TEEP
Internet-Draft
Intended status: Standards Track
Expires: October 16, 2020

H. Tschofenig
Arm Ltd.
M. Pei
Broadcom
D. Wheeler
Intel
D. Thaler
Microsoft
A. Tsukamoto
AIST

April 14, 2020

Trusted Execution Environment Provisioning (TEEP) Protocol
draft-ietf-teep-protocol-02

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 October 16, 2020.

Table of Contents

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, 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", "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.

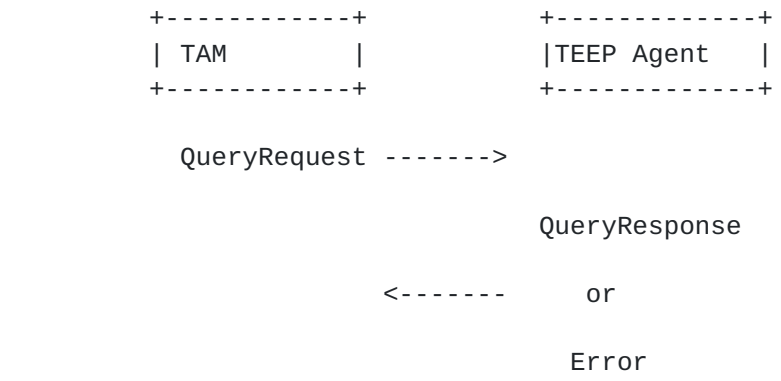
This specification re-uses the terminology defined in [I-D.ietf-teep-architecture].

3. Message Overview

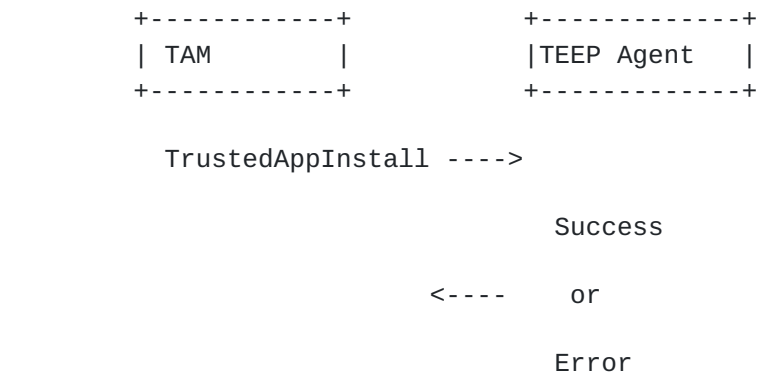
The TEEP protocol consists of a couple of messages exchanged between a TAM and a TEEP Agent via a TEEP Broker. The messages are encoded in CBOR and designed to provide end-to-end security. TEEP protocol messages are signed by the endpoints, i.e., the TAM and the TEEP Agent, but trusted applications may as well be encrypted and signed by the service provider. The TEEP protocol not only re-use CBOR but also the respective security wrapper, namely 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.

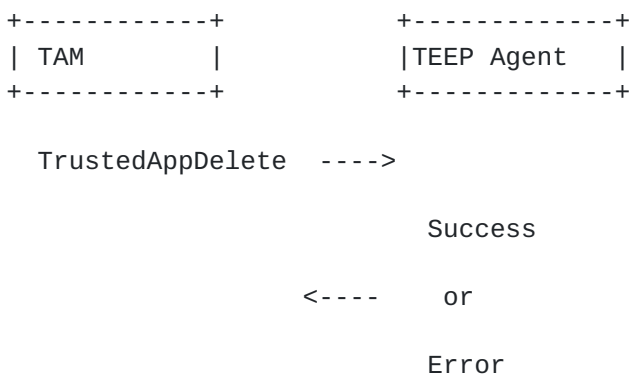
A TAM queries a device's current state with a QueryRequest message. A TEEP Agent will, after authenticating and authorizing the request, report attestation information, list all TAs, and provide information about supported algorithms and extensions in a QueryResponse message. An error message is returned 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.



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.



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.



4. Detailed Messages Specification

TEEP messages are protected by the COSE_Sign1 structure. The TEEP protocol messages are described in CDDL format [RFC8610] below.

```
    teep-message                => (QueryRequest /
                                    QueryResponse /
                                    TrustedAppInstall /
                                    TrustedAppDelete /
                                    Error /
                                    Success ),
}
```

4.1. Creating and Validating TEEP Messages

4.1.1. Creating a TEEP message

To create a TEEP message, the following steps are performed.

1. Create a TEEP message according to the description below and populate it with the respective content.

2. Create a COSE Header containing the desired set of Header Parameters. The COSE Header MUST be valid per the [RFC8152] specification.

3. Create a COSE_Sign1 object using the TEEP message as the COSE_Sign1 Payload; all steps specified in [RFC8152] for creating a COSE_Sign1 object MUST be followed.

4. Prepend the COSE object with the TEEP CBOR tag to indicate that the CBOR-encoded message is indeed a TEEP message.

4.1.2. Validating a TEEP Message

When validating a TEEP message, the following steps are performed. If any of the listed steps fail, then the TEEP message MUST be rejected.

1. Verify that the received message is a valid CBOR object.

2. Remove the TEEP message CBOR tag and verify that one of the COSE CBOR tags follows it.

3. Verify that the message contains a COSE_Sign1 structure.

4. Verify that the resulting COSE Header includes only parameters and values whose syntax and semantics are both understood and supported or that are specified as being ignored when not understood.

5. Follow the steps specified in Section 4 of [RFC8152] ("Signing Objects") for validating a COSE_Sign1 object. The COSE_Sign1 payload is the content of the TEEP message.

6. Verify that the TEEP message is a valid CBOR map and verify the fields of the TEEP message according to this specification.

4.2. QueryRequest

A QueryRequest message is used by the TAM to learn information from the TEEP Agent. The TAM can learn the features supported by the TEEP Agent, including ciphersuites, and protocol versions. Additionally, the TAM can selectively request data items from the TEEP Agent via the request parameter. Currently, the following features are supported:

- Request for attestation information,

- Listing supported extensions,

- Querying installed software (trusted apps), and

- Listing supporting SUIT commands.

Like other TEEP messages, the QueryRequest message is signed, and the relevant CDDL snippet is shown below. The complete CDDL structure is shown in [CDDL].

```
query-request = [
  type: TEEP-TYPE-query-request,
  token: uint,
  options: {
    ? supported-cipher-suites => suite,
    ? nonce => bstr .size (8..64),
    ? version => [ + version ],
    ? oscp-data => bstr,
    * $$query-request-extensions
    * $$teep-option-extensions
  },
  data-item-requested
]
```

The message has the following fields:

type
: The value of (1) corresponds to a QueryRequest message sent from the TAM to the TEEP Agent.

token
: The value in the token parameter is used to match responses to requests. This is particualrly useful when a TAM issues multiple concurrent requests to a TEEP Agent.

request
: The request parameter indicates what information the TAM requests from the TEEP Agent in form of a bitmap. Each value in the bitmap corresponds to an IANA registered information element. This specification defines the following 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. If the TAM requests an attestation token to be returned by the TEEP Agent then it MUST also include the nonce in the message. The nonce is subsequently placed into the EAT token for replay protection.

  trusted_apps (2) With this value the TAM queries the TEEP Agent for all installed TAs.

  extensions (4) 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 (8) 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-suites
The cipher-suites parameter lists the ciphersuite(s) supported by the TAM. Details about the ciphersuite encoding can be found in Section 6.

nonce
The none field is an optional parameter used for ensuring the refreshness of the Entity Attestation Token (EAT) returned with a QueryResponse message. When a nonce is provided in the QueryRequest and an EAT is returned with the QueryResponse message then the nonce contained in this request MUST be copied into the nonce claim found in the EAT token.

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

ocsp_data
The ocsp_data parameter 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.

4.3. QueryResponse

The QueryResponse message is the successful response by the TEEP Agent after receiving a QueryRequest message.

Like other TEEP messages, the QueryResponse message is signed, and the relevant CDDL snippet is shown below. The complete CDDL structure is shown in [CDDL].

```
query-response = [
  type: TEEP-TYPE-query-response,
  token: uint,
  options: {
    ? selected-cipher-suite => suite,
    ? selected-version => version,
    ? eat => bstr,
    ? ta-list  => [ + bstr ],
    ? ext-list => [ + ext-info ],
    * $$query-response-extensions,
    * $$teep-option-extensions
  }
]
```

The message has the following fields:

type
: The value of (2) corresponds to a QueryResponse message sent from the TEEP Agent to the TAM.

token
: The value in the token parameter is used to match responses to requests. The value MUST correspond to the value received with the QueryRequest message.

selected-cipher-suite
: The selected-cipher-suite parameter indicates the selected ciphersuite. Details about the ciphersuite encoding can be found in Section 6.

selected-version
: The selected-version parameter indicates the protocol version selected by the TEEP Agent.

eat
: The eat parameter contains an Entity Attestation Token following the encoding defined in [I-D.ietf-rats-eat].

ta-list
: The ta-list parameter enumerates the trusted applications installed on the device in form of TA_ID byte strings.

ext-list
: The ext-list parameter lists the supported extensions. This document does not define any extensions.

4.4. TrustedAppInstall

The TrustedAppInstall message is used by the TAM to install software (trusted apps) via the TEEP Agent.

Like other TEEP messages, the TrustedAppInstall message is signed, and the relevant CDDL snippet is shown below. The complete CDDL structure is shown in [CDDL].

```
trusted-app-install = [
  type: TEEP-TYPE-trusted-app-install,
  token: uint,
  option: {
    ? manifest-list => [ + SUIT-envelope ],
    * $$trusted-app-install-extensions,
    * $$teep-option-extensions
  }
]
```

The TrustedAppInstall message has the following fields:

type
: The value of (3) corresponds to a TrustedAppInstall 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. 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 responses to requests.

manifest-list
: 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.

4.5. TrustedAppDelete

The TrustedAppDelete message is used by the TAM to remove software (trust apps) from the device.

Like other TEEP messages, the TrustedAppDelete message is signed, and the relevant CDDL snippet is shown below. The complete CDDL structure is shown in [CDDL].

```
trusted-app-delete = [
  type: TEEP-TYPE-trusted-app-delete,
  token: uint,
  option: {
    ? ta-list => [ + bstr ],
    * $$trusted-app-delete-extensions,
    * $$teep-option-extensions
  }
]
```

The TrustedAppDelete message has the following fields:

type
: The value of (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 parameter is used to match responses to requests.

ta-list
: The ta-list parameter enumerates the TAs to be deleted.

4.6. Success

The TEEP protocol defines two implicit success messages and this explicit Success message for the cases where the TEEP Agent cannot return another reply, such as for the TrustedAppInstall and the TrustedAppDelete messages.

Like other TEEP messages, the Success message is signed, and the relevant CDDL snippet is shown below. The complete CDDL structure is shown in [CDDL].

```
teep-success = [
  type: TEEP-TYPE-teep-success,
  token: uint,
  option: {
    ? msg => text,
    * $$teep-success-extensions,
    * $$teep-option-extensions
  }
]
```

The Success message has the following fields:

type
: The value of (5) corresponds to corresponds to a Success message sent from the TEEP Agent to the TAM.

token
: The value in the token parameter is used to match responses to requests.

msg
: The msg parameter contains optional diagnostics information encoded in UTF-8 [RFC3629] returned by the TEEP Agent.

4.7. Error

The Error message is used by the TEEP Agent to return an error.

Like other TEEP messages, the Error message is signed, and the relevant CDDL snippet is shown below. The complete CDDL structure is shown in [CDDL].

```
teep-error = [
  type: TEEP-TYPE-teep-error,
  token: uint,
  err-code: uint,
  options: {
     ? err-msg => text,
     ? cipher-suites => [ + suite ],
     ? versions => [ + version ],
     * $$teep-error--extensions,
     * $$teep-option-extensions
  }
]
```

The Error message has the following fields:

type

The value of (6) corresponds to an Error message sent from the TEEP Agent to the TAM.

token
The value in the token parameter is used to match responses to requests.

err-code
The err-code parameter 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 parameter is a human-readable diagnostic text that MUST be encoded using UTF-8 [RFC3629] using Net-Unicode form [RFC5198].

cipher-suites
The cipher-suites parameter lists the ciphersuite(s) supported by the TEEP Agent. This field is optional but MUST be returned with the ERR_UNSUPPORTED_CRYPTO_ALG error message.

versions
The version parameter enumerates the protocol version(s) supported by the TEEP Agent. This otherwise optional parameter MUST be returned with the ERR_UNSUPPORTED_MSG_VERSION error message.

This specification defines the following initial error messages:

ERR_ILLEGAL_PARAMETER (1)
The TEEP Agent sends this error message when a request contains incorrect fields or fields that are inconsistent with other fields.

ERR_UNSUPPORTED_EXTENSION (2)
The TEEP Agent sends this error message when it recognizes an unsupported extension or unsupported message.

ERR_REQUEST_SIGNATURE_FAILED (3)
The TEEP Agent sends this error message when it fails to verify the signature of the message.

ERR_UNSUPPORTED_MSG_VERSION (4)
The TEEP Agent receives a message but does not support the indicated version.

ERR_UNSUPPORTED_CRYPTO_ALG (5)

The TEEP Agent receives a request message encoded with an unsupported cryptographic algorithm.

ERR_BAD_CERTIFICATE (6)
The TEEP Agent returns this error when processing of a certificate failed. For diagnosis purposes it is RECOMMMENDED to include information about the failing certificate in the error message.

ERR_UNSUPPORTED_CERTIFICATE (7)
The TEEP Agent returns this error when a certificate was of an unsupported type.

ERR_CERTIFICATE_REVOKED (8)
The TEEP Agent returns this error when a certificate was revoked by its signer.

ERR_CERTIFICATE_EXPIRED (9)
The TEEP Agent returns this error when a certificate has expired or is not currently valid.

ERR_INTERNAL_ERROR (10)
The TEEP Agent returns this error when a miscellaneous internal error occurred while processing the request.

ERR_RESOURCE_FULL (11)
This error is reported when a device resource isn't available anymore, such as storage space is full.

ERR_TA_NOT_FOUND (12)
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 (13)
While installing a TA, a TEE will return this error if the TA has already been installed.

ERR_TA_UNKNOWN_FORMAT (14)
The TEEP Agent returns this error when it does not recognize the format of the TA binary.

ERR_TA_DECRYPTION_FAILED (15)
The TEEP Agent returns this error when it fails to decrypt the TA binary.

ERR_TA_DECOMPRESSION_FAILED (16)
The TEEP Agent returns this error when it fails to decompress the TA binary.

ERR_MANIFEST_PROCESSING_FAILED (17)
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 (18)
The TEEP Agent returns this error when it fails to process the provided personalization data.

Additional error code can be registered with IANA.

5. Mapping of TEEP Message Parameters to CBOR Labels

In COSE, arrays and maps use strings, negative integers, and unsigned integers as their keys. Integers are used for compactness of encoding. Since the word "key" is mainly used in its other meaning, as a cryptographic key, this specification uses the term "label" for this usage as a map key.

This specification uses the following mapping:

Name	Label
cipher-suites	1
nonce	2
version	3
ocsp-data	4
selected-cipher-suite	5
selected-version	6
eat	7
ta-list	8
ext-list	9
manifest-list	10
msg	11
err-msg	12

6. 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
1	AES-CCM-16-64-128, HMAC 256/256, X25519, EdDSA
2	AES-CCM-16-64-128, HMAC 256/256, P-256, ES256

7. Security Considerations

This section summarizes the security considerations discussed in this specification:

Cryptographic Algorithms
: TEEP protocol messages exchanged between the TAM and the TEEP Agent are protected using COSE. This specification relies on the cryptographic algorithms provided by COSE. 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 SUIT manifest is re-used and it offers a varity of security features, including digitial 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 SUIT manifest. The personalization data may 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.

8. IANA Considerations

8.1. Media Type Registration

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

Intended usage: COMMON

Restrictions on usage: none

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

Change controller: IETF

8.2. 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 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 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.

8.3. Ciphersuite Registry

IANA is also requested to create a new registry for ciphersuites, as defined in Section 6.

8.4. CBOR Tag Registry

IANA is requested to register a CBOR tag in the "CBOR Tags" registry for use with TEEP messages.

The registry contents is:

- CBOR Tag: TBD1
- Data Item: TEEP Message
- Semantics: TEEP Message, as defined in [[TBD: This RFC]]
- Reference: [[TBD: This RFC]]
- Point of Contact: TEEP working group (teep@ietf.org)

9. References

9.1. 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-03 (work in progress), February 2020.

[I-D.ietf-suit-manifest]
Moran, B., Tschofenig, H., Birkholz, H., and K. Zandberg, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", draft-ietf-suit-manifest-04 (work in progress), March 2020.

[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>.

[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>.

[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>.

9.2. Informative References

[I-D.ietf-teep-architecture]
Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", draft-ietf-teep-architecture-08 (work in progress), April 2020.

[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>.

[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>.

A. Contributors

We would like to thank Brian Witten (Symantec), Tyler Kim (Solacia), Nick Cook (Arm), and Minho Yoo (IoTrust) for their contributions to the Open Trust Protocol (OTrP), which influenced the design of this specification.

B. Acknowledgements

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

We would like to thank Kohei Isobe (TRASIO/SECOM), Kuniyasu Suzaki (TRASIO/AIST), Tsukasa Oi (TRASIO), and Yuichi Takita (SECOM) for their valuable implementation feedback.

We would also like to thank Carsten Bormann and Henk Birkholz for their help with the CDDL.

C. Complete CDDL

```
teep-message = $teep-message-type .within teep-message-framework

SUIT-envelope = bstr ; placeholder

teep-message-framework = [
  type: 0..23 / $teep-type-extension,
  token: uint,
  options: { * teep-option },
```

```
    * int; further integers, e.g. for data-item-requested
  ]

  teep-option = (uint => any)

  ; messages defined below:
  $teep-message-type /= query-request
  $teep-message-type /= query-response
  $teep-message-type /= trusted-app-install
  $teep-message-type /= trusted-app-delete
  $teep-message-type /= teep-error
  $teep-message-type /= teep-success

  ; message type numbers
  TEEP-TYPE-query-request = 1
  TEEP-TYPE-query-response = 2
  TEEP-TYPE-trusted-app-install = 3
  TEEP-TYPE-trusted-app-delete = 4
  TEEP-TYPE-teep-error = 5
  TEEP-TYPE-teep-success = 6

  version = uint .size 4
  ext-info = uint

  ; data items as bitmaps
  data-item-requested = $data-item-requested .within uint .size 8
  attestation = 1
  $data-item-requested /=  attestation
  trusted-apps = 2
  $data-item-requested /= trusted-apps
  extensions = 4
  $data-item-requested /= extensions
  suit-commands = 8
  $data-item-requested /= suit-commands

  query-request = [
    type: TEEP-TYPE-query-request,
    token: uint,
    options: {
      ? supported-cipher-suites => suite,
      ? nonce => bstr .size (8..64),
      ? version => [ + version ],
      ? oscp-data => bstr,
      * $$query-request-extensions
      * $$teep-option-extensions
    },
    data-item-requested
  ]
```

```
   ; ciphersuites as bitmaps
   suite = $TEEP-suite .within uint .size 8

   TEEP-AES-CCM-16-64-128-HMAC256--256-X25519-EdDSA = 1
   TEEP-AES-CCM-16-64-128-HMAC256--256-P-256-ES256  = 2

   $TEEP-suite /= TEEP-AES-CCM-16-64-128-HMAC256--256-X25519-EdDSA
   $TEEP-suite /= TEEP-AES-CCM-16-64-128-HMAC256--256-P-256-ES256

   query-response = [
     type: TEEP-TYPE-query-response,
     token: uint,
     options: {
       ? selected-cipher-suite => suite,
       ? selected-version => version,
       ? eat => bstr,
       ? ta-list  => [ + bstr ],
       ? ext-list => [ + ext-info ],
       * $$query-response-extensions,
       * $$teep-option-extensions
     }
   ]

   trusted-app-install = [
     type: TEEP-TYPE-trusted-app-install,
     token: uint,
     option: {
       ? manifest-list => [ + SUIT-envelope ],
       * $$trusted-app-install-extensions,
       * $$teep-option-extensions
     }
   ]

   trusted-app-delete = [
     type: TEEP-TYPE-trusted-app-delete,
     token: uint,
     option: {
       ? ta-list => [ + bstr ],
       * $$trusted-app-delete-extensions,
       * $$teep-option-extensions
     }
   ]

   teep-success = [
     type: TEEP-TYPE-teep-success,
     token: uint,
     option: {
       ? msg => text,
```

```
      * $$teep-success-extensions,
      * $$teep-option-extensions
    }
  ]

  teep-error = [
    type: TEEP-TYPE-teep-error,
    token: uint,
    options: {
       ? err-msg => text,
       ? cipher-suites => [ + suite ],
       ? versions => [ + version ],
       * $$teep-error--extensions,
       * $$teep-option-extensions
    }
    err-code: uint,
  ]

  cipher-suites = 1
  nonce = 2
  versions = 3
  oscp-data = 4
  selected-cipher-suite = 5
  selected-version = 6
  eat = 7
  ta-list = 8
  ext-list = 9
  manifest-list = 10
  msg = 11
  err-msg = 12
```

Authors' Addresses

Hannes Tschofenig
Arm Ltd.
Absam, Tirol 6067
Austria

Email: hannes.tschofenig@arm.com

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

Email: mingliang.pei@broadcom.com

David Wheeler
Intel
US

Email: david.m.wheeler@intel.com

Dave Thaler
Microsoft
US

Email: dthaler@microsoft.com

Akira Tsukamoto
AIST
JP

Email: akira.tsukamoto@aist.go.jp