RATS
Internet-Draft
Intended status: Informational
Expires: 7 September 2020

H. Tschofenig
S. Frost
M. Brossard
A. Shaw
T. Fossati
Arm Limited
6 March 2020

Arm's Platform Security Architecture (PSA) Attestation Token
draft-tschofenig-rats-psa-token-05

Abstract

The Platform Security Architecture (PSA) is a family of hardware and firmware security specifications, as well as open-source reference implementations, to help device makers and chip manufacturers build best-practice security into products. Devices that are PSA compliant are able to produce attestation tokens as described in this memo, which are the basis for a number of different protocols, including secure provisioning and network access control. This document specifies the PSA attestation token structure and semantics.

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.

Note to Readers

Source for this draft and an issue tracker can be found at https://github.com/thomas-fossati/draft-psa-token (https://github.com/thomas-fossati/draft-psa-token).

Status of This Memo

This Internet-Draft will expire on 7 September 2020.

Copyright Notice

Table of Contents

1. Introduction

Trusted execution environments are now present in many devices, which provide a safe environment to place security sensitive code such as cryptography, secure boot, secure storage, and other essential security functions. These security functions are typically exposed through a narrow and well-defined interface, and can be used by operating system libraries and applications. Various APIs have been developed by Arm as part of the Platform Security Architecture [PSA] framework. This document focuses on the output provided by PSA's Initial Attestation API. Since the tokens are also consumed by services outside the device, there is an actual need to ensure interoperability. Interoperability needs are addressed here 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].

2. Conventions and Definitions

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. An example of RoT is an initial bootloader in ROM, which contains cryptographic functions and credentials, running on a specific hardware platform.

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 trusted code and trusted hardware. (Equivalent to Trusted Execution Environment (TEE), or "secure world".)

NSPE Non Secure Processing Environment, the security domain outside of the SPE, the Application domain, typically containing the application firmware, operating systems, and general hardware. (Equivalent to Rich Execution Environment (REE), or "normal world".)

3. PSA Claims

This section describes the claims to be used in a PSA attestation token.

CDDL [RFC8610] along with text descriptions is used to define each claim independent of encoding. The following CDDL type(s) are reused by different claims:

```
psa-hash-type = bytes .size 32 / bytes .size 48 / bytes .size 64
```

3.1. Caller Claims

3.1.1. Auth Challenge

The Auth Challenge claim is an 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.

This claim MUST be present in a PSA attestation token.

```
psa-nonce-claim = (
    arm_psa_nonce => psa-hash-type
```

```
)
```

3.1.2. Client ID

The Client ID claim 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 value 0 is not permitted. For a definition of the Partition ID, see the PSA Firmware Framework [PSA-FF].

It is essential that this claim is checked in the verification process to ensure that a security domain, i.e., an attestation endpoint, cannot spoof a report from another security domain.

This claim MUST be present in a PSA attestation token.


```
psa-client-id-nspe-type = -2147483648...0
psa-client-id-spe-type = 1..2147483647

psa-client-id-type = psa-client-id-nspe-type / psa-client-id-spe-type

psa-client-id = (
    arm_psa_partition_id => psa-client-id-type
)
```

3.2. Target Identification Claims

3.2.1. Instance ID

The Instance ID claim represents the unique identifier of the device instance. It is a 32 bytes hash of the public key corresponding to the Initial Attestation Key (IAK). If the IAK is a symmetric key then the Instance ID is a hash of the IAK itself. It is encoded as a Universal Entity ID of type RAND [I-D.ietf-rats-eat], i.e., prepending a 0x01 type byte to the key hash. The full definition is in [PSA-SM].

This claim MUST be present in a PSA attestation token.

```
psa-instance-id-type = bytes .size 33

psa-instance-id = (
    arm_psa_UEID => psa-instance-id-type
)
```

3.2.2. Implementation ID

The Implementation ID claim uniquely identifies the underlying immutable PSA RoT. A verification service can use this claim to locate the details of the verification process. Such details include the implementation's origin and associated certification state. The full definition is in [PSA-SM].

This claim MUST be present in a PSA attestation token.

```
psa-implementation-id-type = bytes .size 32

psa-implementation-id = (
    arm_psa_implementation_id => psa-implementation-id-type
)
```


3.2.3. Hardware Version

The Hardware Version claim 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].

```
psa-hardware-version-type = text .regexp "[0-9]{13}"

psa-hardware-version = (
    ? arm_psa_hw_version => psa-hardware-version-type
)
```

3.3. Target State Claims

3.3.1. Security Lifecycle

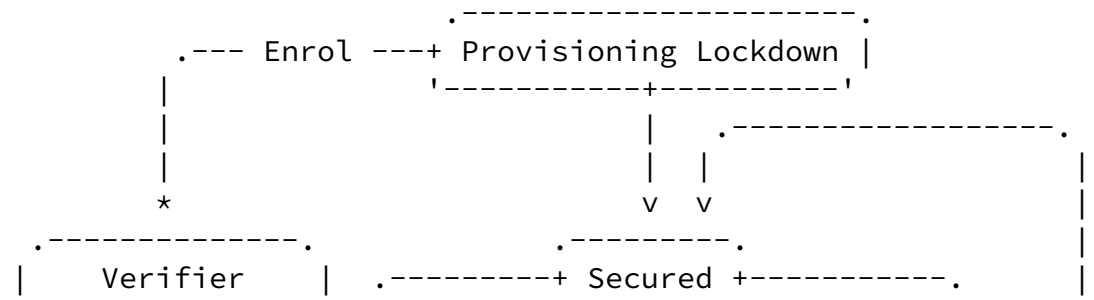
The Security Lifecycle claim represents the current lifecycle state of the PSA RoT. The state is represented by an integer that is divided to convey a major state and a minor state. A major state is mandatory and defined by [PSA-SM]. A minor state is optional and 'IMPLEMENTATION DEFINED'. The PSA security lifecycle state and implementation state are encoded as follows:

* version[15:8] - PSA security lifecycle state, and

* version[7:0] - IMPLEMENTATION DEFINED state.

The PSA lifecycle states are illustrated in Figure 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.

This claim MUST be present in a PSA attestation token.



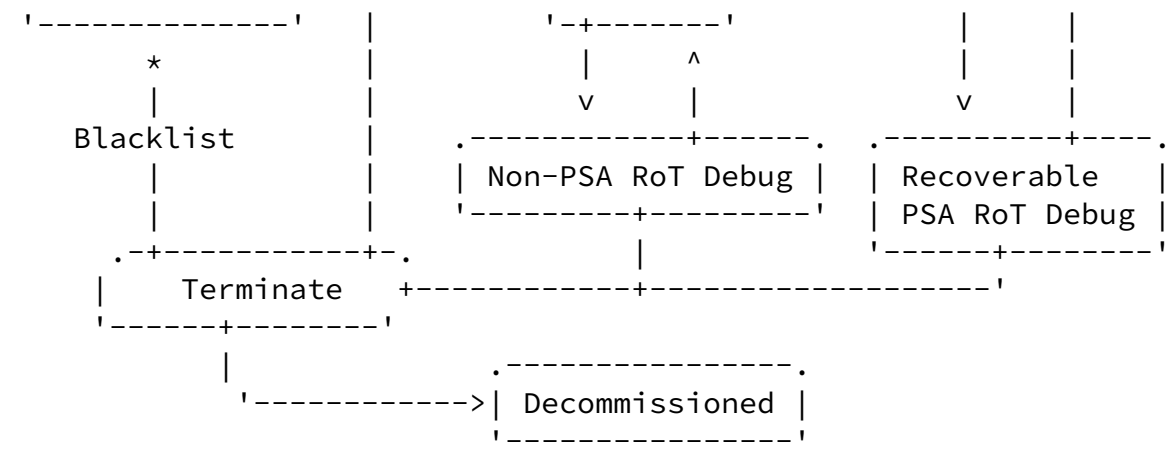


Figure 1: PSA Lifecycle States

```
psa-lifecycle-unknown-type = 0x0000..0x00ff
psa-lifecycle-assembly-and-test-type = 0x1000..0x10ff
psa-lifecycle-psa-rot-provisioning-type = 0x2000..0x20ff
psa-lifecycle-secured-type = 0x3000..0x30ff
psa-lifecycle-non-psa-rot-debug-type = 0x4000..0x40ff
psa-lifecycle-recoverable-psa-rot-debug-type = 0x5000..0x50ff
psa-lifecycle-decommissioned-type = 0x6000..0x60ff

psa-lifecycle-type =
    psa-lifecycle-unknown-type /
    psa-lifecycle-assembly-and-test-type /
    psa-lifecycle-psa-rot-provisioning-type /
    psa-lifecycle-secured-type /
    psa-lifecycle-non-psa-rot-debug-type /
    psa-lifecycle-recoverable-psa-rot-debug-type /
    psa-lifecycle-decommissioned-type

psa-lifecycle = (
    arm_psa_security_lifecycle => psa-lifecycle-type
)
```


3.3.2. Boot Seed

The Boot Seed claim represents a random value created at system boot time that will allow differentiation of reports from different boot sessions.

This claim MUST be present in a PSA attestation token.

```
psa-boot-seed-type = bytes .size 32

psa-boot-seed = (
    arm_psa_boot_seed => psa-boot-seed-type
)
```

3.4. Software Inventory Claims

3.4.1. Software Components

The Software Components claim is a list of software components that includes all the software loaded by the PSA RoT. This claim SHALL be included in attestation tokens produced by an implementation conformant with [PSA-SM]. If the Software Components claim is present, then the No Software Measurement claim (Section 3.4.2) MUST NOT be present.

Each entry in the Software Components list describes one software component using the attributes described in the following subsections. Unless explicitly stated, the presence of an attribute is OPTIONAL.

Note that, as described in [I-D.ietf-rats-architecture], a relying party will typically see the result of the verification process from the Verifier in form of an attestation result, rather than the "naked" PSA token from the attesting endpoint. Therefore, a relying party is not expected to understand the Software Components claim. Instead, it is for the Verifier to check this claim against the available endorsements and provide an answer in form of an "high level" attestation result, which may or may not include the original Software Components claim.

```
psa-software-component = {
    ? 1 => text,         ; measurement type
    2 => psa-hash-type, ; measurement value
    ? 4  => text,        ; version
    5 => psa-hash-type, ; signer id
    ? 6 => text,         ; measurement description
}

psa-software-components = (
    arm_psa_sw_components => [ + psa-software-component ]
)
```

3.4.1.1. Measurement Type

The Measurement Type attribute (key=1) is short string representing the role of this software component.

The following measurement types MAY be used:

* "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

3.4.1.2. Measurement Value

The Measurement Value attribute (key=2) 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.

This attribute MUST be present in a PSA software component.

3.4.1.3. Version

The Version attribute (key=4) is the issued software version in the form of a text string. The value of this attribute will correspond to the entry in the original signed manifest of the component.

3.4.1.4. Signer ID

The Signer ID attribute (key=5) is the hash of a signing authority public key for the software component. The value of this attribute 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 attribute MUST be present in a PSA software component to be compliant with [PSA-SM].

3.4.1.5. Measurement Description

The Measurement Description attribute (key=6) is the 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 attribute will normally be excluded, unless there was an exception to the default measurement described in the profile for a specific component.

3.4.2. No Software Measurements

In the event that the implementation does not contain any software measurements then the Software Components claim Section 3.4.1 can be omitted but instead the token MUST include this claim to indicate this is a deliberate state. The value SHOULD be 1. This claim is intended for devices that are not compliant with [PSA-SM].

```
psa-no-sw-measurements-type = 1

psa-no-sw-measurement = (
    arm_psa_no_sw_measurements => psa-no-sw-measurements-type
)
```

3.5. Verification Claims

3.5.1. Verification Service Indicator

The Verification Service Indicator claim is 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.


```
psa-verification-service-indicator-type = text

psa-verification-service-indicator = (
    ? arm_psa_origination => psa-verification-service-indicator-type
)
```

3.5.2. Profile Definition

The Profile Definition claim contains the name of a document that describes the "profile" of the report. The document name may include versioning. The value for this specification MUST be PSA_IOT_PROFILE_1.

```
psa-profile-type = "PSA_IOT_PROFILE_1"

psa-profile = (
    ? arm_psa_profile_id => psa-profile-type
)
```

4. Token Encoding and Signing

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. For asymmetric key algorithms, the signature structure MUST be COSE-Sign1. For symmetric key algorithms, the structure MUST be COSE-Mac0.

5. Collated CDDL

```
psa-token = {
```

```
        psa-nonce-claim,
        psa-instance-id,
        psa-verification-service-indicator,
        psa-profile,
        psa-implementation-id,
        psa-client-id,
        psa-lifecycle,
        psa-hardware-version,
        psa-boot-seed,
        ( psa-software-components // psa-no-sw-measurement ),
    }

    arm_psa_profile_id = -75000
    arm_psa_partition_id = -75001
    arm_psa_security_lifecycle = -75002
    arm_psa_implementation_id = -75003
```



```
    arm_psa_boot_seed = -75004
    arm_psa_hw_version = -75005
    arm_psa_sw_components = -75006
    arm_psa_no_sw_measurements = -75007
    arm_psa_nonce = -75008
    arm_psa_UEID = -75009
    arm_psa_origination = -75010

    psa-hash-type = bytes .size 32 / bytes .size 48 / bytes .size 64

    psa-boot-seed-type = bytes .size 32

    psa-boot-seed = (
        arm_psa_boot_seed => psa-boot-seed-type
    )

    psa-client-id-nspe-type = -2147483648...0
    psa-client-id-spe-type = 1..2147483647

    psa-client-id-type = psa-client-id-nspe-type / psa-client-id-spe-type

    psa-client-id = (
        arm_psa_partition_id => psa-client-id-type
    )
```

```
psa-hardware-version-type = text .regexp "[0-9]{13}"

psa-hardware-version = (
    ? arm_psa_hw_version => psa-hardware-version-type
)

psa-implementation-id-type = bytes .size 32

psa-implementation-id = (
    arm_psa_implementation_id => psa-implementation-id-type
)

psa-instance-id-type = bytes .size 33

psa-instance-id = (
    arm_psa_UEID => psa-instance-id-type
)

psa-no-sw-measurements-type = 1

psa-no-sw-measurement = (
    arm_psa_no_sw_measurements => psa-no-sw-measurements-type
)
```



```
psa-nonce-claim = (
    arm_psa_nonce => psa-hash-type
)

psa-profile-type = "PSA_IOT_PROFILE_1"

psa-profile = (
    ? arm_psa_profile_id => psa-profile-type
)

psa-lifecycle-unknown-type = 0x0000..0x00ff
psa-lifecycle-assembly-and-test-type = 0x1000..0x10ff
psa-lifecycle-psa-rot-provisioning-type = 0x2000..0x20ff
psa-lifecycle-secured-type = 0x3000..0x30ff
psa-lifecycle-non-psa-rot-debug-type = 0x4000..0x40ff
psa-lifecycle-recoverable-psa-rot-debug-type = 0x5000..0x50ff
psa-lifecycle-decommissioned-type = 0x6000..0x60ff
```

```
psa-lifecycle-type =
    psa-lifecycle-unknown-type /
    psa-lifecycle-assembly-and-test-type /
    psa-lifecycle-psa-rot-provisioning-type /
    psa-lifecycle-secured-type /
    psa-lifecycle-non-psa-rot-debug-type /
    psa-lifecycle-recoverable-psa-rot-debug-type /
    psa-lifecycle-decommissioned-type

psa-lifecycle = (
    arm_psa_security_lifecycle => psa-lifecycle-type
)

psa-software-component = {
    ? 1 => text,         ; measurement type
    2 => psa-hash-type, ; measurement value
    ? 4  => text,        ; version
    5 => psa-hash-type, ; signer id
    ? 6 => text,         ; measurement description
}

psa-software-components = (
    arm_psa_sw_components => [ + psa-software-component ]
)

psa-verification-service-indicator-type = text

psa-verification-service-indicator = (
    ? arm_psa_origination => psa-verification-service-indicator-type
)
```


6. 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 allows signatures based on use of public key cryptography as well as MAC authentication. The token MUST be signed following the structure of the COSE specification [RFC8152]. The COSE type MUST be COSE-Sign1 for public

key signatures or COSE-Mac0 for MAC authentication. Note however that use of MAC authentication is NOT RECOMMENDED due to the associated infrastructure costs for key management and protocol complexities. It may also restrict the ability to interoperate with third parties.

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. Implementations that have privacy requirements must take appropriate measures to ensure that the token is only used to provision anonymous/pseudonym keys.

7. IANA Considerations

IANA is requested to allocate the claims defined in Section 3 to the CBOR Web Token (CWT) Claims registry [IANA-CWT]. The change controller are the authors and the reference is this document.

8. References

8.1. Normative References

[EAN-13] GS1, "International Article Number - EAN/UPC barcodes", 2019, <https://www.gs1.org/standards/barcodes/ean-upc>.

[PSA-FF] Arm, "Platform Security Architecture Firmware Framework 1.0 (PSA-FF)", February 2019, <https://pages.arm.com/psa-resources-ff.html>.

[PSA-SM] Arm, "Platform Security Architecture Security Model 1.0 (PSA-SM)", February 2019, <https://pages.arm.com/psa-resources-sm.html>.

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

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

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

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

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

8.2. Informative References

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., and N. Smith, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-01, 4 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-rats-architecture-01.txt>.

[I-D.ietf-rats-eat]
Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-03, 20 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-rats-eat-03.txt>.

[IANA-CWT] IANA, "CBOR Web Token (CWT) Claims", 2020, <https://www.iana.org/assignments/cwt/cwt.xhtml>.

[PSA] Arm, "Platform Security Architecture Resources", 2019, <https://www.arm.com/why-arm/architecture/platform-security-architecture/psa-resources>.

[TF-M] Linaro, "Trusted Firmware", 2020, <https://www.trustedfirmware.org>.

Appendix A. Reference Implementation

A reference implementation is provided by the Trusted Firmware project [TF-M].

Appendix B. 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+Wi6OpE7BsJt9oqGoKgJSpMHCbuHtfPI1ggx0ZwvLfoWzgD77Q
olASS5z4/6dT3taitXkgMvby1VMIBAiFYINzw0PS81eJqVO42ytZg0oPRKrxfcwfeWG
ied81gRS51IAE=
```


Example of EAT token (base64-encoded):


```
0oRDoQEmoFkCIqk6AAEk+1ggAAECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8
6AAEk+lggAAECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh86AAEk/YSkAlggAA
ECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8EZTMuMS40BVggAAECAwQFBgcIC
QoLDA0ODxAREhMUFRYXGBkaGxwdHh8BYkJMpAJYIAABAgMEBQYHCAkKCwwNDg8QERIT
FBUWFxgZGhscHR4fBGMxLjEFWCAAAQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0
eHwFkUFJvVKQCWCAAAQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHwRjMS4wBV
ggAAECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8BZEFSb1SkAlggAAECAwQFB
gcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8EYzIuMgVYIAABAgMEBQYHCAkKCwwNDg8Q
ERITFBUWFxgZGhscHR4fAWNBcHA6AAEk+RkwADoAAST/WCAAAQIDBAUGBwgJCgsMDQ4
PEBESExQVFhcYGRobHB0eHzoAASUBbHBzYV92ZXJpZmllcjoAAST4IDoAASUAWCEBAA
ECAwQFBgcICQoLDA0ODxAREhMUFRYXGBkaGxwdHh86AAEk93FQU0FfSW9UX1BST0ZJT
EVfMVhAWIYFCO5+jMSOuoctu11pSlQrEyKtDVECPBlw30KfBlAcaDqVEIoMztCm6A4J
ZvIr1j0cAFaXShG6My14d4f7Tw==
```


Same token using extended CBOR diagnostic format:

```
   18(
     [
     / protected / h'a10126' / {
         \ alg \ 1: -7 \ ECDSA 256 \
       } / ,
     / unprotected / {},
     / payload / h'a93a000124fb582000010203040506070809
     0a0b0c0d0e0f10111213141516171819
     1a1b1c1d1e1f3a000124fa582000010203040506070809
     0a0b0c0d0e0f101112131415161718
     191a1b1c1d1e1f3a000124fd84a402582000010203040
     5060708090a0b0c0d0e0f101112131
     4151617181
     91a1b1c1d1e1f0465332e312e340558200001020304050607080
     90a0b0c0d0e0f101112131415161
     718191a1b1
     c1d1e1f0162424ca40258200001020304050607080
     90a0b0c0d0e0f10111213141516171
     8191a1b1c1
     d1e1f0463312e3105582000010203040506070
     8090a0b0c0d0e0f101112131415161
     718191a1b1
     c1d1e1f016450526f54a4025820000102030405060708
     090a0b0c0d0e0f101112131415161
     718191a1b1
     c1d1e1f0463312e3005582000010203040506070
     8090a0b0c0d0e0f101112131415161
     718191a1b1
     c1d1e1f016441526f54a4025820000102030405060708
     090a0b0c0d0e0f10111213141516
     1718191a1b
     1c1d1e1f0463322e3205582000010203040506070
     8090a0b0c0d0e0f101112131415161
     718191a1b1
     c1d1e1f01634170703a000124f91930003a000124ff5820000
     102030405060708090a0b0c0d0e0f
     101112131415161718191a1b1c1d1e1f3a0001
     25016c7073615f76657269666965723a000124f8203a000125005821010001020304
     05060708090a0b0c0d0e0f10111213141516171819
     1a1b1c1d1e1f3a000124f771
     5053415f496f545f50524f46494c455f
     31' / {
        / arm_psa_boot_seed / -75004: h'000102030405060708090a0b0c0d0e0f
        101112131415161718191a1b1c1d1e1f',
        / arm_psa_implementation_id / -75003: h'000102030405060708090a0b
        0c0d0e0f101112131415161718191a1b1c1d1e1f',
        / arm_psa_sw_components / -75006: [
             {
               / measurement / 2: h'000102030405060708090a0b0c0d0e0f1011
               12131415161718191a1b1c1d1e1f',
               / version / 4: "3.1.4",
               / signerID / 5: h'000102030405060708090a0b0c0d0e0f1011121
               31415161718191a1b1c1d1e1f',
               / type / 1: "BL"
             },
             {
               / measurement / 2: h'000102030405060708090a0b0c0d0e0f1011
               12131415161718191a1b1c1d1e1f',
```

```
              / version / 4: "1.1",
              / signerID / 5: h'000102030405060708090a0b0c0d0e0f1011121
              314151617181 91a1b1c1d1e1f',
              / type / 1: "PRoT"
            },
            {
              / measurement / 2: h'000102030405060708090a0b0c0d0e0f1011
              12131415161718191a1b1c1d1e1f',
```



```
              / version / 4: "1.0",
              / signerID / 5: h'000102030405060708090a0b0c0d0e0f1011121
              314151617181 91a1b1c1d1e1f',
              / type / 1: "ARoT"
            },
            {
              / measurement / 2: h'000102030405060708090a0b0c0d0e0f1011
              12131415161718191a1b1c1d1e1f',
              / version / 4: "2.2",
              / signerID / 5: h'000102030405060708090a0b0c0d0e0f1011121
              314151617181 91a1b1c1d1e1f',
              / type / 1: "App"
            }
          ],
        / arm_psa_security_lifecycle / -75002: 12288 / SECURED /,
        / arm_psa_nonce / -75008: h'000102030405060708090a0b0c0d0e0f101
        112131415161718191a1b1c1d1e1f',
        / arm_psa_origination / -75010: "psa_verifier",
        / arm_psa_partition_id / -75001: -1,
        / arm_psa_UEID / -75009: h'0100010203040506070 8090a0b0c0d0e0f10
        111213141516171 8191a1b1c1d1e1f',
        / arm_psa_profile_id / -75000: "PSA_IoT_PROFILE_1"
      }),
      } / ,
    / signature / h'58860508ee7e8cc48eba872dbb5d694a542b1322ad0d51023c1
    970df429f06501c683a95108a0cced0a6e80e0966f22bd63d1c0056974a11ba332d
    787787fb4f'
    ]
  )
```

Contributors

We would like to thank the following colleagues for their

contributions:

* Laurence Lundblade
  Security Theory LLC
  lgl@securitytheory.com

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

* Sergei Trofimov
  Arm Limited
  Sergei.Trofimov@arm.com

Acknowledgments

Thanks to Carsten Bormann for help with the CDDL and Nicholas Wood for ideas and comments.

Authors' Addresses

Hannes Tschofenig
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