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.

Table of Contents

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.
```

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).

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