Workgroup: RATS
Internet-Draft:
draft-tschofenig-rats-aiss-token-00
Published: 22 April 2022
Intended Status: Informational
Expires: 24 October 2022
Authors: H. Tschofenig
Arm Limited
A. Kankaanpää
Synopsys
N. Bowler
Synopsys
T. Khandelwal
Arm Limited

Automatic Integration of Secure Silicon (AISS) Attestation Token

Abstract

This specification defines a profile of the Entity Attestation Token (EAT) for use in special System-on-Chip (SoC) designs that are generated automatically utilizing a methodology currently developed in a DARPA funded project.

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 24 October 2022.

Table of Contents

1. Introduction

The DARPA-funded project Automated Implementation of Secure Silicon (AISS) is aimed at making scalable on-chip security pervasive. The objective is to develop ways to automate the process of adding security into integrated circuits.

If successful, AISS will allow security to be inexpensively incorporated into chip designs with minimal effort and expertise, ultimately making scalable on-chip security ubiquitous. The project seeks to create a novel, automated chip design flow that will allow the security mechanisms to scale consistently with the goals of the design.

As a minimal component, the generated chip designs must offer attestation capabilities.

This specification describes the minimal claim set offered by an attestation token conforming to the Entity Attestation Token (EAT) specification. This attestation token is, on request, provided to a Verifier.

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.

The following term is used in this document:

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.

3. Claims

This section describes the claims to be used in an AISS 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:

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

3.1. Nonce

The Nonce claim is used to carry the challenge provided by the caller to demonstrate freshness of the generated token.

The EAT [I-D.ietf-rats-eat] nonce (claim key 10) is used. The following constraints apply to the nonce-type:

- The length MUST be either 32, 48, or 64 bytes.
- Only a single nonce value is conveyed. Per [I-D.ietf-rats-eat] the array notation is not used for encoding the nonce value.

This claim MUST be present in an AISS attestation token.

```
aiss-nonce = (
    nonce-label => aiss-hash-type
)
```

3.2. Instance ID

The Instance ID claim represents the unique identifier of the attestation key.

The EAT ueid (claim key 256) of type RAND is used. The following constraints apply to the ueid-type:

- The length MUST be 17 bytes.
- The first byte MUST be 0x01 (RAND) followed by the 16-bytes random value, which may be created by hashing the key identifier or may be the key identifier itself.

This claim MUST be present in an AISS attestation token.

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

aiss-instance-id = (
    ueid-label => aiss-instance-id-type
)
```

3.3. Implementation ID

The Implementation ID claim uniquely identifies the implementation of the immutable RoT. A verification service uses this claim to locate the details of the RoT implementation from a manufacturer. Such details are used by a verification service to determine the security properties or certification status of the RoT implementation.

The value and format of the ID is decided by the manufacturer or a particular certification scheme. For example, the ID could take the form of a product serial number, database ID, or other appropriate identifier.

This claim MUST be present in an AISS attestation token.

Note that this identifies the RoT implementation, not a particular instance. The Instance ID claim, see Section 3.2, uniquely identifies an instance.

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

aiss-implementation-id = (
    aiss-implementation-id-label => aiss-implementation-id-type
)
```

3.4. Security Lifecycle

The Security Lifecycle claim represents the current lifecycle state of the RoT. The state is represented by an unsigned integer.

The lifecycle states are illustrated in Figure 1. When the device is deployed, a Verifier can only trust reports when the lifecycle state is in "Secured" and "Non-RoT Debug" states. The states "Testing" and "Provisioning" are utilized during manufacturing. A device is in "Decommisioned" state when it is retired.

This claim MUST be present in an AISS attestation token.

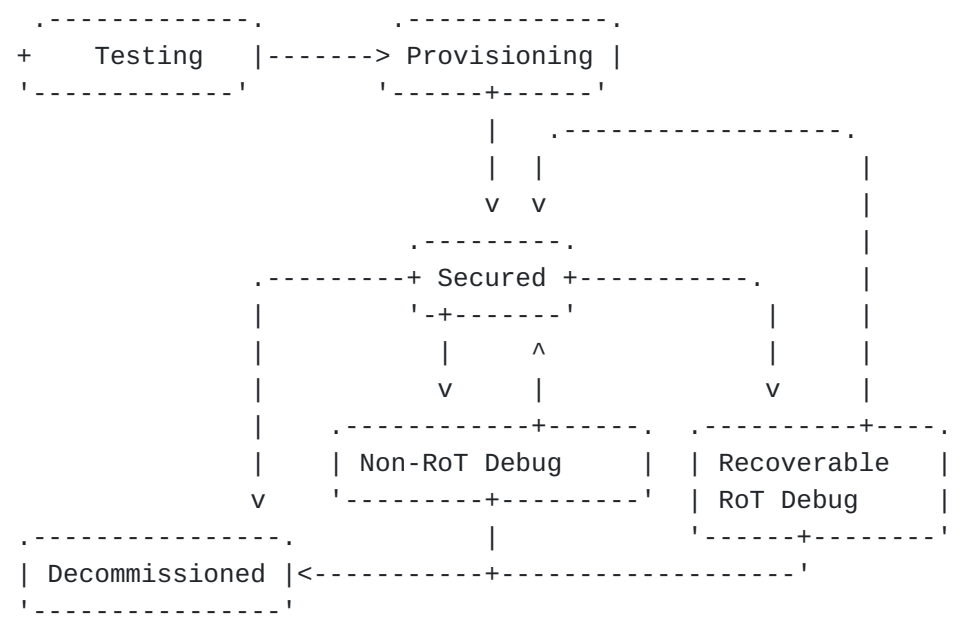

Figure 1: Lifecycle States.

```
aiss-lifecycle-unknown-type = 0
aiss-lifecycle-testing-type = 1
aiss-lifecycle-provisioning-type = 2
aiss-lifecycle-secured-type = 3
aiss-lifecycle-non-rot-debug-type = 4
aiss-lifecycle-recoverable-rot-debug-type = 5
aiss-lifecycle-decommissioned-type = 6

aiss-lifecycle-type =
    aiss-lifecycle-unknown-type /
    aiss-lifecycle-testing-type /
    aiss-lifecycle-provisioning-type /
    aiss-lifecycle-secured-type /
    aiss-lifecycle-non-rot-debug-type /
    aiss-lifecycle-recoverable-rot-debug-type /
    aiss-lifecycle-decommissioned-type

aiss-lifecycle = (
    aiss-lifecycle-label => aiss-lifecycle-type
)
```

3.5. Boot Odometer

The Boot Odometer claim contains a value that represents the number of times the entity or submod has been booted.

The EAT boot-seed-label (claim key TBD) of type unsigned integer is used.

This claim MUST be present in an AISS attestation token.

```
aiss-boot-odometer = (
    aiss-boot-odometer-label => uint
)
```

3.6. Watermark

Watermarking, the process of marking an asset with a known structure, is used to detect intellectual property (IP) theft and overuse. Watermarking in hardware IPs is the mechanism of embedding a unique "code" into IP without altering the original functionality of the design. The ownership of the IP can be later verified when the watermark is extracted.

The Watermark claim contains a code extracted from the watermarking hardware identified by an identifier. This identifier is formated as a type 4 UUID [RFC4122].

This claim MUST be present in an AISS attestation token when the attestation token request asked for a watermark to be present.

```
watermark-type = [
    id:  bstr .size 16,
    watermark:  bytes
]

aiss-watermark = ( watermark-label => watermark-type )
```

3.7. Profile Definition

The Profile Definition claim encodes the unique identifier that corresponds to the EAT profile described by this document. This allows a receiver to assign the intended semantics to the rest of the claims found in the token.

The EAT profile (claim key 265) is used. The following constraints apply to its type:

- The URI encoding MUST be used.

- The value MUST be http://aiss/1.0.0.

This claim MUST be present in an AISS attestation token.

```
aiss-profile-type = "http://aiss/1.0.0"

aiss-profile = (
    profile-label => aiss-profile-type
)
```

4. Token Encoding and Signing

The AISS attestation token is encoded in CBOR [RFC8949] format. Only definite-length string, arrays, and maps are allowed.

Cryptographic protection is accomplished by COSE. The signature structure MUST be COSE_Sign1. Only the use of asymmetric key algorithms is envisioned.

The CWT CBOR tag (61) is not used. An application that needs to exchange PSA attestation tokens can wrap the serialised COSE_Sign1 in a dedicated media type, as for example defined in defined in Section 8.2 or the CoAP Content-Format defined in Section 8.3.

5. Freshness Model

The AISS attestation token supports the freshness models for attestation Evidence based on nonces (Section 10.2 and 10.3 of [I-D.ietf-rats-architecture]) using the nonce claim to convey the nonce supplied by the Verifier. No further assumption on the specific remote attestation protocol is made.

6. Collated CDDL

```
aiss-token = {
    aiss-nonce,
    aiss-instance-id,
    aiss-profile,
    aiss-implementation-id,
    aiss-lifecycle,
    aiss-boot-odometer,
    aiss-watermark,
}

aiss-lifecycle-label = 2500
aiss-implementation-id-label = 2501
aiss-watermark-label = 2502
aiss-boot-odometer-label = 2503

; from EAT
nonce-label = 10
ueid-label = 256
profile-label = 265
aiss-hash-type = bytes .size 32 / bytes .size 48 / bytes .size 64
aiss-nonce = (
    nonce-label => aiss-hash-type
)
aiss-instance-id-type = bytes .size 33

aiss-instance-id = (
    ueid-label => aiss-instance-id-type
)
aiss-implementation-id-type = bytes .size 32

aiss-implementation-id = (
    aiss-implementation-id-label => aiss-implementation-id-type
)
aiss-lifecycle-unknown-type = 0
aiss-lifecycle-testing-type = 1
aiss-lifecycle-provisioning-type = 2
aiss-lifecycle-secured-type = 3
aiss-lifecycle-non-rot-debug-type = 4
aiss-lifecycle-recoverable-rot-debug-type = 5
aiss-lifecycle-decommissioned-type = 6

aiss-lifecycle-type =
    aiss-lifecycle-unknown-type /
    aiss-lifecycle-testing-type /
    aiss-lifecycle-provisioning-type /
    aiss-lifecycle-secured-type /
    aiss-lifecycle-non-rot-debug-type /
    aiss-lifecycle-recoverable-rot-debug-type /
    aiss-lifecycle-decommissioned-type
```

```
aiss-lifecycle = (
    aiss-lifecycle-label => aiss-lifecycle-type
)
aiss-boot-odometer = (
    aiss-boot-odometer-label => uint
)

watermark-type = [
    id:  bstr .size 16,
    watermark:  bytes
]

aiss-watermark = ( watermark-label => watermark-type )

aiss-profile-type = "http://aiss/1.0.0"

aiss-profile = (
    profile-label => aiss-profile-type
)
```

7. Verification

To verify the token, the primary need is to check correct encoding and signing as detailed in Section 4. In particular, the Instance ID claim is used (together with the kid in the COSE header, if present) to assist in locating the public key used to verify the signature covering the token. The key used for verification is supplied to the Verifier by an authorized Endorser along with the corresponding Attester's Instance ID.

In addition, the Verifier will typically operate a policy where values of some of the claims in this profile can be compared to reference values, registered with the Verifier for a given deployment, in order to confirm that the device is endorsed by the manufacturer supply chain. The policy may require that the relevant claims must have a match to a registered reference value.

The protocol used to convey Endorsements and Reference Values to the Verifier is not in scope for this document.

8. IANA Considerations

8.1. Claim Registration

This specification requests IANA to register the following claims in the "CBOR Web Token (CWT) Claims" registry [IANA-CWT].

8.1.1. Security Lifecycle Claim

*Claim Name: aiss-security-lifecycle

*Claim Description: AISS Security Lifecycle

*JWT Claim Name: N/A

*Claim Key: TBD (requested value: 2500)

*Claim Value Type(s): unsigned integer

*Change Controller: [[Authors of this RFC]]

*Specification Document(s): Section 3.4 of [[this RFC]]

8.1.2. Implementation ID Claim

*Claim Name: aiss-implementation-id

*Claim Description: AISS Implementation ID

*JWT Claim Name: N/A

*Claim Key: TBD (requested value: 2501)

*Claim Value Type(s): byte string

*Change Controller: [[Authors of this RFC]]

*Specification Document(s): Section 3.3 of [[this RFC]]

8.1.3. Watermark Claim

*Claim Name: aiss-watermark

*Claim Description: AISS Watermark

*JWT Claim Name: N/A

*Claim Key: TBD (requested value: 2502)

*Claim Value Type(s): byte string

*Change Controller: [[Authors of this RFC]]

*Specification Document(s): Section 3.6 of [[this RFC]]

8.2. Media Type Registration

IANA is requested to register the "application/aiss-attestation-token" media type [RFC2046] in the "Media Types" registry [IANA-MediaTypes] in the manner described in RFC 6838 [RFC6838], which can be used to indicate that the content is an AISS Attestation Token.

*Type name: application

*Subtype name: aiss-attestation-token

*Required parameters: n/a

*Optional parameters: n/a

*Encoding considerations: binary

*Security considerations: See the Security Considerations section of [[this RFC]]

*Interoperability considerations: n/a

*Published specification: [[this RFC]]

*Applications that use this media type: Attesters and Relying Parties sending AISS attestation tokens over HTTP(S), CoAP(S) and other transports.

*Fragment identifier considerations: n/a

*Additional information:

  -Magic number(s): n/a

  -File extension(s): n/a

  -Macintosh file type code(s): n/a

*Person & email address to contact for further information: Hannes Tschofenig, Hannes.Tschofenig@arm.com

*Intended usage: COMMON

*Restrictions on usage: none

*Author: Hannes Tschofenig, Hannes.Tschofenig@arm.com

*Change controller: IESG

*Provisional registration? No

8.3. CoAP Content-Formats Registration

IANA is requested to register the CoAP Content-Format ID for the "application/aiss-attestation-token" media type in the "CoAP Content-Formats" registry [IANA-CoAP-Content-Formats].

8.3.1. Registry Contents

*Media Type: application/aiss-attestation-token

*Encoding: -

*Id: [[To-be-assigned by IANA]]

*Reference: [[this RFC]]

9. References

9.1. Normative References

[I-D.ietf-rats-eat] Lundblade, L., Mandyam, G., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-12, 24 February 2022, <https://www.ietf.org/archive/id/draft-ietf-rats-eat-12.txt>.

[IANA-CWT] IANA, "CBOR Web Token (CWT) Claims", 2022, <https://www.iana.org/assignments/cwt/cwt.xhtml#claims-registry>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[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>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC

6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[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>.

[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>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

9.2. Informative References

[I-D.ietf-rats-architecture] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-15, 8 February 2022, <https://www.ietf.org/archive/id/draft-ietf-rats-architecture-15.txt>.

[IANA-CoAP-Content-Formats] IANA, "CoAP Content-Formats", 2022, <https://www.iana.org/assignments/core-parameters>.

[IANA-MediaTypes] IANA, "Media Types", 2022, <http://www.iana.org/assignments/media-types>.

Appendix A. Example

The following example shows an AISS attestation token for an hypothetical system. The attesting device is in a lifecycle state Section 3.4 of SECURED.

The claims in this example are:

```
{
  / instance-id /         255: h'FF0039A1',
  / nonce /                10: h'AABBCCDD',
  / lifecycle /          2500: 2,
  / implementation-id /  2501: h'CCDDEE',
  / watermark /          2502: h'010203',
  / boot-odometer /      2503: 5,
  / profile-id /          256: "aiss/1.0.0",
}
```

The resulting COSE object is:

```
18(
  [
    / protected /   h'A10126',
    / unprotected / {},
    / payload /     h'A718FF44FF0039A10A44AABBCCDD1909C4021901006
                      A616973732F312E302E301909C543CCDDEE1909C643
                      0102031909C705',
    / signature /   h'9744085E05D875E5EAAEC1598D1DD9E14097CCE4E9A
                      484344D08C9D41244713C700CD4F1CD7E86C0C6397A
                      ABECE40E166EBA5AA92DB11170F69B2DD8E681708E'
  ]
)
```

Acknowledgments

We would like to thank Rob Aitken, Mike Borza, Liam Dillon, Dale Donchin, John Goodenough, and Oleg Raikhman for their feedback.

Work on this document has in part been supported by the DARPA AISS project (grant agreement HR0011-20-9-0043).

Authors' Addresses

Hannes Tschofenig
Arm Limited

Email: Hannes.Tschofenig@arm.com

Arto Kankaanpää
Synopsys

Email: arto.kankaanpaa@synopsys.com

Nick Bowler
Synopsys

Email: nick.bowler@synopsys.com

Tushar Khandelwal
Arm Limited

Email: Tushar.Khandelwal@arm.com