Workgroup: Remote ATtestation ProcedureS
Internet-Draft:
draft-frost-rats-eat-collection-02
Published: 8 December 2022
Intended Status: Standards Track
Expires: 11 June 2023
Authors: S. Frost
Arm

Entity Attestation Token (EAT) Collection Type

Abstract

The default top-level definitions for an EAT [I-D.ietf-rats-eat] assume a hierarchy involving a leading signer within the Attester. Some token use cases do not match that model. This specification defines an extension to EAT allowing the top-level of the token to consist of a collection of otherwise defined tokens.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-frost-rats-eat-collection/.

Discussion of this document takes place on the Remote ATtestation ProcedureS Working Group mailing list (mailto:rats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rats/. Subscribe at https://www.ietf.org/mailman/listinfo/rats/.

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 11 June 2023.

Table of Contents

1. Introduction

An Attestation Token conforming to EAT [I-D.ietf-rats-eat] has a default top level definition for a token to be constructed principally as a claim set within a CBOR Web Token (CWT) [RFC8392] with the associated COSE envelope [STD96] providing at least integrity and authentication. An equivalent JSON encoding for a JWT [RFC7519] in a JWS envelope [RFC7515] is supported as an alternative at the top-level definition. The top level token can be augmented with related claims in a Detached EAT Bundle.

For the use case of transmitting a claim set through a secure channel, the top-level definition can be extended to use an Unprotected CWT Claim Set (UCCS) [I-D.ietf-rats-uccs].

This document outlines an additional top-level extension for which neither of the above top level definitions match exactly: the attestation token consists of a collection of objects, each with

their own integrity and some internally defined relationship through which the integrity of the whole collection can be determined. i.e. there is no top-level signer for the set. The objects may all share the same logical hierarchy in a device or have a hierarchy which is internally defined within the object set.

2. Terminology and Requirements Language

Readers are also expected to be familiar with the terminology from [I-D.ietf-rats-eat] and [I-D.ietf-rats-architecture].

In this document, the structure of data is specified in CDDL [RFC8610] [RFC9165].

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.

3. Design Considerations / Use Cases

Take a device with an attestation system consisting of a platform claim set and a workload claim set, each controlled by different components and with an underlying hardware Root of Trust. The two claim sets are delivered together to make up the overall attestation token. Depending upon the implementation and deployment use case, the signing system can either be entirely centric to the platform RoT or can have separate signers for the two claim sets. In either case, a cryptographic binding is established between the two parts of the token.

A specific manifestation of such a device is one incorporating the Arm Confidential Compute Architecture (CCA) attestation token [Arm-CCA]. In trying to prepare the attestation token using EAT, there were no issues constructing the claim sets or incorporating them into individual CWTs where appropriate. However, in trying to design an 'envelope structure' to convey the two parts as a single report it was found that maintaining EAT compatibility would require very different shaped compound tokens for different models, for example one based on a submod arrangement and another based on a Detached EAT Bundle, though with different 'leading' objects. This would create extra code and explanation in areas where keeping things simple is desirable. There was an alternative approach considered, which stays close to existing thinking on EAT, which would be to create the wrapper from the UCCS EAT extension containing only submods for the respective components. This however stretches the current use case for UCCS beyond its existing description. The RATS WG approach of separating UCCS from the core

EAT specification to be an extension also encourages proposing this further extension.

To support the CCA use case, it is also relevant to consider current attestation technologies which are based on certificate chains (e.g. SPDM, DICE, several key attestation systems). Here also are multiple objects with their own integrity and an internally defined relationship. If attempting to move such a technology to the EAT world, the same challenges apply.

An additional use case beyond the production of tokens from an Attester occurs when using EAT to convey Attestation Results from a Verifier. Attestation results may be separated into different sections depending upon what aspects of Appraisal Policy are applied by the Verifier. For example, the set of validated evidence claims may form one section, while claims reflecting semantic conclusions drawn by an Appraisal Policy could form another section. Given the role of different authorities in concluding result sections, each could have a different signer rather than all results being under a single signature from the Verifier. In this case, a collection can be used to collate all result sections into a single response message. Using a collection simplifies operations if individual sections from the collated result sections need to be later dispersed to different Relying Parties.

A further Attestation Result use case can be seen in the "Below Zero Trust" system described in [I-D.ietf-rats-ar4si] where the AR-augmented Evidence credential has compound form.

4. Token Collection

The proposed extension for the top-level definition is to add a 'Token Collection' type. The contents of the type are a map of CWTs (JWTs). The Detached EAT Bundle top-level entry for EAT is included for completeness, and the UCCS extension can also be embraced, though the use cases for these have not been explored. The identification of collection members and the intra collection integrity mechanism is considered usage specific. A verifier will be expected to extract each of the members of the collection and check their validity both individually and as a set. In addition to entries which have their own integrity, it is also supported to include an unsigned Claims Set, provided that the integrity for that Claims Set is provided within another entry that uses one of the signed forms.

A map was chosen rather than an unbounded array to give the opportunity to add identifying map tags to each entry. The interpretation of the tags will be usage specific, but may correspond to registered identities of specific token types. To

assist a verifier correlate the expected contents a profile entry can be added as the 'profile-label' identity in the map.

See Appendix A for a CDDL description of the proposed extension.

While most of the use cases for collections are for scenarios where there will be at least two entries in a collection, the CDDL allows for >= 1 entries in a collection to allow for the scenario where only one entry is currently available even though the normal set is larger.

4.1. Binder Definition

This specification includes a proposal for a Collection Binder claim (see Figure 1). This claim would be included within any collection entry as a definiton of the integrity mechanism that binds that collection entry to another collection entry. A verifier can use the information within this claim to drive inter collection entry integrity checking. This claim would not be mandatory within a collection entry as a verifier may implement the integrity checking based upon information in the profile alone.

```
; The Collection Binder is a formal declaration of the inter entry
; binding mechanism. It would be included within the body of one or
; more of the collection entries.
Collection-Binder = [
    binder-function,
    [*binder-source-label],
    destination-collection-entry,
    destination-claim
]

; binder-function is the name/id of a hash algorithm
binder-function = JC<text,int>

; By definition, the binder-function is applied to a concatenation
; of the ordered list of source claims.
; If the array is empty, the function is applied to the whole
; contents of the token.
binder-source-label = Claim-Label

destination-collection-entry = collection-entry-label
destination-claim = Claim-Label

Claim-Label = JC<"text", int>
collection-entry-label = JC<text, int>
JC<J,C> = J .feature "json" / C .feature "cbor"
```

Figure 1: EAT collection binder

The attributes within the binder claim are:

- binder-function: the identity of the binding cryptographic function, usually a hash function, applied to the values identified by the binder-source-label array.

- binder-source-label: an array defining the set of claims providing the binding information within the collection entry. It is assumed that the values corresponding to this (ordered) list will be concatenated and have the binder-function applied to produce a binder value. If the array is empty, the entire source collection entry is used as input to the binder-function. This latter condition would normally be applied to a collection entry consisting of a Claim Set.

- destination-collection-entry: this defines the collection entry that will hold (receive) the binding for this (source) collection entry.

- destination-claim: this defines the claim label within destination-collection-entry which will store the binder value.

A verifier can check the binding between two collection entries by computing the binder value for one entry and comparing the result stored within the value of the destination claim (in the destination collection entry).

5. Security Considerations

A verifier for an attestation token must apply a verification process for the full set of entries contained within the Token Collection. This process will be custom to the relevant profile for the Token Collection and take into account any individual verification per entry and/or verification for the objects considered collectively, including the intra token integrity scheme. As there is no overall signature for the Collection, protection against malicious modification must be contained within the entries. It is expected that there exists a cryptographic binding between entries, this can for example be one to many or one to one in a (chain) series. The implementation of creating these bindings may involve passing data across ABIs. This provides an attack vector on the integrity of the collection which must be considered within any threat model. With respect to binder claims, these require integrity protection. This protection can either be provided by the signature on the token entry which contains the binder or, in the case where the entry does not have a signature, by including the binder claim with any other claims when preparing input into the cryptographic binding function. Depending upon the use case and associated threat model, the freshness of entries may need extra consideration.

6. IANA Considerations

In the registry [IANA.cbor-tags], IANA is requested to allocate the tag in Table 1 from the FCFS space, with the present document as the specification reference.

Tag	Data Item	Semantics
TBD399	map	EAT Collection RFCthis

Table 1: EAT Collection

7. References

7.1. Normative References

[I-D.ietf-rats-eat] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-18, 4 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-18>.

[IANA.cbor-tags] IANA, "Concise Binary Object Representation (CBOR) Tags", <https://www.iana.org/assignments/cbor-tags>.

[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/rfc/rfc2119>.

[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/rfc/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/rfc/rfc8610>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/rfc/rfc9165>.

7.2. Informative References

[Arm-CCA] Arm Ltd, "Confidential Compute Architecture", 2022, <https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture>.

[I-D.ietf-rats-ar4si] Voit, E., Birkholz, H., Hardjono, T., Fossati, T., and V. Scarlata, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-ietf-rats-ar4si-03, 6 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-03>.

[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-22, 28 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-architecture-22>.

[I-D.ietf-rats-uccs] Birkholz, H., O'Donoghue, J., Cam-Winget, N., and C. Bormann, "A CBOR Tag for Unprotected CWT Claims Sets", Work in Progress, Internet-Draft, draft-ietf-rats-uccs-03, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-uccs-03>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[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/rfc/rfc8392>.

[STD96] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

Appendix A. CDDL

```
$$EAT-CBOR-Tagged-Token /= Tagged-Collection
$$EAT-CBOR-Untagged-Token /= TL-Collection

Tagged-Collection =  #6.TBD399(TL-Collection)

TL-Collection = {
    ? eat-collection-identifier,
    + all-collection-types
}

eat-collection-identifier = (
    profile-label => general-uri / general-oid
)

all-collection-types = (
    cwt-collection-entries //
    jwt-collection-entries //
    claim-set-collection-entries //
    detatched-eat-bundle-collection-entries
)

cwt-collection-entries = (
    collection-entry-label => CWT-Messages
)

jwt-collection-entries = (
    collection-entry-label => JWT-Messages
)

claim-set-collection-entries = (
    collection-entry-label => JC<json-wrapped-claims-set,
                                 cbor-wrapped-claims-set>
)

detatched-eat-bundle-collection-entries = (
    collection-entry-label => BUNDLE-Messages
)

collection-entry-label = JC<text, int>

; The Collection Binder is a formal declaration of the inter entry
; binding mechanism. It would be included within the body of one or
; more of the collection entries.
Tagged-Collection-Binder =  #6.TBD99(Collection-Binder)
Collection-Binder = [
    binder-function,
    [*binder-source-label],
    destination-collection-entry,
    destination-claim
]
```

```
; binder function is normally the name/id of a hash algorithm
binder-function = JC<text,int>

; by definition, the binder-function is applied to a concatenation
; of the ordered list of source claims
; If the array is empty, the function is applied to the whole
; contents of the token
binder-source-label = Claim-Label

destination-collection-entry = collection-entry-label

destination-claim = Claim-Label
```

Acknowledgments

Thomas Fossati proposed the inclusion of the Binder definiton and collaborated on the CDDL. Yogesh Deshpande provided insightful comments and review for this proposal.

Contributors

Thomas Fossati
Arm Limited

Email: thomas.fossati@arm.com

Author's Address

Simon Frost
Arm

Email: Simon.Frost@arm.com