Workgroup: Remote ATtestation ProcedureS
Internet-Draft: draft-ftbs-rats-msg-wrap-05
Published: 7 November 2023
Intended Status: Standards Track
Expires: 10 May 2024
Authors: H. Birkholz (Fraunhofer SIT)
N. Smith (Intel)
T. Fossati (Linaro)
H. Tschofenig

RATS Conceptual Messages Wrapper

Abstract

This document defines two encapsulation formats for RATS conceptual messages (i.e., evidence, attestation results, endorsements and reference values.)

The first format uses a CBOR or JSON array with two mandatory members, one for the type, another for the value, and a third optional member complementing the type field that says which kind of conceptual message(s) are carried in the value. The other format wraps the value in a CBOR byte string and prepends a CBOR tag to convey the type information.

This document also defines a corresponding CBOR tag, as well as JSON Web Tokens (JWT) and CBOR Web Tokens (CWT) claims. These allow embedding the wrapped conceptual messages into CBOR-based protocols and web APIs, respectively.

Discussion Venues

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

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

Source for this draft and an issue tracker can be found at https://github.com/thomas-fossati/draft-ftbs-rats-msg-wrap.

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 10 May 2024.

Table of Contents

1. Introduction

The RATS architecture defines a handful of conceptual messages (see Section 8 of [RFC9334]), such as evidence and attestation results. Each conceptual message can have multiple claims encoding and serialization formats (Section 9 of [RFC9334]). Throughout their lifetime, RATS conceptual messages are typically transported over different protocols. For example, EAT [I-D.ietf-rats-eat] evidence in a "background check" topological arrangement first flows from Attester to Relying Party, and then from Relying Party to Verifier, over separate protocol legs. Attestation Results for Secure Interactions (AR4SI) [I-D.ietf-rats-ar4si] payloads in "passport" mode would go first from Verifier to Attester and then, at a later point in time and over a different channel, from Attester to Relying Party.

It is desirable to reuse any typing information associated with the messages across such protocol boundaries in order to minimize the cost associated with type registrations and maximize interoperability.

This document defines two encapsulation formats for RATS conceptual messages that aim to achieve the goals stated above.

These encapsulation formats are designed to be:

* Self-describing - which removes the dependency on the framing provided by the embedding protocol (or the storage system) to convey exact typing information.

* Based on media types [RFC6838] - which allows amortising their registration cost across many different usage scenarios.

A protocol designer could use these formats, for example, to convey evidence, endorsements or reference values in certificates and CRLs extensions ([DICE-arch]), to embed attestation results or evidence as first class authentication credentials in TLS handshake messages [I-D.fossati-tls-attestation], to transport attestation-related payloads in RESTful APIs, or for stable storage of attestation results in form of file system objects.

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.

In this document, CDDL [RFC8610] [RFC9165] is used to describe the data formats.

The reader is assumed to be familiar with the vocabulary and concepts defined in [RFC9334].

This document reuses the terms defined in Section 2 of [RFC9193] (e.g., "Content-Type").

3. Conceptual Message Wrapper Encodings

Two types of RATS Conceptual Message Wrapper (CMW) are specified in this document:

1. A CMW using a CBOR or JSON array (Section 3.1);

2. A CMW based on CBOR tags (Section 3.2).

3.1. CMW Array

The CMW array format is defined in Figure 1. (To improve clarity, the Content-Type ABNF is defined separately in Appendix A.)

The CDDL generic JC<> is used where there is a variance between CBOR and JSON. The first argument is the CDDL for JSON and the second is the CDDL for CBOR.

```
cmw-array = [
  type: coap-content-format / media-type
  value: JC<base64-string, bytes>
  ? ind: uint .bits cm-type
]

coap-content-format = uint .size 2
media-type = text .abnf ("Content-Type" .cat Content-Type-ABNF)

base64-string = text .regexp "[A-Za-z0-9_-]+"

cm-type = &(
  reference-values: 0
  endorsements: 1
  evidence: 2
  attestation-results: 3
)
```

Figure 1: CDDL definition of the Array format

It is composed of three members:

type:
Either a text string representing a Content-Type (e.g., an EAT media type [I-D.ietf-rats-eat-media-type]) or an unsigned integer corresponding to a CoAP Content-Format number (Section 12.3 of [RFC7252]).

value:
The RATS conceptual message serialized according to the value defined in the type member.

ind:
An optional bitmap that indicates which conceptual message types are carried in the value field. Any combination (i.e., any value between 1 and 15, included) is allowed. This is useful only if the type is potentially ambiguous and there is no further context available to the CMW consumer to decide. For example, this might be the case if the base media type is not profiled (e.g., application/eat+cwt), if the value field contains multiple conceptual messages with different types (e.g., both reference values and endorsements within the same application/signed-corim+cbor), or if the same profile identifier is shared by different conceptual messages. Future specifications may add new values to the ind field; see Section 7.4.

A CMW array can be encoded as CBOR [STD94] or JSON [RFC8259].

When using JSON, the value field is encoded as Base64 using the URL and filename safe alphabet (Section 5 of [RFC4648]) without padding.

When using CBOR, the value field is encoded as a CBOR byte string.

3.2. CMW CBOR Tags

CBOR Tags used as CMW may be derived from CoAP Content-Format numbers. If a CoAP content format exists for a RATS conceptual message, the TN() transform defined in Appendix B of [RFC9277] can be used to derive a corresponding CBOR tag in range [1668546817, 1668612095].

The RATS conceptual message is first serialized according to the Content-Format number associated with the CBOR tag and then encoded as a CBOR byte string, to which the tag is prepended.

The CMW CBOR Tag is defined in Figure 2.

```
cmw-cbor-tag<bytes> = #6.<cbor-tag-numbers>(bytes)

cbor-tag-numbers = 0..18446744073709551615
```

Figure 2: CDDL definition of the CBOR Tag format

3.2.1. Use of Pre-existing CBOR Tags

If a CBOR tag has been registered in association with a certain RATS conceptual message independently of a CoAP content format (i.e., it is not obtained by applying the TN() transform), it can be readily used as an encapsulation without the extra processing described in Section 3.2.

A consumer can always distinguish tags that have been derived via TN(), which all fall in the [1668546817, 1668612095] range, from tags that are not, and therefore apply the right decapsulation on receive.

3.3. Decapsulation Algorithm

After removing any external framing (for example, the ASN.1 OCTET STRING if the CMW is carried in a certificate extension [DICE-arch]), the CMW decoder does a 1-byte lookahead, as illustrated in the following pseudo code, to decide how to decode the remainder of the byte buffer:

```
func CMWTypeSniff(b []byte) (CMW, error) {
  if len(b) == 0 {
    return Unknown
  }

  if b[0] == 0x82 || b[0] == 0x83 {
    return CBORArray
  } else if b[0] >= 0xc0 && b[0] <= 0xdb {
    return CBORTag
  } else if b[0] == 0x5b {
    return JSONArray
  }

  return Unknown
}
```

4. Examples

The (equivalent) examples in Section 4.1, Section 4.2, and Section 4.3 assume that the Media-Type-Name application/vnd.example.rats-conceptual-msg has been registered alongside a corresponding CoAP Content-Format number 30001. The CBOR tag 1668576818 is derived applying the TN() transform as described in Section 3.2.

The example in Section 4.4 is a signed CoRIM payload with an explicit CM indicator 0b0000_0011 (3), meaning that the wrapped message contains both Reference Values and Endorsements.

4.1. JSON Array

```
[
  "application/vnd.example.rats-conceptual-msg",
  "q82rzQ"
]
```

Note that a CoAP Content-Format number can also be used with the JSON array form. That may be the case when it is known that the receiver can handle CoAP Content-Formats and it is crucial to save bytes.

4.2. CBOR Array

```
[
  30001,
  h'abcdabcd'
]
```

with the following wire representation:

```
82             # array(2)
   19 7531     # unsigned(30001)
   44          # bytes(4)
      abcdabcd # "\xAB\xCD"
```

Note that a Media-Type-Name can also be used with the CBOR array form, for example if it is known that the receiver cannot handle CoAP Content-Formats, or (unlike the case in point) if a CoAP Content-Format number has not been registrered.

```
[
  "application/vnd.example.rats-conceptual-msg",
  h'abcdabcd'
]
```

4.3. CBOR Tag

```
1668576818(h'abcdabcd')
```

with the following wire representation:

```
da 63747632    # tag(1668576818)
   44          # bytes(4)
      abcdabcd # "\xAB\xCD"
```

4.4. CBOR Array with explicit CM indicator

```
[
  "application/signed-corim+cbor",
  h'd28443a10126a1',
  3
]
```

with the following wire representation:

```
83                                  # array(3)
   78 1d                            # text(29)
      6170706c69636174696f6e2f7369676e65642d636f72696d2b63626f72
                                    # "application/signed-corim+cbor"
   47                               # bytes(7)
      d28443a10126a1                # "\xD2\x84C\xA1\u0001&\xA1"
   03                               # unsigned(3)
```

5. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to

RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. Project Veraison

The organization responsible for this implementation is Project Veraison, a Linux Foundation project hosted at the Confidential Computing Consortium.

The software, hosted at https://github.com/veraison/cmw, provides a Golang package that allows encoding and decoding of CMW payloads. The implementation covers all the features presented in this draft. The maturity level is alpha. The license is Apache 2.0. The developers can be contacted on the Zulip channel: https://veraison.zulipchat.com/#narrow/stream/383526-CMW/.

6. Security Considerations

This document defines two encapsulation formats for RATS conceptual messages. The messages themselves and their encoding ensure security protection. For this reason there are no further security requirements raised by the introduction of this encapsulation.

Changing the encapsulation of a payload by an adversary will result in incorrect processing of the encapsulated messages and this will subsequently lead to a processing error.

7. IANA Considerations

RFC Editor: replace "RFCthis" with the RFC number assigned to this document.

7.1. CWT cmw Claim Registration

IANA is requested to add a new cmw claim to the "CBOR Web Token (CWT) Claims" registry [IANA.cwt] as follows:

*Claim Name: cmw

*Claim Description: A RATS Conceptual Message Wrapper

*Claim Key: TBD

*Claim Value Type(s): CBOR Array, or CBOR Tag

*Change Controller: IETF

*Specification Document(s): Section 3.1 and Section 3.2 of RFCthis

The suggested value for the Claim Key is 299.

7.2. JWT cmw Claim Registration

IANA is requested to add a new cmw claim to the "JSON Web Token Claims" sub-registry of the "JSON Web Token (JWT)" registry [IANA.jwt] as follows:

*Claim Name: cmw

*Claim Description: A RATS Conceptual Message Wrapper

*Claim Value Type(s): JSON Array

*Change Controller: IETF

*Specification Document(s): Section 3.1 of RFCthis

7.3. CBOR Tag Registration

IANA is requested to add the following tag to the "CBOR Tags" [IANA.cbor-tags] registry.

CBOR Tag	Data Item	Semantics	Reference
TBD	CBOR array, CBOR tag	RATS Conceptual Message Wrapper	Section 3.1 and Section 3.2 of RFCthis

Table 1

7.4. RATS Conceptual Message Wrapper (CMW) Indicators Registry

This specification defines a new "RATS Conceptual Message Wrapper (CMW) Indicators" registry, with the policy "Expert Review" (Section 4.5 of [BCP26]).

The objective is to have Indicators values registered for all RATS Conceptual Messages (Section 8 of [RFC9334]).

7.4.1. Instructions for the Designated Expert

The expert is instructed to add the values incrementally.

Acceptable values are those corresponding to RATS Conceptual Messages defined by the RATS architecture [RFC9334] and any of its updates.

7.4.2. Structure of Entries

Each entry in the registry must include:

Indicator value:
A number corresponding to the bit position in the cm-ind bitmap.

Conceptual Message name:
A text string describing the RATS conceptual message this indicator corresponds to.

Reference:
A reference to a document, if available, or the registrant.

The initial registrations for the registry are detailed in Table 2.

Indicator value	Conceptual Message name	Reference
0	Reference Values	RFCthis
1	Endorsements	RFCthis
2	Evidence	RFCthis
3	Attestation Results	RFCthis

Table 2: CMW Indicators Registry Initial Contents

8. References

8.1. Normative References

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26,

RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

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

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

[IANA.jwt] IANA, "JSON Web Token (JWT)", <http://www.iana.org/assignments/jwt>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[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/rfc/rfc6838>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[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>.

[RFC9277] Richardson, M. and C. Bormann, "On Stable Storage for Items in Concise Binary Object Representation (CBOR)", RFC 9277, DOI 10.17487/RFC9277, August 2022, <https://www.rfc-editor.org/rfc/rfc9277>.

[STD94] 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/rfc/rfc8949>.

8.2. Informative References

[DICE-arch] Trusted Computing Group, "DICE Attestation Architecture", March 2021, <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-r23-final.pdf>.

[I-D.fossati-tls-attestation] Tschofenig, H., Sheffer, Y., Howard, P., Mihalcea, I., and Y. Deshpande, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-fossati-tls-attestation-04, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-fossati-tls-attestation-04>.

[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-05, 30 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-05>.

[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-22, 14 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-22>.

[I-D.ietf-rats-eat-media-type] Lundblade, L., Birkholz, H., and T. Fossati, "EAT Media Types", Work in Progress, Internet-Draft, draft-ietf-rats-eat-media-type-04, 23 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-media-type-04>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[RFC9193] Keränen, A. and C. Bormann, "Sensor Measurement Lists (SenML) Fields for Indicating Data Value Content-Format",

RFC 9193, DOI 10.17487/RFC9193, June 2022, <https://www.rfc-editor.org/rfc/rfc9193>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

Appendix A. RFC9193 Content-Type ABNF

```
; from RFC9193
Content-Type-ABNF = '

Content-Type   = Media-Type-Name *( *SP ";" *SP parameter )
parameter      = token "=" ( token / quoted-string )

token          = 1*tchar
tchar          = "!" / "#" / "$" / "%" / "&" / "\'" / "*"
               / "+" / "-" / "." / "^" / "_" / "`" / "|" / "~"
               / DIGIT / ALPHA
quoted-string  = %x22 *( qdtext / quoted-pair ) %x22
qdtext         = SP / %x21 / %x23-5B / %x5D-7E
quoted-pair    = "\" ( SP / VCHAR )

Media-Type-Name = type-name "/" subtype-name

type-name = restricted-name
subtype-name = restricted-name

restricted-name = restricted-name-first *126restricted-name-chars
restricted-name-first  = ALPHA / DIGIT
restricted-name-chars  = ALPHA / DIGIT / "!" / "#" /
                         "$" / "&" / "-" / "^" / "_"
restricted-name-chars =/ "." ; Characters before first dot always
                             ; specify a facet name
restricted-name-chars =/ "+" ; Characters after last plus always
                             ; specify a structured syntax suffix

DIGIT     =  %x30-39           ; 0 - 9
POS-DIGIT =  %x31-39           ; 1 - 9
ALPHA     =  %x41-5A / %x61-7A ; A - Z / a - z
SP        =  %x20
VCHAR     =  %x21-7E           ; printable ASCII (no SP)
'
```

Appendix B. Registering and Using CMWs

Figure 3 describes the registration preconditions for using CMWs in either array or CBOR tag forms.

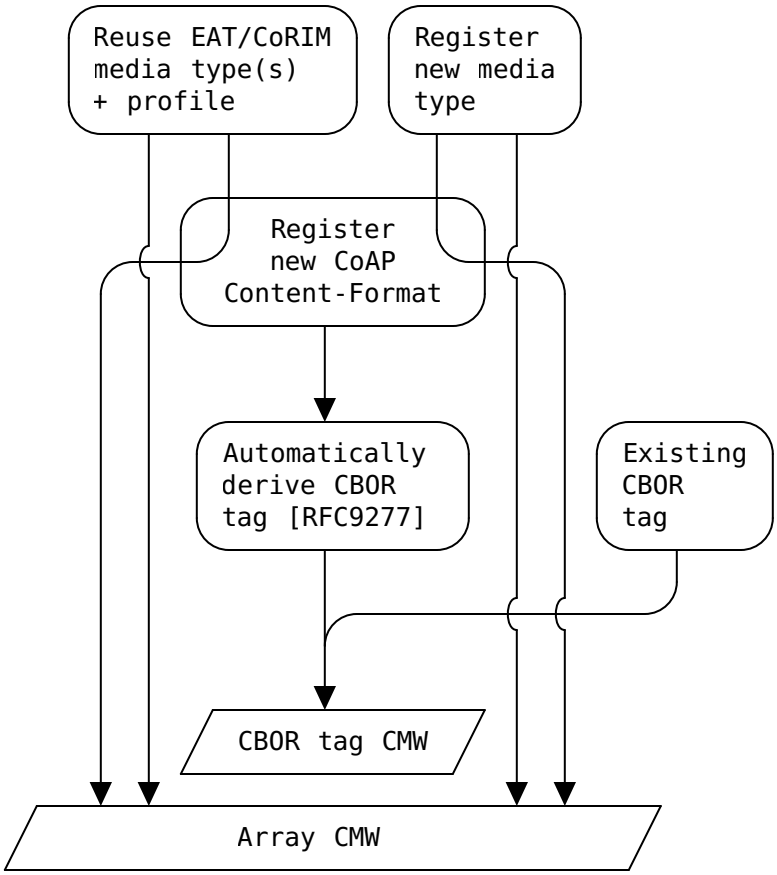

Figure 3: How To CMW

Appendix C. Open Issues

The list of currently open issues for this documents can be found at https://github.com/thomas-fossati/draft-ftbs-rats-msg-wrap/issues.

Note to RFC Editor: please remove before publication.

Acknowledgments

The authors would like to thank Carl Wallace and Carsten Bormann for their reviews and suggestions.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Ned Smith

Intel

Email: ned.smith@intel.com

Thomas Fossati
Linaro

Email: thomas.fossati@linaro.org

Hannes Tschofenig

Email: hannes.tschofenig@gmx.net