

TEEP  
Internet-Draft  
Intended status: Standards Track  
Expires: June 9, 2020

H. Tschofenig  
Arm Ltd.  
M. Pei  
Broadcom  
D. Wheeler  
Intel  
D. Thaler  
Microsoft  
December 7, 2019

**Trusted Execution Environment Provisioning (TEEP) Protocol**  
**draft-ietf-teep-protocol-00**

Abstract

This document specifies a protocol that installs, updates, and deletes Trusted Applications (TAs) in a device with a Trusted Execution Environment (TEE). This specification defines an interoperable protocol for managing the lifecycle of TAs.

The protocol name is pronounced teepee. This conjures an image of a wedge-shaped protective covering for one's belongings, which sort of matches the intent of this protocol.

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 June 9, 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.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">2.</a>	Requirements Language . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Message Overview . . . . .	<a href="#">3</a>
<a href="#">4.</a>	Detailed Messages Specification . . . . .	<a href="#">5</a>
<a href="#">4.1.</a>	QueryRequest . . . . .	<a href="#">5</a>
<a href="#">4.2.</a>	QueryResponse . . . . .	<a href="#">7</a>
<a href="#">4.3.</a>	TrustedAppInstall . . . . .	<a href="#">9</a>
<a href="#">4.4.</a>	TrustedAppDelete . . . . .	<a href="#">9</a>
<a href="#">4.5.</a>	Success . . . . .	<a href="#">10</a>
<a href="#">4.6.</a>	Error . . . . .	<a href="#">10</a>
<a href="#">5.</a>	Ciphersuites . . . . .	<a href="#">12</a>
<a href="#">6.</a>	Security Consideration . . . . .	<a href="#">13</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">14</a>
<a href="#">7.1.</a>	Media Type Registration . . . . .	<a href="#">14</a>
<a href="#">7.2.</a>	Error Code Registry . . . . .	<a href="#">16</a>
<a href="#">7.3.</a>	Ciphersuite Registry . . . . .	<a href="#">17</a>
<a href="#">8.</a>	References . . . . .	<a href="#">17</a>
<a href="#">8.1.</a>	Normative References . . . . .	<a href="#">17</a>
<a href="#">8.2.</a>	Informative References . . . . .	<a href="#">18</a>
<a href="#">Appendix A.</a>	Acknowledgements . . . . .	<a href="#">19</a>
<a href="#">Appendix B.</a>	Contributors . . . . .	<a href="#">19</a>
	Authors' Addresses . . . . .	<a href="#">20</a>

## [1.](#) Introduction

The Trusted Execution Environment (TEE) concept has been designed to separate a regular operating system, also referred as a Rich Execution Environment (REE), from security-sensitive applications. In an TEE ecosystem, different device vendors may use different operating systems in the REE and may use different types of TEEs. When application providers or device administrators use Trusted Application Managers (TAMs) to install, update, and delete Trusted Applications (TAs) on a wide range of devices with potentially different TEEs then an interoperability need arises.



This document specifies the protocol for communicating between a TAM and a TEEP Agent, involving a TEEP Broker.

The Trusted Execution Environment Provisioning (TEEP) architecture document [[I-D.ietf-teep-architecture](#)] has set to provide a design guidance for such an interoperable protocol and introduces the necessary terminology. Note that the term Trusted Application may include more than code; it may also include configuration data and keys needed by the TA to operate correctly.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

This specification re-uses the terminology defined in [[I-D.ietf-teep-architecture](#)].

## 3. Message Overview

The TEEP protocol consists of a couple of messages exchanged between a TAM and a TEEP Agent via a TEEP Broker. The messages are encoded either in JSON or CBOR and designed to provide end-to-end security. TEEP protocol messages are signed and/or encrypted by the endpoints, i.e., the TAM and the TEEP Agent, but trusted applications may as well be encrypted and signed by the service provider. The TEEP protocol not only re-use JSON and CBOR but also the respective security wrappers, namely JOSE (JWS [[RFC7515](#)] and JWE [[RFC7516](#)], to be more specific) and COSE [[RFC8152](#)]. Furthermore, for attestation the Entity Attestation Token (EAT) [[I-D.ietf-rats-eat](#)] and for software updates the SUIT manifest format [[I-D.ietf-suit-manifest](#)] are re-used.

This specification defines six messages.

A TAM queries a device's current state with a QueryRequest message. A TEEP Agent will, after authenticating and authorizing the request, report attestation information, list all TAs, and provide information about supported algorithms and extensions in a QueryResponse message. An error message is returned if the request could not be processed. A TAM will process the QueryResponse message and determine whether subsequent message exchanges to install, update, or delete trusted applications shall be initiated.



+-----+	+-----+
TAM	TEEP Agent
+-----+	+-----+

QueryRequest ----->

QueryResponse

<----- or

Error

With the TrustedAppInstall message a TAM can instruct a TEEP Agent to install a TA. The TEEP Agent will process the message, determine whether the TAM is authorized and whether the TA has been signed by an authorized SP. In addition to the binary, the TAM may also provide personalization data. If the TrustedAppInstall message was processed successfully then a Success message is returned to the TAM, an Error message otherwise.

+-----+	+-----+
TAM	TEEP Agent
+-----+	+-----+

TrustedAppInstall ---->

Success

<---- or

Error

With the TrustedAppDelete message a TAM can instruct a TEEP Agent to delete one or multiple TA(s). A Success message is returned when the operation has been completed successfully, and an Error message otherwise.

+-----+	+-----+
TAM	TEEP Agent
+-----+	+-----+

TrustedAppDelete ---->

Success

<---- or

Error



#### [4.](#) Detailed Messages Specification

For a CBOR-based encoding the following security wrapper is used (described in CDDL format [[I-D.ietf-cbor-cddl](#)]).

```
Outer_Wrapper = {  
    msg-authenc-wrapper      => bstr .cbor  
                               Msg_AuthEnc_Wrapper / nil,  
    teep-message             => (QueryRequest /  
                                QueryResponse /  
                                TrustedAppInstall /  
                                TrustedAppDelete /  
                                Error /  
                                Success ),  
}  
  
msg-authenc-wrapper = 1  
teep-message = 2  
  
Msg_AuthEnc_Wrapper = [ * (COSE_Mac_Tagged /  
                           COSE_Sign_Tagged /  
                           COSE_Mac0_Tagged /  
                           COSE_Sign1_Tagged)]
```

A future version of this specification will also describe the security wrapper for JSON (in CDDL format).

##### [4.1.](#) QueryRequest





```
suite = int

version = int

data_items = (
    attestation: 1,
    trusted_apps: 2,
    extensions: 3,
    suit_commands: 4
)

QueryRequest = (
    TYPE : int,
    TOKEN : bstr,
    REQUEST : [+data_items],
    ? CIPHER_SUITE : [+suite],
    ? NONCE : bstr,
    ? VERSION : [+version],
    ? OCSP_DATA : bstr,
    * $$extensions
)
```

A QueryRequest message is signed by the TAM and has the following fields:

**TYPE** TYPE = 1 corresponds to a QueryRequest message sent from the TAM to the TEEP Agent.

**TOKEN** The value in the TOKEN field is used to match requests to responses.

**REQUEST** The REQUEST field indicates what information the TAM requests from the TEEP Agent in form of a list of integer values. Each integer value corresponds to an IANA registered information element. This specification defines the initial set of information elements:

**attestation (1)** With this value the TAM requests the TEEP Agent to return an entity attestation token (EAT) in the response.

**trusted\_apps (2)** With this value the TAM queries the TEEP Agent for all installed TAs.

**extensions (3)** With this value the TAM queries the TEEP Agent for supported capabilities and extensions, which allows a TAM to discover the capabilities of a TEEP Agent implementation.



suit\_commands (4) With this value the TAM queries the TEEP Agent for supported commands offered by the SUIT manifest implementation.

Further values may be added in the future via IANA registration.

**CIPHER\_SUITE** The CIPHER\_SUITE field lists the ciphersuite(s) supported by the TAM. Details about the ciphersuite encoding can be found in [Section 5](#).

**NONCE** NONCE is an optional field used for ensuring the refreshness of the Entity Attestation Token (EAT) contained in the response.

**VERSION** The VERSION field lists the version(s) supported by the TAM. For this version of the specification this field can be omitted.

**OCSP\_DATA** The OCSP\_DATA field contains a list of OCSP stapling data respectively for the TAM certificate and each of the CA certificates up to the root certificate. The TAM provides OCSP data so that the TEEP Agent can validate the status of the TAM certificate chain without making its own external OCSP service call. OCSP data MUST be conveyed as a DER-encoded OCSP response (using the ASN.1 type OCSPResponse defined in [[RFC2560](#)]). The use of OCSP is optional to implement for both the TAM and the TEEP Agent. A TAM can query the TEEP Agent for the support of this functionality via the capability discovery exchange, as described above.

#### [4.2](#). QueryResponse



```
ta_id = (  
    Vendor_ID = bstr,  
    Class_ID = bstr,  
    Device_ID = bstr,  
    * $$extensions  
)  
  
ext_info = int  
  
QueryResponse = (  
    TYPE : int,  
    TOKEN : bstr,  
    ? SELECTED_CIPHER_SUITE : suite,  
    ? SELECTED_VERSION : version,  
    ? EAT : bstr,  
    ? TA_LIST : [+ta_id],  
    ? EXT_LIST : [+ext_info],  
    * $$extensions  
)
```

The QueryResponse message is signed and encrypted by the TEEP Agent and returned to the TAM. It has the following fields:

**TYPE** TYPE = 2 corresponds to a QueryResponse message sent from the TEEP Agent to the TAM.

**TOKEN** The value in the TOKEN field is used to match requests to responses. The value MUST correspond to the value received with the QueryRequest.

**SELECTED\_CIPHER\_SUITE** The SELECTED\_CIPHER\_SUITE field indicates the selected ciphersuite. Details about the ciphersuite encoding can be found in [Section 5](#).

**SELECTED\_VERSION** The SELECTED\_VERSION field indicates the protocol version selected by the TEEP Agent.

**EAT** The EAT field contains an Entity Attestation Token following the encoding defined in [\[I-D.ietf-rats-eat\]](#).

**TA\_LIST** The TA\_LIST field enumerates the trusted applications installed on the device in form of ta\_ids, i.e., a vendor id/class id/device id triple.

**EXT\_LIST** The EXT\_LIST field lists the supported extensions. This document does not define any extensions.



### **4.3. TrustedAppInstall**

```
TrustedAppInstall = (  
    TYPE : int,  
    TOKEN : bstr,  
    ? MANIFEST_LIST : [+ SUIT_Outer_Wrapper],  
    * $$extensions  
)
```

The TrustedAppInstall message is MACed and encrypted by the TAM and has the following fields:

TYPE TYPE = 3 corresponds to a TrustedAppInstall message sent from the TAM to the TEEP Agent. In case of successful processing, a Success message is returned by the TEEP Agent. In case of an error, an Error message is returned. Note that the TrustedAppInstall message is used for initial TA installation but also for TA updates.

TOKEN The value in the TOKEN field is used to match requests to responses.

TA The MANIFEST\_LIST field is used to convey one or multiple SUIT manifests. A manifest is a bundle of metadata about the trusted app, where to find the code, the devices to which it applies, and cryptographic information protecting the manifest. The manifest may also convey personalization data. TA binaries and personalization data is typically signed and encrypted by the SP. Other combinations are, however, possible as well. For example, it is also possible for the TAM to sign and encrypt the personalization data and to let the SP sign and/or encrypt the TA binary.

### **4.4. TrustedAppDelete**

```
TrustedAppDelete = (  
    TYPE : int,  
    TOKEN : bstr,  
    ? TA_LIST : [+ta_id],  
    * $$extensions  
)
```

The TrustedAppDelete message is MACed and encrypted by the TAM and has the following fields:

TYPE TYPE = 4 corresponds to a TrustedAppDelete message sent from the TAM to the TEEP Agent. In case of successful processing, an





Success message is returned by the TEEP Agent. In case of an error, an Error message is returned.

**TOKEN** The value in the TOKEN field is used to match requests to responses.

**TA\_LIST** The TA\_LIST field enumerates the TAs to be deleted.

#### **4.5. Success**

```
Success = (  
    TYPE : int,  
    TOKEN : bstr,  
    ? MSG : tstr,  
    * $$extensions  
)
```

The Success message is MACed and encrypted by the TEEP Agent and has the following fields:

**TYPE** TYPE = 5 corresponds to a Error message sent from the TEEP Agent to the TAM.

**TOKEN** The value in the TOKEN field is used to match requests to responses.

**MSG** The MSG field contains optional diagnostics information encoded in UTF-8 [[RFC3629](#)] returned by the TEEP Agent.

#### **4.6. Error**

```
Error = (  
    TYPE : int,  
    TOKEN : bstr,  
    ERR_CODE : int,  
    ? ERR_MSG : tstr,  
    ? CIPHER_SUITE : [+suite],  
    ? VERSION : [+version],  
    * $$extensions  
)
```

If possible, the Error message is MACed and encrypted by the TEEP Agent. Unprotected Error messages MUST be handled with care by the TAM due to possible downgrading attacks. It has the following fields:

**TYPE** TYPE = 6 corresponds to a Error message sent from the TEEP Agent to the TAM.



**TOKEN** The value in the TOKEN field is used to match requests to responses.

**ERR\_CODE** The ERR\_CODE field is populated with values listed in a registry (with the initial set of error codes listed below). Only selected messages are applicable to each message.

**ERR\_MSG** The ERR\_MSG message is a human-readable diagnostic message that MUST be encoded using UTF-8 [[RFC3629](#)] using Net-Unicode form [[RFC5198](#)].

**VERSION** The VERSION field enumerates the protocol version(s) supported by the TEEP Agent. This field is optional but MUST be returned with the ERR\_UNSUPPORTED\_MSG\_VERSION error message.

**CIPHER\_SUITE** The CIPHER\_SUITE field lists the ciphersuite(s) supported by the TEEP Agent. This field is optional but MUST be returned with the ERR\_UNSUPPORTED\_CRYPT0\_ALG error message.

This specification defines the following initial error messages. Additional error code can be registered with IANA.

**ERR\_ILLEGAL\_PARAMETER** The TEEP Agent sends this error message when a request contains incorrect fields or fields that are inconsistent with other fields.

**ERR\_UNSUPPORTED\_EXTENSION** The TEEP Agent sends this error message when it recognizes an unsupported extension or unsupported message.

**ERR\_REQUEST\_SIGNATURE\_FAILED** The TEEP Agent sends this error message when it fails to verify the signature of the message.

**ERR\_UNSUPPORTED\_MSG\_VERSION** The TEEP Agent receives a message but does not support the indicated version.

**ERR\_UNSUPPORTED\_CRYPT0\_ALG** The TEEP Agent receives a request message encoded with an unsupported cryptographic algorithm.

**ERR\_BAD\_CERTIFICATE** The TEEP Agent returns this error when processing of a certificate failed. For diagnosis purposes it is RECOMMENDED to include information about the failing certificate in the error message.

**ERR\_UNSUPPORTED\_CERTIFICATE** The TEEP Agent returns this error when a certificate was of an unsupported type.



**ERR\_CERTIFICATE\_REVOKED** The TEEP Agent returns this error when a certificate was revoked by its signer.

**ERR\_CERTIFICATE\_EXPIRED** The TEEP Agent returns this error when a certificate has expired or is not currently valid.

**ERR\_INTERNAL\_ERROR** The TEEP Agent returns this error when a miscellaneous internal error occurred while processing the request.

**ERR\_RESOURCE\_FULL** This error is reported when a device resource isn't available anymore, such as storage space is full.

**ERR\_TA\_NOT\_FOUND** This error will occur when the target TA does not exist. This error may happen when the TAM has stale information and tries to delete a TA that has already been deleted.

**ERR\_TA\_ALREADY\_INSTALLED** While installing a TA, a TEE will return this error if the TA has already been installed.

**ERR\_TA\_UNKNOWN\_FORMAT** The TEEP Agent returns this error when it does not recognize the format of the TA binary.

**ERR\_TA\_DECRYPTION\_FAILED** The TEEP Agent returns this error when it fails to decrypt the TA binary.

**ERR\_TA\_DECOMPRESSION\_FAILED** The TEEP Agent returns this error when it fails to decompress the TA binary.

**ERR\_MANIFEST\_PROCESSING\_FAILED** The TEEP Agent returns this error when manifest processing failures occur that are less specific than **ERR\_TA\_UNKNOWN\_FORMAT**, **ERR\_TA\_UNKNOWN\_FORMAT**, and **ERR\_TA\_DECOMPRESSION\_FAILED**.

**ERR\_PD\_PROCESSING\_FAILED** The TEEP Agent returns this error when it fails to process the provided personalization data.

## 5. Ciphersuites

A ciphersuite consists of an AEAD algorithm, a HMAC algorithm, and a signature algorithm. Each ciphersuite is identified with an integer value, which corresponds to an IANA registered ciphersuite. This document specifies two ciphersuites.

Value | Ciphersuite

-----+-----	
0	AES-CCM-16-64-128, HMAC 256/256, X25519, EdDSA
1	AES-CCM-16-64-128, HMAC 256/256, P-256, ES256



## 6. Security Consideration

This section summarizes the security considerations discussed in this specification:

**Cryptographic Algorithms** This specification relies on the cryptographic algorithms provided by the security wrappers JOSE and COSE, respectively. A companion document makes algorithm recommendations but this document is written in an algorithm-agnostic way. TEEP protocol messages exchanged between the TAM and the TEEP Agent are protected using JWS and JWE (for JSON-encoded messages) and COSE (for CBOR-encoded messages). Public key based authentication is used to by the TEEP Agent to authenticate the TAM and vice versa.

**Attestation** A TAM may rely on the attestation information provided by the TEEP Agent and the Entity Attestation Token is re-used to convey this information. To sign the Entity Attestation Token it is necessary for the device to possess a public key (usually in the form of a certificate) along with the corresponding private key. Depending on the properties of the attestation mechanism it is possible to uniquely identify a device based on information in the attestation information or in the certificate used to sign the attestation token. This uniqueness may raise privacy concerns. To lower the privacy implications the TEEP Agent MUST present its attestation information only to an authenticated and authorized TAM.

**TA Binaries** TA binaries are provided by the SP. It is the responsibility of the TAM to relay only verified TAs from authorized SPs. Delivery of that TA to the TEEP Agent is then the responsibility of the TAM and the TEEP Broker, using the security mechanisms provided by the TEEP protocol. To protect the TA binary the SUIIT manifest is re-used and it offers a variety of security features, including digital signatures and symmetric encryption.

**Personalization Data** An SP or a TAM can supply personalization data along with a TA. This data is also protected by a SUIIT manifest. The personalization data may be itself is (or can be) opaque to the TAM.

**TEEP Broker** The TEEP protocol relies on the TEEP Broker to relay messages between the TAM and the TEEP Agent. When the TEEP Broker is compromised it can drop messages, delay the delivery of messages, and replay messages but it cannot modify those messages. (A replay would be, however, detected by the TEEP Agent.) A compromised TEEP Broker could reorder messages in an attempt to





install an old version of a TA. Information in the manifest ensures that the TEEP Agents are protected against such downgrading attacks based on features offered by the manifest itself.

**CA Compromise** The QueryRequest message from a TAM to the TEEP Agent may include OCSP stapling data for the TAM's signer certificate and for intermediate CA certificates up to the root certificate so that the TEEP Agent can verify the certificate's revocation status.

A certificate revocation status check on a TA signer certificate is OPTIONAL by a TEEP Agent. A TAM is responsible for vetting a TA and before distributing them to TEEP Agents. TEEP Agents will trust a TA signer certificate's validation status done by a TAM.

**CA Compromise** The CA issuing certificates to a TAM or an SP may get compromised. A compromised intermediate CA certificates can be detected by a TEEP Agent by using OCSP information, assuming the revocation information is available. Additionally, it is RECOMMENDED to provide a way to update the trust anchor store used by the device, for example using a firmware update mechanism.

If the CA issuing certificates to devices gets compromised then these devices might be rejected by a TAM, if revocation is available to the TAM.

**Compromised TAM** The TEEP Agent SHOULD use OCSP information to verify the validity of the TAM-provided certificate (as well as the validity of intermediate CA certificates). The integrity and the accuracy of the clock within the TEE determines the ability to determine an expired or revoked certificate since OCSP stapling includes signature generation time, certificate validity dates are compared to the current time.

## **7. IANA Considerations**

### **7.1. Media Type Registration**

IANA is requested to assign a media type for application/teep+json.

Type name: application

Subtype name: teep+json

Required parameters: none

Optional parameters: none



Encoding considerations: Same as encoding considerations of application/json as specified in [Section 11 of \[RFC7159\]](#)

Security considerations: See Security Considerations Section of this document.

Interoperability considerations: Same as interoperability considerations of application/json as specified in [\[RFC7159\]](#)

Published specification: This document.

Applications that use this media type: TEEP protocol implementations

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): N/A

Person to contact for further information: [teep@ietf.org](mailto:teep@ietf.org)

Intended usage: COMMON

Restrictions on usage: none

Author: See the "Authors' Addresses" section of this document

Change controller: IETF

IANA is requested to assign a media type for application/teep+cbor.

Type name: application

Subtype name: teep+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: Same as encoding considerations of application/cbor



Security considerations: See Security Considerations Section of this document.

Interoperability considerations: Same as interoperability considerations of application/cbor as specified in [[RFC7049](#)]

Published specification: This document.

Applications that use this media type: TEEP protocol implementations

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): N/A

Person to contact for further information: [teep@ietf.org](mailto:teep@ietf.org)

Intended usage: COMMON

Restrictions on usage: none

Author: See the "Authors' Addresses" section of this document

Change controller: IETF

## **[7.2.](#) Error Code Registry**

IANA is also requested to create a new registry for the error codes defined in [Section 4](#).

Registration requests are evaluated after a three-week review period on the [teep-reg-review@ietf.org](mailto:teep-reg-review@ietf.org) mailing list, on the advice of one or more Designated Experts [[RFC8126](#)]. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register an error code: example"). Registration requests that are undetermined for a period



longer than 21 days can be brought to the IESG's attention (using the [iesg@ietf.org](mailto:iesg@ietf.org) mailing list) for resolution.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or whether it is useful only for a single extension, and whether the registration description is clear.

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration to the review mailing list.

### **7.3. Ciphersuite Registry**

IANA is also requested to create a new registry for ciphersuites, as defined in [Section 5](#).

## **8. References**

### **8.1. Normative References**

- [I-D.ietf-rats-eat]  
Mandyam, G., Lundblade, L., Ballesteros, M., and J. O'Donoghue, "The Entity Attestation Token (EAT)", [draft-ietf-rats-eat-01](#) (work in progress), July 2019.
- [I-D.ietf-suit-manifest]  
Moran, B., Tschofenig, H., and H. Birkholz, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", [draft-ietf-suit-manifest-02](#) (work in progress), November 2019.
- [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>>.
- [RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", [RFC 2560](#), DOI 10.17487/RFC2560, June 1999, <<https://www.rfc-editor.org/info/rfc2560>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, [RFC 3629](#), DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.





- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 4648](#), DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", [RFC 5198](#), DOI 10.17487/RFC5198, March 2008, <<https://www.rfc-editor.org/info/rfc5198>>.
- [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>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", [RFC 7159](#), DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [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/info/rfc7515>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", [RFC 7516](#), DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", [RFC 7517](#), DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [RFC7518] Jones, M., "JSON Web Algorithms (JWA)", [RFC 7518](#), DOI 10.17487/RFC7518, May 2015, <<https://www.rfc-editor.org/info/rfc7518>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", [RFC 8152](#), DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.

## 8.2. Informative References

- [I-D.ietf-cbor-cddl]  
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", [draft-ietf-cbor-cddl-08](#) (work in progress), March 2019.



[I-D.ietf-teep-architecture]

Pei, M., Tschofenig, H., Wheeler, D., Atyeo, A., and D. Liu, "Trusted Execution Environment Provisioning (TEEP) Architecture", [draft-ietf-teep-architecture-04](#) (work in progress), December 2019.

[I-D.ietf-teep-opentrustprotocol]

Pei, M., Atyeo, A., Cook, N., Yoo, M., and H. Tschofenig, "The Open Trust Protocol (OTrP)", [draft-ietf-teep-opentrustprotocol-03](#) (work in progress), May 2019.

[RFC8126] 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/info/rfc8126>>.

## **[Appendix A.](#) Acknowledgements**

This work is based on the initial version of OTrP [[I-D.ietf-teep-opentrustprotocol](#)] and hence credits go to those who have contributed to it.

We would like to thank Eve Schooler for the suggestion of the protocol name.

## **[Appendix B.](#) Contributors**

We would like to thank the following individuals for their contributions to an earlier version of this specification.

- Brian Witten  
Symantec  
[brian\\_witten@symantec.com](mailto:brian_witten@symantec.com)
- Tyler Kim  
Solacia  
[tylerkim@iotrust.kr](mailto:tylerkim@iotrust.kr)
- Nick Cook  
Arm Ltd.  
[nicholas.cook@arm.com](mailto:nicholas.cook@arm.com)
- Minh Yoo  
IoTrust  
[minho.yoo@iotrust.kr](mailto:minho.yoo@iotrust.kr)



Authors' Addresses

Hannes Tschofenig  
Arm Ltd.  
110 Fulbourn Rd  
Cambridge, CB1 9NJ  
Great Britain

Email: [hannes.tschofenig@arm.com](mailto:hannes.tschofenig@arm.com)

Mingliang Pei  
Broadcom  
350 Ellis St  
Mountain View, CA 94043  
USA

Email: [mingliang.pei@broadcom.com](mailto:mingliang.pei@broadcom.com)

David Wheeler  
Intel  
US

Email: [david.m.wheeler@intel.com](mailto:david.m.wheeler@intel.com)

Dave Thaler  
Microsoft  
US

Email: [dthaler@microsoft.com](mailto:dthaler@microsoft.com)

