Credentials Provisioning and Management via EAP (EAP-CREDS)
draft-pala-eap-creds-02

Abstract

With the increase number of devices, protocols, and applications that rely on strong credentials (e.g., digital certificates, keys, or tokens) for network access, the need for a standardized credentials provisioning and management framework is paramount. The 802.1x architecture allows for entities (e.g., devices, applications, etc.) to authenticate to the network by providing a communication channel where different methods can be used to exchange different types of credentials. However, the need for managing these credentials (i.e., provisioning and renewal) is still a hard problem to solve.

EAP-CREDS, if implemented in Managed Networks (e.g., Cable Modems), could enable our operators to offer a registration and credentials management service integrated in the home WiFi thus enabling visibility about registered devices. During initialization, EAP-CREDS also allows for MUD files or URLs to be transferred between the EAP Peer and the EAP Server, thus giving detailed visibility about devices when they are provisioned with credentials for accessing the networks. The possibility provided by EAP-CREDS can help to secure home or business networks by leveraging the synergies of the security teams from the network operators thanks to the extended knowledge of what and how is registered/authenticated.

This specifications define how to support the provisioning and management of authentication credentials that can be exploited in different environments (e.g., Wired, WiFi, cellular, etc.) to users and/or devices by using EAP together with standard provisioning protocols.

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 December 6, 2019.

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

1. Requirements notation ........................................ 3
2. Introduction .................................................. 4
3. Overview of existing solutions ................................. 4
4. Scope Statement ............................................... 5
5. EAP-CREDS Protocol ............................................ 5
   5.1. EAP-CREDS as tunneled mechanism only .................... 5
   5.2. Message Flow ............................................. 6
   5.3. Phase One: Initialization ................................. 7
   5.4. Phase Two: Provisioning ................................... 9
   5.5. Phase Three: Validation ................................... 10
6. EAP-CREDS Message Format ...................................... 12
   6.1. Message Header .......................................... 13
   6.2. Message Payload ......................................... 13
   6.2.1. General TLV format .................................... 13
   6.2.2. EAP-CREDS defined TLVs ................................ 15
      6.2.2.1. The Action TLV .................................... 15
      6.2.2.2. The Certificate-Data TLV .......................... 16
      6.2.2.3. The Challenge-Data TLV ............................. 17
      6.2.2.4. The Challenge-Response TLV ......................... 17
      6.2.2.5. The Credentials-Information TLV ................... 18
      6.2.2.6. The Credentials-Data TLV .......................... 21
      6.2.2.7. The Error TLV ..................................... 22
      6.2.2.8. The Profile TLV ................................... 23
      6.2.2.9. The Protocol TLV .................................. 24
6.2.2.10. The Provisioning-Data TLV ....... 24
6.2.2.11. The Provisioning-Headers TLV ....... 25
6.2.2.12. The Provisioning-Params TLV ....... 25
6.2.2.13. The Token-Data TLV ............. 28
6.2.2.14. The Version TLV ............... 29
7. EAP-CREDS Messages ..................... 29
  7.1. The EAP-CREDS-Init Message ........... 30
    7.1.1. EAP Server’s Init Message .......... 30
    7.1.2. EAP Peer’s Init Message ........... 30
      7.1.2.1. Bootstrapping Peer’s Trustworthiness .... 31
  7.2. The EAP-CREDS-ProtoFlow Message ... 31
  7.3. The EAP-CREDS-Validate Message ..... 32
      7.3.1. Algorithm Requirements .......... 32
8. Error Handling in EAP-CREDS ............. 32
9. Fragmentation ................................ 32
10. Encapsulating Provisioning Protocols in EAP-CREDS .......... 33
11. EAP-CREDS and Simple Provisioning Protocol (SPP) ........ 33
   11.1. SPP Message Format ................. 34
   11.2. SPP: Registering New Credentials .... 34
   11.3. SPP: Renewing Credentials .......... 35
   11.4. SPP: Removing Credentials .......... 35
   11.5. SPP Password Provisioning .......... 35
   11.6. SPP Certificate Provisioning ....... 35
12. IANA Considerations ..................... 35
   12.1. Provisioning Protocols ............. 36
   12.2. Token Types ....................... 36
   12.3. Credentials Types ................. 36
   12.4. Credentials Algorithms .......... 37
   12.5. Credentials Datatypes ............ 37
   12.6. Credentials Encoding ............ 38
   12.7. Action Types ..................... 38
13. Security Considerations ................ 39
14. Acknowledgments .......................... 39
15. Normative References ..................... 39
Appendix A. EAP-CREDS Example Message Flow for Provisioning Standards .......... 40
   A.1. EAP-CREDS and CMP ................... 40
   A.2. EAP-CREDS and EST ................... 40
   A.3. EAP-CREDS and CMS ................... 40
   A.4. EAP-CREDS and ACME ................... 40
Author’s Address ......................... 40

1. Requirements notation

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].
2. Introduction

Many environments are, today, moving towards requiring strong authentication when it comes to gain access to networks. The .1x architecture provides the network administrators with the possibility to check credentials presented by a device even before providing any IP service to it.

However, the provisioning and management of these credentials is a hard problem to solve and many vendors opt for long-lived credentials that can not be easily revoked, replaced, or simply renewed.

This specification addresses the problem of providing a simple-to-use and simple-to-deploy conduit for credentials management by extending the EAP protocol to support credentials provisioning and management functionality. In particular, the EAP-CREDS method defined here provides a generic framework that can carry the messages for provisioning different types of credentials. The method can be used as a stand-alone method or it can be used as an inner methods of EAP-TTLS, EAP-TEAP, or any other tunnelling method that can provide the required encryption and (at minimum) server-side authentication to deliver server-side generated secrets (e.g., private keys, passwords, secret keys, etc.)

3. Overview of existing solutions

Currently there are many protocols that address credentials lifecycle management. In particular, when it comes to digital certificates, some of the most deployed management protocols are:

- Certificate Management Protocol (CMP) [RFC4210]
- Certificate Management over CMS (CMC) [RFC5272][RFC6402]
- Enrollment over Secure Transport (EST) [RFC7030]
- Automated Certificate Management Environment

However, none of these protocols provide native support for client that do not have IP connectivity yet (e.g., because they do not have network-access credentials, yet). The EAP-CREDS provides the possibility to use such protocols (i.e., message-based) by defining a series of messages that can be used to encapsulate the provisioning messages for the specified protocol. In particular, examples of the message flow for the major provisioning protocols are provided in Annex Appendix A.
In addition to these protocols, EAP-CREDS also defines a series of simple messages that provide a generic enrollment protocol that allows not only certificates but also other types of credentials (e.g., username/password pairs or symmetric secrets) to be delivered to the client as part of the provisioning and/or renewal process. The set of messages that make up the generic provisioning protocol is referred to as the CREDS protocol (not to be confused with the EAP-CREDS).

4. Scope Statement

This document focuses on the definition of the EAP-CREDS method to convey credentials provisioning and managing messages between the client and the AAA server. Moreover, the document defines how to encode messages for the main IETF provisioning protocols.

This document, however, does not provide specifications for how and where the credentials are generated. In particular, the credentials could be generated directly within the AAA server or at a different location (i.e., the Certificate Service Provider or CSP) site. Different authentication mechanisms (e.g., TLS, etc.) can be used to secure the communication between the server’s endpoint and the CSP.

5. EAP-CREDS Protocol

This section outlines the operation of the protocol and message flows. The format of the CREDS messages is given in Section 6.

5.1. EAP-CREDS as tunneled mechanism only

EAP-CREDS requires that an outer mechanism is in place between the Peer and the Server in order to provide authentication and confidentiality of the messages exchanged via EAP-CREDS.

In other words, the EAP-CREDS method assumes that an appropriate protection outer layer has been established to prevent the possibility to leak information or to allow man-in-the-middle attacks.

This choice was taken to simplify the message flow between Peer and Server, and to abstract EAP-CREDS from the secure-channel establishment mechanism. EAP-TLS, EAP-TTLS, and EAP-TEAP are examples of such mechanisms.
5.2. Message Flow

EAP-CREDS message flow is logically subdivided into three different phases:

Phase One (Required). CREDS Initialization. During this phase the Peer and the Server exchange the information needed to select the appropriate credentials management protocol. In particular, the Server sends its initial message of type (‘Init’) and the client replies with the details about what protocols are supported, and additional information such as the version of EAP-CREDS and the supported profiles identifiers.

Phase Two (Required). Provisioning Protocol Flow. In this phase, the Peer and the Server exchange the provisioning protocol’s messages encapsulated in a EAP-CREDS message of type ProtoFlow. The messages use two main TLVs: the first one is the Provisioning Protocol Headers TLV (optional) that carries information that might be normally conveyed via the transport protocol (e.g., HTTP headers), while the second one is the Provisioning Protocol Data (required) and it carries the provisioning protocol’s messages. Phase Two terminates with the Peer sending an empty ProtoFlow message to the Server. The server can decide to repeat phase two again to register new credentials or to renew a separate set of credentials. When no more credentials have to be managed, the Server can start phase three or simply issue an EAP-SUCCESS message that terminates the EAP session.

Phase Three (Optional). Credentials Validation. This optional phase can be initiated by the server and it is used to validate that the Peer has properly installed the credentials and can use them to authenticate itself. Depending on the credentials’ type, the messages can carry a challenge/nonce, the value of the secret/token, or other information. The format of the credentials is supposed to be known by the provider and the device.

In this document we use the following notation in the diagrams to provide information about the cardinality of a data structure (TLV) within a single EAP-CREDS message:
### 5.3. Phase One: Initialization

The following figure provides the message flow for Phase One:

```
+---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+
| Peer|       | Server|       | Peer|       | Server|       | Peer|       | Server|       | Peer|       | Server|       | Peer|       | Server|
+---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+       +---+
         | Outer Tunnel Established                   |         |
<----------------------------->                      |<----------------------------->
[1] EAP-Request/EAP-CREDS(Type=Init)          [2] EAP-Response/EAP-CREDS(Type=Init)
  { [ Version+ ] }                             { [ Version ], Protocol+, [ CredInfo+ ], [ Token ], [ Challenge ], [ Challenge-Rsp ] }
<----------------------------->                      |<----------------------------->
                                                |                                                |
```

During the CREDS initialization, after the establishment of the outer mechanism (e.g., EAP-TLS, EAP-TEAP, EAP-TTLS, etc.), the server MAY decide to start a credentials management session. In order to do that, it sends an empty message to the Client. In particular, it sends a EAP-Request/EAP-CREDS(Type=Init) message with the list of supported EAP-CREDS version via the (possibly multiple) Supported-Version TLV.
The client, sends back a message that carries one ('Supported-Version') TLV to indicate the selected version of EAP-CREDS (i.e. from the list provided by the server) (optional), one or more ('Supported-Protocol') TLV to indicate the list of supported provisioning protocols, followed by the ('CredInfo') TLVs (one or more) that provide the status of the credentials (one or more) that the Network is responsible for. If multiple credentials are configured on the Peer for this Network, then the Peer MUST include one CredInfo TLV for each available credential.

When there are no available credentials, the Peer MAY include an authorization token that can be consumed by the Server for registering new credentials. In particular, the Peer can include the ('Token-Data') TLV to convey the value of the token. The ('Challenge-Data') and ('Challenge-Response') TLVs, instead, can be used to convey a challenge and the response to that challenge based on the authorization information (e.g., maybe a public key hash is present in the Token, then the peer can generate some random data - or use the one from the Server - and generate a signature on that: the value of the signature shall be in the ('Challenge-Response') TLV and should be calculated over the values of the ('Challenge-Data') TLV and the ('Token-Data') TLV).

The server checks that the Peer’s selected protocol, version, and parameters are supported and, if not (or if the server detects an error), it can (a) send a non-recoverable error message to the peer, notify the outer (tunneling) layer, and terminate the EAP-CREDS session, or (b) start phase one again by sending a new ('EAP-CREDS-Init') message that will also carry an ERROR TLV that provides the Peer with the reason the initial response was not acceptable. The server and the peer can repeat phase one until they reach an agreement or the session is terminated.

On the other hand, in case the server can serve the request from the client, EAP-CREDS continues by starting the provisioning process. The determination of the need to start phase two or not is based on the contents of the CredInfo sent by the Peer (e.g., a credential is about to expire or a credential is simply missing). The Server can repeat phase two as many times as needed: each time, the combination of the ('Credentials-Info') TLV (a.k.a. CredInfo) and the ('Action') TLV shall be unique for each session to repeat the same operation on the same credential. In case all credentials for the Network do not need maintenance at this time, the Server can decide to end the EAP-CREDS session (no actions to be taken) and successfully complete the session.

In order to move to phase two (if the server needs to perform management operations on the credentials exposed by the peer), the
Server selects the version, the provisioning protocol, associated parameters, and, optionally, the provider and sends it back to the Peer. Phase Two officially begins with the next message exchanged by the two parties which is an EAP-Request/EAP-CREDS(Type=ProtoFlow) message to the client which includes the selected ('Version'), ('Action'), ('Protocols'), ('CredInfo'), and optionally the ('Provider') TLVs with only one value each. The TLVs cannot be repeated within the same message. If multiple values are detected in the message, the message shall be discarded and the appropriate error message shall be issued by the Peer.

5.4. Phase Two: Provisioning

The following figure provides the message flow for Phase 2:

---

The server starts phase two of the EAP-CREDS protocol by sending an EAP-Request/EAP-CREDS(Type=ProtoFlow) message with the selected protocol’s details to the Peer. This indicates that the Server is ready to initiate the provisioning protocol.
Specifically, the Server MUST include the ‘Version’ and ‘Protocol’ TLVs to indicate the EAP-CREDS version agreed upon by the parties and the specific provisioning protocol to use. The ‘provider’ TLV is optional and MUST be included if a selection was made by the Peer. The ‘provider’ TLV MAY be included in the message even if the Peer did not make a selection.

After that, the Peer sends its first message to the Server by sending the EAP-Response/EAP-CREDS(Type=ProtoFlow) message. This message contains the selected provisioning protocol’s message data and some extra fields (e.g., transport-protocol headers) in the Protocol-Data and Protocol-Headers TLVs respectively.

The Server replies to the Peer’s message with EAP-Request/EAP-CREDS(Type=ProtoFlow) messages until the provisioning protocol reaches an end (the client will send a ‘ProtoFlow’ message with an empty body) or an error condition (the party that detected the error condition will use a ‘ProtoFlow’ message with an ‘Error’ TLV to convey the issue and terminate the protocol).

After the Peer sends the final empty message, the server can start phase three of EAP-CREDS or it can repeat phase two by issuing a new ProtoFlow message with the appropriate TLVs to indicate the target credential and action to perform.

5.5. Phase Three: Validation

The following figure provides the message flow for Phase 3:
Phase three is optional and it is used by the server to request the client to validate (proof) that the new credentials have been installed correctly before issuing the final Success message.

NOTE WELL: Phase Three introduces the dependency on a hashing algorithm to provide an easy way to check the integrity and functionality of a newly installed credentials. This requirement might be updated should the security properties of the selected algorithm be weakened.

In order to do that, the server and the client exchange two messages, i.e. one EAP-Request/EAP-CREDS(Type=Validate) and one EAP-Response/EAP-CREDS(Type=Validate). These messages are used to verify the correctness of the exchanged credentials and the ability by the Peer to use the credentials. The server must include the (‘Credentials-Info’) TLV to provide the indication about which credential the Server intends to validate. The Server MUST also include a randomly generated challenge in the message to the client.

When the client receives the Validate message from the server, it calculates the response to the challenge (based on the type of credentials and the protocol itself) and sends the response back to the server. The Peer MUST calculate the response as follows:

```plaintext
Pala
Expires December 6, 2019
[Page 11]
```
Public-Key - For any public-key based credentials (e.g.,
certificates or raw keys), the response to the challenge is
calculated by generating a signature over the hashed value of the
challenge. The hashing algorithm to be used for this purpose is
specified in Section 7.3.1.

Symmetric Secret - For any symmetric based credentials (e.g.,
password or Key), the response to the challenge is calculated by
using the selected hash function Section 7.3.1 on the
concatenation of (a) the value carried in the corresponding
(‘Challenge-Data’) TLV, and (b) the secret value itself.

Optionally, the client can optionally add a new (‘Challenge-Data’) TLV in its reply to request that the network proves it can calculate responses. This extra challenge is only available for symmetric secrets (it is not used to verify again the identity of the server, but it is used for the server to prove it does have the right symmetric data on its side). This ‘extra’ challenge carries the Peer-Generated challenge information that the server MUST use to calculate the corresponding (‘Challenge-Response’) TLV.

When the server receives the EAP-Response/EAP-CREDS(Type=Validate) with the Challenge TLV in it, it MUST calculate the response to the challenge and send back the response to the client in an EAP-Request/ EAP-CREDS(Type=Validate) with the ChallengeResponse TLV.

In case of issues with the validation of the newly deployed credentials, both the client and the server should consider those credentials invalid (or unusable) and should issue the required failure message(s).

6. EAP-CREDS Message Format

The EAP-CREDS defines the following message types:

- EAP-CREDS/Init
- EAP-CREDS/ProtoFlow
- EAP-CREDS/Validate

Each of these message types have the basic structure as identified in
Section 6.1 and in Section 6.2 and contain zero, one, or more TLVs. The allowed TLVs for the different types of messages are described in
Section 7. The internal structure of the different types of TLVs is
described in Section 6.2.1.
6.1. Message Header

The EAP-CREDS messages consist of the standard EAP header (see Section 4 of [RFC3748]), followed by the version of the EAP-CREDS (4 bits) and a field (4 bits) reserved for future use. The header has the following structure:

```
  +---------------+---------------+---------------+---------------+---------------+
  |     Code      |  Identifier   |             Length            |
  +---------------+---------------+---------------+---------------+
  |     Type      | Flags |  Ver  |         Message Length        |
  +---------------+---------------+---------------+---------------+
  |         Message Length        |               Data           ...
  +---------------+---------------+---------------+---------------+
```

Where the Code, Identifier, Length, and Type fields are all part of the EAP header as defined in [RFC3748].

The Type field in the EAP header is <TBD> for EAP-CREDS.

The Ver (Version) bitfield (4 bits) identifies the version of the EAP-CREDS protocol used for the exchange. This document defines the value to use for this field as '0x1' (0001).

The Flags bitfield (4 bits) is currently not used and it is reserved for future use. The value of the bitfield shall be ignored when the Version ('Ver') field is set to '0x1' (0001).

6.2. Message Payload

The Data part of the message is organized as zero, one, or more TLV objects whose structure is defined in this section. In particular, the general structure of a TLV is described in Section 6.2.1, while the specific structures for the supported TLVs is provided in Section 6.2.2.

6.2.1. General TLV format

Each TLV object has the same basic structure that is defined as follows:
TLV-Type (uint8)

This field is used to indicate the type of data that the TLV carries. The type of TLV determines its internal structure. The supported values for this field are provided in the following table:

Length (uint24)

This field carries the size of the value of the TLV. In particular, the overall size of a TLV (i.e., the header plus the value) can be calculated by adding the size of the header (6 octets) to the value of the Length field (i.e., the size of the TLV’s value).

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Action TLV</td>
</tr>
<tr>
<td>1</td>
<td>Challenge-Data TLV</td>
</tr>
<tr>
<td>2</td>
<td>Challenge-Response TLV</td>
</tr>
<tr>
<td>3</td>
<td>Credentials-Data TLV</td>
</tr>
<tr>
<td>4</td>
<td>Credentials-Info TLV</td>
</tr>
<tr>
<td>5</td>
<td>Error TLV</td>
</tr>
<tr>
<td>6</td>
<td>Profile TLV</td>
</tr>
<tr>
<td>7</td>
<td>Protocol TLV</td>
</tr>
<tr>
<td>8</td>
<td>Provisioning-Headers TLV</td>
</tr>
<tr>
<td>9</td>
<td>Provisioning-Data TLV</td>
</tr>
<tr>
<td>10</td>
<td>Provisioning-Params TLV</td>
</tr>
<tr>
<td>11</td>
<td>Token-Data TLV</td>
</tr>
<tr>
<td>12</td>
<td>Version TLV</td>
</tr>
</tbody>
</table>

Table 2: EAP-CREDS Supported TLVs Types

TLV Value ( > 1 octet )
This field carries data for the identified TLV. The internal structure is determined by the TLV Type field.

6.2.2. EAP-CREDS defined TLVs

EAP-CREDS messages’s payload comprises zero, one, or more TLVs that are encoded in a single EAP-CREDS message. The values for the TLV Type that are supported by this specifications are listed in Table 2.

This section describes the structure of the different supported TLVs and their usage in the different messages.

6.2.2.1. The Action TLV

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TLV Type    |     Flags     |          Action Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TLV Type (uint8)

<TBD> - Action TLV

Flags (uint8)

Reserved

Action Type (uint16)

Provides the indication of the type of action to be performed in the current run of the phase two of EAP-CREDS. The values for this field are provided in Table 11.

Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Table 10.

Value (octet string)

This field carries the data for the credentials.
6.2.2.2. The Certificate-Data TLV

![TLV structure diagram]

TLV Type (uint8)

<TBD> - Certificate-Data TLV

Length (uint24)

Provides the length of the TLV (> 3 octets)

Flags (uint8)

Provides a BITMASK that can be used to provide additional information related to the encapsulated certificate. The bits have the following meaning:

- Bit 0 - If set, the certificate is trusted (Trust Anchor)
- Bit 1 - If set, the certificate is a CA certificate
- Bit 2 - If set, the certificate is self-signed
- Bit 3 - If set, the certificate is a proxy certificate
- Bit 4 - If set, the certificate is an attributes certificate
- Bit 5 - Reserved
- Bit 6 - Reserved
- Bit 7 - Reserved

For a Trusted Root CA, the value of the flags shall be 0x7 (0000 0111). For an intermediate CA certificate that is not implicitly trusted, the value of the flags field should be set to 0x02 (0000 0010). For an End-Entity certificate, the value of the Flags will be 0x0 (0000 0000).
Format (uint8)

Provides the indication of the Format the certificate is in. The allowed values for this field are listed in Section 12.5.

Encoding (uint8)

Provides the indication of the Encoding the certificate is in. The allowed values for this field are listed in Table 10.

Value (octet string)

This field carries the data for the certificate.

6.2.2.3. The Challenge-Data TLV

TLV Type (uint8)

<TBD> - Challenge-Data TLV

Length (uint24)

3 octets

Challenge Data (> 1 octet)

This field carries the data to be used as a challenge when validating newly deployed credentials

6.2.2.4. The Challenge-Response TLV
TLV Type (uint8)

<TBD> - Challenge-Response TLV

Length (uint24)

3 octets

Challenge Response (> 1 octet)

This field carries the data that resulted from the use of the credentials to be validated.

6.2.2.5. The Credentials-Information TLV
The Credential-Information TLV is used by the Peer to provide a description of the installed credentials that are relevant for the network that is being accessed. The TLV is also used by the server to provide indication to the Peer about the required parameters when requesting/updating credentials.

For example, when a set of credentials need to be renewed, the server checks the CredInfo from the Peer and eventually selects the right one for renewal - in the first message of Phase Two, the CredInfo TLV is used by the server to convey (let’s say that the credentials to renew are X.509 based) the requirements for generating the X.509 PKCS#10 request, such as the algorithm (RSA, ECDSA, etc.) further parameters (e.g., the specific curve to use or the Key Size) and the indication if the credentials are to be generated on the server or on the client.

The TLV structure is as follows:

**TLV Type (uint8)**

<TBD> - Credentials-Information TLV

**Length (uint24)**

Provides the total length of the body of the Credential-Information TLV.

**Flags (uint8)**

Provides a BITMASK that can be used to provide information about the status of the credentials (e.g., if the use marks the credentials to be compromised). The bits have the following meaning:

- Bit 0 - If set, the credential is marked as compromised
- Bit 1 - If set, the credential is immutable and cannot be updated
- Bit 2 - Private Key or Secret Immutable, the public part of the credential (e.g., a certificate) can still be updated
- Bit 3 - If set, the credential cannot be updated (both public and private parts)
- Bit 4 - If set, the credential is ready to be used
- Bit 5 - If set, the credential was generated on the server
Bit 6 - If set, the Peer would like to update the credential even if they are not expired

Bit 7 - Reserved

CredType (uint8)

This field provides the description of the type of credential. The type of credentials are listed in Table 7

ProtoID (uint16)

This field indicates the protocol that was used to retrieve the target credential. When the TLV is used in a Request by the Server, this field specifies which protocol should be used for its management. The values for this field are listed in Table 5.

IssuedOn (16 octets)

This field carries the GMT date for when this credential was issued. This field is 16 bytes long (the last byte must be set to '0x00') and contains the NULL-terminated ASCII string that represents the timestamp where the credential was issued. When the value is not set, the field should be set to { 0x00 }. The format of the string is as follows:

YYYYMMDDHHmmssZ

Where:

YYYY - is the 4 digits representation of the year

MM - is the 2 digits representation of the month

DD - is the 2 digits representation of the day of the month

HH - is the 2 digits representation of the hour of the day (24 hour format)

mm - is the 2 digits representation of the minutes of the hour

ss - is the 2 digits representation of the seconds of the minute

Z - is the character ‘Z’
ExpiresOn (16 octets)

This field carries the GMT date for when this credential is to be considered expired. This field is 16 bytes long (the last byte must be set to ‘0x00’) and contains the NULL-terminated ASCII string that represents the timestamp where the credential was issued. The format is the same as the (‘IssuedOn’) field. When the value is not set, the field should be set to { 0x00 }.

Credentials Length (uint16)

Length (in bytes) of the Credentials value. When used with a public-key type of credentials, this is the size of the key (e.g., for an RSA 2048 bit keys, this field should carry the value of 256). When used with a symmetric secret, this field carries the size of the secret (in bytes).

CredIDValue (> 1 octet)

The binary value of the credentials’ identifier. This identifier can be the binary value of the SHA-256 calculated over the certificate, a username, or it could be a simple random handle. As long as the ID allows the server to uniquely (in its context) identify the credentials, the value of this field can be calculated in any way.

6.2.2.6. The Credentials-Data TLV

TLV Type (uint8)

<TBD> - Credentials-Data TLV

Length (uint24)

Provides the length of the TLV (> 3 octets)

Cred Type (uint8)
Provides the indication of the type of credentials. The allowed values for this field are listed in Table 7.

Format (uint8)

Provides the indication of the Format the credentials are in. The allowed values for this field are listed in Section 12.5.

Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Table 10.

Value (octet string)

This field carries the data for the credentials.

6.2.2.7. The Error TLV

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+-------------------------------------------------+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TLV Type</td>
<td>TLV Length</td>
<td></td>
</tr>
<tr>
<td>+-------------------------------------------------+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EAP-CREDS Error Code</td>
<td>Secondary Error Code</td>
<td></td>
</tr>
<tr>
<td>+-------------------------------------------------+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>+-------------------------------------------------+</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TLV Type (uint8)

<TBD> - Challenge-Response-Data TLV

Length (uint24)

3 octets

EAP-CREDS Error Code (2 octets)

This field carries the EAP-CREDS error code. These code are related to the EAP-CREDS operations only and it should not be used to carry the Provisioning-Protocol specific error codes.

Secondary Error Code (2 octets)
This field is used to convey an error at the encapsulation layer (i.e., the provisioning protocol error). Do not use this field to convey EAP-CREDS specific errors.

Description ( > 1 octet)

The Description field is optional (i.e., when the Description Size is set to zero) and carries information about the error that occurred. The message may or may not be used by a user or an automated process for debugging purposes.

6.2.2.8. The Profile TLV

TLV Type (uint8)

<TBD> - Profile Identifying Data TLV

Length (uint24)

4 octets

Profile Identifying Data (octet string)

The Profile Identifying Data is used to provide indication to the other party about which profiles are supported when requesting credentials management.

Also in this case, the data used in this field is left to be interpreted by the end-point and it is orthogonal to EAP-CREDS data types.

An example of values for this field, an end-point could use the string representation (i.e., dotted representation) of the Object Identifier (OID) of the specific profile supported (e.g., could be defined in the Certificate Policy of the credentials’ provider).
6.2.2.9. The Protocol TLV

<table>
<thead>
<tr>
<th>TLV Type</th>
<th>Protocol ID</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;TBD&gt;</td>
<td>&lt;TBD&gt;</td>
<td>&lt;TBD&gt;</td>
</tr>
</tbody>
</table>

TLV Type (uint8)

<TBD> - Protocol TLV

Protocol ID (uint16)

The Protocol ID value carries the id of a supported provisioning protocol. The initial values for the provisioning protocol identifiers can be found in Section 12.1.

Version

The Version (Protocol Version) value represents the specific version of the identified provisioning protocol. When no version is specified for a protocol (i.e., either it does not support multiple versions or it does not matter), the value of this field should be set to ‘0x0’.

6.2.2.10. The Provisioning-Data TLV

<table>
<thead>
<tr>
<th>TLV Type</th>
<th>TLV Length</th>
<th>Provisioning Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;TBD&gt;</td>
<td>&lt;TBD&gt;</td>
<td>&lt;TBD&gt;</td>
</tr>
</tbody>
</table>

TLV Type (uint8)

<TBD> - Provisioning-Data TLV

Length (uint24)

3 octets
Headers Data (> 1 octet)

This field carries the provisioning protocol’s messages.

6.2.2.11. The Provisioning-Headers TLV

TLV Type (uint8)

<TBD> - Provisioning-Headers TLV

Length (uint24)

3 octets

Headers Data (> 1 octet)

This field carries the meta-data (if any) that might be associated with the transport-layer normally used with the provisioning protocol. For example, this TLV can carry the set of HTTP headers required by EST or ACME.

6.2.2.12. The Provisioning-Params TLV

TLV Type (uint8)

<TBD> - Provisioning-Params TLV
Length (uint24)

Provides the length of the TLV (>= 4 octets)

Min Length (uint16)

Provides the indication minimum size of the credentials. This value has meaning depending on the context of the credentials, however sizes are always expressed in bytes.

For example, when used with a symmetric key or a password, the ('Min Length') and ('Max Length') refer to the minimum and maximum size of the password data. The ('Algor OID') field can be omitted in this case.

On the other hand, when referring public-key credentials, this field should carry the size of the modulus of the key. For example, for an RSA 2048 bit keys, the field should carry the value of 256. For an ECDSA that uses the prime256v1 curve, this field should carry the value of 32 and the Algor OID should be the DER representation of the specific value of the curve (i.e., the DER representation of ‘1.2.840.10045.3.1.7’).

Max Length (uint16)

Provides the indication maximum size of the credentials. This value has meaning depending on the context of the credentials, however sizes are always expressed in bytes.

The same considerations apply to this field as well as the ('Min Length') one discussed above.

Flags (uint8)

Provides a BITMASK that can be used to provide information about the status of the credentials (e.g., if the use marks the credentials to be compromised). The bits have the following meaning:

Bit 0 - Credentials (or part of it) are to be generated on the server

Bit 1 - Credentials (or part of it) are to be generated on the peer

Bit 2 - Reserved

Bit 3 - Reserved
Bit 4 - Reserved
Bit 5 - Reserved
Bit 6 - Reserved
Bit 7 - Reserved

When using public-key based credentials, the bits 0 and 1 are mutually exclusive.

When using passwords or shared secrets, if bit 0 is set, then the secret is generated by the server and then sent to the client. On the other hand, if bit 1 is set, then the secret is generated by the peer and then sent to the server. Ultimately, if both bits are set, then the Peer generates the first part of the password and sends it to the Server, while the Server generates the second part of the password and sends it to the Peer. The password to be used for future authentication is the concatenation of the two shares of the password: first the one from the Server, then the one from the Client.

Last but not least, since these passwords/secrets are meant to be used in an automated fashion, there is no restriction around the character set to use or their interpretation. Therefore, we suggest to generate full random passphrases (on client and server) to maximize the randomness carried within the secret.

Algorithm (uint8)

Provides the indication of the algorithm used for the generation of the credentials. The allowed values for this field are listed in Table 8.

Object Identifier (binary; > 1 octet)

Provides the indication of additional parameters that need to be encoded for the credentials. This value is used only when the credentials use public-key cryptography - this field carries additional information about the algorithm to be used. We provide some useful values that can be used as reference:
<table>
<thead>
<tr>
<th>OID Name</th>
<th>Dotted Representation</th>
<th>Binary Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>secp256r1</td>
<td>1.2.840.10045.3.1.7</td>
<td>06 08 2A 86 4B CE 3D 03</td>
</tr>
<tr>
<td>curve</td>
<td></td>
<td>01 07</td>
</tr>
<tr>
<td>secp384r1</td>
<td>1.2.840.10045.3.1.34</td>
<td>06 08 2A 86 4B CE 3D 03</td>
</tr>
<tr>
<td>curve</td>
<td></td>
<td>01 22</td>
</tr>
<tr>
<td>secp521r1</td>
<td>1.2.840.10045.3.1.35</td>
<td>06 08 2A 86 4B CE 3D 03</td>
</tr>
<tr>
<td>curve</td>
<td></td>
<td>01 23</td>
</tr>
<tr>
<td>X25519 curve</td>
<td>1.3.101.110</td>
<td>06 03 2B 65 6E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>06 03 2B 65 6E</td>
</tr>
<tr>
<td>X448 curve</td>
<td>1.3.101.111</td>
<td>06 03 2B 65 6F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>06 03 2B 65 70</td>
</tr>
<tr>
<td>X25519 curve</td>
<td>1.3.101.112</td>
<td>06 03 2B 65 70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>06 03 2B 65 71</td>
</tr>
<tr>
<td>X448 curve</td>
<td>1.3.101.113</td>
<td>06 03 2B 65 71</td>
</tr>
</tbody>
</table>

Table 3: Object Identifiers Examples

6.2.2.13. The Token-Data TLV

| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 |
| +-------------------------------------------+|
| TLV Type | TLV Length | |
| +-------------------------------------------+|
| Token Type | Encoding | Value | ... |
| +-------------------------------------------+

TLV Type (uint8)

<TBD> - Token-Data TLV

Length (uint24)

Provides the length of the TLV (> 3 octets)

Token Type (uint8)

Provides the indication of the type of credentials. The allowed values for this field are listed in Table 6.

Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Table 10.

Pala Expires December 6, 2019 [Page 28]
Value (octet string)

This field carries the data for the credentials.

6.2.2.14. The Version TLV

```
0                   1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TLV Type    |   Version     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TLV Type (uint8)

<TBD> - Version TLV

Version (uint8)

The Version field is a uint8 value and represents the specific version of the EAP-CREDS protocol that are supported by the end point. When multiple versions of EAP-CREDS are supported, multiple (‘Version’) TLVs can be used.

When no version is specified (i.e., either it does not support multiple versions or it does not matter), the value of this field should be set to ‘0x0’.

7. EAP-CREDS Messages

This section describes each message and what TLVs are allowed or required. EAP-CREDS defines the following values for the Message Type (Type):

```
+------------+---------------------+--------------------------------+
|  Message   | Name                 | Description                    |
|    Type    |                     |                                |
+------------+---------------------+--------------------------------+
|     0      | EAP-CREDS-Init      | Initialization Phase           |
|     1      | EAP-CREDS-ProtoFlow | Carries Provisioning Protocol  |
|            |                     | Messages                       |
|     2      | EAP-CREDS-Validate  | Validates newly installed      |
|            |                     | credentials                    |
+------------+---------------------+--------------------------------+
```

Table 4: EAP-CREDS Message Types
7.1. The EAP-CREDS-Init Message

The EAP-CREDS-Init message type is used in Phase One only of EAP-CREDS. The message flow is depicted in Section 5.3. This message supports the following TLVs: Version, Protocol, Credentials-Info, and Error.

7.1.1. EAP Server’s Init Message

EAP-CREDS starts with an (‘EAP-CREDS-Init’) message from the server. This message MAY contain zero, one, or more (‘Version’) TLVs and, optionally, a (‘Challenge-Data’) TLV.

The Server uses one or more (‘Version’) TLVs in the EAP-Request/EAP-CREDS(Type=Init) message to provide the Peer with the list of EAP-CREDS versions supported. If omitted, the implicit version of EAP-CREDS used in the session is one (‘0x1’). If the Server detects multiple occurrences of this TLV in the reply from the Peer, an error shall be issued and the EAP-CREDS session should be terminated.

In case Token-Based registration is enabled on the Server, the Server SHOULD include, in its Init message, a (‘Challenge-Data’) field that can be used by the client to provide reply protection, and (unique) challenge data for proof-of-possession of secrets.

7.1.2. EAP Peer’s Init Message

The Peer MUST reply to the Server’s (‘EAP-CREDS-Init’) message with its own (‘EAP-CREDS-Init’) one. The Peer one Version TLV in the EAP-Response/EAP-CREDS(Type=Init) message to indicate the version of EAP-CREDS that the client wants to use for the session. The Peer MUST also provide the list of supported provisioning protocols (via the (‘Protocol’ TLV), the list and status of the installed credentials (via the (‘Credentials-Info’ TLV). The Peer MAY include authorization data when registering new credentials (e.g., an authorization token or a device certificate).

The Peer MUST include one (‘Credentials-Info’) TLV for each credential the Network is authorized to manage. Typically, a Peer will include only one (‘Credentials-Info’) TLV in its (‘EAP-CREDS-Init’) message, but there might be cases where multiple types of credentials are available (e.g., X.509 certificate and username/password combination) and used depending on the location and other factors.

In case the Peer does not have any credentials available yet, it does not add any (‘Credentials-Info’) TLV – leaving the Server with the only action possible: Registration.
7.1.2.1. Bootstrapping Peer’s Trustworthiness

The Registration process might rely on information exchanged during the Provisioning Process in Phase Two, however, if an authorization mechanism is not available, EAP-CREDS provides a simple mechanism for the Peer to leverage an out-of-band token/passphrase/ott that is available (or can easily be made available) on the Peer and that can be verified by the Server.

In particular, when the Peer wants to register new credentials (and the Server requires the use of authorization because, for example, the Peer does not have any credentials yet) it may need to provide (a) a Token, (b) a challenge value, and (c) a response to the challenge value. To do so, the Peer MUST encode the token in a (‘Token-Data’) TLV, the challenge value in a (‘Challenge-Data’) TLV, and, finally, the response to the challenge in the (‘Challenge-Response’) TLV.

The use of (‘Challenge-Data’) and (‘Challenge-Response’) TLVs is optional, however it is suggested that if a token is used for bootstrapping the trust, it should provide a way to verify a secret associated with it.

It is also very important that the authorization token is disclosed only to authorized servers - the Peer MUST NOT disclose authorization tokens that are not meant for the network that is being accessed. This can be done, usually, by verifying the identity of the Server first (in the outer mechanism) and then verify that the target of the Token is the Server the Client is talking to.

7.2. The EAP-CREDS-ProtoFlow Message

The EAP-CREDS-ProtoFlow message type is used in Phase Two only of EAP-CREDS. The message flow is depicted in Section 5.4. This message type supports the following TLVs: Protocol, Credentials-Info, Provisioning-Headers, Provisioning-Data, Token-Data, and Error.

After the exchange of the initial two (‘EAP-CREDS-Init’) messages, the Server MAY start phase two by issuing an (‘EAP-CREDS-ProtoFlow’) message for the Peer where it encodes all the required details for starting the provisioning process. In particular, the server sends the selected (‘Action’), (‘Protocol’), (‘Credentials-Info’), and metadata to the client in a EAP-Request/EAP-CREDS(Type=ProtoFlow) message.

NOTE WELL: After the initial message, the only TLVs that are allowed in messages coming from the server are the usual (‘Provisioning-Headers’) (‘Provisioning-Data’), and (‘Error’).
The client checks that all the selected parameters are supported for the selected credentials and, if no errors are detected, it sends its first ('EAP-CREDS-ProtoFlow') message to the Server with the ('Provisioning-Headers') and ('Provisioning-Data') TLVs only.

From now on, the conversation between the Peer and the Server continues until an error is detected or the provisioning protocol completes successfully (the client sends an empty ('EAP-CREDS-ProtoFlow') message to the Server to indicate that the protocol completed successfully).

If no other actions, the server MAY continue with phase three or issue a success message and terminate the EAP session.

NOTE WELL: When the SPP protocol is used, the protocol messages that are encoded inside the ('Protocol-Data') TLV are composed of sets of TLVs as defined in this document. The overall message size is provided by the size of the ('Protocol-Data') TLV that encapsulates the SPP-specific TLVs. This design choice provides symmetry in implementing support for SPP when compared to other (available) provisioning protocols.

7.3. The EAP-CREDS-Validate Message

7.3.1. Algorithm Requirements

EAP-CREDS uses the SHA-256 hashing algorithm to verify credentials in phase three of the protocol. Peers and Servers MUST support SHA-256 for this purpose.

8. Error Handling in EAP-CREDS

This section provides a description of the error handling by using the CREDS-Error-TLV in a CREDS message.

9. Fragmentation

Although EAP does not directly support handling of fragmented packets, EAP-CREDS requires that its messages are encapsulated via an outer method that MUST provide fragmentation support.

Because of the outer method requirements in particular, removing any support for fragmented messages in EAP-CREDS removes the duplication of packets (e.g., Acknowledgment Packets) sent across the Peer and the Server, thus resulting in a smaller number of exchanged messages.
10. Encapsulating Provisioning Protocols in EAP-CREDS

In order to use EAP-CREDS together with your favorite provisioning protocol, the messages from the provisioning protocol need to be sent to the other party. In EAP-CREDS, this is done by putting the provisioning protocol messages inside the ('Provisioning-Data') TLV. In case the provisioning protocol uses additional data for its operations (e.g., uses HTTP Headers), this data can be encoded in a separate ('Provisioning-Headers') TLV.

Since the implementation of the provisioning endpoint could happen in a (logically or physically) different component, a method is needed to identify when a provisioning protocol has actually ended. In EAP-CREDS, when the provisioning protocol is over, when an empty message is sent by the Server (not the Peer).

In the first message of Phase Two, the Server provides the client with all the selected parameters for one specific credential that needs attention (or for a new credential) to be registered with the network. In particular, the server provides the ('Version') TLV (optional), the ('Protocol') TLV, the ('Action') TLV, and the ('Provisioning-Params') TLV.

After checking the parameters sent by the Server, if the Peer does not support any of them, it MUST send a message with one single ('Error') TLV with the appropriate error code(s). The server, can then decide if to manage a different set of credentials (if more were reported by the Peer in its Phase One message) or if to terminate the EAP session with an error.

The Peer and the Server exchange ProtoFlow messages until an error is detected and the appropriate error message is sent to the other party. If the error message was sent by the Peer, then the Server MAY send an empty message to the Peer and terminate the session, or it MAY try to restart Phase Two for a different set of credentials or with a different action (same target and action MUST not be repeated in the same EAP-CREDS session).

11. EAP-CREDS and Simple Provisioning Protocol (SPP)

EAP-CREDS supports a Simple Provisioning Protocol (SPP) which comprises a series of messages that enables the management not only of certificates, but also of other types of credentials like username/password pairs, asymmetric keys, and symmetric keys.

The Simple Provisioning Protocol (SPP), described in this section, behaves as any other provisioning protocol: its messages are
encapsulated in the ('Provisioning-Data') TLVs in the second phase of the protocol.

When no ('Credentials-Info') TLVs have been provided by the client, the Server knows that the device does not have valid credentials it wants to use to access the Network. In this case, EAP-CREDS/SPP supports the use of Tokens to kick-off the registration process. The type, format, or encoding of the Token is orthogonal to EAP-CREDS/SPP which treats the token as a black-box field (i.e., it does not try to interpret or parse its contents).

NOTE WELL: During Phase One, the Peer MAY include the ('Token-Data') TLV in its EAP-CREDS-Init message to provide the needed authorization to register a new set of credentials. The Server might not allow the registration of new credentials if the required authorization (i.e., the Token) was not provided during the initialization phase.

In the case where an authorization Token MUST be used and it is not just a bearer token but it can be used (or a secret identified in it) to generate a verifiable proof-of-possession, the Peer can include a ('Challenge-Data') and a ('Challenge-Response') TLVs.

The ('Challenge-Data') TLV MUST be used to convey the challenge data (usually some random value), if any. If the Server provided a ('Challenge-Data') TLV, then the Peer MUST use the same TLV in its message (and to calculate the response).

The ('Challenge-Response') TLV is used, instead, to encode the response to the challenge data that can be generated by the Peer and verified by the Server.

NOTE WELL: The use of the ('Token-Data'), ('Challenge-Data'), and ('Challenge-Response') TLVs in the Peer’s Init message can be used to bootstrap trust between the server and the client.

11.1. SPP Message Format

The SPP Messages are constructed with zero, one, or more TLVs and encoded in the ('Provisioning-Data') TLV. The size of the encapsulating ('Provisioning-Data') TLV provides the size of the whole message.

11.2. SPP: Registering New Credentials
11.3. SPP: Renewing Credentials

11.4. SPP: Removing Credentials

11.5. SPP Password Provisioning

EAP-CREDS/SPP defines the following flow of messages for provisioning symmetric secrets. This method can be used to deliver a username/password combination, a token, or a symmetric key. The message flow is provided.

EAP-CREDS/SPP provides the possibility for shared secret to be generated in different ways:

1. Server-Side Generated
2. Client-Side Generated
3. Both Sides Generate a share

In particular, when initiating the second phase of the protocol, the (‘Provisioning-Params’) TLV is used to specify how to generate the secret.

11.6. SPP Certificate Provisioning

EAP-CREDS/SSP defines the following flow of messages for requesting the provisioning of credentials.

12. IANA Considerations

This document uses a new EAP type, EAP-CREDS, whose value (TBD) MUST be allocated by IANA from the EAP TYPEs subregistry of the RADIUS registry. This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the EAP-CREDS protocol, in accordance with [RFC8126].

The EAP Method Type number for EAP-CREDS needs to be assigned.

This document also requires IANA to create new registries as defined in the following subsections.
12.1. Provisioning Protocols

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unspecified</td>
</tr>
<tr>
<td>1</td>
<td>Simple Provisioning Protocol (SPP)</td>
</tr>
<tr>
<td>2</td>
<td>Basic Certificate Management Protocol (CMP-S)</td>
</tr>
<tr>
<td>3</td>
<td>Full Certificate Management Protocol (CMP-F)</td>
</tr>
<tr>
<td>4</td>
<td>Enrollment over Secure Transport (EST)</td>
</tr>
<tr>
<td>5</td>
<td>Certificate Management over CMS (CMC)</td>
</tr>
<tr>
<td>6</td>
<td>Automatic Certificate Management Environment</td>
</tr>
<tr>
<td></td>
<td>(ACME)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>49141 ... 65534</td>
<td>Vendor Specific</td>
</tr>
</tbody>
</table>

Table 5: EAP-CREDS Inner Protocol Identifiers

Assignment of new values for new cryptosuites MUST be done through IANA with "Specification Required" and "IESG Approval" as defined in [RFC8126].

12.2. Token Types

<table>
<thead>
<tr>
<th>Token Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unspecified</td>
</tr>
<tr>
<td>1</td>
<td>JWT</td>
</tr>
<tr>
<td>2</td>
<td>Kerberos</td>
</tr>
<tr>
<td>3</td>
<td>OAuth</td>
</tr>
<tr>
<td>200..254</td>
<td>Vendor Specific</td>
</tr>
</tbody>
</table>

Table 6: Token Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

12.3. Credentials Types
<table>
<thead>
<tr>
<th>Credentials Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X.509 Certificate</td>
</tr>
<tr>
<td>1</td>
<td>Public Key</td>
</tr>
<tr>
<td>2</td>
<td>Symmetric Key</td>
</tr>
<tr>
<td>3</td>
<td>Username and Password</td>
</tr>
<tr>
<td>4</td>
<td>AKA Subscriber Key</td>
</tr>
<tr>
<td>5</td>
<td>Bearer Token</td>
</tr>
<tr>
<td>6</td>
<td>One-Time Token</td>
</tr>
<tr>
<td>7</td>
<td>API Key</td>
</tr>
</tbody>
</table>

Table 7: Credentials Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

12.4. Credentials Algorithms

<table>
<thead>
<tr>
<th>ID</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>RSA</td>
</tr>
<tr>
<td>2</td>
<td>ECDSA</td>
</tr>
<tr>
<td>3</td>
<td>XMMS</td>
</tr>
<tr>
<td>4</td>
<td>AKA Subscriber Key</td>
</tr>
<tr>
<td>5</td>
<td>OAuth</td>
</tr>
<tr>
<td>6</td>
<td>Kerberos4</td>
</tr>
<tr>
<td>7</td>
<td>Kerberos5</td>
</tr>
<tr>
<td>200-254</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 8: Credentials Algorithms

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

12.5. Credentials Datatypes
Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

12.6. Credentials Encoding

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

12.7. Action Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].
Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

13. Security Considerations

Several security considerations need to be explicitly considered for the system administrators and application developers to understand the weaknesses of the overall architecture.

The most important security consideration when deploying EAP-CREDS is related to the security of the outer channel. In particular, EAP-CREDS assumes that the communication channel has been properly authenticated and that the information exchanged between the Peer and the Server are protected (i.e., confidentiality and integrity).

For example, if certificate-based authentication is used, the server presents a certificate to the peer as part of the trust establishment (or negotiation). The peer SHOULD verify the validity of the EAP server certificate and SHOULD also examine the EAP server name presented in the certificate in order to determine whether the EAP server can be trusted. When performing server certificate validation, implementations MUST provide support for the rules in [RFC5280] for validating certificates against a known trust anchor.

14. Acknowledgments

The authors would like to thank everybody who provided insightful comments and helped in the definition of the deployment considerations.

15. Normative References


Appendix A.  EAP-CREDS Example Message Flow for Provisioning Standards

A.1.  EAP-CREDS and CMP

Describe how to use EAP-CREDS with CMP.

A.2.  EAP-CREDS and EST

Describe how to use EAP-CREDS with EST.

A.3.  EAP-CREDS and CMS

Describe how to use EAP-CREDS with CMS.

A.4.  EAP-CREDS and ACME

Describe how to use EAP-CREDS with ACME.

Author’s Address