Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

A. Fuchs
H. Birkholz
Fraunhofer SIT
I. McDonald
High North Inc
C. Bormann
Universitaet Bremen TZI

July 08, 2016

Time-Based Uni-Directional Attestation
draft-birkholz-tuda-02

Abstract

This memo documents the method and bindings used to conduct time-based uni-directional attestation between distinguishable endpoints over the network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Remote attestation describes the attempt to determine the integrity and trustworthiness of an endpoint -- the attestee -- over a network to another endpoint -- the verifier -- without direct access. One way to do so is based on measurements of software components running on the attestee, where the hash values of all started software components are stored (extended into) a Trust Anchor implemented as a Hardware Security Module (e.g. a Trusted Platform Module or similar) and reported via a signature over these measurements. Protocols that facilitate these Trust Anchor based signatures in order to provide remote attestations are usually bi-directional protocols [PTS], where one entity sends a challenge that is included inside the response to ensure the recentness -- the freshness -- of the attestation information.

In many contexts and scenarios it is not feasible to deploy bi-directional protocols, due to constraints in the underlying communication schemes. Furthermore, many communication schemes do not have a notion of connection, which disallows the usage of connection context related state information. These constraints may make it impossible to deploy challenge-response based schemes to achieve freshness of messages in security protocols. Examples of these constrained environments include broadcast and multicast schemes such as automotive IEEE802.11p as well as communication models that do not maintain connection state over time, such as REST [REST] and SNMP [RFC3411].

This document describes the time-based uni-directional attestation protocol -- TUDA -- that requires only uni-directional communication channels between verifier and attestee. whilst still providing up-to-date information about the integrity and thereby trustworthiness of the attested device. There are two important prerequisites next to the Hardware Security Module (HSM) itself:

- a source of (relative) time (i.e. a tick counter) integrated in the HSM, and

- network access to a trusted time stamp authority (TSA) [RFC3161].

Both prerequisites are mandatory to attest the appropriate freshness of the remotes attestation without bi-directional communication. The attestation scheme of TUDA is based on a set of TUDA information elements that are generated on the attestee and transported to the verifier. TUDA information elements are encoded in the Concise Binary Object Representation, CBOR [RFC7049]. In this document, the composition of the CBOR data items that represent the information

elements is described using the CBOR Data Definition Language, CDDL [I-D.greevenbosch-appsawg-cbor-cddl].

The binding of the attestation scheme used by TUDA to generate the TUDA information elements is specific to the methods provided by the HSM used. As a reference, this document includes pseudo-code that illustrates the production of TUDA information elements using a TPM 1.2 and the corresponding TPM commands specified in [TPM12] as an example. The references to TPM 1.2 commands and corresponding pseudo-code only serves as guidance to enable a better understanding of the attestation scheme and does not imply the use of a specific HSM (excluding, of course, the requirements highlighted above).

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 RFC 2119, BCP 14 [RFC2119].

1.2. Concept

There are significant differences between conventional bi-directional attestation and TUDA regarding both the information elements transmitted between attestee and verifier and the time-frame, in which an attestation can be considered to be fresh (and therefore trustworthy).

In general, remote attestation using a bi-directional communication scheme includes sending a nonce-challenge within a signed attestation token. Using the TPM 1.2 as an example, a corresponding nonce-challenge would be included within the signature created by the TPM_Quote command in order to prove the freshness of the attestation response, see e.g. [PTS].

In contrast, the TUDA protocol would use a combination output of TPM_CertifyInfo and TPM_TickStampBlob. The former provides a proof about the platform's state by attesting that a certain key is bound to said state. The latter provides proof that the platform was in the specified state by using the bound key in a time operation. This combination enables a time-based attestation scheme. This approach is based on the concepts introduced in [SCALE] and [SFKE2008].

The payload of information elements transmitted is based on different methods, because the time-frame, in which an attestation is considered to be fresh (and therefore trustworthy), is defined differently.

The freshness properties of a challenge-response based protocol define the point-of-time of attestation between:

- o the time of transmission of the nonce, and
- o the reception of the response

Given the time-based attestation scheme, the freshness property of TUDA is equivalent to that of bi-directional challenge response attestation, if the point-in-time of attestation lies between:

- o the transmission of a TUDA time-synchronization token, and
- o the typical round-trip time between the verifier and the attestee,

The accuracy of this time-frame is defined by two factors:

- o the time-synchronization between the attestee and the TSA. The time between the two TPM tickstamps give the maximum drift (left and right) to the TSA timestamp, and
- o the drift of local TPM clocks

Since TUDA attestations do not rely upon a verifier provided value (i.e. the nonce), the security guarantees of the protocol only incorporate the TSA and the TPM. As a consequence TUDA attestations can even serve as proof of integrity in audit logs with point in time guarantees, in contrast to classical attestations.

Appendix A contains a realization of TUDA using TPM 1.2 primitives. A realization of TUDA using TPM 2.0 primitives will be added with the next iteration of this document.

1.3. Terminology

This document introduces roles, information elements and types required to conduct TUDA and uses terminology (e.g. specific certificate names) typically seen in the context of attestation or hardware security modules.

1.3.1. Roles

Attestee: the endpoint that is the subject of the attestation to another endpoint.

Verifier: the endpoint that consumes the attestation of another endpoint.

TSA: a Time Stamp Authority [RFC3161]

1.3.2. General Types

Byte: the now customary synonym for octet

Cert: an X.509 certificate represented as a byte-string

PCR-Hash: a hash value of the security posture measurements stored in a TPM Platform Configuration Register (e.g. regarding running software instances) represented as a byte-string

1.3.3. TPM-Specific Terms

AIK: an Attestation Identity Key, a special key type used within a TPM for identity-related operations (such as TPM_Certify or TPM_Quote)

PCR: a Platform Configuration Register that is part of a TPM and is used to securely store and report measurements about security posture

1.3.4. Certificates

TSA-CA: the Certificate Authority that provides the certificate for the TSA represented as a Cert

AIK-CA: the Certificate Authority that provides the certificate for the attestation identity key of the TPM. This is the client platform credential for this protocol. It is a placeholder for a specific CA and AIK-Cert is a placeholder for the corresponding certificate, depending on what protocol was used. The specific protocols are out of scope for this document, see also [AIK-Enrollment] and [IEEE802.1AR].

2. Time-Based Uni-Directional Attestation

A Time-Based Uni-Directional Attestation (TUDA) consists of the following seven information elements. They are used to gain assurance of the Attestee's platform configuration at a certain point in time:

TSA Certificate: The certificate of the Time Stamp Authority that is used in a subsequent synchronization protocol token. This certificate is signed by the TSA-CA.

AIK Certificate (<xref target="AIK-Credential"/>, <xref target="AIK-Enrollment"/>; see <xref target="aik"/>):

A certificate about the Attestation Identity Key (AIK) used. This may or may not also be an [IEEE802.1AR] IDevID or LDevID, depending on their setting of the corresponding identity property.

Synchronization Token: The reference for Attestations are the Tick-Sessions of the TPM. In order to put Attestations into relation with a Real Time Clock (RTC), it is necessary to provide a cryptographic synchronization between the tick session and the RTC. To do so, a synchronization protocol is run with a Time Stamp Authority (TSA).

Restriction Info: The attestation relies on the capability of the TPM to operate on restricted keys. Whenever the PCR values for the machine to be attested change, a new restricted key is created that can only be operated as long as the PCRs remain in their current state.

In order to prove to the Verifier that this restricted temporary key actually has these properties and also to provide the PCR value that it is restricted, the TPM command TPM_CertifyInfo is used. It creates a signed certificate using the AIK about the newly created restricted key.

Measurement Log: Similarly to regular attestations, the Verifier needs a way to reconstruct the PCRs' values in order to estimate the trustworthiness of the device. As such, a list of those elements that were extended into the PCRs is reported. Note though that for certain environments, this step may be optional if a list of valid PCR configurations exists and no measurement log is required.

Implicit Attestation: The actual attestation is then based upon a TPM_TickStampBlob operation using the restricted temporary key that was certified in the steps above. The TPM_TickStampBlob is executed and thereby provides evidence that at this point in time (with respect to the TPM internal tick-session) a certain configuration existed (namely the PCR values associated with the restricted key). Together with the synchronization token this tick-related timing can then be related to the real-time clock.

Concise SWID tags: As an option to better assess the trustworthiness of an Attestee, a Verifier can request the reference hashes (often referred to as golden measurements) of all started software components to compare them with the entries in the measurement log. References hashes regarding installed (and therefore running) software can be provided by the manufacturer via SWID tags. SWID tags are provided by the Attestee using the Concise SWID representation [I-D-birkholz-sacm-coswid] and bundled into a

CBOR array. Ideally, the reference hashes include a signature created by the manufacturer of the software.

These information elements could be sent en bloc, but it is recommended to retrieve them separately to save bandwidth, since these elements have different update cycles. In most cases, retransmitting all seven information elements would result in unnecessary redundancy.

Furthermore, in some scenarios it might be feasible not to store all elements on the Attestee endpoint, but instead they could be retrieved from another location or pre-deployed to the Verifier. It is also feasible to only store public keys at the Verifier and skip the whole certificate provisioning completely in order to save bandwidth and computation time for certificate verification.

2.1. TUDA Information Elements Update Cycles

An endpoint can be in various states and have various information associated with it during its life cycle. For TUDA, a subset of the states (which can include associated information) that an endpoint and its TPM can be in, is important to the attestation process.

- o Some states are persistent, even after reboot. This includes certificates that are associated with the endpoint itself or with services it relies on.

- o Some states are more volatile and change at the beginning of each boot cycle. This includes the TPM-internal Tick-Session which provides the basis for the synchronization token and implicit attestation.

- o Some states are even more volatile and change during an uptime cycle (the period of time an endpoint is powered on, starting with its boot). This includes the content of PCRs of a TPM and thereby also the PCR-restricted keys used during attestation.

Depending on this "lifetime of state", data has to be transported over the wire, or not. E.g. information that does not change due to a reboot typically has to be transported only once between the Attestee and the Verifier.

There are three kinds of events that require a renewed attestation:

- o The Attestee completes a boot-cycle

- o A relevant PCR changes

o Too much time has passed since the last attestation statement

The third event listed above is variable per application use case and can therefore be set appropriately. For usage scenarios, in which the device would periodically push information to be used in an audit-log, a time-frame of approximately one update per minute should be sufficient in most cases. For those usage scenarios, where verifiers request (pull) a fresh attestation statement, an implementation could use the TPM continuously to always present the most freshly created results. To save some utilization of the TPM for other purposes, however, a time-frame of once per ten seconds is recommended, which would leave 80% of utilization for applications.

```
Attestee                                              Verifier
   |                                                     |
 Boot                                                    |
   |                                                     |
 Create Sync-Token                                       |
   |                                                     |
 Create Restricted Key                                   |
 Certify Restricted Key                                  |
   |                                                     |
   | AIK-Cert ------------------------------------------> |
   | Sync-Token ----------------------------------------> |
   | Certify-Info --------------------------------------> |
   | Measurement Log -----------------------------------> |
   | Attestation ---------------------------------------> |
   |                                    Verify Attestation
   |                                                     |
   |       <Time Passed>                                 |
   |                                                     |
   | Attestation ---------------------------------------> |
   |                                    Verify Attestation
   |                                                     |
   |       <Time Passed>                                 |
   |                                                     |
 PCR-Change                                              |
   |                                                     |
 Create Restricted Key                                   |
 Certify Restricted Key                                  |
   |                                                     |
   | Certify-Info --------------------------------------> |
   | Measurement Log -----------------------------------> |
   | Attestation ---------------------------------------> |
   |                                    Verify Attestation
   |                                                     |
 Boot                                                    |
   |                                                     |
```

```
    Create Sync-Token                                           |
      |                                                         |
    Create Restricted Key                                       |
    Certify Restricted Key                                      |
      |                                                         |
      | Sync-Token -------------------------------------------> |
      | Certify-Info -----------------------------------------> |
      | Measurement Log --------------------------------------> |
      | Attestation ------------------------------------------> |
      |                                           Verify Attestation
      |                                                         |
      |       <Time Passed>                                     |
      |                                                         |
      | Attestation ------------------------------------------> |
      |                                           Verify Attestation
      |                                                         |
```

Figure 1: Example sequence of events

3. REST Realization

Each of the seven data items is defined as a media type (Section 5). Representations of resources for each of these media types can be retrieved from URIs that are defined by the respective servers [RFC7320]. As can be derived from the URI, the actual retrieval is via one of the HTTPs ([RFC7230], [RFC7540]) or CoAP [RFC7252]. How a client obtains these URIs is dependent on the application; e.g., CoRE Web links [RFC6690] can be used to obtain the relevant URIs from the self-description of a server, or they could be prescribed by a RESTCONF data model [I-D.ietf-netconf-restconf].

4. SNMP Realization

SNMPv3 [STD62] [RFC3411] is widely available on computers and also constrained devices. To transport the TUDA information elements, an SNMP MIB is defined below which encodes each of the seven TUDA information elements into a table. Each row in a table contains a single read-only columnar SNMP object of datatype OCTET-STRING. The values of a set of rows in each table can be concatenated to reconstitute a CBOR-encoded TUDA information element. The Verifier can retrieve the values for each CBOR fragment by using SNMP GetNext requests to "walk" each table and can decode each of the CBOR-encoded data items based on the corresponding CDDL [I-D.greevenbosch-appsawg-cbor-cddl] definition.

Design Principles:

1. Over time, TUDA attestation values age and should no longer be used. Every table in the TUDA MIB has a primary index with the value of a separate scalar cycle counter object that disambiguates the transition from one attestation cycle to the next.

2. Over time, the measurement log information (for example) may grow large. Therefore, read-only cycle counter scalar objects in all TUDA MIB object groups facilitate more efficient access with SNMP GetNext requests.

3. Notifications are supported by an SNMP trap definition with all of the cycle counters as bindings, to alert a Verifier that a new attestation cycle has occurred (e.g., synchronization data, measurement log, etc. have been updated by adding new rows and possibly deleting old rows).

4.1. Structure of TUDA MIB

The following table summarizes the object groups, tables and their indexes, and conformance requirements for the TUDA MIB:

Group/Table	Cycle	Instance	Fragment	Required
General				x
AIKCert	x	x	x	
TSACert	x	x	x	
SyncToken	x		x	x
Restrict	x			x
Measure	x	x		
VerifyToken	x			x
SWIDTag	x	x	x	

4.1.1. Cycle Index

A tudaV1<Group>CycleIndex is the:

1. first index of a row (element instance or element fragment) in the tudaV1<Group>Table;

2. identifier of an update cycle on the table, when rows were added and/or deleted from the table (bounded by tudaV1<Group>Cycles); and

3. binding in the tudaV1TrapV2Cycles notification for directed polling.

4.1.2. Instance Index

A tudaV1<Group>InstanceIndex is the:

1. second index of a row (element instance or element fragment) in the tudaV1<Group>Table; except for

2. a row in the tudaV1SyncTokenTable (that has only one instance per cycle).

4.1.3. Fragment Index

A tudaV1<Group>FragmentIndex is the:

1. last index of a row (always an element fragment) in the tudaV1<Group>Table; and

2. accomodation for SNMP transport mapping restrictions for large string elements that require fragmentation.

4.2. Relationship to Host Resources MIB

The General group in the TUDA MIB is analogous to the System group in the Host Resources MIB [RFC2790] and provides context information for the TUDA attestation process.

The Verify Token group in the TUDA MIB is analogous to the Device group in the Host MIB and represents the verifiable state of a TPM device and its associated system.

The SWID Tag group (containing a Concise SWID reference hash profile [I-D-birkholz-sacm-coswid]) in the TUDA MIB is analogous to the Software Installed and Software Running groups in the Host Resources MIB [RFC2790].

4.3. Relationship to Entity MIB

The General group in the TUDA MIB is analogous to the Entity General group in the Entity MIB v4 [RFC6933] and provides context information for the TUDA attestation process.

The SWID Tag group in the TUDA MIB is analogous to the Entity Logical group in the Entity MIB v4 [RFC6933].

4.4. Relationship to Other MIBs

The General group in the TUDA MIB is analogous to the System group in MIB-II [RFC1213] and the System group in the SNMPv2 MIB [RFC3418] and provides context information for the TUDA attestation process.

4.5. Definition of TUDA MIB

```
<CODE BEGINS>
TUDA-V1-ATTESTATION-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, Counter32,
    enterprises, NOTIFICATION-TYPE
        FROM SNMPv2-SMI                 -- RFC 2578
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF                -- RFC 2580
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB;        -- RFC 3411

tudaV1MIB MODULE-IDENTITY
    LAST-UPDATED    "201607080000Z"  -- 08 July 2016
    ORGANIZATION
        "Fraunhofer SIT"
    CONTACT-INFO
        "Andreas Fuchs
        Fraunhofer Institute for Secure Information Technology
        Email: andreas.fuchs@sit.fraunhofer.de

        Henk Birkholz
        Fraunhofer Institute for Secure Information Technology
        Email: henk.birkholz@sit.fraunhofer.de

        Ira E McDonald
        High North Inc
        Email: blueroofmusic@gmail.com

        Carsten Bormann
        Universitaet Bremen TZI
        Email: cabo@tzi.org"

    DESCRIPTION
        "The MIB module for monitoring of time-based unidirectional
        attestation information from a network endpoint system,
        based on the Trusted Computing Group TPM 1.2 definition.

        Copyright (C) High North Inc (2016)."
```

```
       REVISION "201607080000Z" -- 08 July 2016
       DESCRIPTION
           "Third version, published as draft-birkholz-tuda-02."

       REVISION "201603210000Z" -- 21 March 2016
       DESCRIPTION
           "Second version, published as draft-birkholz-tuda-01."

       REVISION "201510180000Z" -- 18 October 2015
       DESCRIPTION
           "Initial version, published as draft-birkholz-tuda-00."

           ::= { enterprises fraunhofersit(21616) mibs(1) tudaV1MIB(1) }

   tudaV1MIBNotifications      OBJECT IDENTIFIER ::= { tudaV1MIB 0 }
   tudaV1MIBObjects            OBJECT IDENTIFIER ::= { tudaV1MIB 1 }
   tudaV1MIBConformance        OBJECT IDENTIFIER ::= { tudaV1MIB 2 }

   --
   --  General
   --
   tudaV1General           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 1 }

   tudaV1GeneralCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of TUDA update cycles that have occurred, i.e.,
           sum of all the individual group cycle counters.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1General 1 }

   tudaV1GeneralVersionInfo OBJECT-TYPE
       SYNTAX      SnmpAdminString (SIZE(0..255))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Version information for TUDA MIB, e.g., specific release
           version of TPM 1.2 base specification and release version
           of TPM 1.2 errata specification and manufacturer and model
           TPM module itself."
       DEFVAL      { "" }
       ::= { tudaV1General 2 }

   --
   --  AIK Cert
```

```
   --
   tudaV1AIKCert           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 2 }

   tudaV1AIKCertCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of AIK Certificate chain update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1AIKCert 1 }

   tudaV1AIKCertTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1AIKCertEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "A table of fragments of AIK Certificate data."
       ::= { tudaV1AIKCert 2 }

   tudaV1AIKCertEntry OBJECT-TYPE
       SYNTAX      TudaV1AIKCertEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry for one fragment of AIK Certificate data."
       INDEX       { tudaV1AIKCertCycleIndex,
                     tudaV1AIKCertInstanceIndex,
                     tudaV1AIKCertFragmentIndex }
       ::= { tudaV1AIKCertTable 1 }

   TudaV1AIKCertEntry ::=
       SEQUENCE {
           tudaV1AIKCertCycleIndex         Integer32,
           tudaV1AIKCertInstanceIndex      Integer32,
           tudaV1AIKCertFragmentIndex      Integer32,
           tudaV1AIKCertData               OCTET STRING
       }

   tudaV1AIKCertCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "High-order index of this AIK Certificate fragment.
           Index of an AIK Certificate chain update cycle that has
```

```
           occurred (bounded by the value of tudaV1AIKCertCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1AIKCertEntry 1 }

   tudaV1AIKCertInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Middle index of this AIK Certificate fragment.
           Ordinal of this AIK Certificate in this chain, where the AIK
           Certificate itself has an ordinal of '1' and higher ordinals
           go *up* the certificate chain to the Root CA.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1AIKCertEntry 2 }

   tudaV1AIKCertFragmentIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this AIK Certificate fragment.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1AIKCertEntry 3 }

   tudaV1AIKCertData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "A fragment of CBOR encoded AIK Certificate data."
       DEFVAL      { "" }
       ::= { tudaV1AIKCertEntry 4 }

   --
   --  TSA Cert
   --
   tudaV1TSACert           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 3 }

   tudaV1TSACertCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of TSA Certificate chain update cycles that have
```

```
            occurred.

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1TSACert 1 }

    tudaV1TSACertTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF TudaV1TSACertEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of fragments of TSA Certificate data."
        ::= { tudaV1TSACert 2 }

    tudaV1TSACertEntry OBJECT-TYPE
        SYNTAX      TudaV1TSACertEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one fragment of TSA Certificate data."
        INDEX       { tudaV1TSACertCycleIndex,
                      tudaV1TSACertInstanceIndex,
                      tudaV1TSACertFragmentIndex }
        ::= { tudaV1TSACertTable 1 }

    TudaV1TSACertEntry ::=
        SEQUENCE {
            tudaV1TSACertCycleIndex         Integer32,
            tudaV1TSACertInstanceIndex      Integer32,
            tudaV1TSACertFragmentIndex      Integer32,
            tudaV1TSACertData               OCTET STRING
        }

    tudaV1TSACertCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "High-order index of this TSA Certificate fragment.
            Index of a TSA Certificate chain update cycle that has
            occurred (bounded by the value of tudaV1TSACertCycles).

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1TSACertEntry 1 }

    tudaV1TSACertInstanceIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
```

```
       DESCRIPTION
           "Middle index of this TSA Certificate fragment.
           Ordinal of this TSA Certificate in this chain, where the TSA
           Certificate itself has an ordinal of '1' and higher ordinals
           go *up* the certificate chain to the Root CA.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1TSACertEntry 2 }

   tudaV1TSACertFragmentIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this TSA Certificate fragment.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1TSACertEntry 3 }

   tudaV1TSACertData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "A fragment of CBOR encoded TSA Certificate data."
       DEFVAL      { "" }
       ::= { tudaV1TSACertEntry 4 }

   --
   --  Sync Token
   --
   tudaV1SyncToken         OBJECT IDENTIFIER ::= { tudaV1MIBObjects 4 }

   tudaV1SyncTokenCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of Sync Token update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1SyncToken 1 }

   tudaV1SyncTokenTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1SyncTokenEntry
       MAX-ACCESS  not-accessible
       STATUS      current
```

```
        DESCRIPTION
            "A table of fragments of Sync Token data."
        ::= { tudaV1SyncToken 2 }

    tudaV1SyncTokenEntry OBJECT-TYPE
        SYNTAX      TudaV1SyncTokenEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one fragment of Sync Token data."
        INDEX       { tudaV1SyncTokenCycleIndex,
                      tudaV1SyncTokenFragmentIndex }
        ::= { tudaV1SyncTokenTable 1 }

    TudaV1SyncTokenEntry ::=
        SEQUENCE {
            tudaV1SyncTokenCycleIndex       Integer32,
            tudaV1SyncTokenFragmentIndex    Integer32,
            tudaV1SyncTokenData             OCTET STRING
        }

    tudaV1SyncTokenCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "High-order index of this Sync Token fragment.
            Index of a Sync Token update cycle that has
            occurred (bounded by the value of tudaV1SyncTokenCycles).

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1SyncTokenEntry 1 }

    tudaV1SyncTokenFragmentIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Low-order index of this Sync Token fragment.

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1SyncTokenEntry 2 }

    tudaV1SyncTokenData OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE(0..1024))
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
           "A fragment of CBOR encoded Sync Token data."
       DEFVAL      { "" }
       ::= { tudaV1SyncTokenEntry 3 }

   --
   --  Restriction Info
   --
   tudaV1Restrict          OBJECT IDENTIFIER ::= { tudaV1MIBObjects 5 }

   tudaV1RestrictCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of Restriction Info update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1Restrict 1 }

   tudaV1RestrictTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1RestrictEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "A table of instances of Restriction Info data."
       ::= { tudaV1Restrict 2 }

   tudaV1RestrictEntry OBJECT-TYPE
       SYNTAX      TudaV1RestrictEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry for one instance of Restriction Info data."
       INDEX       { tudaV1RestrictCycleIndex }
       ::= { tudaV1RestrictTable 1 }

   TudaV1RestrictEntry ::=
       SEQUENCE {
           tudaV1RestrictCycleIndex        Integer32,
           tudaV1RestrictData              OCTET STRING
       }

   tudaV1RestrictCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
```

```
           "Index of this Restriction Info entry.
           Index of a Restriction Info update cycle that has
           occurred (bounded by the value of tudaV1RestrictCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1RestrictEntry 1 }


   tudaV1RestrictData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "An instance of CBOR encoded Restriction Info data."
       DEFVAL      { "" }
       ::= { tudaV1RestrictEntry 2 }

   --
   --  Measurement Log
   --
   tudaV1Measure           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 6 }

   tudaV1MeasureCycles OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of Measurement Log update cycles that have
           occurred.

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1Measure 1 }

   tudaV1MeasureInstances OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "Count of Measurement Log instance entries that have
           been recorded (some entries MAY have been pruned).

           DEFVAL intentionally omitted - counter object."
       ::= { tudaV1Measure 2 }

   tudaV1MeasureTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF TudaV1MeasureEntry
       MAX-ACCESS  not-accessible
       STATUS      current
```

```
       DESCRIPTION
           "A table of instances of Measurement Log data."
       ::= { tudaV1Measure 3 }

   tudaV1MeasureEntry OBJECT-TYPE
       SYNTAX      TudaV1MeasureEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry for one instance of Measurement Log data."
       INDEX       { tudaV1MeasureCycleIndex,
                     tudaV1MeasureInstanceIndex }
       ::= { tudaV1MeasureTable 1 }

   TudaV1MeasureEntry ::=
       SEQUENCE {
           tudaV1MeasureCycleIndex         Integer32,
           tudaV1MeasureInstanceIndex      Integer32,
           tudaV1MeasureData               OCTET STRING
       }

   tudaV1MeasureCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "High-order index of this Measurement Log entry.
           Index of a Measurement Log update cycle that has
           occurred (bounded by the value of tudaV1MeasureCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1MeasureEntry 1 }

   tudaV1MeasureInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this Measurement Log entry.
           Ordinal of this instance of Measurement Log data
           (NOT bounded by the value of tudaV1MeasureInstances).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1MeasureEntry 2 }

   tudaV1MeasureData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
```

```
        STATUS      current
        DESCRIPTION
            "A instance of CBOR encoded Measurement Log data."
        DEFVAL      { "" }
        ::= { tudaV1MeasureEntry 3 }

    --
    --  Verify Token
    --
    tudaV1VerifyToken       OBJECT IDENTIFIER ::= { tudaV1MIBObjects 7 }

    tudaV1VerifyTokenCycles OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of Verify Token update cycles that have
            occurred.

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1VerifyToken 1 }

    tudaV1VerifyTokenTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF TudaV1VerifyTokenEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of instances of Verify Token data."
        ::= { tudaV1VerifyToken 2 }

    tudaV1VerifyTokenEntry OBJECT-TYPE
        SYNTAX      TudaV1VerifyTokenEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one instance of Verify Token data."
        INDEX       { tudaV1VerifyTokenCycleIndex }
        ::= { tudaV1VerifyTokenTable 1 }

    TudaV1VerifyTokenEntry ::=
        SEQUENCE {
            tudaV1VerifyTokenCycleIndex     Integer32,
            tudaV1VerifyTokenData           OCTET STRING
        }

    tudaV1VerifyTokenCycleIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
```

```
        STATUS      current
        DESCRIPTION
            "Index of this instance of Verify Token data.
            Index of a Verify Token update cycle that has
            occurred (bounded by the value of tudaV1VerifyTokenCycles).

            DEFVAL intentionally omitted - index object."
        ::= { tudaV1VerifyTokenEntry 1 }

    tudaV1VerifyTokenData OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE(0..1024))
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "A instance of CBOR encoded Verify Token data."
        DEFVAL      { "" }
        ::= { tudaV1VerifyTokenEntry 2 }

    --
    --  SWID Tag
    --
    tudaV1SWIDTag           OBJECT IDENTIFIER ::= { tudaV1MIBObjects 8 }

    tudaV1SWIDTagCycles OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Count of SWID Tag update cycles that have occurred.

            DEFVAL intentionally omitted - counter object."
        ::= { tudaV1SWIDTag 1 }

    tudaV1SWIDTagTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF TudaV1SWIDTagEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of fragments of SWID Tag data."
        ::= { tudaV1SWIDTag 2 }

    tudaV1SWIDTagEntry OBJECT-TYPE
        SYNTAX      TudaV1SWIDTagEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry for one fragment of SWID Tag data."
        INDEX       { tudaV1SWIDTagCycleIndex,
```

```
                     tudaV1SWIDTagInstanceIndex,
                     tudaV1SWIDTagFragmentIndex }
       ::= { tudaV1SWIDTagTable 1 }

   TudaV1SWIDTagEntry ::=
       SEQUENCE {
           tudaV1SWIDTagCycleIndex         Integer32,
           tudaV1SWIDTagInstanceIndex      Integer32,
           tudaV1SWIDTagFragmentIndex      Integer32,
           tudaV1SWIDTagData               OCTET STRING
       }

   tudaV1SWIDTagCycleIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "High-order index of this SWID Tag fragment.
           Index of an SWID Tag update cycle that has
           occurred (bounded by the value of tudaV1SWIDTagCycles).

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1SWIDTagEntry 1 }

   tudaV1SWIDTagInstanceIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Middle index of this SWID Tag fragment.
           Ordinal of this SWID Tag instance in this update cycle.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1SWIDTagEntry 2 }

   tudaV1SWIDTagFragmentIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Low-order index of this SWID Tag fragment.

           DEFVAL intentionally omitted - index object."
       ::= { tudaV1SWIDTagEntry 3 }

   tudaV1SWIDTagData OBJECT-TYPE
       SYNTAX      OCTET STRING (SIZE(0..1024))
       MAX-ACCESS  read-only
```

```
       STATUS      current
       DESCRIPTION
           "A fragment of CBOR encoded SWID Tag data."
       DEFVAL      { "" }
       ::= { tudaV1SWIDTagEntry 4 }

   --
   --  Trap Cycles
   --
   tudaV1TrapV2Cycles NOTIFICATION-TYPE
       OBJECTS {
           tudaV1GeneralCycles,
           tudaV1AIKCertCycles,
           tudaV1TSACertCycles,
           tudaV1SyncTokenCycles,
           tudaV1RestrictCycles,
           tudaV1MeasureCycles,
           tudaV1MeasureInstances,
           tudaV1VerifyTokenCycles,
           tudaV1SWIDTagCycles
       }
       STATUS  current
       DESCRIPTION
           "This trap is sent when the value of any cycle or instance
           counter changes (i.e., one or more tables are updated).

           Note:  The value of sysUpTime in IETF MIB-II (RFC 1213) is
           always included in SNMPv2 traps, per RFC 3416."
       ::= { tudaV1MIBNotifications 1 }

   --
   --  Conformance Information
   --
   tudaV1Compliances           OBJECT IDENTIFIER
       ::= { tudaV1MIBConformance 1 }

   tudaV1ObjectGroups          OBJECT IDENTIFIER
       ::= { tudaV1MIBConformance 2 }

   tudaV1NotificationGroups    OBJECT IDENTIFIER
       ::= { tudaV1MIBConformance 3 }

   --
   --  Compliance Statements
   --
   tudaV1BasicCompliance MODULE-COMPLIANCE
       STATUS  current
       DESCRIPTION
```

```
            "An implementation that complies with this module MUST
            implement all of the objects defined in the mandatory
            group tudaV1BasicGroup."
        MODULE  -- this module
        MANDATORY-GROUPS { tudaV1BasicGroup }

        GROUP   tudaV1OptionalGroup
        DESCRIPTION
            "The optional TUDA MIB objects.
            An implementation MAY implement this group."

        GROUP   tudaV1TrapGroup
        DESCRIPTION
            "The TUDA MIB traps.
            An implementation SHOULD implement this group."
        ::= { tudaV1Compliances 1 }

    --
    --  Compliance Groups
    --
    tudaV1BasicGroup OBJECT-GROUP
        OBJECTS {
            tudaV1GeneralCycles,
            tudaV1GeneralVersionInfo,
            tudaV1SyncTokenCycles,
            tudaV1SyncTokenData,
            tudaV1RestrictCycles,
            tudaV1RestrictData,
            tudaV1VerifyTokenCycles,
            tudaV1VerifyTokenData
        }
        STATUS  current
        DESCRIPTION
            "The basic mandatory TUDA MIB objects."
        ::= { tudaV1ObjectGroups 1 }

    tudaV1OptionalGroup OBJECT-GROUP
        OBJECTS {
            tudaV1AIKCertCycles,
            tudaV1AIKCertData,
            tudaV1TSACertCycles,
            tudaV1TSACertData,
            tudaV1MeasureCycles,
            tudaV1MeasureInstances,
            tudaV1MeasureData,
            tudaV1SWIDTagCycles,
            tudaV1SWIDTagData
        }
```

```
       STATUS  current
       DESCRIPTION
           "The optional TUDA MIB objects."
       ::= { tudaV1ObjectGroups 2 }

   tudaV1TrapGroup NOTIFICATION-GROUP
       NOTIFICATIONS { tudaV1TrapV2Cycles }
       STATUS      current
       DESCRIPTION
           "The recommended TUDA MIB traps - notifications."
       ::= { tudaV1NotificationGroups 1 }

   END
   <CODE ENDS>
```

5. IANA Considerations

This memo includes requests to IANA, including registrations for media type definitions.

TBD

6. Security Considerations

There are Security Considerations. TBD

7. Change Log

Changes from version 01 to version 02:

- o Restructuring of Introduction, highlighting conceptual prerequisites

- o Restructuring of Concept to better illustrate differences to hand-shake based attestation and deciding factors regarding freshness properties

- o Subsection structure added to Terminology

- o Clarification of descriptions of approach (these were the FIXMEs)

- o Correction of RestrictionInfo structure: Added missing signature member

Changes from version 00 to version 01:

Major update to the SNMP MIB and added a table for the Concise SWID profile Reference Hashes that provides additional information to be compared with the measurement logs.

8. Contributors

TBD

9. References

9.1. Normative References

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

9.2. Informative References

[AIK-Credential]
"TCG Credential Profile", 2007, <http://www.trustedcomputinggroup.org/files/temp/642686EC-1D09-3519-AD58BB4C50BD5028/IWG%20Credential_Profiles_V1_R1_14.pdf>.

[AIK-Enrollment]
TCG Infrastructure Working Group, "A CMC Profile for AIK Certificate Enrollment", 2011, <https://www.trustedcomputinggroup.org/files/resource_files/738DF0BB-1A4B-B294-D0AF6AF9CC023163/IWG_CMC_Profile_Cert_Enrollment_v1_r7.pdf>.

[I-D-birkholz-sacm-coswid]
Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identifiers", draft-birkholz-sacm-coswid-00 (work in progress), March 2016.

[I-D.greevenbosch-appsawg-cbor-cddl]
Vigano, C. and H. Birkholz, "CBOR data definition language (CDDL): a notational convention to express CBOR data structures", draft-greevenbosch-appsawg-cbor-cddl-08 (work in progress), March 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-15 (work in progress), July 2016.

[IEEE802.1AR]
IEEE Computer Society, "IEEE Standard for Local and metropolitan area networks -- Secure Device Identity", IEEE Std 802.1AR, 2009.

[PTS] "TCG Attestation PTS Protocol Binding to TNC IF-M", 2011, <http://www.trustedcomputinggroup.org/files/resource_files/508E7E89-1A4B-B294-D06395D5FD7EC4E7/IFM_PTS_v1_0_r28.pdf>.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation, University of California, Irvine, 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

[RFC1213] McCloghrie, K. and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991, <http://www.rfc-editor.org/info/rfc1213>.

[RFC2790] Waldbusser, S. and P. Grillo, "Host Resources MIB", RFC 2790, DOI 10.17487/RFC2790, March 2000, <http://www.rfc-editor.org/info/rfc2790>.

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

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <http://www.rfc-editor.org/info/rfc3411>.

[RFC3418] Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, DOI 10.17487/RFC3418, December 2002, <http://www.rfc-editor.org/info/rfc3418>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <http://www.rfc-editor.org/info/rfc6690>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <http://www.rfc-editor.org/info/rfc6933>.

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

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

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

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <http://www.rfc-editor.org/info/rfc7320>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[SCALE] Fuchs, A., "Improving Scalability for Remote Attestation", Master Thesis (Diplomarbeit), Technische Universitaet Darmstadt, Germany, 2008.

[SFKE2008]
Stumpf, F., Fuchs, A., Katzenbeisser, S., and C. Eckert, "Improving the scalability of platform attestation", ACM Proceedings of the 3rd ACM workshop on Scalable trusted computing, page 1-10, 2008.

[STD62] "Internet Standard 62", STD 62, RFCs 3411 to 3418, December 2002.

[TPM12] "Information technology -- Trusted Platform Module -- Part 1: Overview", ISO/IEC 11889-1, 2009.

Appendix A. Realization with TPM 1.2 functions

A.1. TPM Functions

The following TPM structures, resources and functions are used within this approach. They are based upon the TPM 1.2 specification [TPM12].

A.1.1. Tick-Session and Tick-Stamp

On every boot, the TPM initializes a new Tick-Session. Such a tick-session consists of a nonce that is randomly created upon each boot to identify the current boot-cycle - the phase between boot-time of the device and shutdown or power-off - and prevent replaying of old tick-session values. The TPM uses its internal entropy source that guarantees virtually no collisions of the nonce values between two of such boot cycles.

It further includes an internal timer that is being initialize to Zero on each reboot. From this point on, the TPM increments this timer continuously based upon its internal secure clocking information until the device is powered down or set to sleep. By its hardware design, the TPM will detect attacks on any of those properties.

The TPM offers the function TPM_TickStampBlob, which allows the TPM to create a signature over the current tick-session and two externally provided input values. These input values are designed to serve as a nonce and as payload data to be included in a TickStampBlob: TickstampBlob := sig(TPM-key, currentTicks || nonce || externalData).

As a result, one is able to proof that at a certain point in time (relative to the tick-session) after the provisioning of a certain nonce, some certain externalData was known and provided to the TPM. If an approach however requires no input values or only one input value (such as the use in this document) the input values can be set to well-known value. The convention used within TCG specifications and within this document is to use twenty bytes of zero h'0000000000000000000000000000000000000000' as well-known value.

A.1.2. Platform Configuration Registers (PCRs)

The TPM is a secure cryptoprocessor that provides the ability to store measurements and metrics about an endpoint's configuration and state in a secure, tamper-proof environment. Each of these security relevant metrics can be stored in a volatile Platform Configuration Register (PCR) inside the TPM. These measurements can be conducted

at any point in time, ranging from an initial BIOS boot-up sequence to measurements taken after hundreds of hours of uptime.

The initial measurement is triggered by the Platforms so-called pre-BIOS or ROM-code. It will conduct a measurement of the first loadable pieces of code; i.e.\ the BIOS. The BIOS will in turn measure its Option ROMs and the BootLoader, which measures the OS-Kernel, which in turn measures its applications. This describes a so-called measurement chain. This typically gets recorded in a so-called measurement log, such that the values of the PCRs can be reconstructed from the individual measurements for validation.

Via its PCRs, a TPM provides a Root of Trust that can, for example, support secure boot or remote attestation. The attestation of an endpoint's identity or security posture is based on the content of an TPM's PCRs (platform integrity measurements).

A.1.3. PCR restricted Keys

Every key inside the TPM can be restricted in such a way that it can only be used if a certain set of PCRs are in a predetermined state. For key creation the desired state for PCRs are defined via the PCRInfo field inside the keyInfo parameter. Whenever an operation using this key is performed, the TPM first checks whether the PCRs are in the correct state. Otherwise the operation is denied by the TPM.

A.1.4. CertifyInfo

The TPM offers a command to certify the properties of a key by means of a signature using another key. This includes especially the keyInfo which in turn includes the PCRInfo information used during key creation. This way, a third party can be assured about the fact that a key is only usable if the PCRs are in a certain state.

A.2. Protocol and Procedure

A.2.1. AIK and AIK Certificate

Attestations are based upon a cryptographic signature performed by the TPM using a so-called Attestation Identity Key (AIK). An AIK has the properties that it cannot be exported from a TPM and is used for attestations. Trust in the AIK is established by an X.509 Certificate emitted by a Certificate Authority. The AIK certificate is either provided directly or via a so-called PrivacyCA [AIK-Enrollment].

This element consists of the AIK certificate that includes the AIK's public key used during verification as well as the certificate chain up to the Root CA for validation of the AIK certificate itself.

```
TUDA-Cert = [AIK-Cert, TSA-Cert]; maybe split into two for SNMP
AIK-Cert = Cert
TSA-Cert = Cert
```

Figure 2: TUDA-Cert element in CDDL

The TSA-Cert is a standard certificate of the TSA.

The AIK-Cert may be provisioned in a secure environment using standard means or it may follow the PrivacyCA protocols. Figure 3 gives a rough sketch of this protocol. See [AIK-Enrollment] for more information.

The X.509 Certificate is built from the AIK public key and the corresponding PKCS #7 certificate chain, as shown in Figure 3.

Required TPM functions:

```
| create_AIK_Cert(...) = {
|   AIK = TPM_MakeIdentity()
|   IdReq = CollateIdentityRequest(AIK,EK)
|   IdRes = Call(AIK-CA, IdReq)
|   AIK-Cert = TPM_ActivateIdentity(AIK, IdRes)
| }
|
| /* Alternative */
|
| create_AIK_Cert(...) = {
|   AIK = TPM_CreateWrapKey(Identity)
|   AIK-Cert = Call(AIK-CA, AIK.pubkey)
| }
```

Figure 3: Creating the TUDA-Cert element

A.2.2. Synchronization Token

The reference for Attestations are the Tick-Sessions of the TPM. In order to put Attestations into relation with a Real Time Clock (RTC), it is necessary to provide a cryptographic synchronization between the tick session and the RTC. To do so, a synchronization protocol is run with a Time Stamp Authority (TSA) that consists of three steps:

- The TPM creates a TickStampBlob using the AIK

- This TickstampBlob is used as nonce to the Timestamp of the TSA

- Another TickStampBlob with the AIK is created using the TSA's Timestamp a nonce

The first TickStampBlob is called "left" and the second "right" in a reference to their position on a time-axis.

These three elements, with the TSA's certificate factored out, form the synchronization token

```
TUDA-Synctoken = [
  left: TickStampBlob-Output,
  timestamp: TimeStampToken,
  right: TickStampBlob-Output,
]

TimeStampToken = bytes ; RFC 3161

TickStampBlob-Output = [
  currentTicks: TPM-CURRENT-TICKS,
  sig: bytes,
]

TPM-CURRENT-TICKS = [
  currentTicks: uint
  ? (
    tickRate: uint
    tickNonce: TPM-NONCE
  )
]
; Note that TickStampBlob-Output "right" can omit the values for
;   tickRate and tickNonce since they are the same as in "left"

TPM-NONCE = bytes .size 20
```

Figure 4: TUDA-Sync element in CDDL

Required TPM functions:

```
| dummyDigest = h'0000000000000000000000000000000000000000'
| dummyNonce = dummyDigest
|
| create_sync_token(AIKHandle, TSA) = {
|   ts_left = TPM_TickStampBlob(
|       keyHandle = AIK_Handle,      /*TPM_KEY_HANDLE*/
|       antiReplay = dummyNonce,     /*TPM_NONCE*/
|       digestToStamp = dummyDigest  /*TPM_DIGEST*/)
|
|   ts = TSA_Timestamp(TSA, nonce = hash(ts_left))
|
|   ts_right = TPM_TickStampBlob(
|       keyHandle = AIK_Handle,      /*TPM_KEY_HANDLE*/
|       antiReplay = dummyNonce,     /*TPM_NONCE*/
|       digestToStamp = hash(ts))    /*TPM_DIGEST*/
|
|   TUDA-SyncToken = [[ts_left.ticks, ts_left.sig], ts,
|                     [ts_right.ticks.currentTicks, ts_right.sig]]
|   /* Note: skip the nonce and tickRate field for ts_right.ticks */
| }
```

Figure 5: Creating the Sync-Token element

A.2.3. RestrictionInfo

The attestation relies on the capability of the TPM to operate on restricted keys. Whenever the PCR values for the machine to be attested change, a new restricted key is created that can only be operated as long as the PCRs remain in their current state.

In order to prove to the Verifier that this restricted temporary key actually has these properties and also to provide the PCR value that it is restricted, the TPM command TPM_CertifyInfo is used. It creates a signed certificate using the AIK about the newly created restricted key.

This token is formed from the list of:

- PCR list,
- the newly created restricted public key, and
- the certificate.

```
TUDA-RestrictionInfo = [Composite,
                        restrictedKey_Pub: Pubkey,
                        CertifyInfo]
```

```
PCRSelection = bytes .size (2..4) ; used as bit string

Composite = [
  bitmask: PCRSelection,
  values: [*PCR-Hash],
]

Pubkey = bytes ; may be extended to COSE pubkeys

CertifyInfo = [
  TPM-CERTIFY-INFO,
  sig: bytes,
]

TPM-CERTIFY-INFO = [
  ; we don't encode TPM-STRUCT-VER:
  ; these are 4 bytes always equal to h'01010000'
  keyUsage: uint, ; 4byte? 2byte?
  keyFlags: bytes .size 4, ; 4byte
  authDataUsage: uint, ; 1byte (enum)
  algorithmParms: TPM-KEY-PARMS,
  pubkeyDigest: Hash,
  ; we don't encode TPM-NONCE data, which is 20 bytes, all zero
  parentPCRStatus: bool,
  ; no need to encode pcrinfosize
  pcrinfo: TPM-PCR-INFO,        ; we have exactly one
]

TPM-PCR-INFO = [
    pcrSelection: PCRSelection; /* TPM_PCR_SELECTION */
    digestAtRelease: PCR-Hash;  /* TPM_COMPOSITE_HASH */
    digestAtCreation: PCR-Hash; /* TPM_COMPOSITE_HASH */
]

TPM-KEY-PARMS = [
  ; algorithmID: uint, ; <= 4 bytes -- not encoded, constant for TPM1.2
  encScheme: uint, ; <= 2 bytes
  sigScheme: uint, ; <= 2 bytes
  parms: TPM-RSA-KEY-PARMS,
]

TPM-RSA-KEY-PARMS = [
  ; "size of the RSA key in bits":
  keyLength: uint
  ; "number of prime factors used by this RSA key":
  numPrimes: uint
  ; "This SHALL be the size of the exponent":
  exponentSize: null / uint / biguint
```

```
   ; "If the key is using the default exponent then the exponentSize
   ; MUST be 0" -> we represent this case as null
 ]
```

Figure 6: TUDA-Key element in CDDL

Required TPM functions:

```
| dummyDigest = h'0000000000000000000000000000000000000000'
| dummyNonce = dummyDigest
|
| create_Composite
|
| create_restrictedKey_Pub(pcrsel) = {
|   PCRInfo = {pcrSelection = pcrsel,
|              digestAtRelease = hash(currentValues(pcrSelection))
|              digestAtCreation = dummyDigest}
|   / * PCRInfo is a TPM_PCR_INFO and thus also a TPM_KEY */
|
|   wk = TPM_CreateWrapKey(keyInfo = PCRInfo)
|   wk.keyInfo.pubKey
| }
|
| create_TPM-Certify-Info = {
|   CertifyInfo = TPM_CertifyKey(
|       certHandle = AIK,          /* TPM_KEY_HANDLE */
|       keyHandle = wk,            /* TPM_KEY_HANDLE */
|       antiReply = dummyNonce)    /* TPM_NONCE */
|
|   CertifyInfo.strip()
|   /* Remove those values that are not needed */
| }
```

Figure 7: Creating the pubkey

A.2.4. Measurement Log

Similarly to regular attestations, the Verifier needs a way to reconstruct the PCRs' values in order to estimate the trustworthiness of the device. As such, a list of those elements that were extended into the PCRs is reported. Note though that for certain environments, this step may be optional if a list of valid PCR configurations exists and no measurement log is required.

```
TUDA-Measurement-Log = [*PCR-Event]
PCR-Event = [
  type: PCR-Event-Type,
  pcr: uint,
  template-hash: PCR-Hash,
  filedata-hash: tagged-hash,
  pathname: text; called filename-hint in ima (non-ng)
]

PCR-Event-Type = &(
  bios: 0
  ima: 1
  ima-ng: 2
)

; might want to make use of COSE registry here
; however, that might never define a value for sha1
tagged-hash /= [sha1: 0, bytes .size 20]
tagged-hash /= [sha256: 1, bytes .size 32]
```

A.2.5. Implicit Attestation

The actual attestation is then based upon a TickStampBlob using the restricted temporary key that was certified in the steps above. The TPM-Tickstamp is executed and thereby provides evidence that at this point in time (with respect to the TPM internal tick-session) a certain configuration existed (namely the PCR values associated with the restricted key). Together with the synchronization token this tick-related timing can then be related to the real-time clock.

This element consists only of the TPM_TickStampBlock with no nonce.

```
TUDA-Verifytoken = TickStampBlob-Output
```

Figure 8: TUDA-Verify element in CDDL

Required TPM functions:

```
| imp_att = TPM_TickStampBlob(
|     keyHandle = restrictedKey_Handle,     /*TPM_KEY_HANDLE*/
|     antiReplay = dummyNonce,              /*TPM_NONCE*/
|     digestToStamp = dummyDigest)          /*TPM_DIGEST*/
|
| VerifyToken = imp_att
```

Figure 9: Creating the Verify Token

A.2.6. Attestation Verification Approach

The seven TUDA information elements transport the essential content that is required to enable verification of the attestation statement at the Verifier. The following listings illustrate the verification algorithm to be used at the Verifier in pseudocode. The pseudocode provided covers the entire verification task. If only a subset of TUDA elements changed (see Section 2.1), only the corresponding code listings need to be re-executed.

```
| TSA_pub = verifyCert(TSA-CA, Cert.TSA-Cert)
| AIK_pub = verifyCert(AIK-CA, Cert.AIK-Cert)
```

Figure 10: Verification of Certificates

```
| ts_left = Synctoken.left
| ts_right = Synctoken.right
|
| /* Reconstruct ts_right's omitted values; Alternatively assert == */
| ts_right.currentTicks.tickRate = ts_left.currentTicks.tickRate
| ts_right.currentTicks.tickNonce = ts_left.currentTicks.tickNonce
|
| ticks_left = ts_left.currentTicks
| ticks_right = ts_right.currentTicks
|
| /* Verify Signatures */
| verifySig(AIK_pub, dummyNonce || dummyDigest || ticks_left)
| verifySig(TSA_pub, hash(ts_left) || timestamp.time)
| verifySig(AIK_pub, dummyNonce || hash(timestamp) || ticks_right)
|
| delta_left = timestamp.time -
|     ticks_left.currentTicks * ticks_left.tickRate / 1000
|
| delta_right = timestamp.time -
|     ticks_right.currentTicks * ticks_right.tickRate / 1000
```

Figure 11: Verification of Synchronization Token

```
| compositeHash = hash_init()
| for value in Composite.values:
|     hash_update(compositeHash, value)
| compositeHash = hash_finish(compositeHash)
|
| certInfo = reconstruct_static(TPM-CERTIFY-INFO)
|
| assert(Composite.bitmask == ExpectedPCRBitmask)
| assert(certInfo.pcrinfo.PCRSelection == Composite.bitmask)
| assert(certInfo.pcrinfo.digestAtRelease == compositeHash)
| assert(certInfo.pubkeyDigest == hash(restrictedKey_Pub))
|
| verifySig(AIK_pub, dummyNonce || certInfo)
```

Figure 12: Verification of Restriction Info

```
| for event in Measurement-Log:
|     if event.pcr not in ExpectedPCRBitmask:
|         continue
|     if event.type == BIOS:
|         assert_whitelist-bios(event.pcr, event.template-hash)
|     if event.type == ima:
|         assert(event.pcr == 10)
|         assert_whitelist(event.pathname, event.filedata-hash)
|         assert(event.template-hash ==
|                hash(event.pathname || event.filedata-hash))
|     if event.type == ima-ng:
|         assert(event.pcr == 10)
|         assert_whitelist-ng(event.pathname, event.filedata-hash)
|         assert(event.template-hash ==
|                hash(event.pathname || event.filedata-hash))
|
|     virtPCR[event.pcr] = hash_extend(virtPCR[event.pcr],
|                                      event.template-hash)
|
| for pcr in ExpectedPCRBitmask:
|     assert(virtPCR[pcr] == Composite.values[i++]
```

Figure 13: Verification of Measurement Log

```
| ts = Verifytoken
|
| /* Reconstruct ts's omitted values; Alternatively assert == */
| ts.currentTicks.tickRate = ts_left.currentTicks.tickRate
| ts.currentTicks.tickNonce = ts_left.currentTicks.tickNonce
|
| verifySig(restrictedKey_pub, dummyNonce || dummyDigest || ts)
|
| ticks = ts.currentTicks
|
| time_left = delta_right + ticks.currentTicks * ticks.tickRate / 1000
| time_right = delta_left + ticks.currentTicks * ticks.tickRate / 1000
|
| [time_left, time_right]
```

Figure 14: Verification of Attestation Token

Acknowledgements

Authors' Addresses

Andreas Fuchs
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

Email: andreas.fuchs@sit.fraunhofer.de

Henk Birkholz
Fraunhofer Institute for Secure Information Technology
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Ira E McDonald
High North Inc
PO Box 221
Grand Marais 49839
US

Email: blueroofmusic@gmail.com

Carsten Bormann
Universitaet Bremen TZI
Bibliothekstr. 1
Bremen D-28359
Germany

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