Arm’s Platform Security Architecture (PSA) Attestation Token
draft-tschofenig-rats-psa-token-04

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 IoT systems.

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

At its core, the CWT (COSE Web Token) format is used and populated with a set of claims, in a way similar to EAT (Entity Attestation Token). This specification describes what claims are used by PSA compliant systems 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 May 23, 2020.
Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction ............................................... 3
2. Conventions and Terminology ............................... 4
   2.1. Glossary .............................................. 4
3. Information Model ......................................... 5
   3.1. PSA Lifecycle States ................................. 7
   3.2. PSA Software Components ............................. 7
4. Token Encoding ............................................. 9
5. Claims ...................................................... 9
6. Example .................................................... 11
7. Security and Privacy Considerations ........................ 14
8. IANA Considerations ....................................... 14
9. References ................................................ 14
   9.1. Normative References .................................. 14
   9.2. Informative References ................................ 15
Appendix A. Contributors .................................... 16
Appendix B. Reference Implementation ........................ 16
Authors’ Addresses ........................................... 16

1. Introduction

Modern hardware for Internet of Things devices contain trusted execution environments and in case of the Arm v8-M architecture TrustZone support. On these low end microcontrollers, TrustZone enables the separation between a "normal world" and a "secure world" where security sensitive code resides in the "secure world" and applications running in the "normal world" request secure services using a well-defined API. Various APIs have been developed by Arm as part of the Platform Security Architecture [PSA] programme; 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, there is an actual need for making them interoperable. In this specification these interoperability needs are addressed by describing the exact syntax and semantics of the attestation claims, and defining the way these claims are encoded and cryptographically protected.

Further details on concepts expressed below can be found in the PSA Security Model documentation [PSA-SM].

Figure 1 provides a view of the architectural components and how they interact. Applications on the IoT device communicate with services residing in the "secure world" by means of a well-defined API. The attestation API produces 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 (SPM), 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 mandatory and optional claims in the report.

<table>
<thead>
<tr>
<th>Claim</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth 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. 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. The full definition is in [PSA-SM].</td>
</tr>
<tr>
<td>Verification Service Indicator</td>
<td>No</td>
<td>A hint 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. A verifier may choose to ignore this claim in favor of other information.</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>Uniquely identifies the underlying immutable PSA RoT. A verification service can use this claim to locate the details of</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>negative values represent callers from the NSPE and where positive IDs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>represent callers from the SPE. The full definition of the partition ID is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>given in [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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>represented by an integer that is divided to convey a major state and a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>minor state. A major state is mandatory and defined by [PSA-SM]. A minor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>state is optional and ‘IMPLEMENTATION DEFINED’. The encoding is: version[15:8] - PSA security lifecycle state, and version[7:0] - IMPLEMENTATION DEFINED state. The PSA lifecycle states are listed in Section 3.1. For PSA, a remote verifier can only trust reports from the PSA RoT when it is in SECURED or NON_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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for this instance. It can be used to link the class of chip and PSA RoT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to the data on a certification website. It must be represented as a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>thirteen-digit [EAN-13]</td>
</tr>
<tr>
<td>Boot Seed</td>
<td>Yes</td>
<td>Represents a random value created at system boot time that will allow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>differentiation of reports from different boot sessions.</td>
</tr>
<tr>
<td>Software</td>
<td>Yes (unless</td>
<td>A list of software components</td>
</tr>
</tbody>
</table>
Components     |    the No    | that represent all the software
|                |   Software   | loaded by the PSA Root of Trust.
|                | Measurements | This claim is needed for the
|                |   claim is   | rules outlined in [PSA-SM]. The
|                |  specified)  | software components are further
detailed in Section 3.2.

No Software    | Yes (if no | In the event that the
Measurements   | software    | implementation does not contain
| components    |   components| any software measurements then
| specified)    |              | 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. This claim is
|               |              | intended for devices that are not
|               |              | compliant with [PSA-SM].

Table 1: Information Model of PSA Attestation Claims.

3.1. PSA Lifecycle States

The PSA lifecycle states consist of the following values:

- PSA_LIFECYCLE_UNKNOWN (0x0000u)
- PSA_LIFECYCLE_ASSEMBLY_AND_TEST (0x1000u)
- PSA_LIFECYCLE_PSA_ROT_PROVISIONING (0x2000u)
- PSA_LIFECYCLE_SECURED (0x3000u)
- PSA_LIFECYCLE_NON_PSA_ROT_DEBUG (0x4000u)
- PSA_LIFECYCLE_RECOVERABLE_PSA_ROT_DEBUG (0x5000u)
- PSA_LIFECYCLE_DECOMMISSIONED (0x6000u)

3.2. PSA Software Components

Each software component in the Software Components claim MUST include
the required properties of Table 2.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
</table>

Tschofenig, et al.        Expires May 23, 2020                  
[Page 7]
<table>
<thead>
<tr>
<th></th>
<th>Measurement Type</th>
<th>No</th>
<th>A short string representing the role of this software component (e.g. ‘BL’ for Boot Loader).</th>
</tr>
</thead>
<tbody>
<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 256 bits or stronger.</td>
</tr>
<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. This can be used by a verifier to ensure the components were signed by an expected trusted source. This field must be present to be compliant with [PSA-SM].</td>
</tr>
<tr>
<td>6</td>
<td>Measurement description</td>
<td>No</td>
<td>Description of 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 current 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
- 'TS': a component of a Trusted Subsystem

4. Token Encoding

The report is encoded as a COSE Web Token (CWT) [RFC8392], similar to the Entity Attestation Token (EAT) [I-D.ietf-rats-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

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>-7500 0</td>
<td>Profile Definition</td>
<td>arm_psa_profile_id</td>
<td>Text string</td>
</tr>
<tr>
<td>-7500 1</td>
<td>Client ID</td>
<td>arm_psa_partition_id</td>
<td>Unsigned integer or Negative integer</td>
</tr>
<tr>
<td>-7500 2</td>
<td>Security Lifecycle</td>
<td>arm_psa_security_lifecycle</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>-7500 3</td>
<td>Implementation ID</td>
<td>arm_psa_clean_id</td>
<td>Byte string (&gt;=32 bytes)</td>
</tr>
<tr>
<td>-7500 4</td>
<td>Boot Seed</td>
<td>arm_psa_boot_seed</td>
<td>Byte string (&gt;=32 bytes)</td>
</tr>
<tr>
<td>-7500 5</td>
<td>Hardware Version</td>
<td>arm_psa_hw_version</td>
<td>Text string</td>
</tr>
<tr>
<td>-7500 6</td>
<td>Software Components</td>
<td>arm_psa_sw_components</td>
<td>Array of map entries (compound map claim). See below for allowed key-values.</td>
</tr>
<tr>
<td>-7500 7</td>
<td>No Software Measurements</td>
<td>arm_psa_no_sw_measurements</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>-7500 8</td>
<td>Auth Challenge</td>
<td>arm_psa_nonce</td>
<td>Byte string</td>
</tr>
<tr>
<td>-7500 9</td>
<td>Instance ID</td>
<td>arm_psa_UED</td>
<td>Byte string (the type byte of the UED should be set to 0x01. The type byte is described in [I-D.ietf-rats-eat].)</td>
</tr>
<tr>
<td>-7501 0</td>
<td>Verification Service Indicator</td>
<td>arm_psa_origination</td>
<td>Byte string</td>
</tr>
</tbody>
</table>
When using the Software Components claim each key value MUST correspond to the following types:

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)

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:
  0x8cbadb5fe9f89a7107e8a44ec1b09b7da2a1a82a0252a4c1c26ee1ed7cf
- d:
  0xc74670bcb7e85b3803efb428940492e73e3fe9d4f7b5a8ad5e480cbdbcb554c2

Key using COSE format (base64-encoded):

```text
pSJYIIy621/p+JpxB+Wi6OpE7BsJt9ogGoKgJSpMHCbuhTtfPI1ggx02wvLfoWzgD77Q
olASSS5z4/6dT3taitXkgMvby1VMIBAifYINzw0PS81eJqVO42ytZg0oPRKrxfcwfwG
ied81gRS51IAE=
```

Example of EAT token (base64-encoded):
Same token using extended CBOR diagnostic format:

```plaintext
18{
  /protected / h'\text{a10126}' / {
    \alg / 1: \text{-7} \ ECDSA 256 \ },
  /unprotected / {},
  /payload / h'a93a000124fb5820000102030405060708090a0b0c0d0e0f10000
3141516171819a1b1c1d1e1f3a0000124fa5820000102030405060708090a0b0c0
d0e0f100112123141516171819a1b1c1d1e1f3a0000124fa4a025820000102030405060
708090a0b0c0d0e0f100112123141516171819a1b1c1d1e1f3a0463312e310558200001
02030405060708090a0b0c0d0e0f10000000000102030405060708090a0b0c0d0e0f10
112123141516171819a1b1c1d1e1f0463312e31055820000102030405060708090a0b0
c0d0e0f100112123141516171819a1b1c1d1e1f0463312e3105582000010203040506
0708090a0b0c0d0e0f100112123141516171819a1b1c1d1e1f0463312e3105582000010
2030405060708090a0b0c0d0e0f100112123141516171819a1b1c1d1e1f0463312e310
55820000102030405060708090a0b0c0d0e0f100112123141516171819a1b1c1d1e1f0
1634170703a000124fb91930003a0000124fa5820000102030405060708090a0b0c0
do0f100112123141516171819a1b1c1d1e1f3a000125016c7073615f7665572696669
6723a000124fb8203a00012500582101000102030405060708090a0b0c0d0e0f101112
3141516171819a1b1c1d1e1f3a0000124f7715053415f496f5455024f6494c455f31' /
  /arm_psa_boot_seed / -75004: h'000102030405060708090a0b0c0d0e0f101121
3141516171819a1b1c1d1e1f3a000125016c7073615f76655726966696723a000124fb8
203a00012500582101000102030405060708090a0b0c0d0e0f1011123141516171819a
1b1c1d1e1f3a0000124f7715053415f496f5455024f6494c455f31' /
  /arm_psa_implementation_id / -75003: h'000102030405060708090a0b0c0
0d0e0f1011123141516171819a1b1c1d1e1f3a000125016c7073615f7665572696669
6723a000124fb8203a00012500582101000102030405060708090a0b0c0d0e0f101112
3141516171819a1b1c1d1e1f3a0000124f7715053415f496f5455024f6494c455f31' /
  /arm_psa_sw_components / -75006: [
    /measurement / 2: h'000102030405060708090a0b0c0d0e0f1011123141516171819a
1b1c1d1e1f3a0000124f7715053415f496f5455024f6494c455f31',
    /version / 4: "3.1.4",
}.
```

Internet-Draft            PSA Attestation Token            November 2019

/ signerID / 5: h'000102030405060708090a0b0c0d0e0f101112311415161718191a1b1c1d1e1f',
/ type / 1: "BL"
,
{ / measurement / 2: h'000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f',
/ version / 4: "1.1",
/ signerID / 5: h'000102030405060708090a0b0c0d0e0f101112311415161718191a1b1c1d1e1f',
/ type / 1: "PRoT"
},
{ / measurement / 2: h'000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f',
/ version / 4: "1.0",
/ signerID / 5: h'000102030405060708090a0b0c0d0e0f101112311415161718191a1b1c1d1e1f',
/ type / 1: "ARoT"
},
{ / measurement / 2: h'000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f',
/ version / 4: "2.2",
/ signerID / 5: h'000102030405060708090a0b0c0d0e0f101112311415161718191a1b1c1d1e1f',
/ type / 1: "App"
}

/ arm_psa_security_lifecycle / -75002: 12288 / SECURED /,
/ arm_psa_nonce / -75008: h'000102030405060708090a0b0c0d0e0f101112311415161718191a1b1c1d1e1f',
/ arm_psa_origination / -75010: "psa_verifier",
/ arm_psa_partition_id / -75001: -1,
/ arm_psa_UEID / -75009: h'01000102030405060708090a0b0c0d0e0f101112311415161718191a1b1c1d1e1f',
/ arm_psa_profile_id / -75000: "PSA_IoT_PROFILE_1"
}

/ signature / h'58860508ee7e8cc48eba872dbb5d694a542b1322ad0d51023c1970df429f06501c683a95108a0cced0a6e80e096f22bd63d1c0056974a11ba332d787787fb4f'
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 to single out an individual device for tracking purposes. Implementation must take appropriate measures 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 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 and an SPM, 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
Thomas Fossati
Arm Limited
EMail: thomas.fossati@arm.com