Arm’s Platform Security Architecture (PSA) Attestation Token
draft-tschofenig-rats-psa-token-01

Abstract

The insecurity of IoT systems is a widely known and discussed problem. The Arm Platform Security Architecture (PSA) is being developed to address this challenge by making it easier to build secure systems.

This document specifies token format and claims used in the attestation API of the Arm Platform Security Architecture (PSA).

At its core, the Entity Attestation Token (EAT) format is used and populated with a set of claims. This specification describes what claims are used by the PSA and what has been implemented within Arm Trusted Firmware-M.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 10, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Modern hardware for Internet of Things devices contain trusted execution environments and in case of the Arm v8-M architecture TrustZone support. TrustZone on these low end microcontrollers allows the separation between a normal world and a secure world where security sensitive code resides in the secure world and is executed by applications running on the normal world using a well-defined API.
Various APIs have been developed by Arm as part of the Platform Security Architecture [PSA]; this document focuses on the functionality provided by the attestation API. Since the tokens exposed via the attestation API are also consumed by services outside the device, interoperability needs arise. In this specification these interoperability needs are addressed by a combination of

- a set of claims encoded in CBOR,
- embedded in a CBOR Web Token (CWT),
- protected by functionality offered by the CBOR Object Signing and Encryption (COSE) specification.

Figure 1 shows the architecture graphically. Apps on the IoT device communicate with services on the secure world using a defined API. The attestation API exposes tokens, as described in this document, and those tokens may be presented to network or application services.
2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.1. Glossary

RoT  Root of Trust, the minimal set of software, hardware and data that has to be implicitly trusted in the platform - there is no software or hardware at a deeper level that can verify that the Root of Trust is authentic and unmodified.

SPE  Secure Processing Environment, a platform’s processing environment for software that provides confidentiality and integrity for its runtime state, from software and hardware,
outside of the SPE. Contains the Secure Partition Manager, the Secure Partitions and the trusted hardware.

NSPE  Non Secure Processing Environment, the security domain outside of the SPE, the Application domain, typically containing the application firmware and hardware.

3. Information Model

Table 1 describes the utilized claims.

<table>
<thead>
<tr>
<th>Claim</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenge</td>
<td>Yes</td>
<td>Input object from the caller. For example, this can be a cryptographic nonce, a hash of locally attested data, or both. The length must be 32, 48, or 64 bytes.</td>
</tr>
<tr>
<td>Instance ID</td>
<td>Yes</td>
<td>Represents the unique identifier of the instance. It is a hash of the public key corresponding to the Initial Attestation Key.</td>
</tr>
<tr>
<td>Verification Service</td>
<td>No</td>
<td>Information used by a relying party to locate a validation service for the token. The value is a text string that can be used to locate the service or a URL specifying the address of the service.</td>
</tr>
<tr>
<td>Profile Definition</td>
<td>No</td>
<td>Contains the name of a document that describes the ‘profile’ of the report. The document name may include versioning. The value for this specification is PSA_IOT_PROFILE_1.</td>
</tr>
<tr>
<td>Implementation ID</td>
<td>Yes</td>
<td>Represents the original implementation signer of the attestation key and identifies the contract between the report and verification. A verification service will use this claim to locate the details of the</td>
</tr>
<tr>
<td>Field</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Client ID</td>
<td>Yes</td>
<td>Represents the Partition ID of the caller. It is a signed integer whereby negative values represent callers from the NSPE and where positive IDs represent callers from the SPE. The full definition of the partition ID is defined in the PSA Firmware Framework (PSA-FF) [PSA-FF].</td>
</tr>
<tr>
<td>Security Lifecycle</td>
<td>Yes</td>
<td>Represents the current lifecycle state of the PSA RoT. The state is represented by a 16-bit unsigned integer that is divided to convey a major state and a minor state. A major state is defined by [PSA-SM]. A minor state is ‘IMPLEMENTATION DEFINED’. The encoding is: version[15:8] - PSA lifecycle state, and version[7:0] - IMPLEMENTATION DEFINED state. The PSA lifecycle states are listed below. For PSA, a remote verifier can only trust reports from the PSA RoT when it is in SECURED, NON_PSA_ROT_DEBUG or RECOVERABLE_PSA_ROT_DEBUG major states.</td>
</tr>
<tr>
<td>Hardware version</td>
<td>No</td>
<td>Provides metadata linking the token to the GDSII that went to fabrication for this instance. It can be used to link the class of chip and PSA RoT to the data on a certification website. It must be represented as a thirteen-digit [EAN-13] code.</td>
</tr>
<tr>
<td>Boot Seed</td>
<td>Yes</td>
<td>Represents a random value created at system boot time that will allow differentiation of reports from different system sessions.</td>
</tr>
<tr>
<td>Software Components</td>
<td>Yes (unless the No</td>
<td>A list of software components that represent the entire</td>
</tr>
<tr>
<td></td>
<td>the No</td>
<td></td>
</tr>
</tbody>
</table>
No Software Measurements | Yes (if no software components specified) | In the event that the implementation does not contain any software measurements then the Software Components claim above can be omitted but instead it will be mandatory to include this claim to indicate this is a deliberate state.

| Software Measurements claim is specified | software state of the system. | This claim is recommended in order to comply with the rules outlined in the [PSA-SM]. The software components are further explained below.

Table 1: Information Model of PSA Attestation Claims.

The PSA lifecycle states consist of the following values:

- UNKNOWN (0x1000u)
- PSA_ROT_PROVISIONING (0x2000u)
- SECURED (0x3000u)
- NON_PSA_ROT_DEBUG (0x4000u)
- RECOVERABLE_PSA_ROT_DEBUG (0x5000u)
- PSA_LIFECYCLE_DECOMMISSIONED (0x6000u)

Table 2 shows the structure of each software component entry in the Software Components claim.

<table>
<thead>
<tr>
<th>Key ID</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Measurement Type</td>
<td>No</td>
<td>A short string representing the role of this software component (e.g., ‘BL’ for Boot Loader).</td>
</tr>
<tr>
<td>2</td>
<td>Measurement Value</td>
<td>Yes</td>
<td>Represents a hash of the invariant software component in memory at startup time. The value must be a cryptographic hash of</td>
</tr>
</tbody>
</table>

Tschofenig, et al. Expires October 10, 2019
<table>
<thead>
<tr>
<th></th>
<th>Claim Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Reserved</td>
<td>No</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>Version</td>
<td>No</td>
<td>The issued software version in the form of a text string. The value of this claim will correspond to the entry in the original signed manifest of the component.</td>
</tr>
<tr>
<td>5</td>
<td>Signer ID</td>
<td>No</td>
<td>The hash of a signing authority public key for the software component. The value of this claim will correspond to the entry in the original manifest for the component.</td>
</tr>
<tr>
<td>6</td>
<td>Measurement description</td>
<td>No</td>
<td>Description of the software component, which represents the way in which the measurement value of the software component is computed. The value will be a text string containing an abbreviated description (or name) of the measurement method which can be used to lookup the details of the method in a profile document. This claim will normally be excluded, unless there was an exception to the default measurement described in the profile for a specific component.</td>
</tr>
</tbody>
</table>

Table 2: Software Components Claims.

The following measurement types are currently defined:

- ‘BL’: a Boot Loader
- ‘PROT’: a component of the PSA Root of Trust
- ‘ARoT’: a component of the Application Root of Trust
- ‘App’: a component of the NSPE application
4. Token Encoding

The report is represented as a token, which must be formatted in accordance to the Entity Attestation Token (EAT) [I-D.mandyam-eat]. The token consists of a series of claims declaring evidence as to the nature of the instance of hardware and software. The claims are encoded in CBOR [RFC7049] format.

5. Claims

The token is modelled to include custom values that correspond to the following claims suggested in the EAT specification:

- nonce (mandatory); arm_psa_nonce is used instead
- UEID (mandatory); arm_psa_UEID is used instead
- origination (recommended); arm_psa_origination is used instead

Later revisions of this documents might phase out those custom claims to be replaced by the EAT standard claims.

As noted, some fields must be at least 32 bytes long to provide sufficient cryptographic strength.
<table>
<thead>
<tr>
<th>Claim Key</th>
<th>Claim Description</th>
<th>Claim Name</th>
<th>CBOR Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-75000</td>
<td>Profile Definition</td>
<td>arm_psa_profile_id</td>
<td>Text string</td>
</tr>
<tr>
<td>-75001</td>
<td>Client ID</td>
<td>arm_psa_partition_id</td>
<td>Unsigned integer or Negative integer</td>
</tr>
<tr>
<td>-75002</td>
<td>Security Lifecycle</td>
<td>arm_psa_security_lifecycle</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>-75003</td>
<td>Impl. ID</td>
<td>arm_psa_implementation_id</td>
<td>Byte string (&gt;=32 bytes)</td>
</tr>
<tr>
<td>-75004</td>
<td>Boot Seed</td>
<td>arm_psa_boot_seed</td>
<td>Byte string (&gt;=32 bytes)</td>
</tr>
<tr>
<td>-75005</td>
<td>Hardware Version</td>
<td>arm_psa_hw_version</td>
<td>Text string</td>
</tr>
<tr>
<td>-75006</td>
<td>Software Components</td>
<td>arm_psa_sw_components</td>
<td>Array of map entries. (compound map claim)</td>
</tr>
<tr>
<td>-75007</td>
<td>No Software Measurements</td>
<td>arm_psa_no_sw_measurements</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>-75008</td>
<td>Challenge</td>
<td>arm_psa_nonce</td>
<td>Byte string</td>
</tr>
<tr>
<td>-75009</td>
<td>Instance ID</td>
<td>arm_psa_UEID</td>
<td>Byte string</td>
</tr>
<tr>
<td>-75010</td>
<td>Verification Service Indicator</td>
<td>arm_psa_origination</td>
<td>Byte string or StringOrURI</td>
</tr>
</tbody>
</table>

Each map entry of the software component claim MUST have the following types for each key value:

1. Text string (type)
2. Byte string (measurement, >=32 bytes)
3. Reserved

4. Text string (version)

5. Byte string (signer ID, >=32 bytes)

6. Text string (measurement description)

The following key values will be present in the software components claim: 1 (Type), 2 (Measurement Value), 4 (Version) and 5 (Signer ID). Keys 3 (Reserved) and 6 (Measurement Description) will not be present. Instead of a referenced Measurement Description it is defined that all cases, the software measurement value is taken as a SHA256 hash of the software image, prior to it executing in place.

6. Example

The following example shows an attestation token that was produced for a device that has a single-stage bootloader, and an RTOS with a device management client. From a code point of view, the RTOS and the device management client form a single binary.

EC key using curve P-256 with:
- x: 0xdcf0d0f4bcd5e26a54ee36cad660d283d12abc5f7307de58689e77cd60452e75
- y: 0x8cbadb5fe9f89a7107e5a2e8ea44ec1b09b7da2a1a82a0252a4c1c26ee1ed7cf
- d: 0xc74670bcb7e85b3803efb428940492e73e3fe9d4f7b5a8ad5e480cbdbcb554c2

Key using COSE format (base64-encoded):

```
pSJYIIy621/p+JpxB+Wi6opE7BsJt9ogGoKgLSpMHCBuHitfPI1ggx0ZwvLfoWzgD77QolASS5z4/6dT3taitXkgMvby1VMIBAIfYNzw0PS81eJqVO42ytZg0oPRKrxcWfeWGied81gRS51IAE=
```

Example of EAT token (base64-encoded):

```
```
Same token using extended CBOR diagnostic format:

```
18{
    /protected / h’a10126’ /
    \ alg \ 1: -7 \ ECDSA 256 
},

/unsupported / {}

/payload / h’a93a000124fb58200000102030405060708090a0b0c0d0e0f101112
31415161718191alb1c1d1e1f3a000124fa5820000102030405060708090a0b0c0d0e
0f101112131415161718191alb1c1d1e1f3a000124f84a4025820000102030405060
708090a0b0c0d0e0f101112131415161718191alb1c1d1e1f0465332e312e340505820
00102030405060708090a0b0c0d0e0f101112131415161718191alb1c1d1e1f01624
24ca4025820000102030405060708090a0b0c0d0e0f101112131415161718191alb1c
1d1e1f0463312e31055820000102030405060708090a0b0c0d0e0f101112131415161
718191alb1c1d1e1f016450526f5a4025820000102030405060708090a0b0c0d0e0f10
1112131415161718191alb1c1d1e1f0463312e305055820000102030405060708090
a0b0c0d0e0f101112131415161718191alb1c1d1e1f016441526f5440258200001020
30405060708090a0b0c0d0e0f101112131415161718191alb1c1d1e1f0463322e320
55820000102030405060708090a0b0c0d0e0f101112131415161718191alb1c1d1e1f
01634170703a000124f91930003a000124f5820000102030405060708090a0b0c0d0
e0f101112131415161718191alb1c1d1e1f3a000125016c7037315f7665726966696
723a000124f8203a0001250058201000102030405060708090a0b0c0d0e0f101112
31415161718191alb1c1d1e1f3a000124f7715053415f49654f550244694c455f
31’ {
    /arm_psa_boot_seed / -75004: h’000102030405060708090a0b0c0d0e0f101112
131415161718191alb1c1d1e1f’,
    /arm_psa_implementation_id / -75003: h’000102030405060708090a0b0c
0d0e0f101112131415161718191alb1c1d1e1f’,
    /arm_psa_sw_components / -75006: [ 
        /measurement / 2: h’000102030405060708090a0b0c0d0e0f101112
131415161718191alb1c1d1e1f’,
        /version / 4: “3.1.4”,
    ]
```

Internet-Draft  PSA Attestation Token  April 2019

/ signerID / 5: h'000102030405060708090a0b0c0d0e0f10111231
   415161718191abc1d1e1f',
/ type / 1: "BL"
},
{
/ measurement / 2: h'000102030405060708090a0b0c0d0e0f101112
   131415161718191abc1d1e1f',
/ version / 4: "1.1",
/ signerID / 5: h'000102030405060708090a0b0c0d0e0f10111231
   415161718191abc1d1e1f',
/ type / 1: "PRoT"
},
{
/ measurement / 2: h'000102030405060708090a0b0c0d0e0f101112
   131415161718191abc1d1e1f',
/ version / 4: "1.0",
/ signerID / 5: h'000102030405060708090a0b0c0d0e0f10111231
   415161718191abc1d1e1f',
/ type / 1: "ARoT"
},
{
/ measurement / 2: h'000102030405060708090a0b0c0d0e0f101112
   131415161718191abc1d1e1f',
/ version / 4: "2.2",
/ signerID / 5: h'000102030405060708090a0b0c0d0e0f10111231
   415161718191abc1d1e1f',
/ type / 1: "App"
}

/ arm_psa_security_lifecycle / -75002: 12288 / SECURED /
/ arm_psa_nonce / -75008: h'000102030405060708090a0b0c0d0e0f1011
   2131415161718191abc1d1e1f',
/ arm_psa_origination / -75010: "psa_verifier",
/ arm_psa_partition_id / -75001: -1,
/ arm_psa_UeID / -75009: h'01000102030405060708090a0b0c0d0e0f1011
   1231415161718191abc1d1e1f',
/ arm_psa_profile_id / -75000: "PSA_IoT_PROFILE_1"
}),
}
/ signature / h'58860508ee7e8cc48eba872dbb5d694a542b1322ad0d51023c197
  0df429f06501c683a951080a0cced0a6e80e966f22bd63d1c0056974a11ba332d7877
  87fb4f'
7. Security and Privacy Considerations

This specification re-uses the CWT and the EAT specification. Hence, the security and privacy considerations of those specifications apply here as well.

Since CWTs offer different ways to protect the token this specification profiles those options and only uses public key cryptography. The token MUST be signed following the structure of the COSE specification [RFC8152]. The COSE type MUST be COSE-Sign1.

Attestation tokens contain information that may be unique to a device and therefore they may allow single out an individual device for tracking purposes. Implementation must take to ensure that only those claims are included that fulfil the purpose of the application and that users of those devices consent to the data sharing.

8. IANA Considerations

IANA is requested to allocate the claims defined in Section 5 to the [RFC8392] created CBOR Web Token (CWT) Claims registry [IANA-CWT]. The change controller are the authors and the reference is this document.

9. References

9.1. Normative References


9.2. Informative References


Appendix A. Contributors

We would like to thank the following supporters for their contributions:

* Laurence Lundblade
  Security Theory LLC
  lgl@securitytheory.com

* Tamas Ban
  Arm Limited
  Tamas.Ban@arm.com

Appendix B. Reference Implementation

Trusted Firmware M (TF-M) [TF-M] is the name of the open source project that provides a reference implementation of PSA APIs, created for the latest Arm v8-M microcontrollers with TrustZone technology. TF-M provides foundational firmware components that silicon manufacturers and OEMs can build on (including trusted boot, secure device initialisation and secure function invocation).

Authors’ Addresses

Hannes Tschofenig (editor)
Arm Limited
EMail: hannes.tschofenig@arm.com

Simon Frost
Arm Limited
EMail: Simon.Frost@arm.com

Mathias Brossard
Arm Limited
EMail: Mathias.Brossard@arm.com

Adrian Shaw
Arm Limited
EMail: Adrian.Shaw@arm.com