RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 18, 2021

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

January 14, 2021

A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs
draft-ietf-rats-yang-tpm-charra-05

Abstract

This document defines a YANG RPC and a minimal datastore required to retrieve attestation evidence about integrity measurements from a device following the operational context defined in TPM-based Network Device Remote Integrity Verification. 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.

This Internet-Draft will expire on July 18, 2021.

Table of Contents

1. Introduction

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

primitive operation). Additionally, it is used as a root of trust for storage (RTS) in order to retain shielded secrets and store system measurements using a folding hash function (extend primitive operation).

1.1. Requirements notation

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

2. The YANG Module for Basic Remote Attestation Procedures

One or more TPMs MUST be embedded in the composite device that is providing attestation evidence via the YANG module defined in this document. The ietf-basic-remote-attestation YANG module enables a composite device to take on the role of Claimant and Attester in accordance with the Remote Attestation Procedures (RATS) architecture [I-D.ietf-rats-architecture] and the corresponding challenge-response interaction model defined in the [I-D.ietf-rats-reference-interaction-models] 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. YANG Modules

2.1.1. ietf-tpm-remote-attestation

This YANG module imports modules from [RFC6991], [RFC8348], [I-D.ietf-netconf-keystore], ietf-tcg-algs.yang Section 2.1.2.3.

2.1.1.1. Features

This module supports the following features:

<TPMs> - Indicates that multiple TPMs on the device can support remote attestation, This feature is applicable in cases where multiple line cards, each with its own TPM.

<bios> - Indicates the device supports the retrieval of bios event logs.

<ima> - Indicates the device supports the retrieval of Integrity Measurement Architecture event logs.

<netequip_boot> - Indicates the device supports the retrieval of netequip boot event logs.

2.1.1.2. Identities

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

2.1.1.3. RPCs

2.1.1.3.1. <tpm20-challenge-response-attestation>

This RPC allows a Verifier to request a quote of PCRs from a TPM2.0 compliant cryptoprocessor. Where the feature <TPMs> is active, and one or more <certificate-name> is not provided, all TPM2.0 compliant cryptoprocessors will respond. A YANG tree diagram of this RPC is as follows:

```
  +---x tpm20-challenge-response-attestation {taa:TPM20}?
     +---w input
     |  +---w tpm20-attestation-challenge
     |     +---w nonce-value            binary
     |     +---w tpm20-pcr-selection* []
     |     |  +---w TPM20-hash-algo?   identityref
     |     |  +---w pcr-index*         tpm:pcr
     |     +---w certificate-name*      certificate-name-ref {tpm:TPMs}?
     +--ro output
        +--ro tpm20-attestation-response* []
           +--ro certificate-name       certificate-name-ref
           +--ro TPMS_QUOTE_INFO        binary
           +--ro quote-signature?       binary
           +--ro up-time?               uint32
           +--ro unsigned-pcr-values* []
              +--ro TPM20-hash-algo?   identityref
              +--ro pcr-values* [pcr-index]
                 +--ro pcr-index    pcr
                 +--ro pcr-value?   binary
```

An example of an RPC challenge requesting PCRs 0-7 from a SHA256 bank could look like the following:

```
  <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <tpm20-challenge-response-attestation>
        xmlns="urn:ietf:params:xml:ns:yang:ietf-tpm-remote-attestation">
      <nonce>11010101011001101111100101001010100</nonce>
      <tpm20-pcr-selection>
        <TPM20-hash-algo
            xmlns:taa="urn:ietf:params:xml:ns:yang:ietf-tcg-algs">
          taa:TPM_ALG_SHA256
        </TPM20-hash-algo>
        <pcr-index>0</pcr-index>
        <pcr-index>1</pcr-index>
        <pcr-index>2</pcr-index>
        <pcr-index>3</pcr-index>
        <pcr-index>4</pcr-index>
        <pcr-index>5</pcr-index>
        <pcr-index>6</pcr-index>
        <pcr-index>7</pcr-index>
      </tpm20-pcr-selection>
    </tpm20-challenge-response-attestation>
  </rpc>
```

and a successful response might be formated as follows:

```
  <rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <tpm12-attestation-response
      xmlns="urn:ietf:params:xml:ns:yang:ietf-tpm-remote-attestation">
      <certificate-name
          xmlns:ks=urn:ietf:params:xml:ns:yang:ietf-keystore>
        ks:(instance of Certificate name in the Keystore)
      </certificate-name>
      <TPMS_QUOTE_INFO>
        (raw information from the TPM Quote, this includes a digest
        across the requested PCRs, the nonce, TPM2 time counters.)
      </TPMS_QUOTE_INFO>
      <quote-signature>
          (signature across TPMS_QUOTE_INFO)
      </quote-signature>
    </tpm12-attestation-response>
  </rpc-reply>
```

2.1.1.4. <tpm12-challenge-response-attestation>

This RPC allows a Verifier to request a quote of PCRs from a TPM1.2 compliant cryptoprocessor. Where the feature <TPMs> is active, and one or more <certificate-name> is not provided, all TPM1.2 compliant cryptoprocessors will respond. A YANG tree diagram of this RPC is as follows:

```
   +---x tpm12-challenge-response-attestation {taa:TPM12}?
      +---w input
      |  +---w tpm12-attestation-challenge
      |     +---w pcr-index*          pcr
      |     +---w nonce-value         binary
      |     +---w certificate-name*   certificate-name-ref {tpm:TPMs}?
      +--ro output
         +--ro tpm12-attestation-response* []
            +--ro certificate-name    certificate-name-ref
            +--ro up-time?            uint32
            +--ro TPM_QUOTE2?         binary
```

2.1.1.5. <log-retrieval>

This RPC allows a Verifier to acquire the evidence which was extended into specific PCRs. A YANG tree diagram of this RPC is as follows:

```
   +---x log-retrieval
      +---w input
      |  +---w log-selector* []
      |  |  +---w tpm-name*                  string
      |  |  +---w (index-type)?
      |  |  |  +--:(last-entry)
      |  |  |  |  +---w last-entry-value?    binary
      |  |  |  +--:(index)
      |  |  |  |  +---w last-index-number?   uint64
      |  |  |  +--:(timestamp)
      |  |  |     +---w timestamp?           yang:date-and-time
      |  |  +---w log-entry-quantity?        uint16
      |  +---w log-type        identityref
      +--ro output
         +--ro system-event-logs
            +--ro node-data* []
               +--ro tpm-name?     string
               +--ro up-time?      uint32
               +--ro log-result
                  +--ro (attested_event_log_type)
                     +--:(bios)
                     |  +--ro bios-event-logs
                     |     +--ro bios-event-entry* [event-number]
                     |        +--ro event-number    uint32
                     |        +--ro event-type?     uint32
                     |        +--ro pcr-index?      pcr
                     |        +--ro digest-list* []
                     |        |  +--ro hash-algo?   identityref
                     |        |  +--ro digest*      binary
                     |        +--ro event-size?     uint32
                     |        +--ro event-data*     uint8
```

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

2.1.1.6. Data Nodes

This section provides a high level description of the data nodes containing the configuration and operational objects with the YANG model. For more details, please see the YANG model itself in Section 2.1.1.7.

container <rats-support-structures> - This houses the set of information relating to a device's TPM(s).

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

```
+--rw tpms
   +--rw tpm* [tpm-name]
      +--rw tpm-name                string
      +--ro hardware-based?         boolean
      +--ro tpm-physical-index?     int32 {ietfhw:entity-mib}?
      +--ro tpm-path?               string
      +--ro compute-node            compute-node-ref {tpm:TPMs}?
      +--ro tpm-manufacturer?       string
      +--rw tpm-firmware-version    identityref
      +--rw TPM12-hash-algo?        identityref
      +--rw TPM12-pcrs*             pcr
      +--rw tpm20-pcr-bank* [TPM20-hash-algo]
      |  +--rw TPM20-hash-algo    identityref
      |  +--rw pcr-index*         tpm:pcr
      +--ro tpm-status              enumeration
      +--rw certificates
         +--rw certificate* [certificate-name]
            +--rw certificate-name           string
            +--rw certificate-keystore-ref?   -> /ks:keystore/asymmetric-keys/
asymmetric-key/certificates/certificate/name
            +--rw certificate-type?          enumeration
```

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

```
   +--rw attester-supported-algos
      +--rw tpm12-asymmetric-signing*   identityref {taa:TPM12}?
      +--rw tpm12-hash*                 identityref {taa:TPM12}?
      +--rw tpm20-asymmetric-signing*   identityref {taa:TPM20}?
      +--rw tpm20-hash*                 identityref {taa:TPM20}?
```

container <compute-nodes> - When there is more than one TPM supported, this container maintains the set of information related to the compute associated with a specific TPM. This allows each specific TPM to identify on which <compute-node> it belongs.

```
   +--rw compute-nodes {tpm:TPMs}?
      +--ro compute-node* [node-id]
         +--ro node-id               string
         +--ro node-physical-index?  int32 {ietfhw:entity-mib}?
         +--ro node-name?            string
         +--ro node-location?        string
```

2.1.1.7. YANG Module

```
<CODE BEGINS> file ietf-tpm-remote-attestation@2020-12-17.yang
module ietf-tpm-remote-attestation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-tpm-remote-attestation";
  prefix "tpm";

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

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

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

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

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

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
```

```
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

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

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

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
     'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
     are to be interpreted as described in BCP 14 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

  revision "2020-12-15" {
    description
      "Initial version";
    reference
      "draft-ietf-rats-yang-tpm-charra";
  }

  /*****************/
  /*   Features    */
  /*****************/

  feature TPMs {
    description
      "The device supports the remote attestation of multiple
      TPM based cryptoprocessors.";
  }


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

  typedef pcr {
    type uint8 {
      range "0..31";
    }
    description
```

```
      "Valid index number for a PCR.  At this point 0-31 is viable.";
  }

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

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


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

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

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

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

  identity netequip_boot {
    base attested_event_log_type;
```

```
    description
      "An event type associated with Network Equipment Boot.";
  }

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

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

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

```
  }

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

  grouping TPM12-hash-algo {
    description
      "The cryptographic algorithm used to hash the TPM1.2 PCRs.";
    leaf TPM12-hash-algo {
      must "/tpm:rats-support-structures/tpm:attester-supported-algos"
         + "/tpm:tpm12-hash" {
         error-message "Not a platform supported TPM12-hash-algo";
      }
      type identityref {
        base taa:hash;
      }
      description
        "The hash scheme that is used to hash a TPM1.2 PCR. This
        must be one of those supported by a platform.  This assumes
        that an algorithm other than SHA1 can be supported on some
        TPM1.2 cryptoprocessor variant.";
      default taa:TPM_ALG_SHA1;
    }
  }

  grouping nonce {
    description
      "A nonce to show freshness and to allow the detection
      of replay attacks.";
    leaf nonce-value {
      type binary;
      mandatory true;
```

```
      description
        "This nonce SHOULD be generated via a registered
         cryptographic-strength algorithm. In consequence,
         the length of the nonce depends on the hash algorithm
         used. The algorithm used in this case is independent
         from the hash algorithm used to create the hash-value
         in the response of the attestor.";
    }
  }

  grouping tpm12-pcr-selection {
    description
      "A Verifier can request one or more PCR values using its
       individually created Attestation Key Certificate (AC).
       The corresponding selection filter is represented in this
       grouping.
       Requesting a PCR value that is not in scope of the AC used,
       detailed exposure via error msg should be avoided.";
    leaf-list pcr-index {
      /*  the following XPATH must be updated to ensure that only
            selectable PCRs are allowed in the RPC
      must "/tpm:rats-support-structures/tpm:tpms" +
           "/tpm:tpm[tpm-name = current()]" +
           "/tpm:tpm[TPM12-pcrs = current()]" {
        error-message "Acquiring this PCR index is not supported";
      }
      */
      type pcr;
      description
        "The numbers/indexes of the PCRs. At the moment this is limited
        to 32.";
    }
  }

  grouping tpm20-pcr-selection {
    description
      "A Verifier can acquire one or more PCR values, which are hashed
       together in a TPM2B_DIGEST coming from the TPM2.  The selection
       list of desired PCRs and the Hash Algorithm is represented in
       this grouping.";
    list tpm20-pcr-selection {
      unique "TPM20-hash-algo";
      description
        "Specifies the list of PCRs and Hash Algorithms that can be
        returned within a TPM2B_DIGEST.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
         TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.9.7";
```

```
      uses TPM20-hash-algo;
      leaf-list pcr-index {
        /*  the following XPATH must be updated to ensure that only
            selectable PCRs are allowed in the RPC
        must "/tpm:rats-support-structures/tpm:tpms" +
             "/tpm:tpm[tpm-name = current()]" +
             "/tpm:tpm20-pcr-bank[pcr-index = current()]" {
          error-message "Acquiring this PCR index is not supported";
        }
        */
        type tpm:pcr;
        description
          "The numbers of the PCRs that which are being tracked
          with a hash based on the TPM20-hash-algo.";
      }
    }
  }

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

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

  grouping tpm-name-selector {
    description
      "One or more TPM on a device.";
    leaf-list tpm-name {
      type string;
      config false;
      description
        "Name of one or more unique TPMs on a device.  If this object
        exists, a selection should pull only the objects related to
```

```
        these TPM(s).  If it does not exist, all qualifying TPMs that
        are 'hardware-based' equals true on the device are selected.";
    }
  }

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

  grouping tpm12-attestation {
    description
      "Contains an instance of TPM1.2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester
      information.";
    uses node-uptime;
    leaf TPM_QUOTE2 {
      type binary;
      description
        "Result of a TPM1.2 Quote2 operation. This includes PCRs,
        signatures, locality, the provided nonce and other data which
        can be further parsed to appraise the Attester.";
      reference
        "TPM1.2 commands rev116 July 2007, Section 16.5";
    }
  }

  grouping tpm20-attestation {
    description
      "Contains an instance of TPM2 style signed cryptoprocessor
      measurements.  It is supplemented by unsigned Attester
      information.";
    leaf TPMS_QUOTE_INFO {
      mandatory true;
      type binary;
      description
        "A hash of the latest PCR values (and the hash algorithm used)
        which have been returned from a Verifier for the selected PCRs
        and Hash Algorithms.";
      reference
        "https://www.trustedcomputinggroup.org/wp-content/uploads/
        TPM-Rev-2.0-Part-2-Structures-01.38.pdf  Section 10.12.1";
    }
```

```
    leaf quote-signature {
      type binary;
      description
        "Quote signature returned by TPM Quote.  The signature was
        generated using the key associated with the
        certificate-name.";
    }
    uses node-uptime;
    list unsigned-pcr-values {
      description
        "PCR values in each PCR bank. This might appear redundant with
        the TPM2B_DIGEST, but that digest is calculated across multiple
        PCRs.  Having to verify across multiple PCRs does not
        necessarily make it easy for a Verifier to appraise just the
        minimum set of PCR information which has changed since the last
        received TPM2B_DIGEST.  Put another way, why should a Verifier
        reconstruct the proper value of all PCR Quotes when only a
        single PCR has changed?

        To help this happen, if the Attester does know specific PCR
        values, the Attester can provide these individual values via
        'unsigned-pcr-values'.   By comparing this information to the
        what has previously been validated, it is possible for a
        Verifier to confirm the Attester's signature while eliminating
        significant processing.";
      uses TPM20-hash-algo;
      list pcr-values {
        key pcr-index;
        description
          "List of one PCR bank.";
        leaf pcr-index {
          type pcr;
          description
            "PCR index number.";
        }
        leaf pcr-value {
          type binary;
          description
            "PCR value.";
        }
      }
    }
  }


  grouping log-identifier {
    description
      "Identifier for type of log to be retrieved.";
```

```
    leaf log-type {
      type identityref {
        base attested_event_log_type;
      }
      mandatory true;
      description
        "The corresponding measurement log type identity.";
    }
  }

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

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

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

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

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

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

  grouping netequip-boot-event {
    description
      "Defines an hash log extend event for Network Equipment Boot.";
    leaf event-number {
      type uint64;
      description
        "Unique number for this event for sequencing";
    }
    leaf filename-hint {
      type string;
      description
```

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

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

  grouping event-logs {
    description
      "A selector for the log and its type.";
    choice attested_event_log_type {
      mandatory true;
      description
```

```
        "Event log type determines the event logs content.";
      case bios {
        description
          "BIOS/UEFI event logs";
        container bios-event-logs {
          description
            "BIOS/UEFI event logs";
          uses bios-event-log;
        }
      }
      case ima {
        description
          "IMA event logs.";
        container ima-event-logs {
          description
            "IMA event logs.";
          uses ima-event-log;
        }
      }
      case netequip_boot {
        description
          "Network Equipment Boot event logs";
        container boot-event-logs {
          description
            "Network equipment boot event logs.";
          uses network-equipment-boot-event-log;
        }
      }
    }
  }

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

  rpc tpm12-challenge-response-attestation {
    if-feature "taa:TPM12";
    description
      "This RPC accepts the input for TSS TPM 1.2 commands made to the
       attesting device.";
    input {
      container tpm12-attestation-challenge {
        description
          "This container includes every information element defined
           in the reference challenge-response interaction model for
           remote attestation. Corresponding values are based on
           TPM 1.2 structure definitions";
        uses tpm12-pcr-selection;
```

```
        uses nonce;
        leaf-list certificate-name {
          if-feature "tpm:TPMs";
          must "/tpm:rats-support-structures/tpm:tpms" +
               "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm12']" +
               "/tpm:certificates/" +
               "/tpm:certificate[certificate-name-ref=current()]" {
            error-message "Not an available TPM1.2 AIK certificate.";
          }
          type certificate-name-ref;
          description
            "When populated, the RPC will only get a Quote for the
            TPMs associated with these certificate(s).";
        }
      }
    }
    output {
      list tpm12-attestation-response {
        unique "certificate-name";
        description
          "The binary output of TPM 1.2 TPM_Quote/TPM_Quote2, including
           the PCR selection and other associated attestation evidence
           metadata";
        uses certificate-name-ref {
          description
            "Certificate associated with this tpm12-attestation.";
        }
        uses tpm12-attestation;
      }
    }
  }

  rpc tpm20-challenge-response-attestation {
    if-feature "taa:TPM20";
    description
      "This RPC accepts the input for TSS TPM 2.0 commands of the
       managed device. ComponentIndex from the hardware manager YANG
       module to refer to dedicated TPM in composite devices,
       e.g. smart NICs, is still a TODO.";
    input {
      container tpm20-attestation-challenge {
        description
          "This container includes every information element defined
           in the reference challenge-response interaction model for
           remote attestation. Corresponding values are based on
           TPM 2.0 structure definitions";
        uses nonce;
        uses tpm20-pcr-selection;
```

```
        leaf-list certificate-name {
          if-feature "tpm:TPMs";
          must "/tpm:rats-support-structures/tpm:tpms" +
               "/tpm:tpm[tpm:tpm-firmware-version='taa:tpm20']" +
               "/tpm:certificates/" +
               "/tpm:certificate[certificate-name-ref=current()]" {
            error-message "Not an available TPM2.0 AIK certificate.";
          }
          type certificate-name-ref;
            description
              "When populated, the RPC will only get a Quote for the
              TPMs associated with the certificates.";
        }
      }
    }
    output {
      list tpm20-attestation-response {
        unique "certificate-name";
        description
          "The binary output of TPM2b_Quote in one TPM chip of the
           node which identified by node-id. An TPMS_ATTEST structure
           including a length, encapsulated in a signature";
        uses certificate-name-ref {
          description
            "Certificate associated with this tpm20-attestation.";
        }
        uses tpm20-attestation;
      }
    }
  }

  rpc log-retrieval {
    description
      "Logs Entries are either identified via indices or via providing
       the last line received. The number of lines returned can be
       limited. The type of log is a choice that can be augmented.";
    input {
      list log-selector {
        description
          "Selection of log entries to be reported.";
        uses tpm-name-selector;
        choice index-type {
          description
            "Last log entry received, log index number, or timestamp.";
          case last-entry {
            description
              "The last entry of the log already retrieved.";
            leaf last-entry-value {
```

```
            type binary;
            description
              "Content of an log event which matches 1:1 with a
               unique event record contained within the log.  Log
               entries subsequent to this will be passed to the
               requester.  Note: if log entry values are not unique,
               this MUST return an error.";
          }
        }
        case index {
          description
            "Numeric index of the last log entry retrieved, or
             zero.";
          leaf last-index-number {
            type uint64;
            description
              "The last numeric index number of a log entry.
               Zero means to start at the beginning of the log.
               Entries subsequent to this will be passed to the
               requester.";
          }
        }
        case timestamp {
          leaf timestamp {
            type yang:date-and-time;
            description
              "Timestamp from which to start the extraction.  The
               next log entry subsequent to this timestamp is to
               be sent.";
          }
          description
            "Timestamp from which to start the extraction.";
        }
      }
      leaf log-entry-quantity {
        type uint16;
        description
          "The number of log entries to be returned. If omitted, it
           means all of them.";
      }
    }
    uses log-identifier;
  }

  output {
    container system-event-logs {
      description
        "The requested data of the measurement event logs";
```

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

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

  container rats-support-structures {
    description
      "The datastore definition enabling verifiers or relying
       parties to discover the information necessary to use the
       remote attestation RPCs appropriately.";
    container compute-nodes {
      if-feature "tpm:TPMs";
      description
        "Holds the set device subsystems/components in this composite
         device that support TPM operations.";
      list compute-node {
        key node-id;
        config false;
        min-elements 2;
        description
          "A component in this composite device that
          supports TPM operations.";
        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";
        }
        leaf node-name {
          type string;
          description
            "Name of the compute node.";
        }
        leaf node-location {
          type string;
          description
            "Location of the compute node, such as slot number.";
        }
      }
    }
    container tpms {
      description
        "Holds the set of TPMs within an Attester.";
      list tpm {
        key tpm-name;
        unique "tpm-path";
          description
           "A list of TPMs in this composite device that RATS
           can be conducted with.";
        uses tpm-name;
        leaf hardware-based {
          type boolean;
          config false;
          description
            "Answers the question: is this TPM is a hardware based
            TPM?";
        }
        leaf tpm-physical-index {
          if-feature ietfhw:entity-mib;
          type int32 {
            range "1..2147483647";
          }
          config false;
          description
            "The entPhysicalIndex for the TPM.";
          reference
            "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
        }
        leaf tpm-path {
          type string;
```

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

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

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

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

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

2.1.2. ietf-tcg-algs

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

2.1.2.1. Features

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

2.1.2.2. Identities

There are three types of identities in this model.

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

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

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

2.1.2.3. YANG Module

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

  organization
    "IETF RATS Working Group";

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

  description
    "This module defines a identities for asymmetric algorithms.

     Copyright (c) 2020 IETF Trust and the persons identified
     as authors of the code. All rights reserved.
     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
     itself for full legal notices.
     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
     'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
     are to be interpreted as described in BCP 14 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

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

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

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

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


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

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

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

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

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

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

```
      "TCG_Algorithm_Registry_r1p32_pub Table 2";
  }

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

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

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

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

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

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

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

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

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

  }

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

  }

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

```
  }

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

  }

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

  }

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

  }

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

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

    }

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

  }

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

  }

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

  }
```

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

  }

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

  }

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

  }

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

  }

  identity TPM_ALG_RSASSA {
```

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

  }

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

  }

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

  }

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

```
  }

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

  }

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

  }

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

  }

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

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

  }

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

  }

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

  }

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

```
      ALG_ID: 0x0020";

  }

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

  }

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

  }

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

  }

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

```
  }

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

  }

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

  }

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

  }

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

```
  }

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

  }

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

  }

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

  }

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

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

  }

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

  }

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

  }

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

  }

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

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

  }

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

  }

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

  }

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

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

  }

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

  }

}
<CODE ENDS>
```

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

3. IANA considerations

This document will include requests to IANA:

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

4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

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

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

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

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

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

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

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

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

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

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

5. Acknowledgements

Not yet.

6. Change Log

Changes from version 04 to version 05:

- YANG Dr comments covered

Changes from version 03 to version 04:

- TPM1.2 Quote1 eliminated

- YANG model simplifications so redundant info isn't exposed

Changes from version 02 to version 03:

- moved to tcg-algs
- cleaned up model to eliminate sources of errors
- removed key establishment RPC
- added lots of XPATH which must all be scrubbed still
- Descriptive text added on model contents.

Changes from version 01 to version 02:

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

Changes from version 00 to version 01:

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

7. References

7.1. Normative References

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

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-08 (work in progress), December 2020.

[I-D.ietf-rats-reference-interaction-models]
Birkholz, H., Eckel, M., Newton, C., and L. Chen, "Reference Interaction Models for Remote Attestation Procedures", draft-ietf-rats-reference-interaction-models-01 (work in progress), October 2020.

[I-D.ietf-rats-tpm-based-network-device-attest]
Fedorkow, G., Voit, E., and J. Fitzgerald-McKay, "TPM-based Network Device Remote Integrity Verification", draft-ietf-rats-tpm-based-network-device-attest-06 (work in progress), December 2020.

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

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

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

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

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

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

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

7.2. Informative References

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

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

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

[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>.

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
ThoughtSpot

Email: shwetha.bhandari@thoughtspot.com

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Bill Sulzen
Cisco Systems

Email: bsulzen@cisco.com

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

Email: Frank.Xialiang@huawei.com

Tom Laffey
Hewlett Packard Enterprise

Email: tom.laffey@hpe.com

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

Email: gfedorkow@juniper.net