoided.";
    leaf-list pcr-index {
      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 pcr-list {
      key "TPM2_Algo";
      description
        "Specifies the list of PCRs and Hash Algorithms used for the
         latest returned 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 tpm:TPM2_Algo;
      leaf-list pcr-index {
```

```
        type tpm:pcr;
        description
          "The numbers of the PCRs that are associated with
           the created key.";
      }
    }
  }

  grouping tpm12-attestation-key-identifier {
    description
      "A selector for a suitable key identifier for a TPM 1.2.";
    choice key-identifier {
      description
        "Identifier for the attestation key to use for signing
         attestation evidence.";
      case public-key {
        leaf pub-key-id {
          type binary;
          description
            "The value of the identifier for the public key.";
        }
      }
      case TSS_UUID {
        description
          "Use a YANG agent generated (and maintained) attestation
           key UUID that complies with the TSS_UUID datatype of the TCG
           Software Stack (TSS) Specification, Version 1.10 Golden,
           August 20, 2003.";
        container TSS_UUID-value {
          description
            "A detailed structure that is used to create the
             TPM 1.2 native TSS_UUID as defined in the TCG Software
             Stack (TSS) Specification, Version 1.10 Golden,
             August 20, 2003.";
          leaf ulTimeLow {
            type uint32;
            description
              "The low field of the timestamp.";
          }
          leaf usTimeMid {
            type uint16;
            description
              "The middle field of the timestamp.";
          }
          leaf usTimeHigh {
            type uint16;
            description
              "The high field of the timestamp multiplexed with the
```

```
              version number.";
          }
          leaf bClockSeqHigh {
            type uint8;
            description
              "The high field of the clock sequence multiplexed with
               the variant.";
          }
          leaf bClockSeqLow {
            type uint8;
            description
              "The low field of the clock sequence.";
          }
          leaf-list rgbNode {
            type uint8;
            description
              "The spatially unique node identifier.";
          }
        }
      }
    }
  }

  grouping tpm20-attestation-key-identifier {
    description
      "A selector for a suitable key identifier.";
    choice key-identifier {
      description
        "Identifier for the attestation key to use for signing
         attestation evidence.";
      case public-key {
        leaf pub-key-id {
          type binary;
          description
            "The value of the identifier for the public key.";
        }
      }
      case uuid {
        description
          "Use a YANG agent generated (and maintained) attestation
           key UUID.";
        leaf uuid-value {
          type binary;
          description
            "The UUID identifying the corresponding public key.";
        }
      }
    }
```

```
  }

  grouping certificate-name {
    description
      "An arbitrary name for the identity certificate chain requested.";
    leaf certificate-name {
      type string;
      description
        "An arbitrary name for the identity certificate chain requested.";
    }
  }

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

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

  grouping compute-node-identifier {
    description
      "In a distributed system with multiple compute nodes
       this is the node identified by name and physical-index.";
    leaf node-id {
       type string;
       description
         "ID of the compute node, such as Board Serial Number.";
    }
    leaf node-physical-index {
      if-feature ietfhw:entity-mib;
      type int32 {
```

```
        range "1..2147483647";
      }
      config false;
       description
         "The entPhysicalIndex for the compute node.";
       reference
         "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
    }
  }

  grouping tpm12-pcr-info-short {
    description
      "This structure is for defining a digest at release when the only
       information that is necessary is the release configuration.";
    uses tpm12-pcr-selection;
    leaf locality-at-release {
      type uint8;
      description
        "This SHALL be the locality modifier required to release the
         information (TPM 1.2 type TPM_LOCALITY_SELECTION)";
    }
    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).";
    }
  }

  grouping tpm12-version {
    description
      "This structure provides information relative the version of
      the TPM.";
    list version {
      description
        "This indicates the version of the structure
         (TPM 1.2 type TPM_STRUCT_VER). This MUST be 1.1.0.0.";
      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 revMajor {
        type uint8;
        description
          "Indicates the rev major version of the structure.
           MUST be 0x00.";
      }
      leaf revMinor {
        type uint8;
        description
          "Indicates the rev minor version of the structure.
           MUST be 0x00.";
      }
    }
  }

  grouping tpm12-quote-info-common {
    description
      "These statements are used in bot quote variants of the TPM 1.2";
    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 SHA-1 hash of tpm12-quote-info2'.";
    }
  }

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

```
    uses tpm12-version;
    leaf digest-value {
      type binary;
      description
        "This SHALL be the result of the composite hash algorithm using
         the current values of the requested PCR indices
         (TPM 1.2 type TPM_COMPOSITE_HASH.)";
    }
  }

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

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

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

  grouping tpm12-pcr-composite {
    description
      "The actual values of the selected PCRs (a list of TPM_PCRVALUEs
       (binary) and associated metadata for TPM 1.2.";
    list TPM_PCR_COMPOSITE {
      description
        "The TPM 1.2 TPM_PCRVALUEs for the pcr-indices.";
      uses tpm12-pcr-selection;
      leaf value-size {
        type uint32;
        description
          "This SHALL be the size of the 'tpm12-pcr-value' field
           (not the number of PCRs).";
```

```
      }
      leaf-list tpm12-pcr-value {
        type binary;
        description
          "The list of TPM_PCRVALUEs from each PCR selected in sequence
           of tpm12-pcr-selection.";
      }
    }
  }

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

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

```
  grouping tpm20-attestation {
    description
      "Contains an instance of TPM2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester information.";
    uses certificate-name;
    uses node-uptime;
    uses compute-node-identifier;
    leaf quote {
      type binary;
      description
        "Quote data returned by TPM Quote, including PCR selection,
         PCR digest and etc.";
    }
    leaf quote-signature {
      type binary;
      description
        "Quote signature returned by TPM Quote.";
    }
    list pcr-bank-values {
      /* This often should not be necessary for TPM2, as the information
         if validated will need to be coming from the 'quote' leaf       */
      description
        "PCR values in each PCR bank.";
      uses TPM2_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.";
        }
      }
    }
    container pcr-digest-algo-in-quote {
      uses TPM2_Algo;
      description
        "The hash algorithm for PCR value digest in Quote output.";
    }
  }
```

```
  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 aa:asymmetric-algorithm-type;
        }
        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 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 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-log {
    list ima-event-entry {
      key event-number;
      description
      "Ordered list of ima event logs by event-number";
       uses ima-event;
    }
    description
      "Measurement log created by IMA.";
  }

  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
            "This is an index referencing the TCG Algorithm
             Registry based on TPM_ALG_ID.";
          uses bios-event-log;
        }
      }
      case ima {
        description
          "IMA event logs";
        container ima-event-logs {
          description
            "This is an index referencing the TCG Algorithm
             Registry based on TPM_ALG_ID.";
          uses ima-event-log;
        }
      }
    }
  }

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

  rpc tpm12-challenge-response-attestation {
    if-feature "TPM12";
    description
      "This RPC accepts the input for TSS TPM 1.2 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 tpm1-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;
        uses TPM12_Algo;
        uses tpm12-attestation-key-identifier;
        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.";
        }
        uses tpm-name-selector;
          /* if this scheme is desired, we should define XPATH to limit
           selection to just 'tpm-name' that are '../tpm-specification-version'
           equals 'TPM12' and where '../hardware-based' equals 'true' */
      }
    }
    output {
      list tpm12-attestation-response {
        description
          "The binary output of TPM 1.2 TPM_Quote/TPM_Quote2, including
           the PCR selection and other associated attestation evidence
           metadata";
        uses tpm12-attestation;
      }
    }
  }

  rpc tpm20-challenge-response-attestation {
    if-feature "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;
        list challenge-objects {
          description
            "Nodes to fetch attestation information, PCR selection
             and AK identifier.";
          uses tpm20-pcr-selection;
          uses TPM2_Algo;
          uses tpm20-attestation-key-identifier;
          uses tpm-name-selector;
          /* if this scheme is desired, we should define XPATH to limit
           selection to just 'tpm-name' that are '../tpm-specification-version'
           equals 'TPM2' and where '../hardware-based' equals 'true' */
        }
      }
    }
```

```
    output {
      list tpm20-attestation-response {
        unique "certificate-name";   /* should have XPATH making this mandatory
                                  when there is more than one list entry */
        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 tpm20-attestation;
      }
    }
  }

  rpc basic-trust-establishment {
    description
      "This RPC creates a tpm-resident, non-migratable key to be used
       in TPM_Quote commands, an attestation certificate.";
    input {
      uses nonce;
      uses TPM2_Algo;
      leaf-list tpm-name {
        when "not(../certificate-name)";  /* ensures both are not populated */
        type string;
        description
          "Path to a unique TPM on a device.  If there are no elements in the
          leaf-list, all TPMs which are 'hardware-based' should have keys
          established.";
      }
      uses certificate-name {
        description
          "It is possible to request a new certificate using the old one as a
          reference.";
      }
    }
    output {
      list attestation-certificates {
        description
          "Attestation Certificate data from a TPM identified by the TPM
           name";
        leaf attestation-certificate {
          type ct:end-entity-cert-cms;
          description
            "The binary signed certificate chain data for this identity
             certificate.";
        }
        uses tpm20-attestation-key-identifier;
      }
    }
```

```
  }

  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 "certificate-name";
          description
            "Event logs of a node in a distributed system
             identified by the node name";
          uses node-uptime;
          uses certificate-name;
          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.";
    leaf-list supported-algos {
      config true;
      type identityref {
        base aa:asymmetric-algorithm-type;
      }
```

```
      description
        "Supported algorithms values for an Attester.";
    }
    list compute-nodes {
      config false;
      key node-id;
      uses compute-node-identifier;
      description
        "A list names of hardware components in this composite
         device that RATS can be conducted with.";
      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.";
      }
    }
    list tpms {
      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 {
        config false;
        type boolean;
        description
          "Answers the question: is this TPM is a hardware based TPM?";
      }
      leaf tpm-physical-index {
        if-feature ietfhw:entity-mib;
        config false;
        type int32 {
          range "1..2147483647";
        }
        description
          "The entPhysicalIndex for the TPM.";
        reference
          "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
      }
      leaf tpm-path {
        type string;
        config false;
```

```
        description
          "Path to a unique TPM on a device.  This can change agross reboots.";
      }

      leaf compute-node {
        when "../../compute-nodes";
        config false;
        mandatory true;
        type compute-node-ref;
        description
          "When there is more that one TPM, this indicates for which
          compute node this TPM services.";
      }
      leaf tpm-manufacturer {
        config false;
        type string;
        description
          "TPM manufacturer name.";
      }
      leaf tpm-firmware-version {
        config false;
        type string;
        description
          "TPM firmware version.";
      }
      leaf tpm-specification-version {
        type identityref {
          base cryptoprocessor;
        }
        config false;
        mandatory true;
        description
          "Identifies the cryptoprocessor API set supported";
      }
      leaf tpm-status {
        type string;
        config false;
        description
          "TPM chip self-test status, normal or abnormal.";
      }
      container certificates {
        description
          "The TPM's certificates, including EK certificates
           and AK certificates.";
        list certificate {
          config true;
          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.";
          uses certificate-name;
          leaf certificate-ref {
            type leafref {
              path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
                 + "/ks:certificates/ks:certificate/ks:name";
            }
            description
              "A reference to a specific certificate of an
               asymmetric key in the Keystore.";
               /* Note: It is also possible to import a grouping which allows
               local definition via an imported keystore schema. */
          }
          leaf certificate-type {
            type enumeration {
              enum endorsement-cert {
                value 0;
                description
                  "Endorsement Key (EK) Certificate type.";
              }
              enum initial-attestation-cert {
                value 1;
                description
                  "Initial Attestation key (IAK) Certificate type.";
              }
              enum local-attestation-cert {
                value 2;
                description
                  "Local Attestation Key (LAK) Certificate type.";
              }
            }
            description
              "Type of this certificate";
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

2.3. ietf-asymmetric-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. Therefore a modified version of this draft is included here. Perhaps someone will steward this list as a separate draft.

```
<CODE BEGINS> ietf-asymmetric-algs@2020-06-12.yang
module ietf-asymmetric-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-asymmetric-algs";
  prefix aa;

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     Author:   Eric Voit <mailto:evoit@cisco.com>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
     Author:   Wang Haiguang <wang.haiguang.shieldlab@huawei.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 2020-06-12 {
    description
      "Initial version";
```

```
    reference
      "RFC XXXX: tbd
       initial draft: draft-voit-rats-trusted-path-routing
       concepts from ietf-asymmetric-algs.yang which did not progress to
       WGLC in NETCONF.";
  }

  /*****************/
  /*   Features    */
  /*****************/

  feature TPM12 {
    description
      "This feature indicates support for the TPM 1.2 API.";
  }

  feature TPM20 {
    description
      "This feature indicates support for the TPM 2.0 API.";
  }

  feature iana {
    description
      "This feature indicates support for the IANA algorithms defined
      in Registry xxxxx";
  }

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

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

  identity asymmetric-algorithm-type {
    description
      "Base identity identityerating various asymmetric key algorithms.";
  }

  identity iana-asymmetric-algorithm {
    base asymmetric-algorithm-type;
    description
      "Base identity identityerating various asymmetric key algorithms.";
  }

  identity tpm12-asymmetric-algorithm {
    base asymmetric-algorithm-type;
    description
```

```
      "Base identity identityerating various asymmetric key algorithms.";
    reference
      "TPM-Main-Part-2-TPM-Structures_v1.2_rev116_01032011.pdf
      TPM_ALGORITHM_ID values, page 18";
  }

  identity tpm2-asymmetric-algorithm {
    base asymmetric-algorithm-type;
    description
      "Base identity identityerating various asymmetric key algorithms.";
    reference
      "TPM-Rev-2.0-Part-2-Structures-01.38.pdf
      The TCG Algorithm Registry ID value. Table 9";
  }

  identity rsa {
    base tpm12-asymmetric-algorithm;
    base tpm2-asymmetric-algorithm;
    description
      "RFC 3447 - the RSA algorithm";
  }

  identity rsa1024 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    base rsa;
    description
      "The RSA algorithm using a 1024-bit key.";
    reference
      "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
      }

  identity rsa2048 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    base rsa;
    description
      "The RSA algorithm using a 2048-bit key.";
    reference
      "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
      }

  identity rsa3072 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    base rsa;
    description
      "The RSA algorithm using a 3072-bit key.";
```

```
    reference
      "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
  }

  identity rsa4096 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    base rsa;
    description
      "The RSA algorithm using a 4096-bit key.";
    reference
      "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
  }

  identity rsa7680 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    base rsa;
    description
      "The RSA algorithm using a 7680-bit key.";
    reference
      "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
  }

  identity rsa15360 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    base rsa;
    description
      "The RSA algorithm using a 15360-bit key.";
    reference
      "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
  }

  identity secp192r1 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    description
      "The asymmetric algorithm using a NIST P192 Curve.";
    reference
      "RFC 6090: Fundamental Elliptic Curve Cryptography Algorithms.
       RFC 5480: Elliptic Curve Cryptography Subject Public Key
                 Information.";
  }

  identity secp224r1 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
```

```
    description
      "The asymmetric algorithm using a NIST P224 Curve.";
    reference
      "RFC 6090: Fundamental Elliptic Curve Cryptography Algorithms.
       RFC 5480: Elliptic Curve Cryptography Subject Public Key
                 Information.";
  }

  identity secp256r1 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    description
      "The asymmetric algorithm using a NIST P256 Curve.";
    reference
      "RFC 6090: Fundamental Elliptic Curve Cryptography Algorithms.
       RFC 5480: Elliptic Curve Cryptography Subject Public Key
                 Information.";
  }

  identity secp384r1 {
    base iana-asymmetric-algorithm;
    description
      "The asymmetric algorithm using a NIST P384 Curve.";
    reference
      "RFC 6090: Fundamental Elliptic Curve Cryptography Algorithms.
       RFC 5480: Elliptic Curve Cryptography Subject Public Key
                 Information.";
  }

  identity secp521r1 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    description
      "The asymmetric algorithm using a NIST P521 Curve.";
    reference
      "RFC 6090: Fundamental Elliptic Curve Cryptography Algorithms.
       RFC 5480: Elliptic Curve Cryptography Subject Public Key
                 Information.";
  }

  identity x25519 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    description
      "The asymmetric algorithm using a x.25519 Curve.";
    reference
      "RFC 7748: Elliptic Curves for Security.";
  }
```

```
  identity x448 {
    if-feature "iana";
    base iana-asymmetric-algorithm;
    description
      "The asymmetric algorithm using a x.448 Curve.";
    reference
      "RFC 7748: Elliptic Curves for Security.";
  }

  identity SHA1 {
    if-feature "TPM20 or TPM12";
    base tpm12-asymmetric-algorithm;
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10118-3 - SHA1 algorithm";
  }

  identity HMAC {
    if-feature "TPM20 or TPM12";
    base tpm12-asymmetric-algorithm;
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 9797-2 - Hash Message Authentication Code (HMAC) algorithm
      also RFC2014.
      we need to verify if NMAC implementation isn't different in the two.";
  }

  identity AES {
    if-feature "TPM20 or TPM12";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 18033-3 - the AES algorithm";
  }

  identity AES128 {
    if-feature "TPM12";
    base tpm12-asymmetric-algorithm;
    base AES;
    description
      "ISO/IEC 18033-3 - the AES algorithm, key size 128";
  }

  identity AES192 {
    if-feature "TPM12";
    base tpm12-asymmetric-algorithm;
    base AES;
    description
      "ISO/IEC 18033-3 - the AES algorithm, key size 192";
```

```
  }

  identity AES256 {
    if-feature "TPM12";
    base tpm12-asymmetric-algorithm;
    base AES;
    description
      "ISO/IEC 18033-3 - the AES algorithm, key size 256";
  }

  identity MGF1 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "IEEE Std 1363a -2004 - hash-based mask-generation function";
  }
  identity KEYEDHASH {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "TPM2 KEYEDHASH - an encryption or signing algorithm using a keyed hash";
    }

  identity XOR {
    if-feature "TPM20 or TPM12";
    base tpm12-asymmetric-algorithm;
    base tpm2-asymmetric-algorithm;
    description
        "TPM2 XOR";
  }

  identity SHA256 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10118-3 - the SHA 256 algorithm";
  }

  identity SHA384 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10118-3 - the SHA 384 algorithm";
  }

  identity SHA512 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
```

```
    description
      "ISO/IEC 10118-3 - the SHA 512 algorithm";
  }

  identity NULL {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "TPM2 NULL";
  }

  identity SM3_256 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "GM/T 0004-2012 - SM3_256";
  }

  identity SM4 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "GM/T 0004-2012 - SM4 symmetric block cipher";
  }

  identity RSASSA {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "RFC 3447 - defined in section 8.2 (RSASSAPKCS1-v1_5)";
  }

  identity RSAES {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "RFC 3447 - defined in section 7.2 (RSAES-PKCS1-v1_5)";
  }

  identity RSAPSS {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "RFC 3447 - defined in section 8.1 (RSASSA PSS)";
  }

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

```
    base tpm2-asymmetric-algorithm;
    description
      "RFC 3447 - defined in section 7.1 (RSASSA OAEP)";
  }

  identity ECDSA {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 14888-3 - elliptic curve cryptography (ECC)";
  }

  identity ECDH {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "NIST SP800-56A - secret sharing using ECC";
  }

  identity ECDAA {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "TPM2 - elliptic-curve based anonymous signing scheme";
  }

  identity SM2 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "A GM/T 0003.1-2012, GM/T 0003.2-2012, GM/T 0003.3-2012,
      GM/T 0003.5-2012   SM2";
  }

  identity ECSCHNORR {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "TPM2 - elliptic-curve based Schnorr signature";
  }

  identity ECMQV {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "NIST SP800-56A - two-phase elliptic-curve key";
  }
```

```
  identity KDF1_SP800_56A {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "NIST SP800-56A - concatenation key derivation function,
      (approved alternative1) section 5.8.1";
  }

  identity KDF2 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "IEEE 1363a-2004 - key derivation function KDF2 section 13.2";
  }

  identity KDF1_SP800_108 {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "NIST SP800-108 - Section 5.1 KDF in Counter Mode";
  }

  identity ECC {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 15946-1 - prime field ECC";
  }

  identity SYMCIPHER {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "TPM2 - object type for a symmetric block cipher";
  }

  identity CAMELLIA {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 18033-3 - the  Camellia algorithm";
  }

  identity CTR {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10116 - Counter mode";
```

```
  }

  identity OFB {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10116 - Output Feedback mode";
  }

  identity CBC {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10116 - Cipher Block Chaining mode";
  }

  identity CFB {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10116 - Cipher Feedback mode";
  }

  identity ECB {
    if-feature "TPM20";
    base tpm2-asymmetric-algorithm;
    description
      "ISO/IEC 10116 - Electronic Codebook mode";
  }

}
<CODE ENDS>
```

3. IANA considerations

This document will include requests to IANA:

To be defined yet.

4. Security Considerations

There are always some.

5. Acknowledgements

Not yet.

6. Change Log

Changes from version 01 to version 02:

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

Changes from version 00 to version 01:

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

7. References

7.1. Normative References

[I-D.birkholz-rats-reference-interaction-model]
Birkholz, H. and M. Eckel, "Reference Interaction Models for Remote Attestation Procedures", draft-birkholz-rats-reference-interaction-model-02 (work in progress), January 2020.

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-15 (work in progress), May 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>.

7.2. Informative References

[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.

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

Shwetha Bhandari
Cisco Systems

Email: shwethab@cisco.com

Bill Sulzen
Cisco Systems

Email: bsulzen@cisco.com

Eric Voit
Cisco Systems

Email: evoit@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

RATS
Internet-Draft
Intended status: Informational
Expires: 14 December 2020

A. Shaw
H. Tschofenig
S. Trofimov
S. Frost
T. Fossati
arm

12 June 2020

Restful Attested Resources
draft-shaw-rats-rear-00

Abstract

This memo describes a REST interface based on the RATS architecture that can be used to retrieve attested system state, for example the reading of a security critical sensor. The objective is to present a common vocabulary of data formats and basic protocol transactions that can be pieced together into a cohesive interface that is capable of serving different attestation workflows.

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 14 December 2020.

Table of Contents

1. Introduction 3
1.1. Use Cases 3
1.2. Document Organisation 4
1.3. Conventions used in this document 4
2. Abstract Mechanism 4
2.1. Attester Interface 4
2.1.1. Resource Validation 5
2.2. Verifier Interface 5
2.2.1. Attestation Result Validation 6
2.3. Example Compositions 6
2.3.1. Background Check with Nonce-based Freshness 6
2.3.2. Background Check with Timestamp-based Freshness . . . 7
2.3.3. Passport with Timestamp-based Freshness 7
2.3.4. Timestamp-based Uni-directional 8
3. REST Instantiation 9
3.1. Basic Data Formats 9
3.1.1. Resource 9
3.1.2. Nonce 10
3.1.3. Timestamp 10
3.1.4. Evidence 10
3.1.5. Attestation Result 10
3.2. Request and Response Payloads 10
3.2.1. Requesting an Attested Resource 10
3.2.2. Attested Resource 10
3.2.3. Request for Attestation Result 11
3.2.4. Verifier Response 11
3.3. Interaction Model 12
3.3.1. Channel Security Considerations 12
3.3.2. URLs 12
3.3.3. Methods 12
3.3.4. Multicast Support 13
3.3.5. Examples 13
4. Discovery 17
4.1. Resource Directory 17
4.1.1. Attested Resource Registration 17
4.1.2. Verifier Resource Registration 19
5. IANA Considerations 19
6. Privacy Considerations 20
7. Security Considerations 20
7.1. Model Architecture for the Origin 20
Acknowledgments 20
References 20

Normative References 20
Informative References 22
Authors' Addresses 22

1. Introduction

This memo describes a REST [Fielding] interface based on the RATS architecture [I-D.ietf-rats-architecture] that can be used to retrieve attested system state, for example the reading of a security critical sensor.

We present a simple vocabulary of data formats and basic protocol transactions that can be pieced together into a cohesive interface capable of serving different attestation workflows. At a minimum, we want to cater for the "background check" and "passport" topological models, and for freshness of attestation based on nonces as well as timestamps.

The obvious advantage of sharing a uniform interface across different actors is it creates an ecosystem in which variability is minimised and so is the need to add complex and often fragile logics into the deployed components, e.g., data format and protocol translation. Besides, using the familiar REST toolbox provides additional benefits in terms of developer friendliness as well as code base and infrastructure reuse (e.g., web caching).

1.1. Use Cases

The primary use case is that of a device that needs to provide application state to third parties with strong authenticity.

This is a common situation in critical infrastructure systems where an actuator device needs some assurance that the sensing equipment is in pristine state before acting on its signals. Here, the sensor would expose its safety critical samples via an attested resource whose authenticity can be verified by the actuator.

Another potential application is a fleet controller that needs to know the current state of its dependent devices to inform its next actions (e.g., scheduling a firmware update campaign). Here, the dependent devices uniformly expose the same resource (e.g., the list of currently installed software components) to the controller, which can decide, based on the information provided, which devices need a certain security patch.

Many more use cases exist.

1.2. Document Organisation

The remainder of this document describes:

* An abstract protocol that allows a device to expose arbitrary attested system state, which can be consumed by third parties (Section 2);
* An instantiation of said abstract protocol as a set of uniform data formats and interaction primitives based on the REST paradigm for both HTTP [RFC7230] and CoAP [RFC7252] (Section 3);
* A way to advertise and discover said capability (Section 4).

1.3. Conventions used in this document

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. Abstract Mechanism

The protocol principals are the three RATS actors: the attester (A), the relying party (RP) and the verifier (V).

It is assumed that A either directly owns a resource, r, or has a direct trust relationship with the resource owner.

In the following, "n" and "t" are freshness indicators: "n" is an initiator provided nonce, "t" is a timestamp sourced by the responder. When using timestamp based freshness, producers' and consumers' clocks MUST be synchronised.

2.1. Attester Interface

The interface to the Attester is illustrated in Figure 1.

X is any entity interacting with the Attester, typically a Relying Party, which wants to retrieve an attested resource.

A function "E(n_X, r, t_A)" is used by A to compute an evidence report binding the device status to the resource ("r") together with the freshness indicators "n_X" and "t_A". Typically, only one of "n_X" or "t_A" will be present.

"E()" outputs an EAT token [I-D.ietf-rats-eat], "E", carrying a "nonce" claim that is used as described in the following.

The binding between "n_X", "t_A" and "r" is obtained by hashing their concatenation, "H(n_X || r || t_A)", and storing the result in the "nonce" claim which is then cryptographically signed by the Attester as part of the produced evidence, "E". The presence of any freshness indicator (i.e., "n_X" or "t_A") is optional. For the purpose of computing "E", a nil freshness indicator is replaced by the zero-length string, "". If "t_A != nil", then its value needs to be sent back to the requester as an additional explicit protocol entity.

Optionally, an attestation result "R" computed on evidence "E" MAY be returned by an Attester that acts as a forwarder for a Verifier.

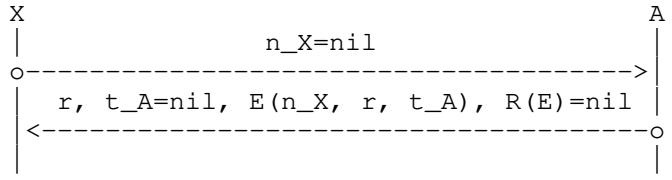


Figure 1: Attester Interface

2.1.1. Resource Validation

Given an Appraisal Policy for Evidence "APE" and an Appraisal Policy for Attestation Result "APR", X accepts "r" if and only if:

* "E | APE => true"
* "E.nonce == H(n_X || r || t_A)"

If "R(E)!=nil", two further conditions MUST hold:

* "R(E) | APR => true"
* "R.nonce == H("" || E || "")"

Note that not all the appraisal operations are computed directly by X. For example, "E | APE" is typically delegated to a trusted Verifier.

2.2. Verifier Interface

The interface to the Verifier is illustrated in Figure 2.

Y is any entity interacting with the Verifier, e.g., a Relying Party or an Attester, which supplies an evidence and receives an attestation result.

The function "R(n_Y, E, t_V)" is used by V to compute the attestation result over "E" using an implicit Appraisal Policy for Evidence "APE". The result is cryptographically signed by V and bound to any available freshness indicator.

"R()" outputs an EAT token [I-D.ietf-rats-eat], "R", carrying at a minimum:

* a "result" claim carrying a boolean value that reflects the validity of the submitted evidence given the Appraisal Policy for Evidence used by the Verifier;
* a "nonce" claim that is used as described in the following.

The token MAY contain further information associated with the evidence validation process.

The binding between "n_Y", "t_V" and "E" is obtained by hashing their concatenation, "H(n_Y || E || t_V)", and storing the result in the "nonce" claim which is then cryptographically signed by the Verifier as part of the produced attestation result, "R". The presence of any freshness indicator (i.e., "n_Y" or "t_V") is optional. For the purpose of computing "R", a nil freshness indicator is replaced by the zero-length string, "".

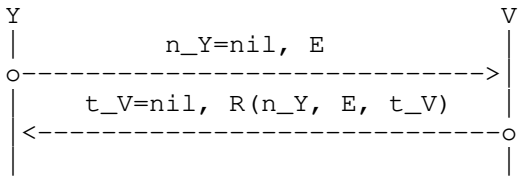


Figure 2: Verifier Interface

2.2.1. Attestation Result Validation

Given an Appraisal Policy for Attestation Result "APR", Y accepts "R" if and only if:

* "R(E) | APR => true"
* "R.nonce == H(n_Y || E || t_V)"

2.3. Example Compositions

2.3.1. Background Check with Nonce-based Freshness

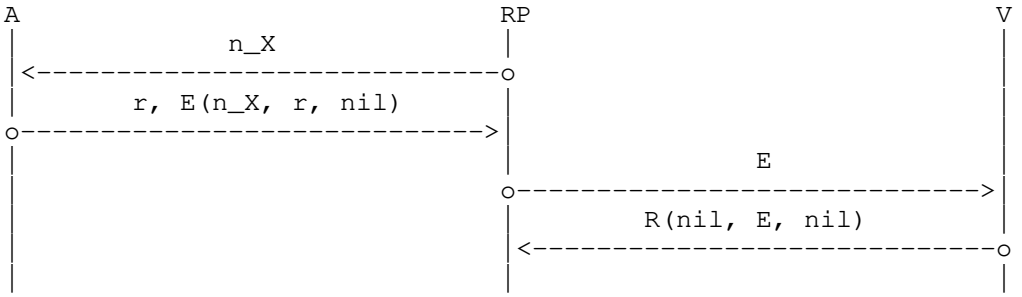


Figure 3: Background Check with Nonce-based Freshness

RP accepts "r" if and only if:

* "E | APE => true"
* "E.nonce == H(n_X || r || "")"
* "R | APR => true", or equivalently "R.result == true"
* "R.nonce == H("" || E || "")"

2.3.2. Background Check with Timestamp-based Freshness

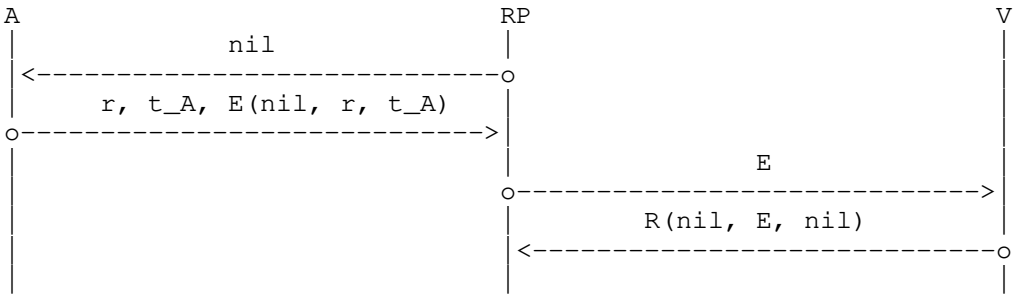


Figure 4: Background Check with Timestamp-based Freshness

RP accepts r if and only if:

* "R | APR => true", or equivalently "R.result == true"
* "R.nonce == H("" || E || "")"
* "E | APE => true"
* "E.nonce == H("" || r || t_A)"

2.3.3. Passport with Timestamp-based Freshness

The idea is that whenever the state of r changes, the Attester will "self-issue" an evidence for the changed resource using a locally sourced timestamp ("t_A") as the freshness indicator.

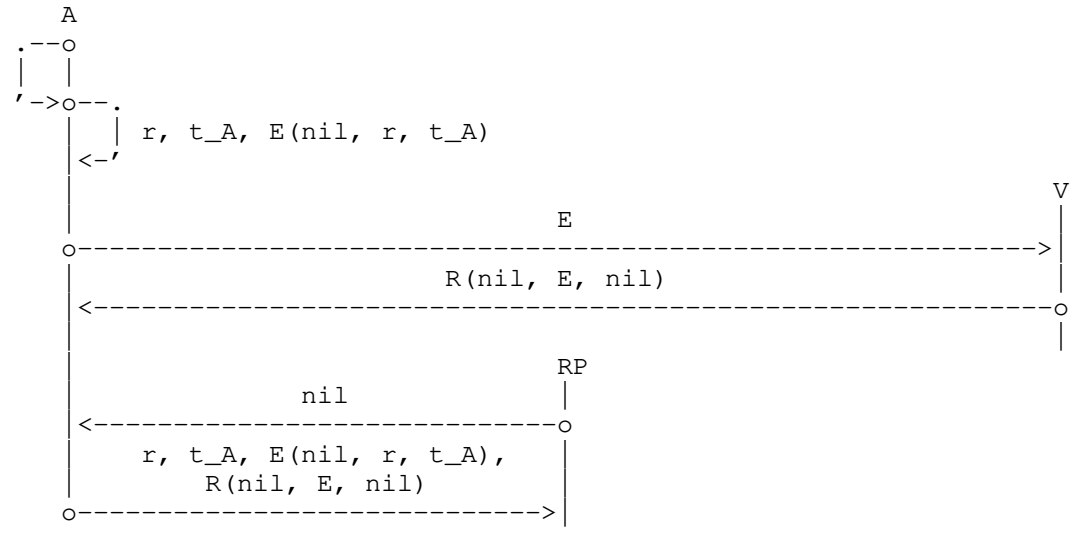


Figure 5: Passport with Timestamp-based Freshness

RP accepts r if and only if:

* "R | APR => true"
* "R.nonce == H("" || E || "")"
* "E.nonce == H("" || r || t_A)"

2.3.4. Timestamp-based Uni-directional

If the transport allows it, timestamp-based uni-directional attestation protocols, e.g., TUDA [I-D.birkholz-rats-tuda], can also be constructed from the presented primitives. For example, using CoAP Observe [RFC7641] the interaction pattern in Figure 6, with an initial trigger and subsequent automatic updates on resource status change, can be naturally implemented.

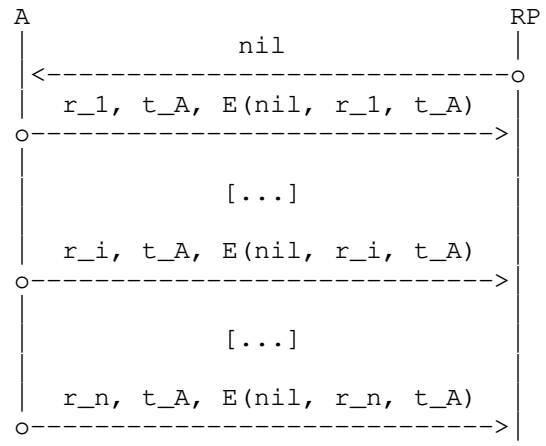


Figure 6: Timestamp-based Uni-directional

3. REST Instantiation

Four new MIME types are defined for the requests and responses among the three actors that have been identified in the abstract mechanism. The MIME types are composed of the basic data types defined in Section 3.1.

3.1. Basic Data Formats

* The resource to be attested;
* A caller provided nonce;
* A locally sourced timestamp;
* The evidence produced by the Attester, and
* The attestation result produced by the Verifier.

These basic types are described by the following CDDL rules, which reuse the eat-token definition from [I-D.ietf-rats-eat].

3.1.1. Resource

An "ANY DEFINED BY"-like payload with type set to the original MIME type, either Content-Type (HTTP) or Content-Format (CoAP), of the resource representation.

```
resource-type = (
  typ tstr / uint,
  val any,
)
```

3.1.2. Nonce

```
nonce-type = bstr
```

3.1.3. Timestamp

```
timestamp-type = tdate / time
```

3.1.4. Evidence

An EAT token signed by the attester bound to the relying party request and the attested resource state.

```
evidence-type = eat-token
```

3.1.5. Attestation Result

An EAT token signed by the verifier and bound to an evidence.

```
attestation-result-type = eat-token
```

3.2. Request and Response Payloads

3.2.1. Requesting an Attested Resource

MIME type "application/rats-attested-resource-request"

CoAP Content-Format: TBD-rats-attested-resource-request-CT

```
nonce-key = 0 / "n_X"

attested-resource-request = {
    ? nonce-key => nonce-type,
}
```

This type is used in a POST request to an attested resource.

3.2.2. Attested Resource

MIME type "application/rats-attested-resource"

CoAP Content-Format: TBD-rats-attested-resource-CT

```
resource-key = 1 / "r"
t-A-key = 2 / "t_A"
evidence-key = 3 / "E"
attestation-result-key = 4 / "R"

attested-resource = {
    resource-key => resource-type,
    ? t-A-key => timestamp-type,
    evidence-key => evidence-type,
    ? attestation-result-key => attestation-result-type,
}
```

This type is used in a successful response to a request to an attested resource endpoint.

Note that an attestation result is only present when the Passport model is used.

Note also that the fact that the inner resource representation is embedded within the "application/rats-attested-resource" envelope suppresses the ability to do content negotiation on it, i.e., the inner representation format is unilaterally chosen by the origin.

3.2.3. Request for Attestation Result

MIME type "application/rats-attestation-result-request"

CoAP Content-Format: TBD-rats-attestation-result-request-CT

```
n-Y-key = 5 / "n_Y"

attestation-result-request = {
    ? n-Y-key => nonce-type,
    evidence-key => evidence-type,
}
```

This type is used in a POST request to a verifier endpoint.

3.2.4. Verifier Response

MIME type "application/rats-attestation-result-response"

CoAP Content-Format: TBD-rats-attestation-result-response-CT

```
t-V-key = 6 / "n_Y"

attestation-result-response = {
    ? t-V-key => timestamp-type,
    attestation-result-key => attestation-result-type,
}
```

This type is used in a successful response to a POST request to a verifier endpoint.

3.3. Interaction Model

(For now) we only describe a synchronous, RPC-like transaction model, including the slight variant with a one-off trigger presented in Section 2.3.4.

This might be not suited for devices that sit behind a NAT/firewall box, or those that have to go through extended sleep cycles in order to save energy. For this kind of devices, we assume in-network support in the form of store-and-forward nodes (e.g., LwM2M queue mode, specialised border routers, etc.).

3.3.1. Channel Security Considerations

Unless the channel can be considered free from passive and active attackers at all times, all transactions are to be carried over a secure transport (i.e., HTTPS or COAPS).

3.3.2. URLs

In the spirit of [RFC7320], no specific URL format is mandated. An application is free to specify the URL scheme of its liking for the exposed attested resources.

When an origin exposes the same underlying state both as nonce- and timestamp-based resources, these are identified by two separate URIs.

The verifier function is exposed via an URI that accepts evidence in form of "application/rats-attestation-result-request" typed requests and returns attestation results in form of "application/rats-attestation-result-response" typed responses.

3.3.3. Methods

As per usual REST conventions, the guiding principles are:

* POST is used for all requests involving a payload;
* GET is used for requests without a payload.

The only example of the latter is when retrieving an "Attested Resource" using the timestamp-based freshness model. Any other request uses POST.

3.3.3.1. Response Codes and Caching

The possible status codes are:

* HTTP
  - 200 (OK) for successful GET. This response is cacheable; origins can use Cache-Control (max-age) and ETag headers in order to instruct on-path caches.
  - 201 (Created) for a successful POST. This response is not cacheable.
* CoAP
  - 2.05 (Content) for successful GET. This response is cacheable; origins can use Max-Age and ETag Options to instruct on-path caches;
  - 2.01 (Created) for successful POST. This response is not cacheable.

Otherwise, a suitable error response (i.e., HTTP 4xx/5xx, CoAP 4.nn/5.nn) is returned.

3.3.4. Multicast Support

TODO (This is a CoAP only feature.)

3.3.5. Examples

A few examples are given to illustrate the different interaction models using both CoAP and HTTP transports.

3.3.5.1. Background Check with Nonce Based Freshness

* RP - Attester (CoAP)

```
     >> Request:
      POST coap://device.example/my-attested-resource
      Content-Format: TBD-application/rats-attested-resource-request-CT
      Accept: application/rats-attested-resource
      Payload:
      {
          "n_X": "bm9uY2Uh"
      }

     << Response:
      2.01 Created
      ETag: "xyzzy"
      Content-format: TBD-application/rats-attested-resource-CT
      Payload:
      {
          "r" : {
              "typ": "text/plain",
              "val": "foobar"
          },
          "E": "eyJhbGciO...RfrKmTWk"
      }
```

* RP - Verifier (HTTP)

```
     >> Request:
      POST /my-verify
      Host: verifier.example
      Content-Type: application/rats-attestation-result-request
      Accept: application/rats-attestation-result-response

      {
          "E": "eyJhbGciO...RfrKmTWk"
      }

     << Response:
      HTTP/1.1 201 Created
      ETag: "abccb"
      Content-format: application/rats-attestation-result-response
      Payload:
      {
          "R": "eyJhbGciO...8j5EDGYc"
      }
```

3.3.5.2. Background Check with Timestamp Based Freshness

* RP - Attester (CoAP) with POST

```
>> Request:
 POST coap://device.example/my-attested-resource
 Content-Format: TBD-application/rats-attested-resource-request-CT
 Accept: TBD-application/rats-attested-resource-CT
 Payload:
 { }

<< Response:
 2.01 Created
 ETag: "xyzzy"
 Content-format: TBD-application/rats-attested-resource-CT
 Payload:
 {
     "r" : {
         "typ": "text/plain",
         "val": "foobar"
     },
     "t_A": "2020-04-01T21:02:31Z",
     "E": "eyJhbGciO...z0ikw9Aa"
 }
```

* RP - Attester (CoAP) with GET

```
>> Request:
 GET coap://device.example/my-attested-resource
 Accept: TBD-application/rats-attested-resource-CT

<< Response:
 2.05 Content
 ETag: "xyzzy"
 Max-Age: 3600
 Content-format: TBD-application/rats-attested-resource-CT
 Payload:
 {
     "r" : {
         "typ": "text/plain",
         "val": "foobar"
     },
     "t_A": "2020-04-01T21:02:31Z",
     "E": "eyJhbGciO...z0ikw9Aa"
 }
```

* RP - Verifier (HTTP) is the same as Section 3.3.5.1.

3.3.5.3. Passport Model

* Attester - Verifier (CoAP)

```
     >> Request:
      POST coap://verifier.example/my-verify
      Content-Format: application/rats-attestation-result-request
      Accept: application/rats-attestation-result-response
      Payload:
      {
          "E": "eyJhbGciO...RfrKmTWk"
      }

     << Response:
      2.01 Created
      ETag: "jkllk"
      Content-format: application/rats-attestation-result-response
      Payload:
      {
          "R": "eyJhbGciO...Z0IKW9aA"
      }
```

* Relying Party - Attester (CoAP) with POST

```
     >> Request:
      POST coap://device.example/my-attested-resource
      Content-Format: TBD-application/rats-attested-resource-request-CT
      Accept: TBD-application/rats-attested-resource-CT
      Payload:
      { }

     << Response:
      2.01 Created
      ETag: "qwerty"
      Content-format: TBD-application/rats-attested-resource-CT
      Payload:
      {
           "r": {
               "type": "text/plain",
               "val": "foobar"
           },
           "t_A": "2020-04-01T21:02:31Z",
           "E": "eyJhbGciO...RfrKmTWk",
           "R": "eyJhbGciO...Z0IKW9aA"
       }
```

* Relying Party - Attester (CoAP) with GET

```
>> Request:
 GET coap://device.example/my-attested-resource
 Accept: TBD-application/rats-attested-resource-CT

<< Response:
 2.05 Content
 ETag: "qwerty"
 Max-Age: 3600
 Content-format: TBD-application/rats-attested-resource-CT
 Payload:
 {
      "r": {
          "type": "text/plain",
          "val": "foobar"
      },
      "t_A": "2020-04-01T21:02:31Z",
      "E": "eyJhbGciO...RfrKmTWk",
      "R": "eyJhbGciO...Z0IKW9aA"
  }
```

4. Discovery

4.1. Resource Directory

The following describes the new link format attribute values needed for registering attested resources as well as verification endpoints to a Resource Directory [I-D.ietf-core-resource-directory].

The same attribute values can be used by RD clients to discover attestation related resources.

4.1.1. Attested Resource Registration

An attested resource is registered with:

* an interface description (if=) with value "rats.if.timestamp" or "rats.if.nonce" depending on the supported freshness model, which determines the access method (i.e., POST+nonce vs GET);
* a content format (ct=) with value "TBD-application/rats-attested-resource-CT";
* an inner content format (ict=) that reflects the "type" field of the returned "resource";
* a resource type (rt=) that reflects the nature of the inner resource.

If a resource has both a "plain" and an "attested" variant, then the link value corresponding to the "attested" resource can be associated to its "plain" twin by means of the link relationship "attested-variant".

TBD: Should we have rats.if.timestamp variants for GET and POST? Alternative includes: 1) let the client probe and server return 405/4.05 if the requested variant is not supported; 2) add another attribute that explicitly states which request methods are supported.

4.1.1.1. Examples

The following example shows a registrant endpoint with the name "node1" registering an attested heart rate sensor resource to an RD.

The location /rd is an example RD location discovered in a previous .well-known/core query.

```
  >> Request:
    POST /rd?ep=node1 HTTP/1.1
    Host: rd.example
    Content-Type: application/link-format

    </sensors/attested-heartrate>;
      if="rats.if.timestamp";
      rt="heart-rate-zoladz";
      ct=TBD-application/rats-attested-resource-CT;
      ict=0

  << Response:
    HTTP/1.1 201 Created
    Location: /rd/4520
```

The following example shows a registrant endpoint with the name "node1" registering a temperature sensor resource along with its attested twin to an RD.

The "attested-variant" link relation establishes the semantics of the link between /sensors/temp and /sensors/attested-temp: the latter being an attested version of the former. Note, in particular, that the resource type (rt=) of the linked resource is inherited by the attested twin. Missing an explicit inner content format (ict=) the content type of the inner resource representation can be assumed to be that of the linked resource. The interface description (if=) "rats.if.nonce" says that the access to the attested resource happens by supplying a nonce through a POST.

```
     >> Request:
       POST /rd?ep=node1 HTTP/1.1
       Host: rd.example
       Content-Type: application/link-format

       </sensors/temp>;
         ct=41;
         rt="temperature-c";
         if="sensor",
       </sensors/attested-temp>;
         anchor="/sensors/temp";
         rel="attested-variant";
         if="rats.if.nonce";
         ct=TBD-application/rats-attested-resource-CT;
         ict=41

     << Response:
       HTTP/1.1 201 Created
       Location: /rd/4521
```

4.1.2. Verifier Resource Registration

A Verifier resource is registered with:

* An "rt" with value "rats.verifier";
* A "ct" with value "TBD-application/rats-attestation-result-response-CT"

4.1.2.1. Examples

```
     >> Request:
       POST /rd?ep=node1 HTTP/1.1
       Host: rd.example
       Content-Type: application/link-format

       </my-verifier>;
         ct=application/rats-attestation-result-response;
         rt="rats.verifier"

     << Response:
       HTTP/1.1 201 Created
       Location: /rd/4522
```

5. IANA Considerations

TODO

6. Privacy Considerations

TODO

7. Security Considerations

7.1. Model Architecture for the Origin

The model architecture for the origin of the attested resource is illustrated in Figure 7. The REST client (an user agent of a relying party or verifier) interfaces directly with a REST front-end (a CoAP or HTTP server stack) running in the Rich Execution Environment (REE), for example a Linux operating system. The REST front-end is paired with a back-end Trusted Application (TA) running in the Trusted Execution Environment (TEE). The TA has exclusive control over some "resource" (e.g., a sensor that feeds back into some kind of critical infrastructure control system) and can talk to the attestation service hosted inside the TEE to request EAT tokens.

In this model, it is critical that the attestation service can only be used by the intended TA or, failing that, that the identity of the calling TA can be securely proved to the relying party or verifier. An example of the latter is the Client ID claim used in PSA attestation [I-D.tschofenig-rats-psa-token].

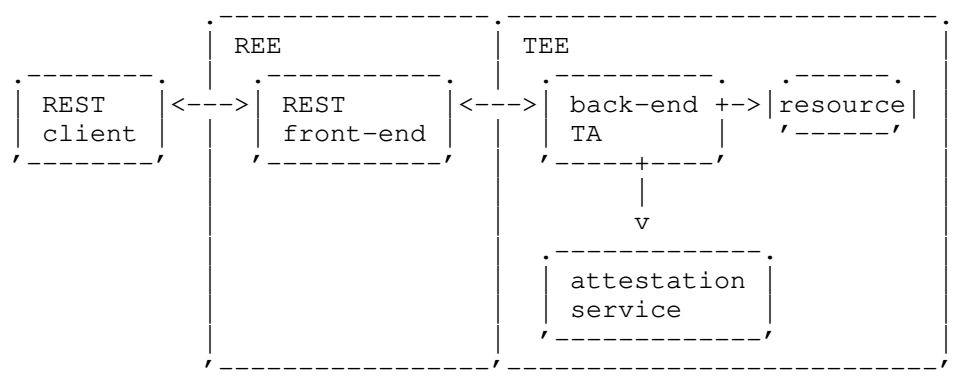


Figure 7: Model Security Architecture

Acknowledgments

TBD

References

Normative References

[I-D.ietf-core-resource-directory]
Shelby, Z., Koster, M., Bormann, C., Stok, P., and C. Amsuess, "CoRE Resource Directory", Work in Progress, Internet-Draft, draft-ietf-core-resource-directory-24, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-core-resource-directory-24.txt>.

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-04, 21 May 2020, <http://www.ietf.org/internet-drafts/draft-ietf-rats-architecture-04.txt>.

[I-D.ietf-rats-eat]
Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-03, 20 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-rats-eat-03.txt>.

[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>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[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>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <https://www.rfc-editor.org/info/rfc7320>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[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>.

Informative References

[Fielding] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation, University of California, Irvine, 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

[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-02, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-birkholz-rats-tuda-02.txt>.

[I-D.tschofenig-rats-psa-token] Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", Work in Progress, Internet-Draft, draft-tschofenig-rats-psa-token-05, 6 March 2020, <http://www.ietf.org/internet-drafts/draft-tschofenig-rats-psa-token-05.txt>.

Authors' Addresses

Adrian Shaw
arm

Email: Adrian.Shaw@arm.com

Hannes Tschofenig
arm

Email: Hannes.Tschofenig@arm.com

Sergei Trofimov
arm

Email: Sergei.Trofimov@arm.com

Simon Frost
arm

Email: Simon.Frost@arm.com

Thomas Fossati
arm

Email: Thomas.Fossati@arm.com

RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 27, 2020

E. Voit
Cisco

June 25, 2020

Trusted Path Routing
draft-voit-rats-trustworthy-path-routing-00

Abstract

There are end-users who believe encryption technologies like IPSec alone are insufficient to protect the confidentiality of their highly sensitive traffic flows. These end-users want their flows to traverse devices which have been freshly appraised and verified. This specification describes Trusted Path Routing. Trusted Path Routing protects sensitive flows as they transit a network by forwarding traffic to/from sensitive subnets across network devices recently appraised as trustworthy.

Table of Contents

1. Introduction 2
2. Terminology 3
2.1. Terms 3
2.2. Requirements Notation 4
3. Protocol Independent Definitions 4
3.1. Trusted Path Routing Service 4
3.2. Network Topology Assembly 4
3.3. Link Appraisal 5
3.4. Trustworthiness Vector 6
3.5. Attestation Results 8
3.6. Stamped Passport 9
3.7. Appraising the Stamped Passport 11
4. Implementable Solution 13
4.1. Prerequisites 13
4.2. Protocol Bindings 14
5. YANG Module 16
6. Security Considerations 22
7. References 23
7.1. Normative References 23
7.2. Informative References 24
Appendix A. Acknowledgements 24
Appendix B. Change Log 25
Appendix C. Open Questions 25
Author's Address 26

1. Introduction

There are end-users who believe encryption technologies like IPSec alone are insufficient to protect the confidentiality of their highly sensitive traffic flows. These customers want their highly sensitive flows to be transported over only network devices recently verified as trustworthy.

With the inclusion of TPM based cryptoprocessors into network devices, it is now possible for network providers to identify potentially compromised devices as well as potentially exploitable (or even exploited) vulnerabilities. Using this knowledge, it then becomes possible to redirect sensitive flows around these devices.

Trusted Path Routing provides a method of establishing Trusted Topologies which only include trust-verified network devices. Membership in a Trusted Topology is established and maintained via an

exchange of Stamped Passports at the link layer between peering network devices. As links to Attesting Devices are appraised as meeting at least a minimum set of formally defined Trustworthiness Levels, the links are then included as members of this Trusted Topology. Routing protocols like [I-D.ietf-lsr-flex-algo] can then used to propagate topology state throughout a network. IP Packets to and from end-user designated Sensitive Subnets are then forwarded into this Trusted Topology at each network boundary.

The specification works under the following assumptions:

- All network devices supports the TPM remote attestation profile as laid out in [RATS-Device]
- A routing protocol capable of maintaining multiple topologies connects the network devices which span the network domain.
- One or more Verifiers continuously appraise the set of network devices in that network domain, and these Verifiers can return the Attestation Results back to the attesting network device.

2. Terminology

2.1. Terms

The following terms are imported from [RATS-Arch]: Attester, Evidence, Passport, Relying Party, and Verifier.

Newly defined terms for this document:

Attested Device - a device where a Verifier's most recent appraisal of Evidence has returned a Trustworthiness Vector.

Stamped Passport - a bundle of Evidence which includes at least signed Attestation Results from a Verifier, and two independent TPM quotes from an Attester.

Sensitive Subnet - an IP address range where IP packets to or from that range must only have their IP headers and encapsulated payloads accessible/visible only by Attested Devices.

Transparently-Transited Device - a network device within an IGP domain where any packets passed into that IGP domain are completely opaque at Layer 3 and above.

Trusted Topology - a topology which includes only Attested Devices and Transparently-Transited Devices.

Trustworthiness Level - a specific quanta of trustworthiness which can be assigned by a Verifier.

Trustworthiness Vector - a set of Trustworthiness Levels assigned during a single assessment cycle by a Verfier using Evidence and Claims related to an Attested Device. The vector is included within Attestation Results.

2.2. 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.

3. Protocol Independent Definitions

3.1. Trusted Path Routing Service

An end user identifies sensitive IP subnets where flows with applications using these IP subnets need enhanced privacy guarantees. Trusted Path Routing passes flows to/from these Sensitive Subnets over a Trusted Topology able to meet these guarantees. The Trusted Topology itself consists of the interconnection of network devices where each potentially transited device has passed a recent trustworthiness appraisal.

Different guarantees of end-to-end trustworthiness appraisal may be offered to network users. These guarantees are network operator specific, but might include options such as:

- all transited devices are currently boot integrity verified
- all transited devices are from a specific set of vendors and are running known software containing the latest patches
- no guarantees provided

3.2. Network Topology Assembly

To be included in a Trusted Topology, Evidence of trustworthiness is shared between network device peers (such as routers). Upon receiving and appraising this Evidence as part of link layer authentication, the network device peer decides if this link should be added as an active adjacency for the Trusted Topology.

When enough links have been successfully added, a Trusted Topology will come into existence as routing protocols flood the adjacency information across the network domain.

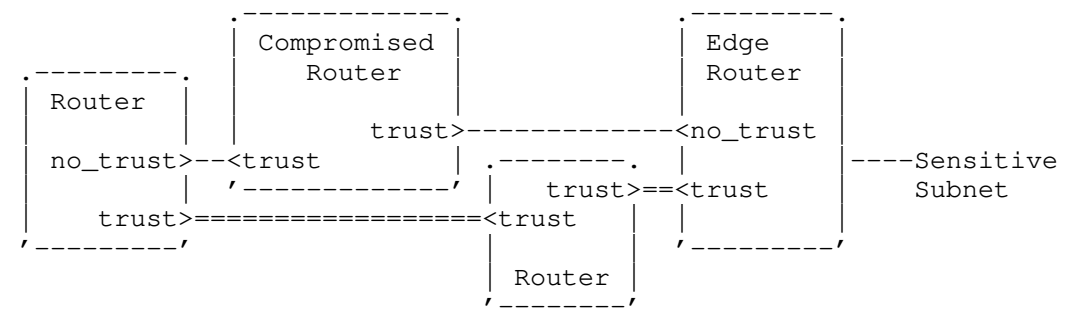


Figure 1: Trusted Path Topology Assembly

Traffic exchanged with Sensitive Subnets can then be forwarded into that Trusted Topology from all edges of the network domain.

3.3. Link Appraisal

Critical to the establishment and maintenance of a Trusted Topology is the Stamped Passport. A Stamped Passport is comprised of Evidence from both an Attester and a Verifier. Stamped Passports are exchanged in both directions between peering network devices over a link layer protocols like 802.1x or MACSEC. As link layer protocols will continuously re-authenticate the link, this allows fresh Evidence to be constantly appraised by either side of the connection.

Each Stamped Passport will include the most recent Verifier provided Attestation Results, as well as the most recent TPM Quote for that Attester. Upon receiving this information as part of link layer authentication, the Relying Party Router appraises the results and decides if this link should be added to a Trusted Topology.

Figure 2 describes this flow of information using the time definitions described in [RATS-Arch], and the information flows defined in Section 7 of [RATS-Interactions].

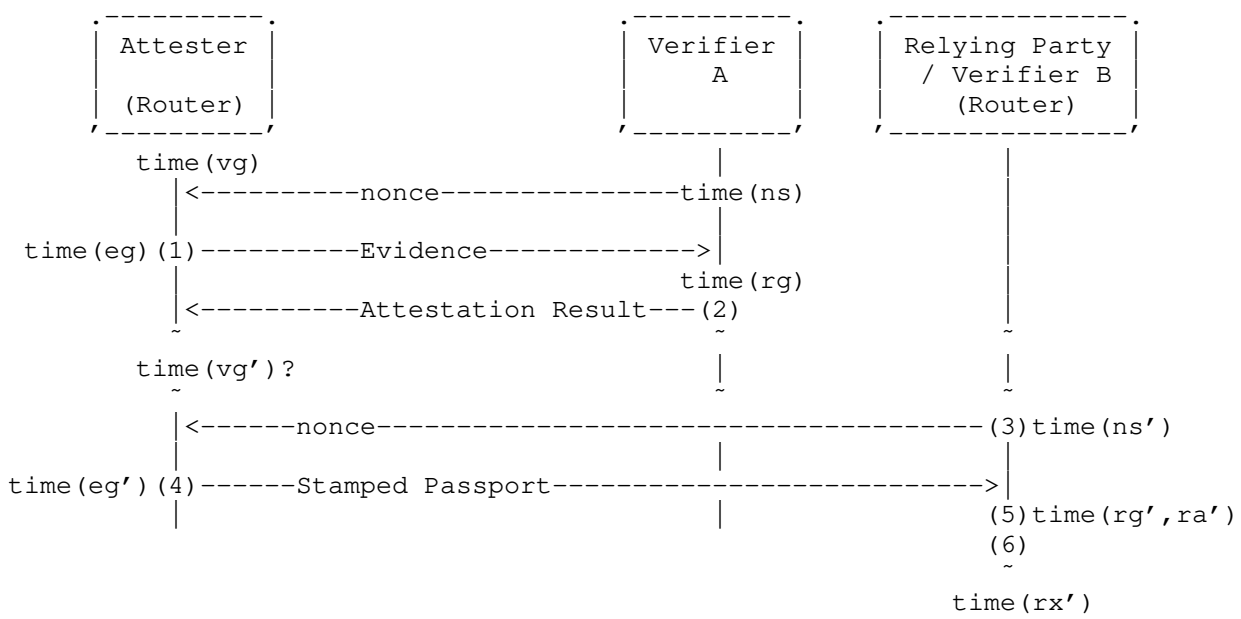


Figure 2: Trusted Path Timing

Specific of each of these information flows, included what happens at the items numbered (1) through (5) are described in Section 3.6.

3.4. Trustworthiness Vector

For Trusted Path Routing to operate, fresh Attestation Results need to be communicated by a Verifier back to the Attester. These Attestation Results must be encoded in a way which is known and actionable.

To support this requirement, specific levels of appraised trustworthiness have been defined; it is these Trustworthiness Levels which are asserted as Attestation Results by a Verifier. It is out of the scope of this document for the Verifier to provide proof or logic on how the assertion was derived.

Following are the set of available Trustworthiness Levels:

Trustworthiness Level	Definition
hw-authentic	A Verifier has appraised an Attester as having authentic hardware
fw-authentic	A Verifier has appraised an Attester as having authentic firmware
hw-verification-fail	A Verifier has appraised an Attester has failed its hardware or firmware verification
identity-verified	A Verifier has appraised and verified an Attester's unique identity
identity-fail	A Verifier has been unable to assess or verify an Attester's unique identity
boot-verified	A Verifier has appraised an Attester as Boot Integrity Verified
boot-verification-fail	A Verifier has appraised an Attester has failed its Boot Integrity verification
files-verified	A Verifier has appraised an Attester's file system, and asserts that it recognizes relevant files
file-blacklisted	A Verifier has found a file on an Attester which should not be present

A quick look at the list above shows that multiple Trustworthiness Level will often be applicable at single point in time. To support this, the Attestation Results will include a single Trustworthiness Vector consisting of a set of Trustworthiness Levels. The establishment of this Trustworthiness Vector follows the following logic on the Verifier:

```
Start: TPM Quote Received, log received, or appraisal timer expired

Step 0: set Trustworthiness Vector = Null

Step 1: Is there sufficient fresh signed evidence to appraise?
   (yes) - No Action
   (no) -  Goto Step 6

Step 2: Appraise Hardware Integrity
   (if hw-verification-fail) - push onto vector, go to Step 6
   (if hw-authentic) - push onto vector
   (if fw-authentic) - push onto vector
   (if not evaluated, or insufficient data to conclude: take no action)

Step 3: Appraise attester identity
   (if identity-verified) - push onto vector
   (if identity-fail) - push onto vector
   (if not evaluated, or insufficient data to conclude: take no action)

Step 4: Appraise boot integrity
   (if boot-verified) - push onto vector
   (if boot-verification-fail) - push onto vector
   (if not evaluated, or insufficient data to conclude: take no action)

Step 5: Appraise filesystem integrity
   (if files-verified) - push onto vector
   (if file-blacklisted) - push onto vector
   (if not evaluated, or insufficient data to conclude: take no action)

Step 6: Assemble Attestation Results, and push to Attester

End
```

3.5. Attestation Results

As Evidence changes, a new Trustworthiness Vector needs to be returned to the Attester as Attestation Results. But this Trustworthiness Vector is not all that needs to be returned. Following is a YANG tree for all the returned objects. Each of these objects will later be used as Evidence by another Verifier which is co-resident with the Relying Party.

```
module: ietf-attestation-results-vector
  +--rw attestation-results!
     +--rw trustworthiness-vector*        identityref
     +--rw (tpm-specification-version)?
     |  +--:(TPM2.0) {tpm:TPM20}?
     |  |  +--rw TPM2B_DIGEST              binary
     |  |  +--rw pcr-list* [TPM2_Algo]
     |  |  |  +--rw TPM2_Algo    identityref
     |  |  |  +--rw pcr-index*   tpm:pcr
     |  |  +--rw clock                     uint64
     |  |  +--rw reset-counter             uint32
     |  |  +--rw restart-counter           uint32
     |  |  +--rw safe                      boolean
     |  +--:(TPM1.2) {tpm:TPM12}?
     |     +--rw pcr-index*                pcr
     |     +--rw tpm12-pcr-value*          binary
     |     +--rw timestamp                 yang:date-and-time
     +--rw public-key-format               identityref
     +--rw public-key                      binary
     +--rw public-key-algorithm-type       identityref
     +--rw verifier-signature-key-name?    string
     +--rw verifier-signature              binary
```

Figure 3: Attestation Results Tree

Looking at the objects above, if the Attester has a TPM2, then the values of the TPM PCRs are included (i.e., <TPM2B_DIGEST>, <TPM2_Algo>, and <pcr-index>), as are the timing counters from the TPM (i.e., <clock>, <reset-counter>, <restart-counter>, and <safe>).

Likewise if the Attester has a TPM1.2, the TPM PCR values of the <pcr-index> and <pcr-value> are included. Timing information comes from the Verifier itself via the <timestamp> object.

For both the TPM1.2 and the TPM2, there are other Attestation Results which are sent. These are the Attester's TPM key (i.e., <public-key>, <public-key-format>, and <public-key-algorithm-type>). This key later will allow the Relying Party router to appraise a subsequent TPM Quote. It is this signature which allows the Trustworthiness Vector to be later provably associated with a recent TPM Quote.

3.6. Stamped Passport

The Attestation Results are not the only item which a Relying Party needs to consider during its appraisal. A provably recent TPM Quote from the Attester must also be included. With these two items, the

resulting Stamped Passports formats described below must be converted and passed over EAP. If an Attester includes a TPM2, the objects are:

```
    YANG structure for a TPM2 Stamped Passport
       +--ro latest-tpm-quote
       |  +--ro quote              binary
       |  +--ro quote-signature    binary
       +--ro latest-attestation-results
          +--ro trustworthiness-vector*        identityref
          +--ro TPM2B_DIGEST                   binary
          +--ro pcr-list* [TPM2_Algo]
          |  +--ro TPM2_Algo    identityref
          |  +--ro pcr-index*   tpm:pcr
          +--ro clock                          uint64
          +--ro reset-counter                  uint32
          +--ro restart-counter                uint32
          +--ro safe                           boolean
          +--ro public-key-format              identityref
          +--ro public-key                     binary
          +--ro public-key-algorithm-type      identityref
          +--ro verifier-signature-key-name?   string
          +--ro verifier-signature             binary
```

And if the Attester is a TPM1.2, the object are:

```
    YANG structure for a TPM1.2 Stamped Passport
       +--ro latest-tpm-quote
       |  +--ro version* []
       |  |  +--ro major?      uint8
       |  |  +--ro minor?      uint8
       |  |  +--ro revMajor?   uint8
       |  |  +--ro revMinor?   uint8
       |  +--ro digest-value?   binary
       +--ro latest-tpm12-attestation-results
          +--ro trustworthiness-vector*        identityref
          +--ro pcr-index*                     pcr
          +--ro tpm12-pcr-value*               binary
          +--ro timestamp                      yang:date-and-time
          +--ro public-key-format              identityref
          +--ro public-key                     binary
          +--ro public-key-algorithm-type      identityref
          +--ro verifier-signature-key-name?   string
          +--ro verifier-signature             binary
```

With either of these passport formats, if the <latest-tpm-quote> is verifiably fresh, then the state of the Attester can be appraised by a network peer.

3.7. Appraising the Stamped Passport

When it receives a Stamped Passport, a Verifier co-resident with the Relying Party on a network peer can make nuanced decisions about how to handle traffic coming from that link. For example, when the Attester's TPM hardware identity credentials can be verified, it might choose to accept link layer connections and forward generic Internet traffic.

Additionally, if the Attester's Trustworthiness Vector is acceptable to the Relying Party, and it hasn't been too long since the Verifier has provided a Stamped Passport, the Relying Party can include that link in a Trusted Topology.

As the process described above repeats across the set of links within a network domain, Trusted Topologies can be extended and maintained. Traffic to and from Sensitive Subnets is then identified at the edges of the network domain and passed into this Trusted Topology.

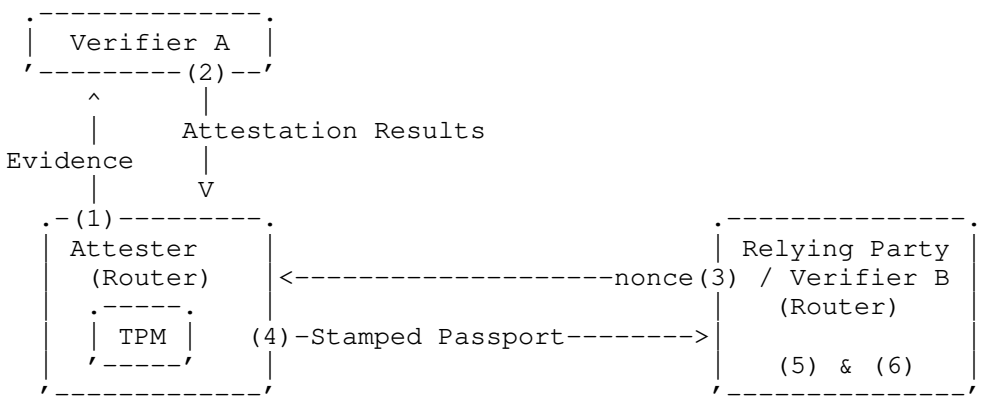


Figure 4: Stamped Passport Generation and Appraisal

In Figure 4 above, Evidence from a TPM is generated and signed by that TPM. This Evidence is appraised by Verifier A, and the Attester is given a Trustworthiness Vector which is signed and returned as Attestation Results to the Attester. Later, when a request comes in from a Relying Party, the Attester assembles and returns three independently signed elements of Evidence. These three comprise the Stamped Passport which when taken together allow Verifier B to appraise and set the current Trustworthiness Vector of the Attester.

More details on the mechanisms used in the construction and verification of the Stamped Passport are listed below. These numbers match to the numbered steps of Figure 4:

1. An Attester sends a signed TPM Quote which includes PCR measurements to Verifier A at time(eg).

2. Verifier A appraises (1), then sends the following items back to that Attester as Attestation Results:

    1. the Trustworthiness Vector of an Attester,

    2. the PCR state information from the TPM Quote of (1),

    3. time information associated with the TPM Quote of (1),

    4. the Public Attestation Key which it used to validate the TPM Quote of (1), and

    5. a Verifier signature across (2.1) though (2.4).

3. At time(eg') a nonce known to the Relying Party is sent to the Attester .

4. The Attester generates and sends a Stamped Passport. This Stamped Passport includes:

    1. The Attestation Results from (2)

    2. New signed, verifiably fresh PCR measurements from time(eg'), which incorporates the nonce from (3).

5. On receipt of (4), the Relying Party makes its determination of how the Stamped Passport will impact adjacencies within a Trusted Topology. The decision process is:

    1. Verify that (4.2) includes the nonce from (3).

    2. Use a local certificate to validate the signature (4.1).

    3. Use the Attestation Results provided public key info of (2.4) to validate the signatures of (4.2).

    4. Failure of (5.1) through (5.3) means the link does not meet minimum validation criteria, therefore appraise the link as having a null Trustworthiness Vector. Jump to step (6).

    5. If all PCR values from (2.2) equal those (4.2), then Relying Party can accept (2.1) as the link's Trustworthiness Vector. Jump to step (6).

6. If the PCR state information of (2.2) doesn't equal (4.2), and not much time has passed between time(eg) and time(eg'), the Relying Party accepts any previous Trustworthiness Vector. (Note: rather than accepting, it is also viable to attempt to acquire a new Stamped Passport. Where [stream-subscription] is used, it should only be a few seconds before a new Attestation Results are delivered to an Attester via (2).)

7. When the PCR state information is different, and there is a large or uncertain time gap between time(eg) and time(eg'), the link should be assigned a null Trustworthiness Vector.

6. Take action based on Verifier B's appraised Trustworthiness Vector:

1. Include the link within any Trusted Topology for which that Trustworthiness Vector is qualified.

2. Remove the link from any Trusted Topology for which that Trustworthiness Vector is not qualified.

4. Implementable Solution

This section defines one set of protocols which can be used for Trusted Path Routing. The protocols include [MACSEC] or [IEEE-802.1X], ISIS [I-D.ietf-lsr-flex-algo], YANG subscriptions [RFC8639], and [RFC3748] methods. Other alternatives are also viable.

4.1. Prerequisites

- A Trusted Topology such as one established by ISIS exists in an IGP domain for the forwarding of Sensitive Subnet traffic. This Topology will carry traffic across a set of devices which currently meet at a defined set of Trustworthiness Vectors.

- Customer designated Sensitive Subnets and their requested Trustworthiness Vectors have been identified and associated with external interfaces to/from the edge of a network. Traffic to a Sensitive Subnet can be passed into the Trusted Topology.

- Verifiers A and B are able to verify [TPM1.2] or [TPM2.0] signatures of an Attester.

- Verifier B trusts information signed by Verifier A. Verifier B has also been pre-provisioned with certificates or public keys necessary to confirm that Stamped Passports came from Verifier A

- Within a network, a Relying Party is able to use affinity to include/exclude links as part of the Trusted Topology based on this appraisal.

4.2. Protocol Bindings

The numbering in below matches to the steps in Figure 4.

Step (1)

There are two alternatives for Verifier A to acquires Evidence including a TPM Quote from the Attester:

- Subscription to the <attestation> stream defined in [stream-subscription]. Note: this method is recommended as it will minimize the interval between when a PCR change is made in a TPM, and when the PCR change appraisal is incorporated within a subsequent Stamped Passport.

- The RPCs <tpm20-challenge-response-attestation> or <tpm12-challenge-response-attestation> defined in device [RATS-YANG]

Step (2)

The delivery of these Attestation Results back to the Attester MAY be done via an operational datastore write to the YANG module <ietf-attestation-results-vector>.

Step (3)

At time(ns') a Relying Party makes a Link Layer authentication request to an Attester via a either [MACSEC] or [IEEE-802.1X]. This connection request must include [RFC3748] credentials. Specifics of the EAP mapping to the Stamped Passport is tbd.

Step (4)

Upon receipt of (3), a Stamped Passport is generated as per Section 3.6, and sent to the Relying Party. Note that with [MACSEC] or [IEEE-802.1X], steps (3) & (4) will repeat periodically independently of any subsequent iteration (1) and (2). This allows for periodic reauthentication of the link layer in a way not bound to the updating of Verifier A's Attestation Results.

Step (5)

Upon receipt of (4), the Relying Party appraises the Stamped Passport as per Section 3.6. Following are relevant mappings which replace

generic steps from Section 3.6 with specific objects available with a TPM1.2 or TPM2.0.

TPM2.0 - Bindings/details
(5.5): If the <TPM2B_DIGEST>, <TPML_PCR_SELECTION>, <reset-counter>, <restart-counter> and <safe> are equal between the Attestation Results and the TPM Quote at time(eg') then Relying Party can accept (2.1) as the link's Trustworthiness Vector. Jump to step (6). (5.6): If the <reset-counter>, <restart-counter> and <safe> are equal between the Attestation Results and the TPM Quote at time(eg'), and the <clock> object from time(eg') has not incremented by an unacceptable number of seconds since the Attestation Result, then Relying Party can accept (2.1) as the link's Trustworthiness Vector. Jump to step (6). (5.7): Assign the link a null Trustworthiness Vector.

TPM1.2 - Bindings/details
(5.5): If the <pcr-index>'s and <tpm12-pcr-value>'s are equal between the Attestation Results and the TPM Quote at time(eg'), then Relying Party can accept (2.1) as the link's Trustworthiness Vector. Jump to step (6). (5.6): If the time hasn't incremented an unacceptable number of seconds from the Attestation Results <timestamp> and the system clock of the Relying Party, then Relying Party can accept (2.1) as the link's Trustworthiness Vector. Jump to step (6). (5.7): Assign the link a null Trustworthiness Vector.

Step (6)

After the Trustworthiness Vector has been validated or reset, based on the link's Trustworthiness Vector, the Relying Party may adjust the link affinity of the corresponding ISIS [I-D.ietf-lsr-flex-algo] topology. ISIS will then replicate the link state across the IGP domain. Traffic will then avoid links which do not have a qualifying Trustworthiness Vector.

5. YANG Module

This YANG module imports modules from [RATS-YANG], [crypto-types] and [RFC6021].

```
<CODE BEGINS> ietf-attestation-results-vector@2020-06-23.yang
module ietf-attestation-results-vector {
  yang-version 1.1;
  namespace
     "urn:ietf:params:xml:ns:yang:ietf-attestation-results-vector";
  prefix arv;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-tpm-remote-attestation {
    prefix tpm;
    reference
      "draft-ietf-rats-yang-tpm-charra";
  }
  import ietf-asymmetric-algs {
    prefix aa;
  }
  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC XXXX: Common YANG Data Types for Cryptography
      (currently draft-ietf-netconf-crypto-types)";
  }

  organization "IETF";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rats/>
     WG List:  <mailto:rats@ietf.org>

     Editor:   Eric Voit
               <mailto:evoit@cisco.com>";

  description
    "This module contains conceptual YANG specifications for
    subscribing to attestation streams being generated from TPM chips.

    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; see the RFC
    itself for full legal notices.";

  revision 2020-06-23 {
    description
      "Initial version.";
    reference
      "draft-voit-rats-trustworthy-path-routing";
  }


  /*
   * IDENTITIES
   */

  identity trustworthiness-level {
    description
      "Base identity for a Verifier that uses its Appraisal Policy for
      Evidence to establish a trustworthiness level.";
  }

  identity trustworthiness-pass {
    description
      "A trustworthiness-level which successfully meets an Appraisal Policy for
      Evidence.";
  }

  identity trustworthiness-fail {
    description
      "A trustworthiness-level which hit Appraisal Policy for Evidence
      necessary to fail an evaluation.  Note: this failure might or might not
      consider whether sufficient Evidence has been provided.  In other words
      having insufficient evidence might not drive the setting of this failing
      trustworthiness-level.";
  }

  identity boot-verified {
    base trustworthiness-pass;
    description
      "A Verifier has appraised an Attester as Boot Integrity Verified.";
  }

  identity boot-verification-fail {
    base trustworthiness-fail;
    description
```

```
      "A Verifier has appraised an Attester has failed its Boot Integrity
      verification.";
  }

  identity hw-authentic {
    base trustworthiness-pass;
    description
      "A Verifier has appraised an Attester as having authentic hardware.";
  }

  identity fw-authentic {
    base trustworthiness-pass;
    description
      "A Verifier has appraised an Attester as having authentic firmware.";
  }

  identity hw-verification-fail {
    base trustworthiness-fail;
    description
      "A Verifier has appraised an Attester has failed its hardware or
      firmware verification.";
  }
  identity identity-verified {
    base trustworthiness-pass;
    description
      "A Verifier has appraised and verified an Attester's unique identity.";
  }

  identity identity-fail {
    base trustworthiness-fail;
    description
      "A Verifier has been unable to assess or verify an Attester's unique
      identity";
  }

  identity files-verified {
    base trustworthiness-pass;
    description
      "A Verifier has appraised an Attester's file system, and asserts that
      it recognizes relevant files.";
  }

  identity file-blacklisted {
    base trustworthiness-fail;
    description
      "A Verifier has found a file on an Attester which should not be
      present.";
  }
```

```
  grouping TPM20-unsigned-internals {
    description
      "The unsigned extract of a TPM2 Quote.";
    leaf TPM2B_DIGEST {
      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
        identified within TPML_PCR_SELECTION.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
        TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.12.1";
    }
    uses tpm:tpm20-pcr-selection {
      description
        "Specifies the list of PCRs and Hash Algorithms used for the
        latest returned TPM2B_DIGEST. Identifying
        this object simplifies Stamped Passport troubleshooting if the
        same PCRs and Hash algorithms are not used when attempting to
        correlate independent TPM2B_DIGESTs.";
    }
    leaf clock {
      mandatory true;
      type uint64;
      description
        "Clock is a monotonically increasing counter that advances whenever
         power is applied to a TPM2. The value of Clock is incremented each
         millisecond.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
         TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.11.2";
    }
    leaf reset-counter {
      mandatory true;
      type uint32;
      description
        "This counter increments on each TPM Reset.  The most common
        TPM Reset would be due to a hardware power cycle.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
         TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.11.3";
    }
    leaf restart-counter {
      mandatory true;
      type uint32;
      description
        "This counter shall increment by one for each TPM Restart or
```

```
        TPM Resume. The restartCount shall be reset to zero on a TPM
        Reset.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
         TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.11.4";
    }
    leaf safe {
      mandatory true;
      type boolean;
      description
        "This parameter is set to YES when the value reported in Clock
        is guaranteed to be unique for the current Owner. It is set to
        NO when the value of Clock may have been reported in a previous
        attestation or access.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
        TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.11.5";
    }
  }

  grouping TPM12-unsigned-internals-extended {
    description
      "The unsigned extract of a TPM12 Quote, with extra content from the
      Verifier specific to a TPM12.";
    uses tpm:tpm12-pcr-selection;
    leaf-list tpm12-pcr-value {
      type binary;
      description
        "The list of TPM_PCRVALUEs from each PCR selected in sequence
        of tpm12-pcr-selection.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
         TPM-Main-Part-2-TPM-Structures_v1.2_rev116_01032011.pdf
         Section 10.9.7";
    }
    leaf timestamp {
      type yang:date-and-time;
      mandatory true;
      description
        "The timestamp of the Verifier's appraisal.  This can be used by
        a Relying Party to determine the freshness of the attestation
        results.";
    }
  }

  /*
   * DATA NODES
   */
```

```
  container attestation-results {
    presence
      "Indicates that Verifier has appraised the security posture of the
      Attester, and returned the results within this container.  If the
      Attester believes this information is no longer fresh, this container
      should automatically be deleted.";
    description
      "Retains the most recent Attestation Results for this Attester.
      It must only be written by a Verfier which is to be trusted by a
      Relying Party.";
    leaf-list trustworthiness-vector {
      type identityref {
        base trustworthiness-level;
      }
      ordered-by system;
      description
        "One or more Trustworthiness Levels assigned which expose the
        Verifier's evaluation of the Evidence associated with the
        'tpmt-signature'.";
    }
    choice tpm-specification-version {
      description
        "Identifies the cryptoprocessor API set which drove the Attestation
        Results.";
      case TPM2.0 {
        if-feature "tpm:TPM20";
        description
          "The Attestation Results are from a TPM2.";
        uses TPM20-unsigned-internals;
      }
      case TPM1.2 {
        if-feature "tpm:TPM12";
        description
          "The most recent Attestation Results from a TPM1.2.";
        uses TPM12-unsigned-internals-extended;
      }
    }
    uses ct:public-key-grouping {
      description
        "In order to avoid having to provision AIK certificates on a Relying
        Party network device, it is possible to send the AIK public key as
        from the Verifier as part of the passport. This is safe because the
        key is signed by the Verifier (hence vouching for its validity.)
        The two objects within this group allow the Verifier to include this
        information as part of the Attestation Results.";
    }
    leaf public-key-algorithm-type {
      mandatory true;
```

```
      type identityref {
        base aa:asymmetric-algorithm-type;
      }
      description
        "Indicates what kind of algorithm is used with the Attester's
        Public Key Value.";
    }
    leaf verifier-signature-key-name {
      type string;
      description
        "Name of the key the Verifier used to sign the results.";
    }
    leaf verifier-signature {
      type binary;
      mandatory true;
      description
        "Signature of the Verifier across all the objects within the
        attestation-results container.  The signature will assume the
        sequence of objects as defined in the YANG model schema.";
    }
    leaf verifier-key-algorithm-type {
      mandatory true;
      type identityref {
        base aa:asymmetric-algorithm-type;
      }
      description
        "Indicates what kind of algorithm was used for the
        'verifier-signature'.";
    }
  }
}
<CODE ENDS>
```

6. Security Considerations

Verifiers are limited to the Evidence available for appraisal from a Router. Although the state of the art is improving, some exploits may not be visible via Evidence.

Only security measurements which are placed into PCRs are capable of being exposed via TPM Quote at time(eg')

Successful attacks on an Verifier have the potential of affecting traffic on the Trusted Topology.

For Trusted Path Routing, links which are part of the FlexAlgo are visible across the entire IGP domain. Therefore a compromised device will know when it is being bypassed.

Access control for the objects in Figure 3 should be tightly controlled so that it becomes difficult for the Stamped Passport to become a denial of service vector.

7. References

7.1. Normative References

[crypto-types]
"Common YANG Data Types for Cryptography", May 2020, <https://datatracker.ietf.org/doc/draft-ietf-netconf-crypto-types/>.

[RATS-Arch]
"Remote Attestation Procedures Architecture", March 2020, <https://tools.ietf.org/html/draft-ietf-rats-architecture-02>.

[RATS-YANG]
"A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs", June 2020, <https://datatracker.ietf.org/doc/draft-ietf-rats-yang-tpm-charra/>.

[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>.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <https://www.rfc-editor.org/info/rfc6021>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[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

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-07 (work in progress), April 2020.

[IEEE-802.1X]
Parsons, G., "802.1AE: MAC Security (MACsec)", January 2020, <https://standards.ieee.org/standard/802_1X-2010.html>.

[MACSEC] Seaman, M., "802.1AE: MAC Security (MACsec)", January 2006, <https://1.ieee802.org/security/802-1ae/>.

[RATS-Device]
"Network Device Remote Integrity Verification", n.d., <https://datatracker.ietf.org/doc/draft-ietf-rats-tpm-based-network-device-attest>.

[RATS-Interactions]
"Reference Interaction Models for Remote Attestation Procedures", June 2020, <https://ietf-rats.github.io/draft-birkholz-rats-reference-interaction-model/draft-birkholz-rats-reference-interaction-model.html#section-7>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[stream-subscription]
"Attestation Event Stream Subscription", June 2020, <https://datatracker.ietf.org/doc/draft-birkholz-rats-network-device-subscription>.

Appendix A. Acknowledgements

Shwetha Bhandari, Henk Birkholz, Chennakesava Reddy Gaddam, Sujal Sheth, Peter Psenak, Nancy Cam Winget, Ned Smith, Guy Fedorkow.

Appendix B. Change Log

[THIS SECTION TO BE REMOVED BY THE RFC EDITOR.]

v02-v00 of draft-voit-rats-trustworthy-path-routing-00

- file rename was due to an IETF tool submission glitch
- The Attester's AIK is included within the Stamped Passport. This eliminates the need to provision to AIK certificate on the Relying Party.
- Removed Centralized variant
- Added timing diagram, and moved content around to match

v01-v02 of draft-voit-rats-trusted-path-routing

- Extracted the attestation stream, and placed into draft-birkholz-rats-network-device-subscription
- Introduced the Trustworthiness Vector

v00-v01 of draft-voit-rats-trusted-path-routing

- Move all FlexAlgo terminology to Section 4.2. This allows Section 3.6 to be more generic.
- Edited Figure 1 so that (4) points to the egress router.
- Added text freshness mechanisms, and articulated configured subscription support.
- Minor YANG model clarifications.
- Added a few open questions which Frank thinks interesting to work.

Appendix C. Open Questions

(1) When there is no available Trusted Topology?

Do we need functional requirements on how to handle traffic to/from Sensitive Subnets when no Trusted Topology exists between IGP edges? The network typically can make this unnecessary. For example it is possible to construct a local IPSec tunnel to make untrusted devices appear as Transparently-Transited Devices. This way Secure Subnets could be tunneled between FlexAlgo nodes where an end-to-end path

doesn't currently exist. However there still is a corner case where all IGP egress points are not considered sufficiently trustworthy.

(2) Extension of the Stamped Passport?

We might move to 'verifier-certificate' and 'verifier-certificate-name' based on WG desire to include more information in the Stamped Passport. The format used could be extracted from ietf-keystore.yang, grouping keystore-grouping.

Author's Address

Eric Voit
Cisco Systems, Inc.

Email: evoit@cisco.com