

Network Working Group
INTERNET-DRAFT
Updates: 5247
Category: Standards Track
<draft-dekok-emu-eap-session-id-01.txt>
23 July 2019

DeKok, Alan
FreeRADIUS

EAP Session-Id Derivation
draft-dekok-emu-eap-session-id-01.txt

Abstract

EAP Session-Id derivation has not been defined for EAP-SIM, EAP-AKA, and EAP-AKA' when using the fast re-authentication exchange instead of full authentication. This document updates [RFC5247] to define those derivations for EAP-SIM, and EAP-AKA. Since [AKAP] defines the Session-ID for EAP-AKA', the definition for EAP-AKA' is not included here. [RFC5247] also does not define Session-Id derivation for PEAP. A definition is given here which follows the definition for other TLS-based EAP methods.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on July 22, 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 (<http://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. Introduction	4
1.1. Requirements Language	4
2. Updates to RFC 5247 Appendix A	5
2.1. EAP-AKA	5
2.2. EAP-SIM	5
2.3. Rationale	7
2.4. Session-Id for PEAP	7
3. Security Considerations	7
4. IANA Considerations	8
5. References	8
5.1. Normative References	8
5.2. Informative References	8

1. Introduction

EAP [RFC3748] Session-Id derivation has not been defined for EAP-SIM, EAP-AKA, and EAP-AKA' when using the fast re-authentication exchange instead of full authentication. [RFC5247] defines the Session-Id for these EAP methods, but that derivation is only applicable for the full authentication case.

The IEEE is defining FILS authentication [FILS], which needs the EAP Session-Id for in order for the EAP Re-authentication Protocol (ERP) [RFC5296] to work, it would be important to get this resolved with a clearly defined and agreed derivation rules to allow fast re-authentication cases to be used to derive ERP key hierarchy.

Further, [RFC5247] did not define Session-Id for PEAP. We correct that deficiency here.

1.1. 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.

2. Updates to RFC 5247 Appendix A

This section updates [RFC5247] Appendix A to define Session-Id for fast re-authentication exchange for EAP-AKA and EAP-SIM. It further defines Session-ID derivation for PEAP.

2.1. EAP-AKA

For EAP-AKA, [RFC5247] Appendix A says:

EAP-AKA

EAP-AKA is defined in [RFC4187]. The EAP-AKA Session-Id is the concatenation of the EAP Type Code (0x17) with the contents of the RAND field from the AT_RANDOM attribute, followed by the contents of the AUTN field in the AT_AUTN attribute:

$$\text{Session-Id} = 0x17 \parallel \text{RAND} \parallel \text{AUTN}$$

It should say:

EAP-AKA

EAP-AKA is defined in [RFC4187]. When using full authentication, the EAP-AKA Session-Id is the concatenation of the EAP Type Code (0x17) with the contents of the RAND field from the AT_RANDOM attribute, followed by the contents of the AUTN field in the AT_AUTN attribute:

$$\text{Session-Id} = 0x17 \parallel \text{RAND} \parallel \text{AUTN}$$

When using fast re-authentication, the EAP-AKA Session-Id is the concatenation of the EAP Type Code (0x17) with the contents of the NONCE_S field from the AT_NONCE_S attribute, followed by the contents of the MAC field from the AT_MAC attribute from EAP-Request/AKA-Reauthentication:

$$\text{Session-Id} = 0x17 \parallel \text{NONCE_S} \parallel \text{MAC}$$

2.2. EAP-SIM

Similarly for EAP-SIM, it says:

EAP-SIM

EAP-SIM is defined in [RFC4186]. The EAP-SIM Session-Id is the concatenation of the EAP Type Code (0x12) with the contents of the

RAND field from the AT_RANDOM attribute, followed by the contents of the NONCE_MT field in the AT_NONCE_MT attribute:

Session-Id = 0x12 || RAND || NONCE_MT

The Peer-Id is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length octets from the beginning, however. Note that the contents are used as they are transmitted, regardless of whether the transmitted identity was a permanent, pseudonym, or fast EAP re-authentication identity. The Server-Id is the null string (zero length).

It should say:

EAP-SIM

EAP-SIM is defined in [RFC4186]. The EAP-SIM Session-Id is the concatenation of the EAP Type Code (0x12) with the contents of the RAND field from the AT_RANDOM attribute, followed by the contents of the NONCE_MT field in the AT_NONCE_MT attribute. RFC 4186 says that EAP server should obtain "n" GSM triplets where "n=2" or "n=3".

For "n=2", the Session-Id is therefore defined as

Session-Id = 0x12 || RAND1 || RAND2 || NONCE_MT

which is 49 octets in length.

For "n=3", the Session-Id is therefore defined as

Session-Id = 0x12 || RAND1 || RAND2 || RAND3 || NONCE_MT

which is 65 octets in length.

The Peer-Id is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length octets from the beginning, however. Note that the contents are used as they are transmitted, regardless of whether the transmitted identity was a permanent, pseudonym, or fast EAP re-authentication identity. The Server-Id is the null string (zero length).

When using fast re-authentication, the EAP-SIM Session-Id is the concatenation of the EAP Type Code (0x12) with the contents of the NONCE_S field from the AT_NONCE_S attribute, followed by the

contents of the MAC field from the AT_MAC attribute from EAP-Request/AKA-Reauthentication:

Session-Id = 0x12 || NONCE_S || MAC

which is 33 octets in length.

2.3. Rationale

[RFC5247] was supposed to define exported parameters for existing EAP methods in Appendix A. The way Session-Id was defined for EAP-AKA and EAP-SIM works only for the full authentication case, i.e., it cannot be used when the optional fast re-authentication case is used since the used parameters (RAND, AUTN, NONCE_MT) are not used in the fast re-authentication case. Based on [RFC4187] Section 5.2, and similar text in [RFC4186], NONCE_S corresponds to RAND and MAC in EAP-Request/AKA-Reauthentication corresponds to AUTN. That would seem to imply that the Session-Id could be defined using NONCE_S and MAC instead of RAND and AUTN/NONCE_MT.

2.4. Session-Id for PEAP

[RFC5247] did not define Session-Id definition for Microsoft's Protected EAP (PEAP). Similar to the definition in [RFC5216] Section 2.3, we define it as:

Session-Id = 0x19 || client.random || server.random

This definition is already in wide-spread use in multiple PEAP implementations.

Note that this definition for Session-Id only applies when TLS 1.2 or earlier is used. A different derivation is defined for TLS 1.3.

3. Security Considerations

This specification defines EAP Session-Ids for ERP with EAP-SIM and EAP-AKA. It therefore enables ERP key hierarchy establishment using fast re-authentication with EAP-SIM and EAP-AKA.

There are no known security issues from using the NONCE_S and MAC as defined above.

This specification also defines the EAP Session-Id for PEAP. That derivation has no known security issues.

4. IANA Considerations

There are no actions for IANA. RFC EDITOR: This section may be removed before publication.

5. References

5.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

[RFC3748]

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC5216]

Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008

[RFC5247]

Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008,

[RFC5296]

Narayanan, V. and L. Dondeti, "EAP Extensions for EAP Re-authentication Protocol (ERP)", RFC 5296, August 2008.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.

[FELS]

"IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 1: Fast Initial Link Setup", IEEE Std 802.11ai-2016, 2016.

5.2. Informative References

[RFC4186]

Haverinen, H. (Ed), Salowey, J., "Extensible Authentication

Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, January 2006.

[RFC4187]

Arkko, J., Haverinen, H., "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, January 2006.

[AKAP]

Arkko, J., et al, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", draft-arkko-eap-rfc5448bis-04.txt, January 2019.

Acknowledgments

The issue corrected in this specification was first reported by Jouni Malinen in a technical errata at https://www.rfc-editor.org/errata_search.php?rfc=5247

The text in this document follows his suggestions.

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

Network Working Group
Internet-Draft
Updates: 5216 (if approved)
Intended status: Standards Track
Expires: 23 April 2022

J. Preuß Mattsson
M. Sethi
Ericsson
20 October 2021

Using EAP-TLS with TLS 1.3 (EAP-TLS 1.3)
draft-ietf-emu-eap-tls13-21

Abstract

The Extensible Authentication Protocol (EAP), defined in RFC 3748, provides a standard mechanism for support of multiple authentication methods. This document specifies the use of EAP-Transport Layer Security (EAP-TLS) with TLS 1.3 while remaining backwards compatible with existing implementations of EAP-TLS. TLS 1.3 provides significantly improved security and privacy, and reduced latency when compared to earlier versions of TLS. EAP-TLS with TLS 1.3 (EAP-TLS 1.3) further improves security and privacy by always providing forward secrecy, never disclosing the peer identity, and by mandating use of revocation checking, when compared to EAP-TLS with earlier versions of TLS. This document also provides guidance on authentication, authorization, and resumption for EAP-TLS in general (regardless of the underlying TLS version used). This document updates RFC 5216.

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 23 April 2022.

Copyright Notice

Copyright (c) 2021 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. Introduction	3
1.1. Requirements and Terminology	4
2. Protocol Overview	5
2.1. Overview of the EAP-TLS Conversation	5
2.1.1. Authentication	6
2.1.2. Ticket Establishment	7
2.1.3. Resumption	9
2.1.4. Termination	11
2.1.5. No Peer Authentication	14
2.1.6. Hello Retry Request	15
2.1.7. Identity	16
2.1.8. Privacy	17
2.1.9. Fragmentation	18
2.2. Identity Verification	18
2.3. Key Hierarchy	19
2.4. Parameter Negotiation and Compliance Requirements	20
2.5. EAP State Machines	21
3. Detailed Description of the EAP-TLS Protocol	22
4. IANA considerations	22
5. Security Considerations	22
5.1. Security Claims	22
5.2. Peer and Server Identities	23
5.3. Certificate Validation	23
5.4. Certificate Revocation	23
5.5. Packet Modification Attacks	24
5.6. Authorization	25
5.7. Resumption	26
5.8. Privacy Considerations	28
5.9. Pervasive Monitoring	29
5.10. Discovered Vulnerabilities	29
5.11. Cross-Protocol Attacks	30
6. References	30
6.1. Normative References	30
6.2. Informative references	31
Appendix A. Updated References	36
Acknowledgments	36
Contributors	36

Authors' Addresses	36
------------------------------	----

1. Introduction

The Extensible Authentication Protocol (EAP), defined in [RFC3748], provides a standard mechanism for support of multiple authentication methods. EAP-Transport Layer Security (EAP-TLS) [RFC5216] specifies an EAP authentication method with certificate-based mutual authentication utilizing the TLS handshake protocol for cryptographic algorithms and protocol version negotiation and establishment of shared secret keying material. EAP-TLS is widely supported for authentication and key establishment in IEEE 802.11 [IEEE-802.11] (Wi-Fi) and IEEE 802.1AE [IEEE-802.1AE] (MACsec) networks using IEEE 802.1X [IEEE-802.1X] and it's the default mechanism for certificate based authentication in 3GPP 5G [TS.33.501] and MulteFire [MulteFire] networks. Many other EAP methods such as EAP-FAST [RFC4851], EAP-TTLS [RFC5281], TEAP [RFC7170], and PEAP [PEAP] depend on TLS and EAP-TLS.

EAP-TLS [RFC5216] references TLS 1.0 [RFC2246] and TLS 1.1 [RFC4346], but can also work with TLS 1.2 [RFC5246]. TLS 1.0 and 1.1 are formally deprecated and prohibited to negotiate and use [RFC8996]. Weaknesses found in TLS 1.2, as well as new requirements for security, privacy, and reduced latency have led to the specification of TLS 1.3 [RFC8446], which obsoletes TLS 1.2 [RFC5246]. TLS 1.3 is in large parts a complete remodeling of the TLS handshake protocol including a different message flow, different handshake messages, different key schedule, different cipher suites, different resumption mechanism, different privacy protection, and different record padding. This means that significant parts of the normative text in the previous EAP-TLS specification [RFC5216] are not applicable to EAP-TLS with TLS 1.3. Therefore, aspects such as resumption, privacy handling, and key derivation need to be appropriately addressed for EAP-TLS with TLS 1.3.

This document updates [RFC5216] to define how to use EAP-TLS with TLS 1.3. When older TLS versions are negotiated, RFC 5216 applies to maintain backwards compatibility. However, this document does provide additional guidance on authentication, authorization, and resumption for EAP-TLS regardless of the underlying TLS version used. This document only describes differences compared to [RFC5216]. When EAP-TLS is used with TLS 1.3, some references are updated as specified in Appendix A. All message flow are example message flows specific to TLS 1.3 and do not apply to TLS 1.2. Since EAP-TLS couples the TLS handshake state machine with the EAP state machine it is possible that new versions of TLS will cause incompatibilities that introduce failures or security issues if they are not carefully integrated into the EAP-TLS protocol. Therefore, implementations MUST limit the maximum TLS version they use to 1.3, unless later versions are explicitly enabled by the administrator.

This document specifies EAP-TLS 1.3 and does not specify how other TLS-based EAP methods use TLS 1.3. The specification for how other TLS-based EAP methods use TLS 1.3 is left to other documents such as [I-D.ietf-emu-tls-eap-types].

In addition to the improved security and privacy offered by TLS 1.3, there are other significant benefits of using EAP-TLS with TLS 1.3. Privacy, which in EAP-TLS means that no information about the underlying peer identity is disclosed, is mandatory and achieved without any additional round-trips. Revocation checking is mandatory and simplified with OCSP stapling, and TLS 1.3 introduces more possibilities to reduce fragmentation when compared to earlier versions of TLS.

1.1. Requirements and Terminology

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.

Readers are expected to be familiar with the terms and concepts used in EAP-TLS [RFC5216] and TLS [RFC8446]. The term EAP-TLS peer is used for the entity acting as EAP peer and TLS client. The term EAP-TLS server is used for the entity acting as EAP server and TLS server.

This document follows the terminology from [I-D.ietf-tls-rfc8446bis] where the master secret is renamed to the main secret and the exporter_master_secret is renamed to the exporter_secret.

2. Protocol Overview

2.1. Overview of the EAP-TLS Conversation

This section updates Section 2.1 of [RFC5216] by amending it in accordance with the following discussion.

If the TLS implementation correctly implements TLS version negotiation, EAP-TLS will automatically leverage that capability. The EAP-TLS implementation needs to know which version of TLS was negotiated to correctly support EAP-TLS 1.3 as well as to maintain backward compatibility with EAP-TLS 1.2.

TLS 1.3 changes both the message flow and the handshake messages compared to earlier versions of TLS. Therefore, much of Section 2.1 of [RFC5216] does not apply for TLS 1.3. Except for Sections 2.2 and 5.7, this update applies only when TLS 1.3 is negotiated. When TLS 1.2 is negotiated, then [RFC5216] applies.

TLS 1.3 introduces several new handshake messages including HelloRetryRequest, NewSessionTicket, and KeyUpdate. In general, these messages will be handled by the underlying TLS libraries and are not visible to EAP-TLS, however, there are a few things to note:

- * The HelloRetryRequest is used by the server to reject the parameters offered in the ClientHello and suggest new parameters. When this message is encountered it will increase the number of round trips used by the protocol.
- * The NewSessionTicket message is used to convey resumption information and is covered in Sections 2.1.2 and 2.1.3.
- * The KeyUpdate message is used to update the traffic keys used on a TLS connection. EAP-TLS does not encrypt significant amounts of data so this functionality is not needed. Implementations SHOULD NOT send this message, however some TLS libraries may automatically generate and process this message.
- * Early Data MUST NOT be used in EAP-TLS. EAP-TLS servers MUST NOT send an early_data extension and clients MUST NOT send an EndOfEarlyData message.
- * Post-handshake authentication MUST NOT be used in EAP-TLS. Clients MUST NOT send a "post_handshake_auth" extension and Servers MUST NOT request post-handshake client authentication.

After receiving an EAP-Request packet with EAP-Type=EAP-TLS as described in [RFC5216] the conversation will continue with the TLS handshake protocol encapsulated in the data fields of EAP-Response and EAP-Request packets. When EAP-TLS is used with TLS version 1.3, the formatting and processing of the TLS handshake SHALL be done as specified in version 1.3 of TLS. This update only lists additional and different requirements, restrictions, and processing compared to [RFC8446] and [RFC5216].

2.1.1. Authentication

This section updates Section 2.1.1 of [RFC5216] by amending it in accordance with the following discussion.

The EAP-TLS server MUST authenticate with a certificate and SHOULD require the EAP-TLS peer to authenticate with a certificate. Certificates can be of any type supported by TLS including raw public keys. Pre-Shared Key (PSK) authentication SHALL NOT be used except for resumption. The full handshake in EAP-TLS with TLS 1.3 always provides forward secrecy by exchange of ephemeral "key_share" extensions in the ClientHello and ServerHello (e.g., containing ephemeral ECDHE public keys). SessionID is deprecated in TLS 1.3, see Sections 4.1.2 and 4.1.3 of [RFC8446]. TLS 1.3 introduced early application data which like all application data (other than the protected success indication described below) is not used in EAP-TLS; see Section 4.2.10 of [RFC8446] for additional information on the "early_data" extension. Resumption is handled as described in Section 2.1.3. As a protected success indication [RFC3748] the EAP-TLS server always sends TLS application data 0x00, see Section 2.5. Note that a TLS implementation MAY not allow the EAP-TLS layer to control in which order things are sent and the application data MAY therefore be sent before a NewSessionTicket. TLS application data 0x00 is therefore to be interpreted as success after the EAP-Request that contains TLS application data 0x00. After the EAP-TLS server has sent an EAP-Request containing the TLS application data 0x00 and received an EAP-Response packet of EAP-Type=EAP-TLS and no data, the EAP-TLS server sends EAP-Success.

Figure 1 shows an example message flow for a successful EAP-TLS full handshake with mutual authentication (and neither HelloRetryRequest nor post-handshake messages are sent).

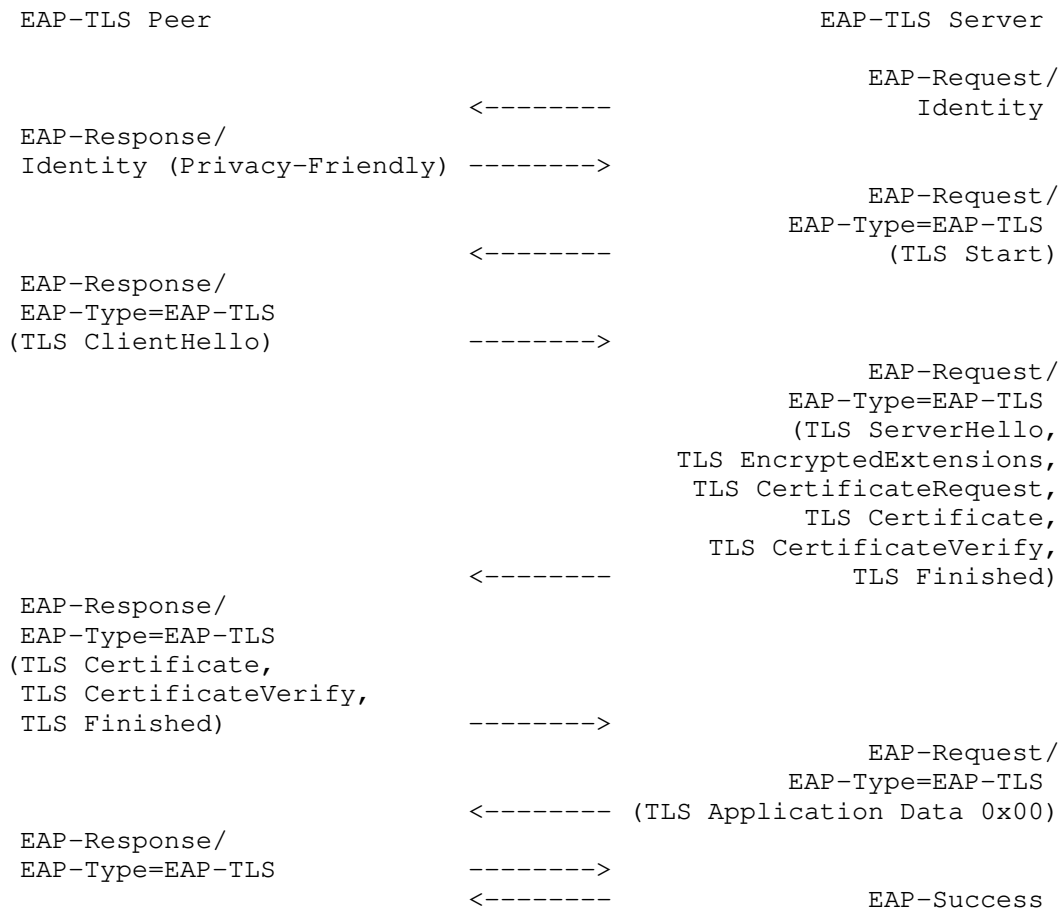


Figure 1: EAP-TLS mutual authentication

2.1.2. Ticket Establishment

This is a new section when compared to [RFC5216].

To enable resumption when using EAP-TLS with TLS 1.3, the EAP-TLS server MUST send one or more post-handshake NewSessionTicket messages (each associated with a PSK, a PSK identity, a ticket lifetime, and other parameters) in the initial authentication. Note that TLS 1.3 [RFC8446] limits the ticket lifetime to a maximum of 604800 seconds (7 days) and EAP-TLS servers MUST respect this upper limit when issuing tickets. The NewSessionTicket is sent after the EAP-TLS server has received the client Finished message in the initial authentication. The NewSessionTicket can be sent in the same flight as the TLS server Finished or later. The PSK associated with the

ticket depends on the client Finished and cannot be pre-computed (so as to be sent in the same flight as the TLS server Finished) in handshakes with client authentication. The NewSessionTicket message MUST NOT include an "early_data" extension. If the "early_data" extension is received then it MUST be ignored. Servers should take into account that fewer NewSessionTickets will likely be needed in EAP-TLS than in the usual HTTPS connection scenario. In most cases a single NewSessionTicket will be sufficient. A mechanism by which clients can specify the desired number of tickets needed for future connections is defined in [I-D.ietf-tls-ticketrequests].

Figure 2 shows an example message flow for a successful EAP-TLS full handshake with mutual authentication and ticket establishment of a single ticket.

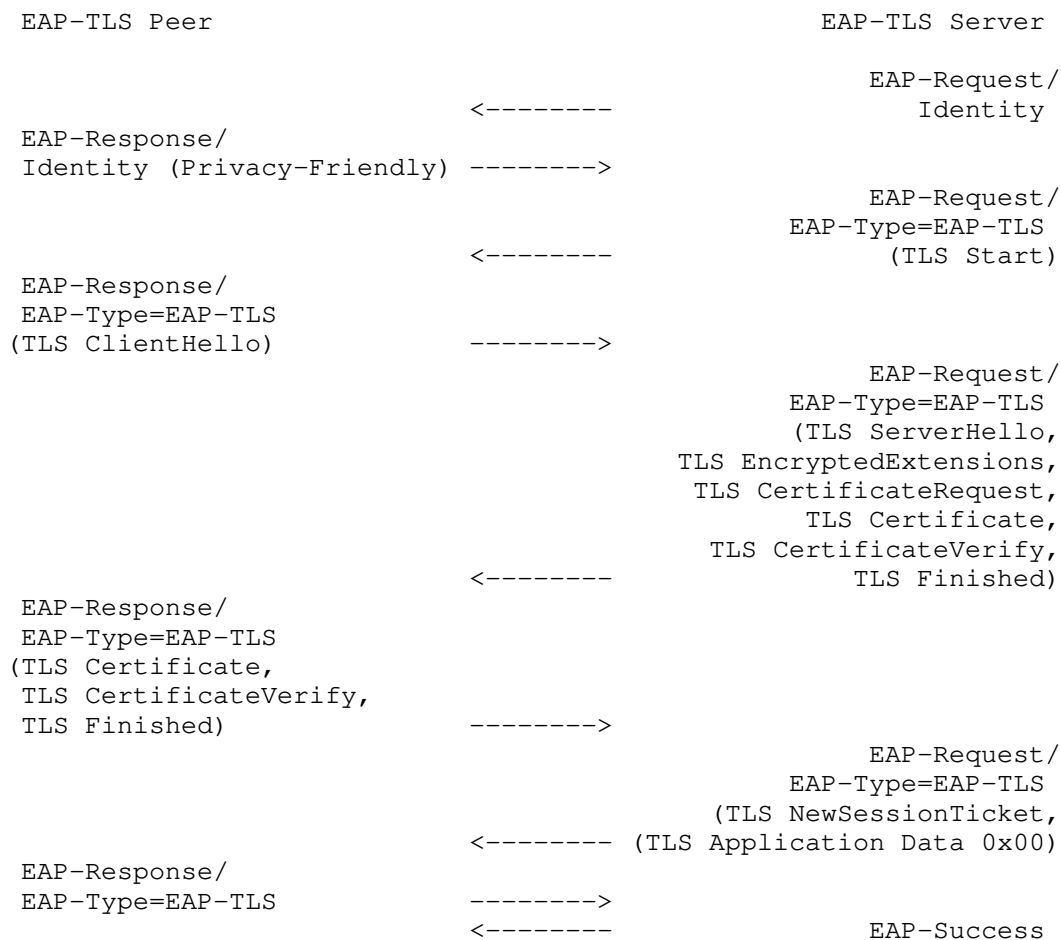


Figure 2: EAP-TLS ticket establishment

2.1.3. Resumption

This section updates Section 2.1.2 of [RFC5216] by amending it in accordance with the following discussion.

EAP-TLS is typically used with client authentication and typically fragments the TLS flights into a large number of EAP requests and EAP responses. Resumption significantly reduces the number of round-trips and enables the EAP-TLS server to omit database lookups needed during a full handshake with client authentication. TLS 1.3 replaces the session resumption mechanisms in earlier versions of TLS with a new PSK exchange. When EAP-TLS is used with TLS version 1.3, EAP-TLS SHALL use a resumption mechanism compatible with version 1.3 of TLS.

For TLS 1.3, resumption is described in Section 2.2 of [RFC8446]. If the client has received a NewSessionTicket message from the EAP-TLS server, the client can use the PSK identity associated with the ticket to negotiate the use of the associated PSK. If the EAP-TLS server accepts it, then the resumed session has been deemed to be authenticated, and securely associated with the prior authentication or resumption. It is up to the EAP-TLS peer to use resumption, but it is RECOMMENDED that the EAP-TLS peer use resumption if it has a valid ticket that has not been used before. It is left to the EAP-TLS server whether to accept resumption, but it is RECOMMENDED that the EAP-TLS server accept resumption if the ticket which was issued is still valid. However, the EAP-TLS server MAY choose to require a full handshake. In the case a full handshake is required, the negotiation proceeds as if the session was a new authentication, and the resumption attempt is ignored. The requirements of Sections 2.1.1 and 2.1.2 then apply in their entirety. As described in Appendix C.4 of [RFC8446], reuse of a ticket allows passive observers to correlate different connections. EAP-TLS peers and EAP-TLS servers SHOULD follow the client tracking preventions in Appendix C.4 of [RFC8446].

It is RECOMMENDED to use a Network Access Identifiers (NAIs) with the same realm during resumption and the original full handshake. This requirement allows EAP packets to be routed to the same destination as the original full handshake. If this recommendation is not followed, resumption is likely impossible. When NAI reuse can be done without privacy implications, it is RECOMMENDED to use the same NAI in the resumption, as was used in the original full handshake [RFC7542]. For example, the NAI @realm can safely be reused since it does not provide any specific information to associate a user's resumption attempt with the original full handshake. However, reusing the NAI P2ZIM2F+OEVAO21nNWg2bVpgNnU=@realm enables an on-path

attacker to associate a resumption attempt with the original full handshake. The TLS PSK identity is typically derived by the TLS implementation and may be an opaque blob without a routable realm. The TLS PSK identity on its own is therefore unsuitable as a NAI in the Identity Response.

Figure 3 shows an example message flow for a subsequent successful EAP-TLS resumption handshake where both sides authenticate via a PSK provisioned via an earlier NewSessionTicket and where the server provisions a single new ticket.

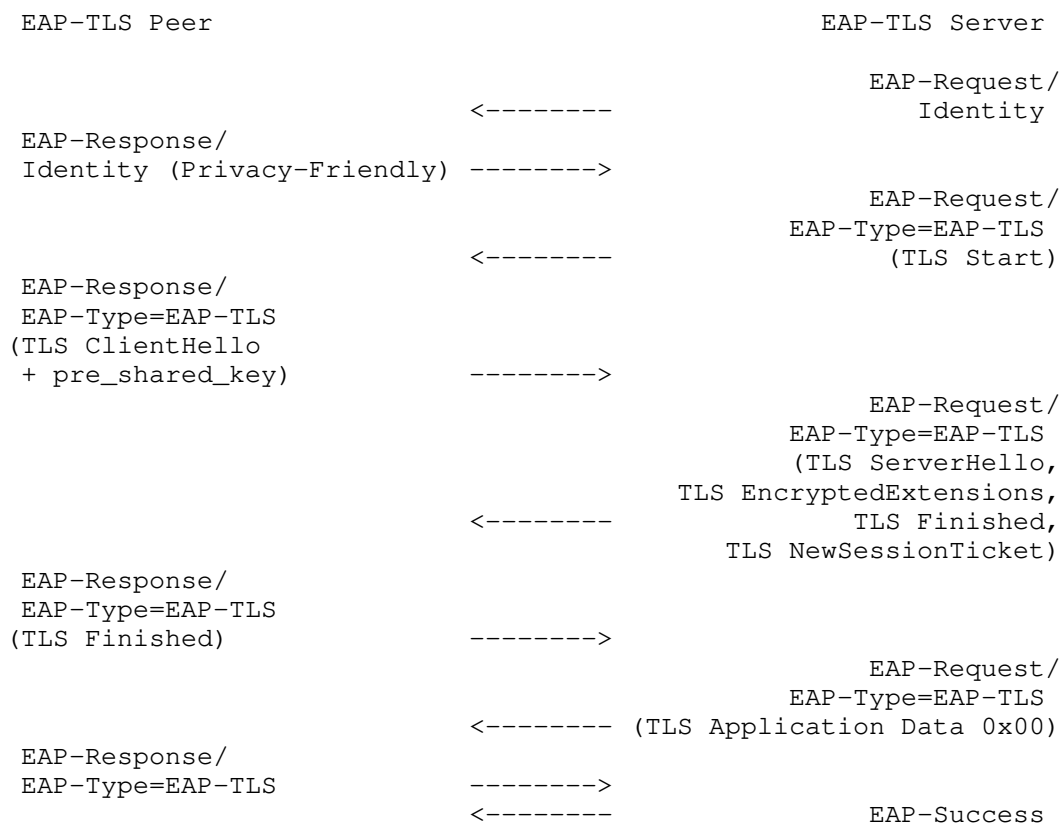


Figure 3: EAP-TLS resumption

As specified in Section 2.2 of [RFC8446], the EAP-TLS peer SHOULD supply a "key_share" extension when attempting resumption, which allows the EAP-TLS server to potentially decline resumption and fall back to a full handshake. If the EAP-TLS peer did not supply a "key_share" extension when attempting resumption, the EAP-TLS server needs to send HelloRetryRequest to signal that additional information

is needed to complete the handshake, and the EAP-TLS peer needs to send a second ClientHello containing that information. Providing a "key_share" and using the "psk_dhe_ke" pre-shared key exchange mode is also important in order to limit the impact of a key compromise. When using "psk_dhe_ke", TLS 1.3 provides forward secrecy meaning that compromise of the PSK used for resumption does not compromise any earlier connections. The "psk_dh_ke" key-exchange mode **MUST** be used for resumption unless the deployment has a local requirement to allow configuration of other mechanisms.

2.1.4. Termination

This section updates Section 2.1.3 of [RFC5216] by amending it in accordance with the following discussion.

TLS 1.3 changes both the message flow and the handshake messages compared to earlier versions of TLS. Therefore, some normative text in Section 2.1.3 of [RFC5216] does not apply for TLS 1.3. The two paragraphs below replace the corresponding paragraphs in Section 2.1.3 of [RFC5216] when EAP-TLS is used with TLS 1.3. The other paragraphs in Section 2.1.3 of [RFC5216] still apply with the exception that SessionID is deprecated.

If the EAP-TLS peer authenticates successfully, the EAP-TLS server **MUST** send an EAP-Request packet with EAP-Type=EAP-TLS containing TLS records conforming to the version of TLS used. The message flow ends with a protected success indication from the EAP-TLS server, followed by an EAP-Response packet of EAP-Type=EAP-TLS and no data from the EAP-TLS peer, followed by EAP-Success from the server.

If the EAP-TLS server authenticates successfully, the EAP-TLS peer **MUST** send an EAP-Response message with EAP-Type=EAP-TLS containing TLS records conforming to the version of TLS used.

Figures 4, 5, and 6 illustrate message flows in several cases where the EAP-TLS peer or EAP-TLS server sends a TLS Error alert message. In earlier versions of TLS, error alerts could be warnings or fatal. In TLS 1.3, error alerts are always fatal and the only alerts sent at warning level are "close_notify" and "user_canceled", both of which indicate that the connection is not going to continue normally, see [RFC8446].

In TLS 1.3 [RFC8446], error alerts are not mandatory to send after a fatal error condition. Failure to send TLS Error alerts means that the peer or server would have no way of determining what went wrong. EAP-TLS 1.3 strengthens this requirement. Whenever an implementation encounters a fatal error condition, it **MUST** send an appropriate TLS Error alert.

Figure 4 shows an example message flow where the EAP-TLS server rejects the ClientHello with an error alert. The EAP-TLS server can also partly reject the ClientHello with a HelloRetryRequest, see Section 2.1.6.



Figure 4: EAP-TLS server rejection of ClientHello

Figure 5 shows an example message flow where EAP-TLS server authentication is unsuccessful and the EAP-TLS peer sends a TLS Error alert.

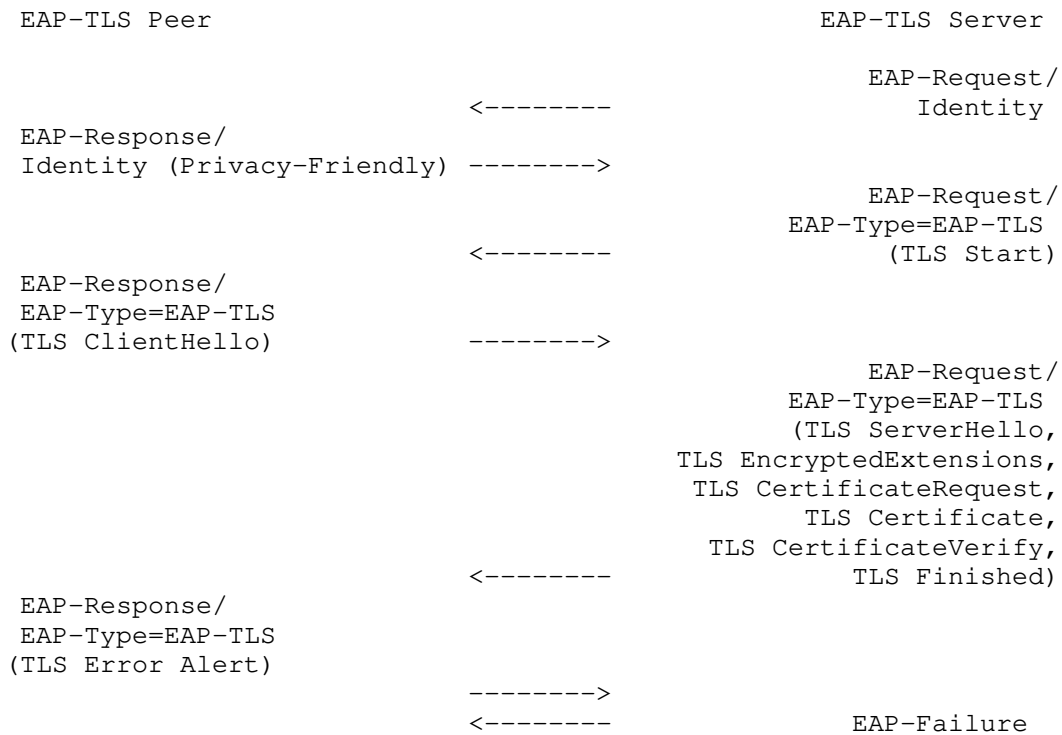


Figure 5: EAP-TLS unsuccessful EAP-TLS server authentication

Figure 6 shows an example message flow where the EAP-TLS server authenticates to the EAP-TLS peer successfully, but the EAP-TLS peer fails to authenticate to the EAP-TLS server and the server sends a TLS Error alert.

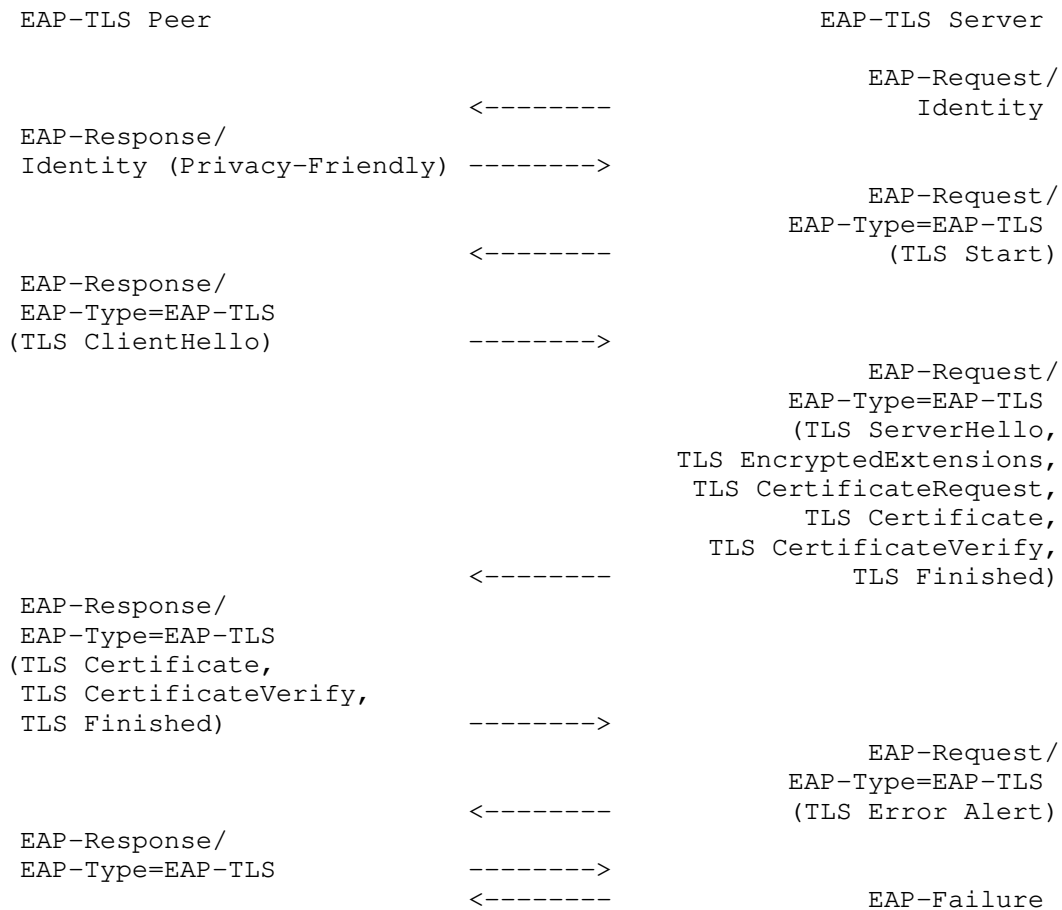


Figure 6: EAP-TLS unsuccessful client authentication

2.1.5. No Peer Authentication

This is a new section when compared to [RFC5216].

Figure 7 shows an example message flow for a successful EAP-TLS full handshake without peer authentication (e.g., emergency services, as described in [RFC7406]).

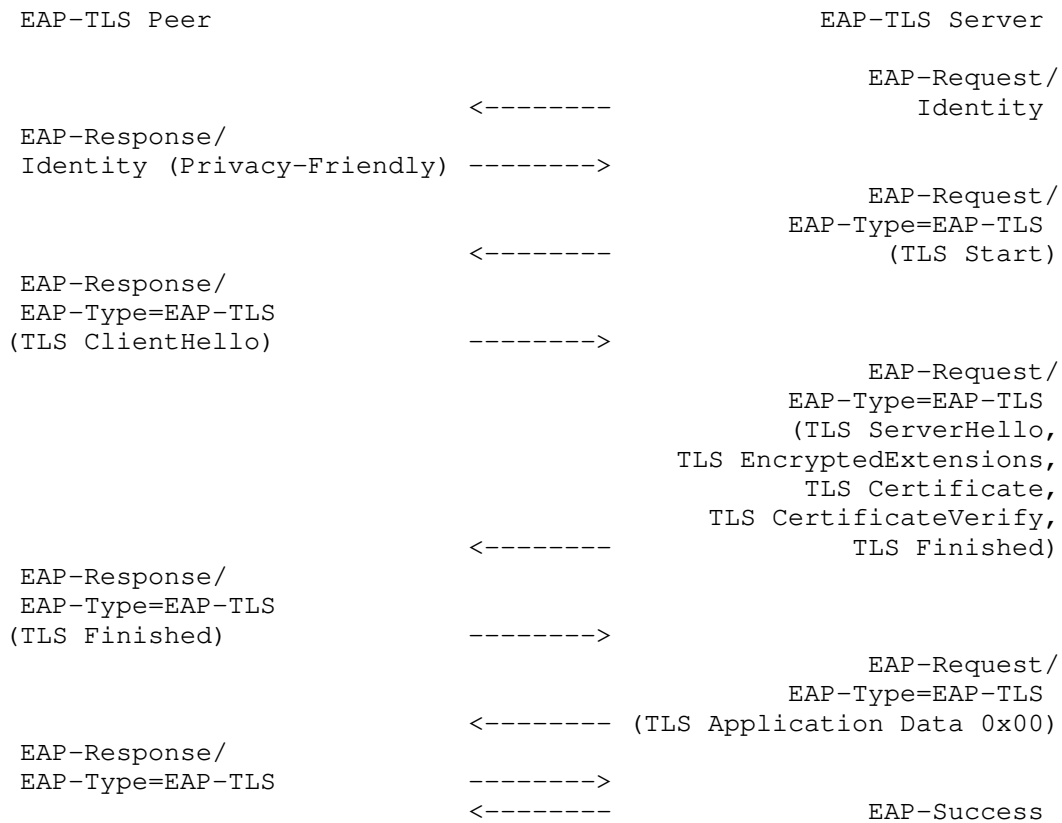


Figure 7: EAP-TLS without peer authentication

2.1.6. Hello Retry Request

This is a new section when compared to [RFC5216].

As defined in TLS 1.3 [RFC8446], EAP-TLS servers can send a HelloRetryRequest message in response to a ClientHello if the EAP-TLS server finds an acceptable set of parameters but the initial ClientHello does not contain all the needed information to continue the handshake. One use case is if the EAP-TLS server does not support the groups in the "key_share" extension (or there is no "key_share" extension), but supports one of the groups in the "supported_groups" extension. In this case the client should send a new ClientHello with a "key_share" that the EAP-TLS server supports.

Figure 8 shows an example message flow for a successful EAP-TLS full handshake with mutual authentication and HelloRetryRequest. Note the extra round-trip as a result of the HelloRetryRequest.

```

EAP-TLS Peer                                EAP-TLS Server

                                           EAP-Request/
                                           Identity
EAP-Response/                               <-----
Identity (Privacy-Friendly) ----->

                                           EAP-Request/
                                           EAP-Type=EAP-TLS
                                           (TLS Start)
EAP-Response/                               <-----
EAP-Type=EAP-TLS                           ----->
(TLS ClientHello)

                                           EAP-Request/
                                           EAP-Type=EAP-TLS
                                           (TLS HelloRetryRequest)
EAP-Response/                               <-----
EAP-Type=EAP-TLS                           ----->
(TLS ClientHello)

                                           EAP-Request/
                                           EAP-Type=EAP-TLS
                                           (TLS ServerHello,
                                           TLS EncryptedExtensions,
                                           TLS CertificateRequest,
                                           TLS Certificate,
                                           TLS CertificateVerify,
                                           TLS Finished)
EAP-Response/                               ----->
EAP-Type=EAP-TLS                           (TLS Application Data 0x00)
(TLS Certificate,
TLS CertificateVerify,
TLS Finished)

                                           EAP-Request/
                                           EAP-Type=EAP-TLS
                                           (TLS Application Data 0x00)
EAP-Response/                               <-----
EAP-Type=EAP-TLS                           ----->
                                           <-----
                                           EAP-Success

```

Figure 8: EAP-TLS with Hello Retry Request

2.1.7. Identity

This is a new section when compared to [RFC5216].

It is RECOMMENDED to use anonymous NAIs [RFC7542] in the Identity Response as such identities are routable and privacy-friendly. While opaque blobs are allowed by [RFC3748], such identities are NOT

RECOMMENDED as they are not routable and should only be considered in local deployments where the EAP-TLS peer, EAP authenticator, and EAP-TLS server all belong to the same network. Many client certificates contain an identity such as an email address, which is already in NAI format. When the client certificate contains a NAI as subject name or alternative subject name, an anonymous NAI SHOULD be derived from the NAI in the certificate, see Section 2.1.8. More details on identities are described in Sections 2.1.3, 2.1.8, 2.2, and 5.8.

2.1.8. Privacy

This section updates Section 2.1.4 of [RFC5216] by amending it in accordance with the following discussion.

EAP-TLS 1.3 significantly improves privacy when compared to earlier versions of EAP-TLS. EAP-TLS 1.3 forbids cipher suites without confidentiality which means that TLS 1.3 is always encrypting large parts of the TLS handshake including the certificate messages.

EAP-TLS peer and server implementations supporting TLS 1.3 MUST support anonymous Network Access Identifiers (NAIs) (Section 2.4 in [RFC7542]) and a client supporting TLS 1.3 MUST NOT send its username in cleartext in the Identity Response. Following [RFC7542], it is RECOMMENDED to omit the username (i.e., the NAI is @realm), but other constructions such as a fixed username (e.g., anonymous@realm) or an encrypted username (e.g., xCZINCPTK5+7y81CrSYbPg+RKPE30TrYLn4AQc4AC2U=@realm) are allowed. Note that the NAI MUST be a UTF-8 string as defined by the grammar in Section 2.2 of [RFC7542].

The HelloRequest message used for privacy in EAP-TLS 1.2 does not exist in TLS 1.3 but as the certificate messages in TLS 1.3 are encrypted, there is no need to send an empty certificate_list and perform a second handshake for privacy (as needed by EAP-TLS with earlier versions of TLS). When EAP-TLS is used with TLS version 1.3 the EAP-TLS peer and EAP-TLS server SHALL follow the processing specified by version 1.3 of TLS. This means that the EAP-TLS peer only sends an empty certificate_list if it does not have an appropriate certificate to send, and the EAP-TLS server MAY treat an empty certificate_list as a terminal condition.

EAP-TLS with TLS 1.3 is always used with privacy. This does not add any extra round-trips and the message flow with privacy is just the normal message flow as shown in Figure 1.

2.1.9. Fragmentation

This section updates Section 2.1.5 of [RFC5216] by amending it in accordance with the following discussion.

Including ContentType (1 byte), ProtocolVersion (2 bytes), and length (2 bytes) headers a single TLS record may be up to 16645 octets in length. EAP-TLS fragmentation support is provided through addition of a flags octet within the EAP-Response and EAP-Request packets, as well as a (conditional) TLS Message Length field of four octets. Implementations **MUST NOT** set the L bit in unfragmented messages, but **MUST** accept unfragmented messages with and without the L bit set.

Some EAP implementations and access networks may limit the number of EAP packet exchanges that can be handled. To avoid fragmentation, it is **RECOMMENDED** to keep the sizes of EAP-TLS peer, EAP-TLS server, and trust anchor certificates small and the length of the certificate chains short. In addition, it is **RECOMMENDED** to use mechanisms that reduce the sizes of Certificate messages. For a detailed discussion on reducing message sizes to prevent fragmentation, see [I-D.ietf-emu-eaptlscert].

2.2. Identity Verification

This section updates Section 2.2 of [RFC5216] by amending it in accordance with the following discussion. The guidance in this section is relevant for EAP-TLS in general (regardless of the underlying TLS version used).

The EAP peer identity provided in the EAP-Response/Identity is not authenticated by EAP-TLS. Unauthenticated information **MUST NOT** be used for accounting purposes or to give authorization. The authenticator and the EAP-TLS server **MAY** examine the identity presented in EAP-Response/Identity for purposes such as routing and EAP method selection. EAP-TLS servers **MAY** reject conversations if the identity does not match their policy. Note that this also applies to resumption, see Sections 2.1.3, 5.6, and 5.7.

The EAP server identity in the TLS server certificate is typically a fully qualified domain name (FQDN) in the SubjectAltName (SAN) extension. Since EAP-TLS deployments may use more than one EAP server, each with a different certificate, EAP peer implementations **SHOULD** allow for the configuration of one or more trusted root certificates (CA certificate) to authenticate the server certificate and one or more server names to match against the SubjectAltName (SAN) extension in the server certificate. If any of the configured names match any of the names in the SAN extension then the name check passes. To simplify name matching, an EAP-TLS deployment can assign

a name to represent an authorized EAP server and EAP Server certificates can include this name in the list of SANs for each certificate that represents an EAP-TLS server. If server name matching is not used, then it degrades the confidence that the EAP server with which it is interacting is authoritative for the given network. If name matching is not used with a public root CA, then effectively any server can obtain a certificate which will be trusted for EAP authentication by the Peer. While this guidance to verify domain names is new, and was not mentioned in [RFC5216], it has been widely implemented in EAP-TLS peers. As such, it is believed that this section contains minimal new interoperability or implementation requirements on EAP-TLS peers and can be applied to earlier versions of TLS.

The process of configuring a root CA certificate and a server name is non-trivial and therefore automated methods of provisioning are RECOMMENDED. For example, the eduroam federation [RFC7593] provides a Configuration Assistant Tool (CAT) to automate the configuration process. In the absence of a trusted root CA certificate (user configured or system-wide), EAP peers MAY implement a trust on first use (TOFU) mechanism where the peer trusts and stores the server certificate during the first connection attempt. The EAP peer ensures that the server presents the same stored certificate on subsequent interactions. Use of a TOFU mechanism does not allow for the server certificate to change without out-of-band validation of the certificate and is therefore not suitable for many deployments including ones where multiple EAP servers are deployed for high availability. TOFU mechanisms increase the susceptibility to traffic interception attacks and should only be used if there are adequate controls in place to mitigate this risk.

2.3. Key Hierarchy

This section updates Section 2.3 of [RFC5216] by replacing it in accordance with the following discussion.

TLS 1.3 replaces the TLS pseudorandom function (PRF) used in earlier versions of TLS with HKDF and completely changes the Key Schedule. The key hierarchies shown in Section 2.3 of [RFC5216] are therefore not correct when EAP-TLS is used with TLS version 1.3. For TLS 1.3 the key schedule is described in Section 7.1 of [RFC8446].

When EAP-TLS is used with TLS version 1.3 the Key_Material and Method-Id SHALL be derived from the exporter_secret using the TLS exporter interface [RFC5705] (for TLS 1.3 this is defined in Section 7.5 of [RFC8446]). Type is the value of the EAP Type field defined in Section 2 of [RFC3748]. For EAP-TLS the Type field has value 0x0D.

```
Type = 0x0D
Key_Material = TLS-Exporter("EXPORTER_EAP_TLS_Key_Material",
                             Type, 128)
Method-Id    = TLS-Exporter("EXPORTER_EAP_TLS_Method-Id",
                             Type, 64)
Session-Id   = Type || Method-Id
```

The MSK and EMSK are derived from the Key_Material in the same manner as with EAP-TLS [RFC5216], Section 2.3. The definitions are repeated below for simplicity:

```
MSK          = Key_Material(0, 63)
EMSK         = Key_Material(64, 127)
```

Other TLS based EAP methods can use the TLS exporter in a similar fashion, see [I-D.ietf-emu-tls-eap-types].

[RFC5247] deprecates the use of IV. Thus, RECV-IV and SEND-IV are not exported in EAP-TLS with TLS 1.3. As noted in [RFC5247], lower layers use the MSK in a lower-layer-dependent manner. EAP-TLS with TLS 1.3 exports the MSK and does not specify how it is used by lower layers.

Note that the key derivation MUST use the length values given above. While in TLS 1.2 and earlier it was possible to truncate the output by requesting less data from the TLS-Exporter function, this practice is not possible with TLS 1.3. If an implementation intends to use only a part of the output of the TLS-Exporter function, then it MUST ask for the full output and then only use the desired part. Failure to do so will result in incorrect values being calculated for the above keying material.

By using the TLS exporter, EAP-TLS can use any TLS 1.3 implementation which provides a public API for the exporter. Note that when TLS 1.2 is used with the EAP-TLS exporter [RFC5705] it generates the same key material as in EAP-TLS [RFC5216].

2.4. Parameter Negotiation and Compliance Requirements

This section updates Section 2.4 of [RFC5216] by amending it in accordance with the following discussion.

TLS 1.3 cipher suites are defined differently than in earlier versions of TLS (see Section B.4 of [RFC8446]), and the cipher suites discussed in Section 2.4 of [RFC5216] can therefore not be used when EAP-TLS is used with TLS version 1.3.

When EAP-TLS is used with TLS version 1.3, the EAP-TLS peers and EAP-TLS servers MUST comply with the compliance requirements (mandatory-to-implement cipher suites, signature algorithms, key exchange algorithms, extensions, etc.) defined in Section 9 of [RFC8446]. In EAP-TLS with TLS 1.3, only cipher suites with confidentiality SHALL be supported.

While EAP-TLS does not protect any application data except for the 0x00 byte that serves as protected success indication, the negotiated cipher suites and algorithms MAY be used to secure data as done in other TLS-based EAP methods.

2.5. EAP State Machines

This is a new section when compared to [RFC5216] and only applies to TLS 1.3. [RFC4137] offers a proposed state machine for EAP.

TLS 1.3 [RFC8446] introduces post-handshake messages. These post-handshake messages use the handshake content type and can be sent after the main handshake. Examples of post-handshake messages are NewSessionTicket, which is used for resumption and KeyUpdate, which is not useful and not expected in EAP-TLS. After sending TLS Finished, the EAP-TLS server may send any number of post-handshake messages in one or more EAP-Requests.

To provide a protected success result indication and to decrease the uncertainty for the EAP-TLS peer, the following procedure MUST be followed:

When an EAP-TLS server has successfully processed the TLS client Finished and sent its last handshake message (Finished or a post-handshake message), it sends an encrypted TLS record with application data 0x00. The encrypted TLS record with application data 0x00 is a protected success result indication, as defined in [RFC3748]. After sending an EAP-Request that contains the protected success result indication, the EAP-TLS server must not send any more EAP-Request and may only send an EAP-Success. The EAP-TLS server MUST NOT send an encrypted TLS record with application data 0x00 alert before it has successfully processed the client finished and sent its last handshake message.

TLS Error alerts SHOULD be considered a failure result indication, as defined in [RFC3748]. Implementations following [RFC4137] set the alternate indication of failure variable altReject after sending or receiving an error alert. After sending or receiving a TLS Error alert, the EAP-TLS server may only send an EAP-Failure. Protected TLS Error alerts are protected failure result indications, unprotected TLS Error alerts are not.

The keying material can be derived after the TLS server Finished has been sent or received. Implementations following [RFC4137] can then set the eapKeyData and aaaEapKeyData variables.

The keying material can be made available to lower layers and the authenticator after the authenticated success result indication has been sent or received. Implementations following [RFC4137] can set the eapKeyAvailable and aaaEapKeyAvailable variables.

3. Detailed Description of the EAP-TLS Protocol

No updates to Section 3 of [RFC5216].

4. IANA considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the EAP-TLS 1.3 protocol in accordance with [RFC8126].

This document requires IANA to add the following labels to the TLS Exporter Label Registry defined by [RFC5705]. These labels are used in derivation of Key_Material and Method-Id as defined in Section 2.3:

Value	DTLS-OK	Recommended	Note
EXPORTER_EAP_TLS_Key_Material	N	Y	
EXPORTER_EAP_TLS_Method-Id	N	Y	

Table 1: TLS Exporter Label Registry

5. Security Considerations

The security considerations of TLS 1.3 [RFC8446] apply to EAP-TLS 1.3

5.1. Security Claims

Using EAP-TLS with TLS 1.3 does not change the security claims for EAP-TLS as given in Section 5.1 of [RFC5216]. However, it strengthens several of the claims as described in the following updates to the notes given in Section 5.1 of [RFC5216].

[1] Mutual authentication: By mandating revocation checking of certificates, the authentication in EAP-TLS with TLS 1.3 is stronger as authentication with revoked certificates will always fail.

[2] Confidentiality: The TLS 1.3 handshake offers much better confidentiality than earlier versions of TLS. EAP-TLS with TLS 1.3 mandates use of cipher suites that ensure confidentiality. TLS 1.3 also encrypts certificates and some of the extensions. When using EAP-TLS with TLS 1.3, the use of privacy is mandatory and does not cause any additional round-trips.

[3] Cryptographic strength: TLS 1.3 only defines strong algorithms without major weaknesses and EAP-TLS with TLS 1.3 always provides forward secrecy, see [RFC8446]. Weak algorithms such as 3DES, CBC mode, RC4, SHA-1, MD5, P-192, and RSA-1024 have not been registered for use in TLS 1.3.

[4] Cryptographic Negotiation: The TLS layer handles the negotiation of cryptographic parameters. When EAP-TLS is used with TLS 1.3, EAP-TLS inherits the cryptographic negotiation of AEAD algorithm, HKDF hash algorithm, key exchange groups, and signature algorithm, see Section 4.1.1 of [RFC8446].

5.2. Peer and Server Identities

No updates to section 5.2 of [RFC5216]. Note that Section 2.2 has additional discussion on identities.

5.3. Certificate Validation

No updates to section 5.3 of [RFC5216]. In addition to section 5.3 of [RFC5216], guidance on server certificate validation can be found in [RFC6125].

5.4. Certificate Revocation

This section updates Section 5.4 of [RFC5216] by amending it in accordance with the following discussion.

There are a number of reasons (e.g., key compromise, CA compromise, privilege withdrawn, etc.) why EAP-TLS peer, EAP-TLS server, or sub-CA certificates have to be revoked before their expiry date. Revocation of the EAP-TLS server's certificate is complicated by the fact that the EAP-TLS peer may not have Internet connectivity until authentication completes.

When EAP-TLS is used with TLS 1.3, the revocation status of all the certificates in the certificate chains MUST be checked (except the trust anchor). An implementation may use Certificate Revocation List (CRL), Online Certificate Status Protocol (OCSP), or other standardized/proprietary methods for revocation checking. Examples of proprietary methods are non-standard formats for distribution of revocation lists as well as certificates with very short lifetime.

EAP-TLS servers supporting TLS 1.3 MUST implement Certificate Status Requests (OCSP stapling) as specified in [RFC6066] and Section 4.4.2.1 of [RFC8446]. It is RECOMMENDED that EAP-TLS peers and EAP-TLS servers use OCSP stapling for verifying the status of the EAP-TLS server's certificate chain. When an EAP-TLS peer uses Certificate Status Requests to check the revocation status of the EAP-TLS server's certificate chain it MUST treat a CertificateEntry (except the trust anchor) without a valid CertificateStatus extension as invalid and abort the handshake with an appropriate alert. The OCSP status handling in TLS 1.3 is different from earlier versions of TLS, see Section 4.4.2.1 of [RFC8446]. In TLS 1.3 the OCSP information is carried in the CertificateEntry containing the associated certificate instead of a separate CertificateStatus message as in [RFC6066]. This enables sending OCSP information for all certificates in the certificate chain (except the trust anchor).

To enable revocation checking in situations where EAP-TLS peers do not implement or use OCSP stapling, and where network connectivity is not available prior to authentication completion, EAP-TLS peer implementations MUST also support checking for certificate revocation after authentication completes and network connectivity is available. An EAP peer implementation SHOULD NOT trust the network (and any services) until it has verified the revocation status of the server certificate after receiving network connectivity. An EAP peer MUST use a secure transport to verify the revocation status of the server certificate. An EAP peer SHOULD NOT send any other traffic before revocation checking for the server certificate is complete.

5.5. Packet Modification Attacks

This section updates Section 5.5 of [RFC5216] by amending it in accordance with the following discussion.

As described in [RFC3748] and Section 5.5 of [RFC5216], the only information that is integrity and replay protected in EAP-TLS are the parts of the TLS Data that TLS protects. All other information in the EAP-TLS message exchange including EAP-Request and EAP-Response headers, the identity in the identity response, EAP-TLS packet header fields, Type, and Flags, EAP-Success, and EAP-Failure can be modified, spoofed, or replayed.

Protected TLS Error alerts are protected failure result indications and enables the EAP-TLS peer and EAP-TLS server to determine that the failure result was not spoofed by an attacker. Protected failure result indications provide integrity and replay protection but MAY be unauthenticated. Protected failure results do not significantly improve availability as TLS 1.3 treats most malformed data as a fatal error.

5.6. Authorization

This is a new section when compared to [RFC5216]. The guidance in this section is relevant for EAP-TLS in general (regardless of the underlying TLS version used).

EAP servers will usually require the EAP peer to provide a valid certificate and will fail the connection if one is not provided. Some deployments may permit no peer authentication for some or all connections. When peer authentication is not used, EAP-TLS server implementations MUST take care to limit network access appropriately for unauthenticated peers and implementations MUST use resumption with caution to ensure that a resumed session is not granted more privilege than was intended for the original session. An example of limiting network access would be to invoke a vendor's walled garden or quarantine network functionality.

EAP-TLS is typically encapsulated in other protocols, such as PPP [RFC1661], RADIUS [RFC2865], Diameter [RFC6733], or PANA [RFC5191]. The encapsulating protocols can also provide additional, non-EAP information to an EAP-TLS server. This information can include, but is not limited to, information about the authenticator, information about the EAP-TLS peer, or information about the protocol layers above or below EAP (MAC addresses, IP addresses, port numbers, Wi-Fi SSID, etc.). EAP-TLS servers implementing EAP-TLS inside those protocols can make policy decisions and enforce authorization based on a combination of information from the EAP-TLS exchange and non-EAP information.

As noted in Section 2.2, the identity presented in EAP-Response/Identity is not authenticated by EAP-TLS and is therefore trivial for an attacker to forge, modify, or replay. Authorization and accounting MUST be based on authenticated information such as information in the certificate or the PSK identity and cached data provisioned for resumption as described in Section 5.7. Note that the requirements for Network Access Identifiers (NAIs) specified in Section 4 of [RFC7542] still apply and MUST be followed.

EAP-TLS servers MAY reject conversations based on non-EAP information provided by the encapsulating protocol, for example, if the MAC address of the authenticator does not match the expected policy.

In addition to allowing configuration of one or more trusted root certificates (CA certificate) to authenticate the server certificate and one or more server names to match against the SubjectAltName (SAN) extension, EAP peer implementations MAY allow binding the configured acceptable SAN to a specific CA (or CAs) that should have issued the server certificate to prevent attacks from rogue or compromised CAs.

5.7. Resumption

This is a new section when compared to [RFC5216]. The guidance in this section is relevant for EAP-TLS in general (regardless of the underlying TLS version used).

There are a number of security issues related to resumption that are not described in [RFC5216]. The problems, guidelines, and requirements in this section therefore applies to EAP-TLS when it is used with any version of TLS.

When resumption occurs, it is based on cached information at the TLS layer. To perform resumption securely, the EAP-TLS peer and EAP-TLS server need to be able to securely retrieve authorization information such as certificate chains from the initial full handshake. This document uses the term "cached data" to describe such information. Authorization during resumption MUST be based on such cached data. The EAP-TLS peer and EAP-TLS server MAY perform fresh revocation checks on the cached certificate data. Any security policies for authorization MUST be followed also for resumption. The certificates may have been revoked since the initial full handshake and the authorizations of the other party may have been reduced. If the cached revocation data is not sufficiently current, the EAP-TLS peer or EAP-TLS server MAY force a full TLS handshake.

There are two ways to retrieve the cached data from the original full handshake. The first method is that the EAP-TLS server and client cache the information locally. The cached information is identified by an identifier. For TLS versions before 1.3, the identifier can be the session ID, for TLS 1.3, the identifier is the PSK identity. The second method for retrieving cached information is via [RFC5077] or [RFC8446], where the EAP-TLS server avoids storing information locally and instead encapsulates the information into a ticket which is sent to the client for storage. This ticket is encrypted using a key that only the EAP-TLS server knows. Note that the client still needs to cache the original handshake information locally and will

obtain it while determining the session ID or PSK identity to use for resumption. However, the EAP-TLS server is able to decrypt the ticket or PSK to obtain the original handshake information.

The EAP-TLS server or EAP client MUST cache data during the initial full handshake sufficient to allow authorization decisions to be made during resumption. If cached data cannot be retrieved securely, resumption MUST NOT be done.

The above requirements also apply if the EAP-TLS server expects some system to perform accounting for the session. Since accounting must be tied to an authenticated identity, and resumption does not supply such an identity, accounting is impossible without access to cached data. Therefore, systems which expect to perform accounting for the session SHOULD cache an identifier which can be used in subsequent accounting.

As suggested in [RFC8446], EAP-TLS peers MUST NOT store resumption PSKs or tickets (and associated cached data) for longer than 604800 seconds (7 days), regardless of the PSK or ticket lifetime. The EAP-TLS peer MAY delete them earlier based on local policy. The cached data MAY also be removed on the EAP-TLS server or EAP-TLS peer if any certificate in the certificate chain has been revoked or has expired. In all such cases, an attempt at resumption results in a full TLS handshake instead.

Information from the EAP-TLS exchange (e.g., the identity provided in EAP-Response/Identity) as well as non-EAP information (e.g., IP addresses) may change between the initial full handshake and resumption. This change creates a "time-of-check time-of-use" (TOCTOU) security vulnerability. A malicious or compromised user could supply one set of data during the initial authentication, and a different set of data during resumption, potentially allowing them to obtain access that they should not have.

If any authorization, accounting, or policy decisions were made with information that has changed between the initial full handshake and resumption, and if change may lead to a different decision, such decisions MUST be reevaluated. It is RECOMMENDED that authorization, accounting, and policy decisions are reevaluated based on the information given in the resumption. EAP-TLS servers MAY reject resumption where the information supplied during resumption does not match the information supplied during the original authentication. If a safe decision is not possible, EAP-TLS servers SHOULD reject the resumption and continue with a full handshake.

Section 2.2 and 4.2.11 of [RFC8446] provides security considerations for TLS 1.3 resumption.

5.8. Privacy Considerations

This is a new section when compared to [RFC5216].

TLS 1.3 offers much better privacy than earlier versions of TLS as discussed in Section 2.1.8. In this section, we only discuss the privacy properties of EAP-TLS with TLS 1.3. For privacy properties of TLS 1.3 itself, see [RFC8446].

EAP-TLS sends the standard TLS 1.3 handshake messages encapsulated in EAP packets. Additionally, the EAP-TLS peer sends an identity in the first EAP-Response. The other fields in the EAP-TLS Request and the EAP-TLS Response packets do not contain any cleartext privacy-sensitive information.

Tracking of users by eavesdropping on identity responses or certificates is a well-known problem in many EAP methods. When EAP-TLS is used with TLS 1.3, all certificates are encrypted, and the username part of the identity response is not revealed (e.g., using anonymous NAIs). Note that even though all certificates are encrypted, the server's identity is only protected against passive attackers while the client's identity is protected against both passive and active attackers. As with other EAP methods, even when privacy-friendly identifiers or EAP tunneling is used, the domain name (i.e., the realm) in the NAI is still typically visible. How much privacy-sensitive information the domain name leaks is highly dependent on how many other users are using the same domain name in the particular access network. If all EAP-TLS peers have the same domain, no additional information is leaked. If a domain name is used by a small subset of the EAP-TLS peers, it may aid an attacker in tracking or identifying the user.

Without padding, information about the size of the client certificate is leaked from the size of the EAP-TLS packets. The EAP-TLS packets sizes may therefore leak information that can be used to track or identify the user. If all client certificates have the same length, no information is leaked. EAP-TLS peers SHOULD use record padding, see Section 5.4 of [RFC8446] to reduce information leakage of certificate sizes.

If anonymous NAIs are not used, the privacy-friendly identifiers need to be generated with care. The identities MUST be generated in a cryptographically secure way so that it is computationally infeasible for an attacker to differentiate two identities belonging to the same user from two identities belonging to different users in the same realm. This can be achieved, for instance, by using random or pseudo-random usernames such as random byte strings or ciphertexts and only using the pseudo-random usernames a single time. Note that

the privacy-friendly usernames also MUST NOT include substrings that can be used to relate the identity to a specific user. Similarly, privacy-friendly username MUST NOT be formed by a fixed mapping that stays the same across multiple different authentications.

An EAP-TLS peer with a policy allowing communication with EAP-TLS servers supporting only TLS 1.2 without privacy and with a static RSA key exchange is vulnerable to disclosure of the EAP-TLS peer username. An active attacker can in this case make the EAP-TLS peer believe that an EAP-TLS server supporting TLS 1.3 only supports TLS 1.2 without privacy. The attacker can simply impersonate the EAP-TLS server and negotiate TLS 1.2 with static RSA key exchange and send an TLS alert message when the EAP-TLS peer tries to use privacy by sending an empty certificate message. Since the attacker (impersonating the EAP-TLS server) does not provide a proof-of-possession of the private key until the Finished message when a static RSA key exchange is used, an EAP-TLS peer may inadvertently disclose its identity (username) to an attacker. Therefore, it is RECOMMENDED for EAP-TLS peers to not use EAP-TLS with TLS 1.2 and static RSA based cipher suites without privacy. This implies that an EAP-TLS peer SHOULD NOT continue the EAP authentication attempt if a TLS 1.2 EAP-TLS server sends an EAP-TLS/Request with a TLS alert message in response to an empty certificate message from the peer.

5.9. Pervasive Monitoring

This is a new section when compared to [RFC5216].

Pervasive monitoring refers to widespread surveillance of users. In the context of EAP-TLS, pervasive monitoring attacks can target EAP-TLS peer devices for tracking them (and their users) as and when they join a network. By encrypting more information, mandating the use of privacy, and always providing forward secrecy, EAP-TLS with TLS 1.3 offers much better protection against pervasive monitoring. In addition to the privacy attacks discussed above, surveillance on a large scale may enable tracking of a user over a wide geographical area and across different access networks. Using information from EAP-TLS together with information gathered from other protocols increases the risk of identifying individual users.

5.10. Discovered Vulnerabilities

This is a new section when compared to [RFC5216].

Over the years, there have been several serious attacks on earlier versions of Transport Layer Security (TLS), including attacks on its most commonly used ciphers and modes of operation. [RFC7457] summarizes the attacks that were known at the time of publishing and

BCP 195 [RFC7525] [RFC8996] provides recommendations and requirements for improving the security of deployed services that use TLS. However, many of the attacks are less serious for EAP-TLS as EAP-TLS only uses the TLS handshake and does not protect any application data. EAP-TLS implementations MUST mitigate known attacks. EAP-TLS implementations need to monitor and follow new EAP and TLS related security guidance and requirements such as [RFC8447] and [I-D.ietf-tls-md5-sha1-deprecate].

5.11. Cross-Protocol Attacks

This is a new section when compared to [RFC5216].

Allowing the same certificate to be used in multiple protocols can potentially allow an attacker to authenticate via one protocol, and then "resume" that session in another protocol. Section 2.2 above suggests that certificates typically have one or more FQDNs in the SAN extension. However, those fields are for EAP validation only, and do not indicate that the certificates are suitable for use on WWW (or other) protocol server on the named host.

Section 2.1.3 above suggests that authorization rules should be re-applied on resumption, but does not mandate this behavior. As a result, this cross-protocol resumption could allow the attacker to bypass authorization policies, and to obtain undesired access to secured systems. Along with making sure that appropriate authorization information is available and used during resumption, using different certificates and resumption caches for different protocols is RECOMMENDED to help keep different protocol usages separate.

6. References

6.1. Normative References

- [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>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <<https://www.rfc-editor.org/info/rfc5216>>.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <<https://www.rfc-editor.org/info/rfc5705>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <<https://www.rfc-editor.org/info/rfc7542>>.
- [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>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

6.2. Informative references

- [IEEE-802.1X] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks -- Port-Based Network Access Control", IEEE Standard 802.1X-2020 , February 2020.

- [IEEE-802.11] Institute of Electrical and Electronics Engineers, "IEEE Standard for Information technologyTelecommunications and information exchange between systems Local and metropolitan area networksSpecific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11-2020 , February 2021.
- [IEEE-802.1AE] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks -- Media Access Control (MAC) Security", IEEE Standard 802.1AE-2018 , December 2018.
- [TS.33.501] 3GPP, "Security architecture and procedures for 5G System", 3GPP TS 33.501 17.3.0, September 2021.
- [MultaFire] MultaFire, "MultaFire Release 1.1 specification", 2019.
- [PEAP] Microsoft Corporation, "[MS-PEAP]: Protected Extensible Authentication Protocol (PEAP)", June 2021.
- [RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994, <<https://www.rfc-editor.org/info/rfc1661>>.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <<https://www.rfc-editor.org/info/rfc2246>>.
- [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>>.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.

- [RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, DOI 10.17487/RFC3280, April 2002, <<https://www.rfc-editor.org/info/rfc3280>>.
- [RFC4137] Vollbrecht, J., Eronen, P., Petroni, N., and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC 4137, DOI 10.17487/RFC4137, August 2005, <<https://www.rfc-editor.org/info/rfc4137>>.
- [RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, DOI 10.17487/RFC4282, December 2005, <<https://www.rfc-editor.org/info/rfc4282>>.
- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <<https://www.rfc-editor.org/info/rfc4346>>.
- [RFC4851] Cam-Winget, N., McGrew, D., Salowey, J., and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, DOI 10.17487/RFC4851, May 2007, <<https://www.rfc-editor.org/info/rfc4851>>.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <<https://www.rfc-editor.org/info/rfc5077>>.
- [RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <<https://www.rfc-editor.org/info/rfc5191>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <<https://www.rfc-editor.org/info/rfc5247>>.

- [RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <<https://www.rfc-editor.org/info/rfc5281>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <<https://www.rfc-editor.org/info/rfc6733>>.
- [RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <<https://www.rfc-editor.org/info/rfc7170>>.
- [RFC7406] Schulzrinne, H., McCann, S., Bajko, G., Tschofenig, H., and D. Kroesenberg, "Extensions to the Emergency Services Architecture for Dealing With Unauthenticated and Unauthorized Devices", RFC 7406, DOI 10.17487/RFC7406, December 2014, <<https://www.rfc-editor.org/info/rfc7406>>.
- [RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <<https://www.rfc-editor.org/info/rfc7457>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <<https://www.rfc-editor.org/info/rfc7593>>.

- [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>>.
- [RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <<https://www.rfc-editor.org/info/rfc8447>>.
- [RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <<https://www.rfc-editor.org/info/rfc8996>>.
- [I-D.ietf-tls-md5-sha1-deprecate]
Velvindron, L., Moriarty, K., and A. Ghedini, "Deprecating MD5 and SHA-1 signature hashes in (D)TLS 1.2", Work in Progress, Internet-Draft, draft-ietf-tls-md5-sha1-deprecate-09, 20 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-md5-sha1-deprecate-09.txt>>.
- [I-D.ietf-emu-eaptls-cert]
Sethi, M., Mattsson, J., and S. Turner, "Handling Large Certificates and Long Certificate Chains in TLS-based EAP Methods", Work in Progress, Internet-Draft, draft-ietf-emu-eaptls-cert-08, 20 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-emu-eaptls-cert-08.txt>>.
- [I-D.ietf-tls-ticket-requests]
Pauly, T., Schinazi, D., and C. A. Wood, "TLS Ticket Requests", Work in Progress, Internet-Draft, draft-ietf-tls-ticket-requests-07, 3 December 2020, <<https://www.ietf.org/archive/id/draft-ietf-tls-ticket-requests-07.txt>>.
- [I-D.ietf-emu-tls-eap-types]
DeKok, A., "TLS-based EAP types and TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-emu-tls-eap-types-03, 22 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-emu-tls-eap-types-03.txt>>.
- [I-D.ietf-tls-rfc8446bis]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-02, 23 August 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-rfc8446bis-02.txt>>.

Appendix A. Updated References

All the following references in [RFC5216] are updated as specified below when EAP-TLS is used with TLS 1.3.

All references to [RFC2560] are updated to refer to [RFC6960].

All references to [RFC3280] are updated to refer to [RFC5280].
References to Section 4.2.1.13 of [RFC3280] are updated to refer to
Section 4.2.1.12 of [RFC5280].

All references to [RFC4282] are updated to refer to [RFC7542].
References to Section 2.1 of [RFC4282] are updated to refer to
Section 2.2 of [RFC7542].

Acknowledgments

The authors want to thank Bernard Aboba, Jari Arkko, Terry Burton, Alan DeKok, Ari Keraenen, Benjamin Kaduk, Jouni Malinen, Oleg Pekar, Eric Rescorla, Jim Schaad, Joseph Salowey, Martin Thomson, Vesa Torvinen, Hannes Tschofenig, and Heikki Vatiainen for comments and suggestions on the draft. Special thanks to the document shepherd Joseph Salowey.

Contributors

Alan DeKok, FreeRADIUS

Authors' Addresses

John Preuß Mattsson
Ericsson
SE-164 40 Stockholm
Sweden

Email: john.mattsson@ericsson.com

Mohit Sethi
Ericsson
FI-02420 Jorvas
Finland

Email: mohit@piuha.net

Network Working Group
Internet-Draft
Updates: 5448,4187 (if approved)
Intended status: Informational
Expires: November 11, 2021

J. Arkko
V. Lehtovirta
V. Torvinen
Ericsson
P. Eronen
Independent
May 10, 2021

Improved Extensible Authentication Protocol Method for 3GPP Mobile
Network Authentication and Key Agreement (EAP-AKA')
draft-ietf-emu-rfc5448bis-10

Abstract

The 3GPP Mobile Network Authentication and Key Agreement (AKA) is an authentication mechanism for devices wishing to access mobile networks. RFC 4187 (EAP-AKA) made the use of this mechanism possible within the Extensible Authentication Protocol (EAP) framework. RFC 5448 (EAP-AKA') was an improved version of EAP-AKA.

This document is the most recent specification of EAP-AKA', including, for instance, details and references about related to operating EAP-AKA' in 5G networks.

EAP-AKA' differs from EAP-AKA by providing a key derivation function that binds the keys derived within the method to the name of the access network. The key derivation function has been defined in the 3rd Generation Partnership Project (3GPP). EAP-AKA' allows its use in EAP in an interoperable manner. EAP-AKA' also updates the algorithm used in hash functions, as it employs SHA-256 / HMAC-SHA-256 instead of SHA-1 / HMAC-SHA-1 as in EAP-AKA.

This version of EAP-AKA' specification specifies the protocol behaviour for both 4G and 5G deployments, whereas the previous version only did this for 4G.

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 <http://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 November 11, 2021.

Copyright Notice

Copyright (c) 2021 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 (<http://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. Introduction	3
2. Requirements Language	5
3. EAP-AKA'	5
3.1. AT_KDF_INPUT	8
3.2. AT_KDF	11
3.3. Key Derivation	13
3.4. Hash Functions	15
3.4.1. PRF'	15
3.4.2. AT_MAC	15
3.4.3. AT_CHECKCODE	15
3.5. Summary of Attributes for EAP-AKA'	16
4. Bidding Down Prevention for EAP-AKA	18
4.1. Summary of Attributes for EAP-AKA	20
5. Peer Identities	20
5.1. Username Types in EAP-AKA' Identities	20
5.2. Generating Pseudonyms and Fast Re-Authentication Identities	21
5.3. Identifier Usage in 5G	22
5.3.1. Key Derivation	23
5.3.2. EAP Identity Response and EAP-AKA' AT_IDENTITY Attribute	24
6. Exported Parameters	24
7. Security Considerations	25
7.1. Privacy	28

7.2. Discovered Vulnerabilities	30
7.3. Pervasive Monitoring	32
7.4. Security Properties of Binding Network Names	33
8. IANA Considerations	34
8.1. Type Value	34
8.2. Attribute Type Values	34
8.3. Key Derivation Function Namespace	34
9. References	35
9.1. Normative References	35
9.2. Informative References	37
Appendix A. Changes from RFC 5448	40
Appendix B. Changes to RFC 4187	41
Appendix C. Changes from Previous Version of This Draft	41
Appendix D. Importance of Explicit Negotiation	45
Appendix E. Test Vectors	46
Contributors	50
Acknowledgments	51
Authors' Addresses	51

1. Introduction

The 3GPP Mobile Network Authentication and Key Agreement (AKA) is an authentication mechanism for devices wishing to access mobile networks. [RFC4187] (EAP-AKA) made the use of this mechanism possible within the Extensible Authentication Protocol (EAP) framework [RFC3748].

[RFC5448] (EAP-AKA') was an improved version of EAP-AKA. EAP-AKA' was defined in RFC 5448 and updated EAP-AKA RFC 4187.

This document is the most recent specification of EAP-AKA', including, for instance, details and references about related to operating EAP-AKA' in 5G networks. RFC 5448 is not obsolete, but the most recent and fully backwards compatible specification is in this document.

EAP-AKA' is commonly implemented in mobile phones and network equipment. It can be used for authentication to gain network access via Wireless LAN networks and, with 5G, also directly to mobile networks.

EAP-AKA' differs from EAP-AKA by providing a different key derivation function. This function binds the keys derived within the method to the name of the access network. This limits the effects of compromised access network nodes and keys. EAP-AKA' also updates the algorithm used for hash functions.

The EAP-AKA' method employs the derived keys CK' and IK' from the 3GPP specification [TS-3GPP.33.402] and updates the used hash function to SHA-256 [FIPS.180-4] and HMAC to HMAC-SHA-256. Otherwise, EAP-AKA' is equivalent to EAP-AKA. Given that a different EAP method type value is used for EAP-AKA and EAP-AKA', a mutually supported method may be negotiated using the standard mechanisms in EAP [RFC3748].

Note that any change of the key derivation must be unambiguous to both sides in the protocol. That is, it must not be possible to accidentally connect old equipment to new equipment and get the key derivation wrong or attempt to use wrong keys without getting a proper error message. See Appendix D for further information.

Note also that choices in authentication protocols should be secure against bidding down attacks that attempt to force the participants to use the least secure function. See Section 4 for further information.

The changes from RFC 5448 to this specification are as follows:

- o Update the reference on how the Network Name field is constructed in the protocol. This update ensures that EAP-AKA' is compatible with 5G deployments. RFC 5448 referred to the Release 8 version of [TS-3GPP.24.302] and this update points to the first 5G version, Release 15.
- o Specify how EAP and EAP-AKA' use identifiers in 5G. Additional identifiers are introduced in 5G, and for interoperability, it is necessary that the right identifiers are used as inputs in the key derivation. In addition, for identity privacy it is important that when privacy-friendly identifiers in 5G are used, no trackable, permanent identifiers are passed in EAP-AKA' either.
- o Specify session identifiers and other exported parameters, as those were not specified in [RFC5448] despite requirements set forward in [RFC5247] to do so. Also, while [RFC5247] specified session identifiers for EAP-AKA, it only did so for the full authentication case, not for the case of fast re-authentication.
- o Update the requirements on generating pseudonym usernames and fast re-authentication identities to ensure identity privacy.
- o Describe what has been learned about any vulnerabilities in AKA or EAP-AKA'.
- o Describe the privacy and pervasive monitoring considerations related to EAP-AKA'.

- o Summaries of the attributes have been added.

Some of the updates are small. For instance, for the first update, the reference update does not change the 3GPP specification number, only the version. But this reference is crucial in correct calculation of the keys resulting from running the EAP-AKA' method, so an update of the RFC with the newest version pointer may be warranted.

Note: Any further updates in 3GPP specifications that affect, for instance, key derivation is something that EAP-AKA' implementations need to take into account. Upon such updates there will be a need to both update this specification and the implementations.

It is an explicit non-goal of this draft to include any other technical modifications, addition of new features or other changes. The EAP-AKA' base protocol is stable and needs to stay that way. If there are any extensions or variants, those need to be proposed as standalone extensions or even as different authentication methods.

The rest of this specification is structured as follows. Section 3 defines the EAP-AKA' method. Section 4 adds support to EAP-AKA to prevent bidding down attacks from EAP-AKA'. Section 5 specifies requirements regarding the use of peer identities, including how 5G identifiers are used in the EAP-AKA' context. Section 6 specifies what parameters EAP-AKA' exports out of the method. Section 7 explains the security differences between EAP-AKA and EAP-AKA'. Section 8 describes the IANA considerations and Appendix A and Appendix B explains what updates to RFC 5448 EAP-AKA' and RFC 4187 EAP-AKA have been made in this specification. Appendix D explains some of the design rationale for creating EAP-AKA'. Finally, Appendix E provides test vectors.

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.

3. EAP-AKA'

EAP-AKA' is an EAP method that follows the EAP-AKA specification [RFC4187] in all respects except the following:

- o It uses the Type code 0x32, not 0x17 (which is used by EAP-AKA).

- o It carries the AT_KDF_INPUT attribute, as defined in Section 3.1, to ensure that both the peer and server know the name of the access network.
- o It supports key derivation function negotiation via the AT_KDF attribute (Section 3.2) to allow for future extensions.
- o It calculates keys as defined in Section 3.3, not as defined in EAP-AKA.
- o It employs SHA-256 / HMAC-SHA-256, not SHA-1 / HMAC-SHA-1 [FIPS.180-4] (Section 3.4 [RFC2104]).

Figure 1 shows an example of the authentication process. Each message AKA'-Challenge and so on represents the corresponding message from EAP-AKA, but with EAP-AKA' Type code. The definition of these messages, along with the definition of attributes AT_RANDOM, AT_AUTN, AT_MAC, and AT_RES can be found in [RFC4187].

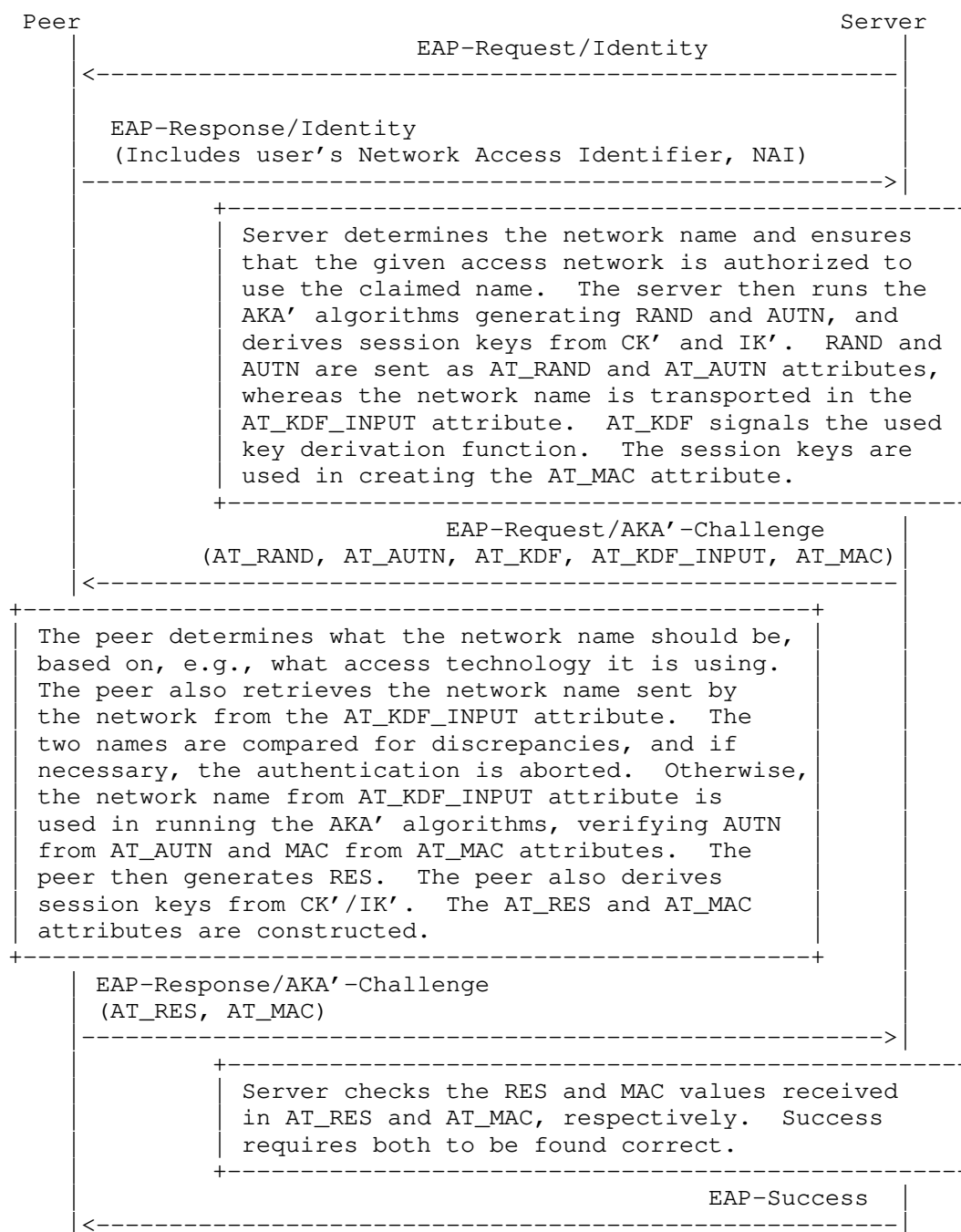
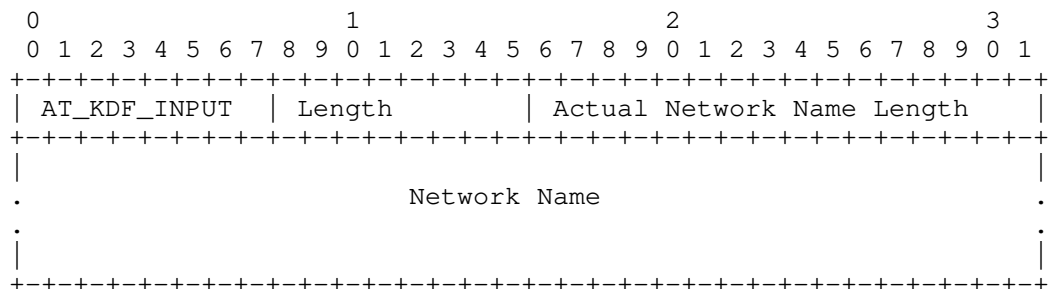


Figure 1: EAP-AKA' Authentication Process

EAP-AKA' can operate on the same credentials as EAP-AKA and employ the same identities. However, EAP-AKA' employs different leading characters than EAP-AKA for the conventions given in Section 4.1.1 of [RFC4187] for International Mobile Subscriber Identifier (IMSI) based usernames. For 4G networks, EAP-AKA' MUST use the leading character "6" (ASCII 36 hexadecimal) instead of "0" for IMSI-based permanent usernames. For 5G networks, leading character "6" is not used for IMSI-based permanent user names. Identifier usage in 5G is specified in Section 5.3. All other usage and processing of the leading characters, usernames, and identities is as defined by EAP-AKA [RFC4187]. For instance, the pseudonym and fast re-authentication usernames need to be constructed so that the server can recognize them. As an example, a pseudonym could begin with a leading "7" character (ASCII 37 hexadecimal) and a fast re-authentication username could begin with "8" (ASCII 38 hexadecimal). Note that a server that implements only EAP-AKA may not recognize these leading characters. According to Section 4.1.4 of [RFC4187], such a server will re-request the identity via the EAP- Request/AKA-Identity message, making obvious to the peer that EAP-AKA and associated identity are expected.

3.1. AT_KDF_INPUT

The format of the AT_KDF_INPUT attribute is shown below.



The fields are as follows:

AT_KDF_INPUT

This is set to 23.

Length

The length of the attribute, calculated as defined in [RFC4187], Section 8.1.

Actual Network Name Length

This is a 2 byte actual length field, needed due to the requirement that the previous field is expressed in multiples of 4 bytes per the usual EAP-AKA rules. The Actual Network Name Length field provides the length of the network name in bytes.

Network Name

This field contains the network name of the access network for which the authentication is being performed. The name does not include any terminating null characters. Because the length of the entire attribute must be a multiple of 4 bytes, the sender pads the name with 1, 2, or 3 bytes of all zero bits when necessary.

Only the server sends the AT_KDF_INPUT attribute. The value is sent as specified in [TS-3GPP.24.302] for both non-3GPP access networks and for 5G access networks. Per [TS-3GPP.33.402], the server always verifies the authorization of a given access network to use a particular name before sending it to the peer over EAP-AKA'. The value of the AT_KDF_INPUT attribute from the server MUST be non-empty, with a greater than zero length in the Actual Network Name Length field. If AT_KDF_INPUT attribute is empty, the peer behaves as if AUTN had been incorrect and authentication fails. See Section 3 and Figure 3 of [RFC4187] for an overview of how authentication failures are handled.

In addition, the peer MAY check the received value against its own understanding of the network name. Upon detecting a discrepancy, the peer either warns the user and continues, or fails the authentication process. More specifically, the peer SHOULD have a configurable policy that it can follow under these circumstances. If the policy indicates that it can continue, the peer SHOULD log a warning message or display it to the user. If the peer chooses to proceed, it MUST use the network name as received in the AT_KDF_INPUT attribute. If the policy indicates that the authentication should fail, the peer behaves as if AUTN had been incorrect and authentication fails.

The Network Name field contains a UTF-8 string. This string MUST be constructed as specified in [TS-3GPP.24.302] for "Access Network Identity". The string is structured as fields separated by colons (:). The algorithms and mechanisms to construct the identity string depend on the used access technology.

On the network side, the network name construction is a configuration issue in an access network and an authorization check in the authentication server. On the peer, the network name is constructed

based on the local observations. For instance, the peer knows which access technology it is using on the link, it can see information in a link-layer beacon, and so on. The construction rules specify how this information maps to an access network name. Typically, the network name consists of the name of the access technology, or the name of the access technology followed by some operator identifier that was advertised in a link-layer beacon. In all cases, [TS-3GPP.24.302] is the normative specification for the construction in both the network and peer side. If the peer policy allows running EAP-AKA' over an access technology for which that specification does not provide network name construction rules, the peer SHOULD rely only on the information from the AT_KDF_INPUT attribute and not perform a comparison.

If a comparison of the locally determined network name and the one received over EAP-AKA' is performed on the peer, it MUST be done as follows. First, each name is broken down to the fields separated by colons. If one of the names has more colons and fields than the other one, the additional fields are ignored. The remaining sequences of fields are compared, and they match only if they are equal character by character. This algorithm allows a prefix match where the peer would be able to match "", "FOO", and "FOO:BAR" against the value "FOO:BAR" received from the server. This capability is important in order to allow possible updates to the specifications that dictate how the network names are constructed. For instance, if a peer knows that it is running on access technology "FOO", it can use the string "FOO" even if the server uses an additional, more accurate description, e.g., "FOO:BAR", that contains more information.

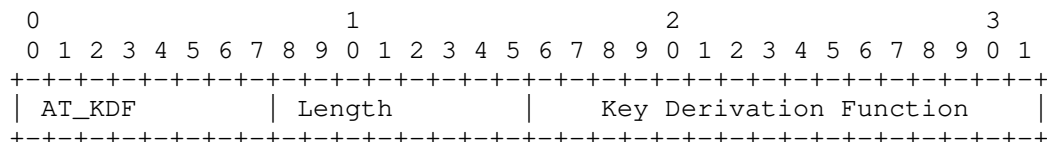
The allocation procedures in [TS-3GPP.24.302] ensure that conflicts potentially arising from using the same name in different types of networks are avoided. The specification also has detailed rules about how a client can determine these based on information available to the client, such as the type of protocol used to attach to the network, beacons sent out by the network, and so on. Information that the client cannot directly observe (such as the type or version of the home network) is not used by this algorithm.

The AT_KDF_INPUT attribute MUST be sent and processed as explained above when AT_KDF attribute has the value 1. Future definitions of new AT_KDF values MUST define how this attribute is sent and processed.

3.2. AT_KDF

AT_KDF is an attribute that the server uses to reference a specific key derivation function. It offers a negotiation capability that can be useful for future evolution of the key derivation functions.

The format of the AT_KDF attribute is shown below.



The fields are as follows:

AT_KDF

This is set to 24.

Length

The length of the attribute, calculated as defined in [RFC4187], Section 8.1. For AT_KDF, the Length field MUST be set to 1.

Key Derivation Function

An enumerated value representing the key derivation function that the server (or peer) wishes to use. Value 1 represents the default key derivation function for EAP-AKA', i.e., employing CK' and IK' as defined in Section 3.3.

Servers MUST send one or more AT_KDF attributes in the EAP-Request/ AKA'-Challenge message. These attributes represent the desired functions ordered by preference, the most preferred function being the first attribute.

Upon receiving a set of these attributes, if the peer supports and is willing to use the key derivation function indicated by the first attribute, the function is taken into use without any further negotiation. However, if the peer does not support this function or is unwilling to use it, it does not process the received EAP-Request/ AKA'-Challenge in any way except by responding with the EAP-Response/ AKA'-Challenge message that contains only one attribute, AT_KDF with the value set to the selected alternative. If there is no suitable alternative, the peer behaves as if AUTN had been incorrect and authentication fails (see Figure 3 of [RFC4187]). The peer fails the

authentication also if there are any duplicate values within the list of AT_KDF attributes (except where the duplication is due to a request to change the key derivation function; see below for further information).

Upon receiving an EAP-Response/AKA'-Challenge with AT_KDF from the peer, the server checks that the suggested AT_KDF value was one of the alternatives in its offer. The first AT_KDF value in the message from the server is not a valid alternative since the peer should have accepted it without further negotiation. If the peer has replied with the first AT_KDF value, the server behaves as if AT_MAC of the response had been incorrect and fails the authentication. For an overview of the failed authentication process in the server side, see Section 3 and Figure 2 of [RFC4187]. Otherwise, the server re-sends the EAP-Response/AKA'-Challenge message, but adds the selected alternative to the beginning of the list of AT_KDF attributes and retains the entire list following it. Note that this means that the selected alternative appears twice in the set of AT_KDF values. Responding to the peer's request to change the key derivation function is the only legal situation where such duplication may occur.

When the peer receives the new EAP-Request/AKA'-Challenge message, it MUST check that the requested change, and only the requested change, occurred in the list of AT_KDF attributes. If so, it continues with processing the received EAP-Request/AKA'-Challenge as specified in [RFC4187] and Section 3.1 of this document. If not, it behaves as if AT_MAC had been incorrect and fails the authentication. If the peer receives multiple EAP-Request/AKA'-Challenge messages with differing AT_KDF attributes without having requested negotiation, the peer MUST behave as if AT_MAC had been incorrect and fail the authentication.

Note that the peer may also request sequence number resynchronization [RFC4187]. This happens after AT_KDF negotiation has already completed. That is, the EAP-Request/AKA'-Challenge and, possibly, the EAP-Response/AKA'-Challenge message are exchanged first to come up with a mutually acceptable key derivation function, and only then the possible AKA'-Synchronization-Failure message is sent. The AKA'-Synchronization-Failure message is sent as a response to the newly received EAP-Request/AKA'-Challenge which is the last message of the AT_KDF negotiation. Note that if the first proposed KDF is acceptable, then last message is at the same time the first EAP-Request/AKA'-Challenge message. The AKA'-Synchronization-Failure message MUST contain the AUTS parameter as specified in [RFC4187] and a copy the AT_KDF attributes as they appeared in the last message of the AT_KDF negotiation. If the AT_KDF attributes are found to differ from their earlier values, the peer and server MUST behave as if AT_MAC had been incorrect and fail the authentication.

3.3. Key Derivation

Both the peer and server MUST derive the keys as follows.

AT_KDF parameter has the value 1

In this case, MK is derived and used as follows:

```
MK = PRF'(IK'|CK',"EAP-AKA'"|Identity)
K_encr = MK[0..127]
K_aut  = MK[128..383]
K_re   = MK[384..639]
MSK    = MK[640..1151]
EMSK   = MK[1152..1663]
```

Here [n..m] denotes the substring from bit n to m, including bits n and m. PRF' is a new pseudo-random function specified in Section 3.4. The first 1664 bits from its output are used for K_encr (encryption key, 128 bits), K_aut (authentication key, 256 bits), K_re (re-authentication key, 256 bits), MSK (Master Session Key, 512 bits), and EMSK (Extended Master Session Key, 512 bits). These keys are used by the subsequent EAP-AKA' process. K_encr is used by the AT_ENCR_DATA attribute, and K_aut by the AT_MAC attribute. K_re is used later in this section. MSK and EMSK are outputs from a successful EAP method run [RFC3748].

IK' and CK' are derived as specified in [TS-3GPP.33.402]. The functions that derive IK' and CK' take the following parameters: CK and IK produced by the AKA algorithm, and value of the Network Name field comes from the AT_KDF_INPUT attribute (without length or padding).

The value "EAP-AKA'" is an eight-characters-long ASCII string. It is used as is, without any trailing NUL characters.

Identity is the peer identity as specified in Section 7 of [RFC4187], and Section 5.3.2 in this document for the 5G cases.

When the server creates an AKA challenge and corresponding AUTN, CK, CK', IK, and IK' values, it MUST set the Authentication Management Field (AMF) separation bit to 1 in the AKA algorithm [TS-3GPP.33.102]. Similarly, the peer MUST check that the AMF separation bit is set to 1. If the bit is not set to 1, the peer behaves as if the AUTN had been incorrect and fails the authentication.

On fast re-authentication, the following keys are calculated:

```
MK = PRF'(K_re, "EAP-AKA' re-auth"|Identity|counter|NONCE_S)
MSK  = MK[0..511]
EMSK = MK[512..1023]
```

MSK and EMSK are the resulting 512-bit keys, taking the first 1024 bits from the result of PRF'. Note that K_encr and K_aut are not re-derived on fast re-authentication. K_re is the re-authentication key from the preceding full authentication and stays unchanged over any fast re-authentication(s) that may happen based on it. The value "EAP-AKA' re-auth" is a sixteen-characters-long ASCII string, again represented without any trailing NUL characters. Identity is the fast re-authentication identity, counter is the value from the AT_COUNTER attribute, NONCE_S is the nonce value from the AT_NONCE_S attribute, all as specified in Section 7 of [RFC4187]. To prevent the use of compromised keys in other places, it is forbidden to change the network name when going from the full to the fast re-authentication process. The peer SHOULD NOT attempt fast re-authentication when it knows that the network name in the current access network is different from the one in the initial, full authentication. Upon seeing a re-authentication request with a changed network name, the server SHOULD behave as if the re-authentication identifier had been unrecognized, and fall back to full authentication. The server observes the change in the name by comparing where the fast re-authentication and full authentication EAP transactions were received at the Authentication, Authorization, and Accounting (AAA) protocol level.

AT_KDF has any other value

Future variations of key derivation functions may be defined, and they will be represented by new values of AT_KDF. If the peer does not recognize the value, it cannot calculate the keys and behaves as explained in Section 3.2.

AT_KDF is missing

The peer behaves as if the AUTN had been incorrect and MUST fail the authentication.

If the peer supports a given key derivation function but is unwilling to perform it for policy reasons, it refuses to calculate the keys and behaves as explained in Section 3.2.

3.4. Hash Functions

EAP-AKA' uses SHA-256 / HMAC-SHA-256, not SHA-1 / HMAC-SHA-1 (see [FIPS.180-4] [RFC2104]) as in EAP-AKA. This requires a change to the pseudo-random function (PRF) as well as the AT_MAC and AT_CHECKCODE attributes.

3.4.1. PRF'

The PRF' construction is the same one IKEv2 uses (see Section 2.13 of [RFC7296]; this is the same function as was defined [RFC4306] that RFC 5448 referred to). The function takes two arguments. K is a 256-bit value and S is a byte string of arbitrary length. PRF' is defined as follows:

$$\text{PRF}'(K, S) = T1 \mid T2 \mid T3 \mid T4 \mid \dots$$

where:

T1 = HMAC-SHA-256 (K, S | 0x01)
T2 = HMAC-SHA-256 (K, T1 | S | 0x02)
T3 = HMAC-SHA-256 (K, T2 | S | 0x03)
T4 = HMAC-SHA-256 (K, T3 | S | 0x04)
...

PRF' produces as many bits of output as is needed. HMAC-SHA-256 is the application of HMAC [RFC2104] to SHA-256.

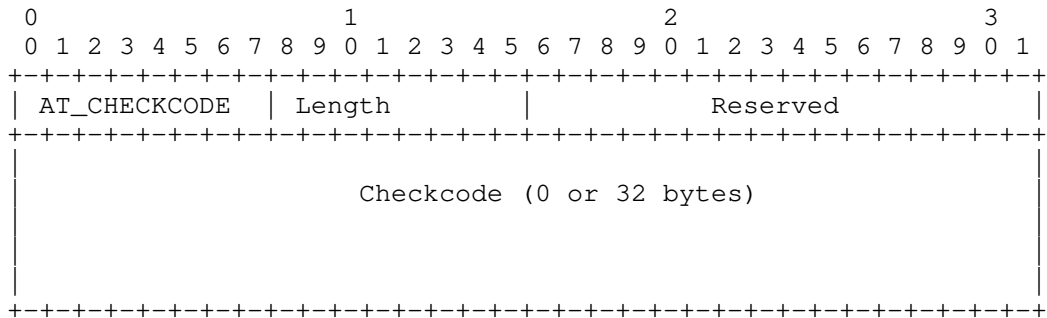
3.4.2. AT_MAC

When used within EAP-AKA', the AT_MAC attribute is changed as follows. The MAC algorithm is HMAC-SHA-256-128, a keyed hash value. The HMAC-SHA-256-128 value is obtained from the 32-byte HMAC-SHA-256 value by truncating the output to the first 16 bytes. Hence, the length of the MAC is 16 bytes.

Otherwise, the use of AT_MAC in EAP-AKA' follows Section 10.15 of [RFC4187].

3.4.3. AT_CHECKCODE

When used within EAP-AKA', the AT_CHECKCODE attribute is changed as follows. First, a 32-byte value is needed to accommodate a 256-bit hash output:



Second, the checkcode is a hash value, calculated with SHA-256 [FIPS.180-4], over the data specified in Section 10.13 of [RFC4187].

3.5. Summary of Attributes for EAP-AKA'

Table 1 provides a guide to which attributes may be found in which kinds of messages, and in what quantity.

Messages are denoted with numbers in parentheses as follows:

- (1) EAP-Request/AKA-Identity,
- (2) EAP-Response/AKA-Identity,
- (3) EAP-Request/AKA-Challenge,
- (4) EAP-Response/AKA-Challenge,
- (5) EAP-Request/AKA-Notification,
- (6) EAP-Response/AKA-Notification,
- (7) EAP-Response/AKA-Client-Error
- (8) EAP-Request/AKA-Reauthentication,
- (9) EAP-Response/AKA-Reauthentication,
- (10) EAP-Response/AKA-Authentication-Reject, and
- (11) EAP-Response/AKA-Synchronization-Failure.

The column denoted with "E" indicates whether the attribute is a nested attribute that MUST be included within AT_ENCR_DATA.

In addition, the numbered columns indicate the quantity of the attribute within the message as follows:

"0" indicates that the attribute MUST NOT be included in the message,

"1" indicates that the attribute MUST be included in the message,

"0-1" indicates that the attribute is sometimes included in the message,

"0+" indicates that zero or more copies of the attribute MAY be included in the message,

"1+" indicates that there MUST be at least one attribute in the message but more than one MAY be included in the message, and

"0*" indicates that the attribute is not included in the message in cases specified in this document, but MAY be included in the future versions of the protocol.

The attribute table is shown below. The table is largely the same as in the EAP-AKA attribute table ([RFC4187] Section 10.1), but changes how many times AT_MAC may appear in EAP-Response/AKA'-Challenge message as it does not appear there when AT_KDF has to be sent from the peer to the server. The table also adds the AT_KDF and AT_KDF_INPUT attributes.

Attribute	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	E
AT_PERMANENT_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_ANY_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_FULLAUTH_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_IDENTITY	0	0-1	0	0	0	0	0	0	0	0	0	N
AT_RAND	0	0	1	0	0	0	0	0	0	0	0	N
AT_AUTN	0	0	1	0	0	0	0	0	0	0	0	N
AT_RES	0	0	0	1	0	0	0	0	0	0	0	N
AT_AUTS	0	0	0	0	0	0	0	0	0	0	1	N
AT_NEXT_PSEUDONYM	0	0	0-1	0	0	0	0	0	0	0	0	Y
AT_NEXT_REAUTH_ID	0	0	0-1	0	0	0	0	0-1	0	0	0	Y
AT_IV	0	0	0-1	0*	0-1	0-1	0	1	1	0	0	N
AT_ENCR_DATA	0	0	0-1	0*	0-1	0-1	0	1	1	0	0	N
AT_PADDING	0	0	0-1	0*	0-1	0-1	0	0-1	0-1	0	0	Y
AT_CHECKCODE	0	0	0-1	0-1	0	0	0	0-1	0-1	0	0	N
AT_RESULT_IND	0	0	0-1	0-1	0	0	0	0-1	0-1	0	0	N
AT_MAC	0	0	1	0-1	0-1	0-1	0	1	1	0	0	N
AT_COUNTER	0	0	0	0	0-1	0-1	0	1	1	0	0	Y
AT_COUNTER_TOO_SMALL	0	0	0	0	0	0	0	0	0-1	0	0	Y
AT_NONCE_S	0	0	0	0	0	0	0	1	0	0	0	Y
AT_NOTIFICATION	0	0	0	0	1	0	0	0	0	0	0	N
AT_CLIENT_ERROR_CODE	0	0	0	0	0	0	1	0	0	0	0	N
AT_KDF	0	0	1+	0+	0	0	0	0	0	0	1+	N
AT_KDF_INPUT	0	0	1	0	0	0	0	0	0	0	0	N

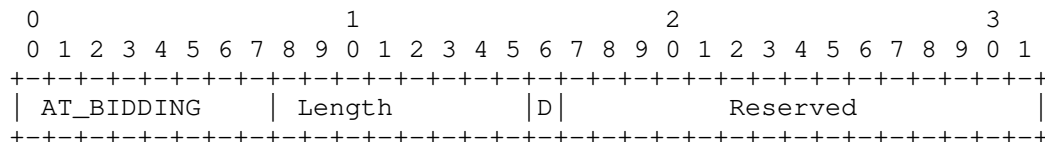
Table 1: The attribute table

4. Bidding Down Prevention for EAP-AKA

As discussed in [RFC3748], negotiation of methods within EAP is insecure. That is, a man-in-the-middle attacker may force the endpoints to use a method that is not the strongest that they both support. This is a problem, as we expect EAP-AKA and EAP-AKA' to be negotiated via EAP.

In order to prevent such attacks, this RFC specifies a new mechanism for EAP-AKA that allows the endpoints to securely discover the capabilities of each other. This mechanism comes in the form of the AT_BIDDING attribute. This allows both endpoints to communicate their desire and support for EAP-AKA' when exchanging EAP-AKA messages. This attribute is not included in EAP-AKA' messages. It is only included in EAP-AKA messages. (Those messages are protected with the AT_MAC attribute.) This approach is based on the assumption that EAP-AKA' is always preferable (see Section 7). If during the EAP-AKA authentication process it is discovered that both endpoints would have been able to use EAP-AKA', the authentication process SHOULD be aborted, as a bidding down attack may have happened.

The format of the AT_BIDDING attribute is shown below.



The fields are as follows:

AT_BIDDING

This is set to 136.

Length

The length of the attribute, calculated as defined in [RFC4187], Section 8.1. For AT_BIDDING, the Length MUST be set to 1.

D

This bit is set to 1 if the sender supports EAP-AKA', is willing to use it, and prefers it over EAP-AKA. Otherwise, it should be set to zero.

Reserved

This field MUST be set to zero when sent and ignored on receipt.

The server sends this attribute in the EAP-Request/AKA-Challenge message. If the peer supports EAP-AKA', it compares the received value to its own capabilities. If it turns out that both the server and peer would have been able to use EAP-AKA' and preferred it over EAP-AKA, the peer behaves as if AUTN had been incorrect and fails the authentication (see Figure 3 of [RFC4187]). A peer not supporting EAP-AKA' will simply ignore this attribute. In all cases, the attribute is protected by the integrity mechanisms of EAP-AKA, so it cannot be removed by a man-in-the-middle attacker.

Note that we assume (Section 7) that EAP-AKA' is always stronger than EAP-AKA. As a result, this specification does not provide protection against bidding "down" attacks in the other direction, i.e., attackers forcing the endpoints to use EAP-AKA'.

4.1. Summary of Attributes for EAP-AKA

The appearance of the AT_BIDDING attribute in EAP-AKA exchanges is shown below, using the notation from Section 3.5:

Attribute	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	E
AT_BIDDING	0	0	1	0	0	0	0	0	0	0	0	N

5. Peer Identities

EAP-AKA' peer identities are as specified in [RFC4187] Section 4.1, with the addition of some requirements specified in this section.

EAP-AKA' includes optional identity privacy support that can be used to hide the cleartext permanent identity and thereby make the subscriber's EAP exchanges untraceable to eavesdroppers. EAP-AKA' can also use the privacy friendly identifiers specified for 5G networks.

The permanent identity is usually based on the IMSI. Exposing the IMSI is undesirable, because as a permanent identity it is easily trackable. In addition, since IMSIs may be used in other contexts as well, there would be additional opportunities for such tracking.

In EAP-AKA', identity privacy is based on temporary usernames, or pseudonym usernames. These are similar to but separate from the Temporary Mobile Subscriber Identities (TMSI) that are used on cellular networks.

5.1. Username Types in EAP-AKA' Identities

Section 4.1.1.3 of [RFC4187] specified that there are three types of usernames: permanent, pseudonym, and fast re-authentication usernames. This specification extends this definition as follows. There are four types of usernames:

(1) Regular usernames. These are external names given to EAP-AKA' peers. The regular usernames are further subdivided into to categories:

(a) Permanent usernames, for instance IMSI-based usernames.

(b) Privacy-friendly temporary usernames, for instance 5G GUTI (5G Globally Unique Temporary Identifier) or 5G privacy identifiers (see Section 5.3.2), for instance SUCI (Subscription Concealed Identifier).

(2) EAP-AKA' pseudonym usernames. For example, 2s7ah6n9q@example.com might be a valid pseudonym identity. In this example, 2s7ah6n9q is the pseudonym username.

(3) EAP-AKA' fast re-authentication usernames. For example, 43953754@example.com might be a valid fast re-authentication identity and 43953754 the fast re-authentication username.

The permanent, privacy-friendly temporary, and pseudonym usernames are only used on full authentication, and fast re-authentication usernames only on fast re-authentication. Unlike permanent usernames and pseudonym usernames, privacy friendly temporary usernames and fast re-authentication usernames are one-time identifiers, which are not re-used across EAP exchanges.

5.2. Generating Pseudonyms and Fast Re-Authentication Identities

This section provides some additional guidance for implementations for producing secure pseudonyms and fast re-authentication identities. It does not impact backwards compatibility, because each server consumes only the identities it itself generates. However, adherence to the guidance will provide better security.

As specified by [RFC4187] Section 4.1.1.7, pseudonym usernames and fast re-authentication identities are generated by the EAP server, in an implementation-dependent manner. RFC 4187 provides some general requirements on how these identities are transported, how they map to the NAI syntax, how they are distinguished from each other, and so on.

However, to enhance privacy some additional requirements need to be applied.

The pseudonym usernames and fast re-authentication identities MUST be generated in a cryptographically secure way so that that it is computationally infeasible for an attacker to differentiate two identities belonging to the same user from two identities belonging to different users. This can be achieved, for instance, by using random or pseudo-random identifiers such as random byte strings or ciphertexts. See also [RFC4086] for guidance on random number generation.

Note that the pseudonym and fast re-authentication usernames also MUST NOT include substrings that can be used to relate the username to a particular entity or a particular permanent identity. For instance, the usernames can not include any subscriber-identifying part of an IMSI or other permanent identifier. Similarly, no part of the username can be formed by a fixed mapping that stays the same

across multiple different pseudonyms or fast re-authentication identities for the same subscriber.

When the identifier used to identify a subscriber in an EAP-AKA' authentication exchange is a privacy-friendly identifier that is used only once, the EAP-AKA' peer MUST NOT use a pseudonym provided in that authentication exchange in subsequent exchanges more than once. To ensure that this does not happen, EAP-AKA' server MAY decline to provide a pseudonym in such authentication exchanges. An important case where such privacy-friendly identifiers are used is in 5G networks (see Section 5.3).

5.3. Identifier Usage in 5G

In EAP-AKA', the peer identity may be communicated to the server in one of three ways:

- o As a part of link layer establishment procedures, externally to EAP.
- o With the EAP-Response/Identity message in the beginning of the EAP exchange, but before the selection of EAP-AKA'.
- o Transmitted from the peer to the server using EAP-AKA' messages instead of EAP-Response/Identity. In this case, the server includes an identity requesting attribute (AT_ANY_ID_REQ, AT_FULLAUTH_ID_REQ or AT_PERMANENT_ID_REQ) in the EAP-Request/AKA-Identity message; and the peer includes the AT_IDENTITY attribute, which contains the peer's identity, in the EAP-Response/AKA-Identity message.

The identity carried above may be a permanent identity, privacy friendly identity, pseudonym identity, or fast re-authentication identity as defined in Section 5.1.

5G supports the concept of privacy identifiers, and it is important for interoperability that the right type of identifier is used.

5G defines the Subscription Permanent Identifier (SUPI) and Subscription Concealed Identifier (SUCI) [TS-3GPP.23.501] [TS-3GPP.33.501] [TS-3GPP.23.003]. SUPI is globally unique and allocated to each subscriber. However, it is only used internally in the 5G network, and is privacy sensitive. The SUCI is a privacy preserving identifier containing the concealed SUPI, using public key cryptography to encrypt the SUPI.

Given the choice between these two types of identifiers, EAP-AKA' ensures interoperability as follows:

- o Where identifiers are used within EAP-AKA' -- such as key derivation -- specify what values exactly should be used, to avoid ambiguity (see Section 5.3.1).
- o Where identifiers are carried within EAP-AKA' packets -- such as in the AT_IDENTITY attribute -- specify which identifiers should be filled in (see Section 5.3.2).

In 5G, the normal mode of operation is that identifiers are only transmitted outside EAP. However, in a system involving terminals from many generations and several connectivity options via 5G and other mechanisms, implementations and the EAP-AKA' specification need to prepare for many different situations, including sometimes having to communicate identities within EAP.

The following sections clarify which identifiers are used and how.

5.3.1. Key Derivation

In EAP-AKA', the peer identity is used in the Section 3.3 key derivation formula.

The identity needs to be represented in exact correct format for the key derivation formula to produce correct results.

If the AT_KDF_INPUT parameter contains the prefix "5G:", the AT_KDF parameter has the value 1, and this authentication is not a fast re-authentication, then the peer identity used in the key derivation MUST be as specified in Annex F.3 of [TS-3GPP.33.501] and Clause 2.2 of [TS-3GPP.23.003]. This is in contrast to [RFC5448], which used the identity as communicated in EAP and represented as a NAI. Also, in contrast to [RFC5448], in 5G EAP-AKA' does not use the "0" or "6" prefix in front of the identifier.

For an example of the format of the identity, see Clause 2.2 of [TS-3GPP.23.003].

In all other cases, the following applies:

The identity used in the key derivation formula MUST be exactly the one sent in EAP-AKA' AT_IDENTITY attribute, if one was sent, regardless of the kind of identity that it may have been. If no AT_IDENTITY was sent, the identity MUST be the exactly the one sent in the generic EAP Identity exchange, if one was made.

If no identity was communicated inside EAP, then the identity is the one communicated outside EAP in link layer messaging.

In this case, the used identity MUST be the identity most recently communicated by the peer to the network, again regardless of what type of identity it may have been.

5.3.2. EAP Identity Response and EAP-AKA' AT_IDENTITY Attribute

The EAP authentication option is only available in 5G when the new 5G core network is also in use. However, in other networks an EAP-AKA' peer may be connecting to other types of networks and existing equipment.

When the EAP server is in a 5G network, the 5G procedures for EAP-AKA' apply. When EAP server is defined to be in a 5G network is specified in [TS-3GPP.33.501].

Note: Currently, the following conditions are specified: when the EAP peer uses the 5G Non-Access Stratum (NAS) protocol [TS-3GPP.24.501] or when the EAP peer attaches to a network that advertises 5G connectivity without NAS [TS-3GPP.23.501]. Possible future conditions may also be specified by 3GPP.

When the 5G procedures for EAP-AKA' apply, EAP identity exchanges are generally not used as the identity is already made available on previous link layer exchanges.

In this situation, the EAP Identity Response and EAP-AKA' AT_IDENTITY attribute are handled as specified in Annex F.2 of [TS-3GPP.33.501].

When used in EAP-AKA', the format of the SUCI MUST be as specified in [TS-3GPP.23.003] Section 28.7.3, with the semantics defined in [TS-3GPP.23.003] Section 2.2B. Also, in contrast to [RFC5448], in 5G EAP-AKA' does not use the "0" or "6" prefix in front of the identifier.

For an example of an IMSI in NAI format, see [TS-3GPP.23.003] Section 28.7.3.

Otherwise, the peer SHOULD employ IMSI, SUPI, or a NAI as it is configured to use.

6. Exported Parameters

When not using fast re-authentication, the EAP-AKA' Session-Id is the concatenation of the EAP Type Code (0x32, one byte) with the contents of the RAND field from the AT_RAND attribute, followed by the contents of the AUTN field in the AT_AUTN attribute :

Session-Id = 0x32 || RAND || AUTN

When using fast re-authentication, the EAP-AKA' Session-Id is the concatenation of the EAP Type Code (0x32) with the contents of the NONCE_S field from the AT_NONCE_S attribute, followed by the contents of the MAC field from the AT_MAC attribute from EAP-Request/AKA-Reauthentication:

$$\text{Session-Id} = 0x32 \parallel \text{NONCE_S} \parallel \text{MAC}$$

The Peer-Id is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length bytes from the beginning. Note that the contents are used as they are transmitted, regardless of whether the transmitted identity was a permanent, pseudonym, or fast EAP re-authentication identity. If no AT_IDENTITY attribute was exchanged, the exported Peer-Id is the identity provided from the EAP Identity Response packet. If no EAP Identity Response was provided either, the exported Peer-Id is the null string (zero length).

The Server-Id is the null string (zero length).

7. Security Considerations

A summary of the security properties of EAP-AKA' follows. These properties are very similar to those in EAP-AKA. We assume that HMAC SHA-256 is at least as secure as HMAC SHA-1 (see also [RFC6194]). This is called the SHA-256 assumption in the remainder of this section. Under this assumption, EAP-AKA' is at least as secure as EAP-AKA.

If the AT_KDF attribute has value 1, then the security properties of EAP-AKA' are as follows:

Protected ciphersuite negotiation

EAP-AKA' has no ciphersuite negotiation mechanisms. It does have a negotiation mechanism for selecting the key derivation functions. This mechanism is secure against bidding down attacks from EAP-AKA' to EAP-AKA. The negotiation mechanism allows changing the offered key derivation function, but the change is visible in the final EAP-Request/AKA'-Challenge message that the server sends to the peer. This message is authenticated via the AT_MAC attribute, and carries both the chosen alternative and the initially offered list. The peer refuses to accept a change it did not initiate. As a result, both parties are aware that a change is being made and what the original offer was.

Per assumptions in Section 4, there is no protection against bidding down attacks from EAP-AKA to EAP-AKA', should EAP-AKA'

somehow be considered less secure some day than EAP-AKA. Such protection was not provided in RFC 5448 implementations and consequently neither does this specification provide it. If such support is needed, it would have to be added as a separate new feature.

In general, it is expected that the current negotiation capabilities in EAP-AKA' are sufficient for some types of extensions, including adding Perfect Forward Secrecy ([I-D.ietf-emu-aka-pfs]) and perhaps others. But as with how EAP-AKA' itself came about, some larger changes may require a new EAP method type. One example of such change would be the introduction of new algorithms.

Mutual authentication

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Integrity protection

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good (most likely better) as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details. The only difference is that a stronger hash algorithm and keyed MAC, SHA-256 / HMAC-SHA-256, is used instead of SHA-1 / HMAC-SHA-1.

Replay protection

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Confidentiality

The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Key derivation

EAP-AKA' supports key derivation with an effective key strength against brute force attacks equal to the minimum of the length of the derived keys and the length of the AKA base key, i.e., 128 bits or more. The key hierarchy is specified in Section 3.3.

The Transient EAP Keys used to protect EAP-AKA packets (K_{encr} , K_{aut} , K_{re}), the MSK, and the EMSK are cryptographically separate. If we make the assumption that SHA-256 behaves as a pseudo-random function, an attacker is incapable of deriving any non-trivial information about any of these keys based on the other keys. An attacker also cannot calculate the pre-shared secret from IK , CK , IK' , CK' , K_{encr} , K_{aut} , K_{re} , MSK, or EMSK by any practically feasible means.

EAP-AKA' adds an additional layer of key derivation functions within itself to protect against the use of compromised keys. This is discussed further in Section 7.4.

EAP-AKA' uses a pseudo-random function modeled after the one used in IKEv2 [RFC7296] together with SHA-256.

Key strength

See above.

Dictionary attack resistance

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Fast reconnect

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details. Note that implementations MUST prevent performing a fast reconnect across method types.

Cryptographic binding

Note that this term refers to a very specific form of binding, something that is performed between two layers of authentication. It is not the same as the binding to a particular network name. The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect, i.e., as it is not a tunnel method, this property is not applicable to it. Refer to [RFC4187], Section 12 for further details.

Session independence

The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Fragmentation

The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Channel binding

EAP-AKA', like EAP-AKA, does not provide channel bindings as they're defined in [RFC3748] and [RFC5247]. New skippable attributes can be used to add channel binding support in the future, if required.

However, including the Network Name field in the AKA' algorithms (which are also used for other purposes than EAP-AKA') provides a form of cryptographic separation between different network names, which resembles channel bindings. However, the network name does not typically identify the EAP (pass-through) authenticator. See Section 7.4 for more discussion.

7.1. Privacy

[RFC6973] suggests that the privacy considerations of IETF protocols be documented.

The confidentiality properties of EAP-AKA' itself have been discussed above under "Confidentiality".

EAP-AKA' uses several different types of identifiers to identify the authenticating peer. It is strongly RECOMMENDED to use the privacy-friendly temporary or hidden identifiers, i.e., the 5G GUTI or SUCI, pseudonym usernames, and fast re-authentication usernames. The use of permanent identifiers such as the IMSI or SUPI may lead to an ability to track the peer and/or user associated with the peer. The use of permanent identifiers such as the IMSI or SUPI is strongly NOT RECOMMENDED.

As discussed in Section 5.3, when authenticating to a 5G network, only the SUCI identifier is normally used. The use of EAP-AKA' pseudonyms in this situation is at best limited, because the SUCI already provides a stronger mechanism. In fact, the re-use of the same pseudonym multiple times will result in a tracking opportunity for observers that see the pseudonym pass by. To avoid this, the peer and server need to follow the guidelines given in Section 5.2.

When authenticating to a 5G network, per Section 5.3.1, both the EAP-AKA' peer and server need to employ the permanent identifier, SUPI, as an input to key derivation. However, this use of the SUPI is only internal. As such, the SUPI need not be communicated in EAP messages. Therefore, SUPI MUST NOT be communicated in EAP-AKA' when authenticating to a 5G network.

While the use of SUCI in 5G networks generally provides identity privacy, this is not true if the null-scheme encryption is used to construct the SUCI (see [TS-3GPP.33.501] Annex C). The use of this scheme turns the use of SUCI equivalent to the use of SUPI or IMSI. The use of the null scheme is NOT RECOMMENDED where identity privacy is important.

The use of fast re-authentication identities when authenticating to a 5G network does not have the same problems as the use of pseudonyms, as long as the 5G authentication server generates the fast re-authentication identifiers in a proper manner specified in Section 5.2.

Outside 5G, the peer can freely choose between the use of permanent, pseudonym, or fast re-authentication identifiers:

- o A peer that has not yet performed any EAP-AKA' exchanges does not typically have a pseudonym available. If the peer does not have a pseudonym available, then the privacy mechanism cannot be used, and the permanent identity will have to be sent in the clear.

The terminal SHOULD store the pseudonym in non-volatile memory so that it can be maintained across reboots. An active attacker that impersonates the network may use the AT_PERMANENT_ID_REQ attribute ([RFC4187] Section 4.1.2) to learn the subscriber's IMSI. However, as discussed in [RFC4187] Section 4.1.2, the terminal can refuse to send the cleartext permanent identity if it believes that the network should be able to recognize the pseudonym.

- o When pseudonyms and fast re-authentication identities are used, the peer relies on the properly created identifiers by the server.

It is essential that an attacker cannot link a privacy-friendly identifier to the user in any way or determine that two identifiers belong to the same user as outlined in Section 5.2. The pseudonym usernames and fast re-authentication identities MUST NOT be used for other purposes (e.g., in other protocols).

If the peer and server cannot guarantee that SUCI can be used or pseudonyms will be available, generated properly, and maintained reliably, and identity privacy is required then additional protection

from an external security mechanism such as tunneled EAP methods such as TLS [RFC5281] or TEAP [RFC7170] may be used. The benefits and the security considerations of using an external security mechanism with EAP-AKA are beyond the scope of this document.

Finally, as with other EAP methods, even when privacy-friendly identifiers or EAP tunneling is used, typically the domain part of an identifier (e.g., the home operator) is visible to external parties.

7.2. Discovered Vulnerabilities

There have been no published attacks that violate the primary secrecy or authentication properties defined for Authentication and Key Agreement (AKA) under the originally assumed trust model. The same is true of EAP-AKA'.

However, there have been attacks when a different trust model is in use, with characteristics not originally provided by the design, or when participants in the protocol leak information to outsiders on purpose, and there have been some privacy-related attacks.

For instance, the original AKA protocol does not prevent supplying keys by an insider to a third party as done in, e.g., by Mjolsnes and Tsay in [MT2012] where a serving network lets an authentication run succeed, but then misuses the session keys to send traffic on the authenticated user's behalf. This particular attack is not different from any on-path entity (such as a router) pretending to send traffic, but the general issue of insider attacks can be a problem, particularly in a large group of collaborating operators.

Another class of attacks is the use of tunneling of traffic from one place to another, e.g., as done by Zhang and Fang in [ZF2005] to leverage security policy differences between different operator networks, for instance. To gain something in such an attack, the attacker needs to trick the user into believing it is in another location. If policies between different locations differ, for instance, in some location it is not required to encrypt all payload traffic, the attacker may trick the user into opening a vulnerability. As an authentication mechanism, EAP-AKA' is not directly affected by most such attacks. EAP-AKA' network name binding can also help alleviate some of the attacks. In any case, it is recommended that EAP-AKA' configuration not be dependent on the location of where a request comes from, unless the location information can be cryptographically confirmed, e.g., with the network name binding.

Zhang and Fang also looked at Denial-of-Service attacks [ZF2005]. A serving network may request large numbers of authentication runs for

a particular subscriber from a home network. While resynchronization process can help recover from this, eventually it is possible to exhaust the sequence number space and render the subscriber's card unusable. This attack is possible for both native AKA and EAP-AKA'. However, it requires the collaboration of a serving network in an attack. It is recommended that EAP-AKA' implementations provide means to track, detect, and limit excessive authentication attempts to combat this problem.

There have also been attacks related to the use of AKA without the generated session keys (e.g., [BT2013]). Some of those attacks relate to the use of originally man-in-the-middle vulnerable HTTP Digest AKAv1 [RFC3310]. This has since then been corrected in [RFC4169]. The EAP-AKA' protocol uses session keys and provides channel binding, and as such, is resistant to the above attacks except where the protocol participants leak information to outsiders.

Basin et al [Basin2018] have performed formal analysis and concluded that the AKA protocol would have benefited from additional security requirements, such as key confirmation.

In the context of pervasive monitoring revelations, there were also reports of compromised long term pre-shared keys used in SIM and AKA [Heist2015]. While no protocol can survive the theft of key material associated with its credentials, there are some things that alleviate the impacts in such situations. These are discussed further in Section 7.3.

Arapinis et al ([Arapinis2012]) describe an attack that uses the AKA resynchronization protocol to attempt to detect whether a particular subscriber is on a given area. This attack depends on the ability of the attacker to have a false base station on the given area, and the subscriber performing at least one authentication between the time the attack is set up and run.

Borgaonkar et al discovered that the AKA resynchronization protocol may also be used to predict the authentication frequency of a subscribers if non-time-based SQN generation scheme is used [Borgaonkar2018]. The attacker can force the re-use of the keystream that is used to protect the SQN in the AKA resynchronization protocol. The attacker then guesses the authentication frequency based on the lowest bits of two XORed SQNs. The researchers' concern was that the authentication frequency would reveal some information about the phone usage behavior, e.g., number of phone calls made or number of SMS messages sent. There are a number of possible triggers for authentication, so such information leak is not direct, but can be a concern. The impact of the attack is also different depending on whether time or non-time-based SQN generation scheme is used.

Similar attacks are possible outside AKA in the cellular paging protocols where the attacker can simply send application layer data, short messages or make phone calls to the intended victim and observe the air-interface (e.g., [Kune2012] and [Shaik2016]). Hussain et. al. demonstrated a slightly more sophisticated version of the attack that exploits the fact that 4G paging protocol uses the IMSI to calculate the paging timeslot [Hussain2019]. As this attack is outside AKA, it does not impact EAP-AKA'.

Finally, bad implementations of EAP-AKA' may not produce pseudonym usernames or fast re-authentication identities in a manner that is sufficiently secure. While it is not a problem with the protocol itself, following the recommendations in Section 5.2 mitigate this concern.

7.3. Pervasive Monitoring

As required by [RFC7258], work on IETF protocols needs to consider the effects of pervasive monitoring and mitigate them when possible.

As described in Section 7.2, after the publication of RFC 5448, new information has come to light regarding the use of pervasive monitoring techniques against many security technologies, including AKA-based authentication.

For AKA, these attacks relate to theft of the long-term shared secret key material stored on the cards. Such attacks are conceivable, for instance, during the manufacturing process of cards, through coercion of the card manufacturers, or during the transfer of cards and associated information to an operator. Since the publication of reports about such attacks, manufacturing and provisioning processes have gained much scrutiny and have improved.

In particular, it is crucial that manufacturers limit access to the secret information and the cards only to necessary systems and personnel. It is also crucial that secure mechanisms be used to store and communicate the secrets between the manufacturer and the operator that adopts those cards for their customers.

Beyond these operational considerations, there are also technical means to improve resistance to these attacks. One approach is to provide Perfect Forward Secrecy (PFS). This would prevent any passive attacks merely based on the long-term secrets and observation of traffic. Such a mechanism can be defined as a backwards-compatible extension of EAP-AKA', and is pursued separately from this specification [I-D.ietf-emu-aka-pfs]. Alternatively, EAP-AKA' authentication can be run inside a PFS-capable tunneled

authentication method. In any case, the use of some PFS-capable mechanism is recommended.

7.4. Security Properties of Binding Network Names

The ability of EAP-AKA' to bind the network name into the used keys provides some additional protection against key leakage to inappropriate parties. The keys used in the protocol are specific to a particular network name. If key leakage occurs due to an accident, access node compromise, or another attack, the leaked keys are only useful when providing access with that name. For instance, a malicious access point cannot claim to be network Y if it has stolen keys from network X. Obviously, if an access point is compromised, the malicious node can still represent the compromised node. As a result, neither EAP-AKA' nor any other extension can prevent such attacks; however, the binding to a particular name limits the attacker's choices, allows better tracking of attacks, makes it possible to identify compromised networks, and applies good cryptographic hygiene.

The server receives the EAP transaction from a given access network, and verifies that the claim from the access network corresponds to the name that this access network should be using. It becomes impossible for an access network to claim over AAA that it is another access network. In addition, if the peer checks that the information it has received locally over the network-access link layer matches with the information the server has given it via EAP-AKA', it becomes impossible for the access network to tell one story to the AAA network and another one to the peer. These checks prevent some "lying NAS" (Network Access Server) attacks. For instance, a roaming partner, R, might claim that it is the home network H in an effort to lure peers to connect to itself. Such an attack would be beneficial for the roaming partner if it can attract more users, and damaging for the users if their access costs in R are higher than those in other alternative networks, such as H.

Any attacker who gets hold of the keys CK and IK, produced by the AKA algorithm, can compute the keys CK' and IK' and, hence, the Master Key (MK) according to the rules in Section 3.3. The attacker could then act as a lying NAS. In 3GPP systems in general, the keys CK and IK have been distributed to, for instance, nodes in a visited access network where they may be vulnerable. In order to reduce this risk, the AKA algorithm MUST be computed with the AMF separation bit set to 1, and the peer MUST check that this is indeed the case whenever it runs EAP-AKA'. Furthermore, [TS-3GPP.33.402] requires that no CK or IK keys computed in this way ever leave the home subscriber system.

The additional security benefits obtained from the binding depend obviously on the way names are assigned to different access networks. This is specified in [TS-3GPP.24.302]. See also [TS-3GPP.23.003]. Ideally, the names allow separating each different access technology, each different access network, and each different NAS within a domain. If this is not possible, the full benefits may not be achieved. For instance, if the names identify just an access technology, use of compromised keys in a different technology can be prevented, but it is not possible to prevent their use by other domains or devices using the same technology.

8. IANA Considerations

IANA should update the Extensible Authentication Protocol (EAP) Registry and the EAP-AKA and EAP-SIM Parameters so that entries pointing to RFC 5448 will point to this RFC instead.

8.1. Type Value

EAP-AKA' has the EAP Type value 0x32 in the Extensible Authentication Protocol (EAP) Registry under Method Types. Per Section 6.2 of [RFC3748], this allocation can be made with Designated Expert and Specification Required.

8.2. Attribute Type Values

EAP-AKA' shares its attribute space and subtypes with EAP-SIM [RFC4186] and EAP-AKA [RFC4187]. No new registries are needed.

However, a new Attribute Type value (23) in the non-skippable range has been assigned for AT_KDF_INPUT (Section 3.1) in the EAP-AKA and EAP-SIM Parameters registry under Attribute Types.

Also, a new Attribute Type value (24) in the non-skippable range has been assigned for AT_KDF (Section 3.2).

Finally, a new Attribute Type value (136) in the skippable range has been assigned for AT_BIDDING (Section 4).

8.3. Key Derivation Function Namespace

IANA has also created a new namespace for EAP-AKA' AT_KDF Key Derivation Function Values. This namespace exists under the EAP-AKA and EAP-SIM Parameters registry. The initial contents of this namespace are given below; new values can be created through the Specification Required policy [RFC8126].

Value	Description	Reference
0	Reserved	[RFC Editor: Refer to this RFC]
1	EAP-AKA' with CK'/IK'	[RFC Editor: Refer to this RFC]
2-65535	Unassigned	

9. References

9.1. Normative References

- [TS-3GPP.23.003]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 16)",
3GPP Technical Specification 23.003 version 16.5.0,
December 2020.
- [TS-3GPP.23.501]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture and procedures for 5G System; (Release 16)", 3GPP Technical Specification 23.501 version 16.7.0, December 2020.
- [TS-3GPP.24.302]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks; Stage 3; (Release 16)", 3GPP Technical Specification 24.302 version 16.4.0, July 2020.
- [TS-3GPP.24.501]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks; Stage 3; (Release 16)", 3GPP Draft Technical Specification 24.501 version 16.7.0, December 2020.
- [TS-3GPP.33.102]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture (Release 16)",
3GPP Technical Specification 33.102 version 16.0.0, July 2020.

- [TS-3GPP.33.402]
3GPP, "3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses (Release 16)", 3GPP Technical Specification 33.402 version 16.0.0, July 2020.
- [TS-3GPP.33.501]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture and procedures for 5G System (Release 16)", 3GPP Technical Specification 33.501 version 16.5.0, December 2020.
- [FIPS.180-4]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-4, August 2015, <<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>>.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [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>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <<https://www.rfc-editor.org/info/rfc4187>>.
- [RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <<https://www.rfc-editor.org/info/rfc7542>>.
- [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>>.

- [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

- [TS-3GPP.35.208] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions fl, fl*, f2, f3, f4, f5 and f5*; Document 4: Design Conformance Test Data (Release 14)", 3GPP Technical Specification 35.208 version 15.0.0, October 2018.
- [FIPS.180-1] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, April 1995, <<http://www.itl.nist.gov/fipspubs/fip180-1.htm>>.
- [FIPS.180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, <<http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>>.
- [RFC3310] Niemi, A., Arkko, J., and V. Torvinen, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", RFC 3310, DOI 10.17487/RFC3310, September 2002, <<https://www.rfc-editor.org/info/rfc3310>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4169] Torvinen, V., Arkko, J., and M. Naslund, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA) Version-2", RFC 4169, DOI 10.17487/RFC4169, November 2005, <<https://www.rfc-editor.org/info/rfc4169>>.
- [RFC4186] Haverinen, H., Ed. and J. Salowey, Ed., "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, DOI 10.17487/RFC4186, January 2006, <<https://www.rfc-editor.org/info/rfc4186>>.

- [RFC4284] Adrangi, F., Lortz, V., Bari, F., and P. Eronen, "Identity Selection Hints for the Extensible Authentication Protocol (EAP)", RFC 4284, DOI 10.17487/RFC4284, January 2006, <<https://www.rfc-editor.org/info/rfc4284>>.
- [RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, DOI 10.17487/RFC4306, December 2005, <<https://www.rfc-editor.org/info/rfc4306>>.
- [RFC5113] Arkko, J., Aboba, B., Korhonen, J., Ed., and F. Bari, "Network Discovery and Selection Problem", RFC 5113, DOI 10.17487/RFC5113, January 2008, <<https://www.rfc-editor.org/info/rfc5113>>.
- [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <<https://www.rfc-editor.org/info/rfc5247>>.
- [RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <<https://www.rfc-editor.org/info/rfc5281>>.
- [RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, DOI 10.17487/RFC5448, May 2009, <<https://www.rfc-editor.org/info/rfc5448>>.
- [RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <<https://www.rfc-editor.org/info/rfc6194>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <<https://www.rfc-editor.org/info/rfc7170>>.

- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [I-D.ietf-emu-aka-pfs] Arkko, J., Norrman, K., and V. Torvinen, "Perfect-Forward Secrecy for the Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' PFS)", draft-ietf-emu-aka-pfs-05 (work in progress), October 2020.
- [Heist2015] Scahill, J. and J. Begley, "The great SIM heist", February 2015, in <https://firstlook.org/theintercept/2015/02/19/great-sim-heist/>.
- [MT2012] Mjolsnes, S. and J-K. Tsay, "A vulnerability in the UMTS and LTE authentication and key agreement protocols", October 2012, in Proceedings of the 6th international conference on Mathematical Methods, Models and Architectures for Computer Network Security: computer network security.
- [BT2013] Beekman, J. and C. Thompson, "Breaking Cell Phone Authentication: Vulnerabilities in AKA, IMS and Android", August 2013, in 7th USENIX Workshop on Offensive Technologies, WOOT '13.
- [ZF2005] Zhang, M. and Y. Fang, "Breaking Cell Phone Authentication: Vulnerabilities in AKA, IMS and Android", March 2005, IEEE Transactions on Wireless Communications, Vol. 4, No. 2.
- [Basin2018] Basin, D., Dreier, J., Hirsch, L., Radomirovic, S., Sasse, R., and V. Stettle, "A Formal Analysis of 5G Authentication", August 2018, arXiv:1806.10360.
- [Arapinis2012] Arapinis, M., Mancini, L., Ritter, E., Ryan, M., Golde, N., and R. Borgaonkar, "New Privacy Issues in Mobile Telephony: Fix and Verification", October 2012, CCS'12, Raleigh, North Carolina, USA.

[Borgaonkar2018]

Borgaonkar, R., Hirschi, L., Park, S., and A. Shaik, "New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols", 2018 in IACR Cryptology ePrint Archive.

[Kune2012]

Kune, D., Koelndorfer, J., and Y. Kim, "Location leaks on the GSM air interface", 2012 in the proceedings of NDSS '12 held 5-8 February, 2012 in San Diego, California.

[Shaik2016]

Shaik, A., Seifert, J., Borgaonkar, R., Asokan, N., and V. Niemi, "Practical attacks against privacy and availability in 4G/LTE mobile communication systems", 2012 in the proceedings of NDSS '16 held 21-24 February, 2016 in San Diego, California.

[Hussain2019]

Hussain, S., Echeverria, M., Chowdhury, O., Li, N., and E. Bertino, "Privacy Attacks to the 4G and 5G Cellular Paging Protocols Using Side Channel Information", in the Proceedings of NDSS '19, held 24-27 February, 2019, in San Diego, California.

Appendix A. Changes from RFC 5448

The changes consist first of all, referring to a newer version of [TS-3GPP.24.302]. The new version includes an updated definition of the Network Name field, to include 5G.

Secondly, identifier usage for 5G has been specified in Section 5.3. Also, the requirements on generating pseudonym usernames and fast re-authentication identities have been updated from the original definition in RFC 5448, which referenced RFC 4187. See Section 5.

Thirdly, exported parameters for EAP-AKA' have been defined in Section 6, as required by [RFC5247], including the definition of those parameters for both full authentication and fast re-authentication.

The security, privacy, and pervasive monitoring considerations have been updated or added. See Section 7.

The references to [RFC2119], [RFC7542], [RFC7296], [RFC8126], [FIPS.180-1] and [FIPS.180-2] have been updated to their most recent versions and language in this document changed accordingly. However, this is merely an update to a newer RFC but the actual protocol functions are the same as defined in the earlier RFCs.

Similarly, references to all 3GPP technical specifications have been updated to their 5G (Release 16) versions or otherwise most recent version when there has not been a 5G-related update.

Finally, a number of clarifications have been made, including a summary of where attributes may appear.

Appendix B. Changes to RFC 4187

In addition to specifying EAP-AKA', this document mandates also a change to another EAP method, EAP-AKA that was defined in RFC 4187. This change was mandated already in RFC 5448 but repeated here to ensure that the latest EAP-AKA' specification contains the instructions about the necessary bidding down feature in EAP-AKA as well.

The changes to RFC 4187 relate only to the bidding down prevention support defined in Section 4. In particular, this document does not change how the Master Key (MK) is calculated or any other aspect of EAP-AKA. The provisions in this specification for EAP-AKA' do not apply to EAP-AKA, outside Section 4.

Appendix C. Changes from Previous Version of This Draft

RFC Editor: Please delete this section at the time of publication.

The -00 version of the working group draft is merely a republication of an earlier individual draft.

The -01 version of the working group draft clarifies updates relationship to RFC 4187, clarifies language relating to obsoleting RFC 5448, clarifies when the 3GPP references are expected to be stable, updates several past references to their more recently published versions, specifies what identifiers should be used in key derivation formula for 5G, specifies how to construct the network name in manner that is compatible with both 5G and previous versions, and has some minor editorial changes.

The -02 version of the working group draft added specification of peer identity usage in EAP-AKA', added requirements on the generation of pseudonym and fast re-authentication identifiers, specified the format of 5G-identifiers when they are used within EAP-AKA', defined privacy and pervasive surveillance considerations, clarified when 5G-related procedures apply, specified what Peer-Id value is exported when no AT_IDENTITY is exchanged within EAP-AKA', and made a number of other clarifications and editorial improvements. The security considerations section also includes a summary of vulnerabilities

brought up in the context of AKA or EAP-AKA', and discusses their applicability and impacts in EAP-AKA'.

The -03 version of the working group draft corrected some typos, referred to the 3GPP specifications for the SUPI and SUCI formats, updated some of the references to newer versions, and reduced the strength of some of the recommendations in the security considerations section from keyword level to normal language (as they are just deployment recommendations).

The -04 version of the working group draft rewrote the abstract and some of the introduction, corrected some typos, added sentence to the abstract about obsoleting RFC 5448, clarified the use of the language when referring to AT_KDF values vs. AT_KDF attribute number, provided guidance on random number generation, clarified the dangers relating to the use of permanent user identities such as IMSIs, aligned the key derivation function/mechanism terminology, aligned the key derivation/generation terminology, aligned the octet/byte terminology, clarified the text regarding strength of SHA-256, added some cross references between sections, instructed IANA to change registries to point to this RFC rather than RFC 5448, and changed Pasi's listed affiliation.

The -05 version of the draft corrected the Section 7.1 statement that SUCI must not be communicated in EAP-AKA'; this statement was meant to say SUPI must not be communicated. That was a major bug, but hopefully one that previous readers understood was a mistake!

The -05 version also changed keyword strengths for identifier requests in different cases in a 5G network, to match the 3GPP specifications (see Section 5.3.2).

Tables of where attributes may appear has been added to the -05 version of the document, see Section 3.5 and Section 4.1. The tables are based on the original table in RFC 4187.

Other changes in the -05 version included the following:

- o The attribute appearance table entry for AT_MAC in EAP-Response/ AKA-Challenge has been specified to be 0-1 because it does not appear when AT_KDF has to be sent; this was based on implementor feedback.
- o Added information about attacks against the re-synchronization protocol and other attacks recently discussed in academic conferences.

- o Clarified length field calculations and the AT_KDF negotiation procedure.
- o The treatment of AT_KDF attribute copy in the EAP-Response/AKA'-Synchronization-Failure message was clarified in Section 3.2.
- o Updated and added several references
- o Switched to use of hexadecimal for EAP Type Values for consistency with other documents.
- o Made editorial clarifications to a number places in the document.

The version -06 included changes to updates of references to newer versions on IANA considerations guidelines, NAIs, and IKEv2.

The version -07 includes the following changes, per AD and last call review comments:

- o The use of pseudonyms has been clarified in Section 7.1.
- o The document now clarifies that it specifies behaviour both for 4G and 5G.
- o The implications of collisions between "Access Network ID" (4G) and "Serving Network Name" (5G) have been explained in Section 3.1.
- o The ability of the bidding down protection to protect bidding down only in the direction from EAP-AKA' to EAP-AKA but the other way around has been noted in Section 7.
- o The implications of the attack described by [Borgaonkar2018] have been updated.
- o Section 3.1 now specifies more clearly that zero-length network name is not allowed.
- o Section 3.1 refers to the network name that is today specified in [TS-3GPP.24.302] for both 4G (non-3GPP access) and 5G.
- o Section 7 now discusses cryptographic agility.
- o The document now is clear that any change to key aspects of 3GPP specifications, such as key derivation for AKA, would affect this specification and implementations.

- o References have been updated to the latest Release 15 versions, that are now stable.
- o Tables have been numbered.
- o Adopted a number of other editorial corrections.

The version -08 includes the following changes:

- o Alignment of the 3GPP TS Annex and this draft, so that each individual part of the specification is stated in only one place. This has lead to this draft referring to bigger parts of the 3GPP specification, instead of spelling out the details within this document. Note that this alignment change is a proposal at this stage, and will be discussed in the upcoming 3GPP meeting.
- o Relaxed the language on using only SUCI in 5G. While that is the mode of operation expected to be used, [TS-3GPP.33.501] does not prohibit other types of identifiers.

The version -09 includes the following changes:

- o Updated the language relating to obsoleting/updating RFC 5448; there was an interest to ensure that RFC 5448 stays a valid specification also in the future, owing to existing implementations.
- o Clarified that the leading digit "6" is not used in 5G networks.
- o Updated the language relating to when 5G-specific procedures are in effect, to support new use cases 3GPP has defined.
- o Updated the reference in Section 3.3, as the identities are different in the 5G case.
- o Clarified that the use of the newer reference to IKEv2 RFC did not change the actual PRF' function from RFC 5448.
- o Clarified that the Section 5.2 text does not impact backwards compatibility.
- o Corrected the characterization of the attack from [ZF2005].
- o Mentioned 5G GUTIs as one possible 5G-identifier in Section 5.1.
- o Updated the references to Release 16. These specifications are stable in 3GPP.

Version -10 is the final version and made changes per IESG and directorate review comments. These changes were editorial. One duplicate requirement in Section 5.3.1 was removed, and some references were added for tunnel methods discussion in Section 7.1. The language about exported parameters was clarified in Section 6.

Appendix D. Importance of Explicit Negotiation

Choosing between the traditional and revised AKA key derivation functions is easy when their use is unambiguously tied to a particular radio access network, e.g., Long Term Evolution (LTE) as defined by 3GPP or evolved High Rate Packet Data (eHRPD) as defined by 3GPP2. There is no possibility for interoperability problems if this radio access network is always used in conjunction with new protocols that cannot be mixed with the old ones; clients will always know whether they are connecting to the old or new system.

However, using the new key derivation functions over EAP introduces several degrees of separation, making the choice of the correct key derivation functions much harder. Many different types of networks employ EAP. Most of these networks have no means to carry any information about what is expected from the authentication process. EAP itself is severely limited in carrying any additional information, as noted in [RFC4284] and [RFC5113]. Even if these networks or EAP were extended to carry additional information, it would not affect millions of deployed access networks and clients attaching to them.

Simply changing the key derivation functions that EAP-AKA [RFC4187] uses would cause interoperability problems with all of the existing implementations. Perhaps it would be possible to employ strict separation into domain names that should be used by the new clients and networks. Only these new devices would then employ the new key derivation function. While this can be made to work for specific cases, it would be an extremely brittle mechanism, ripe to result in problems whenever client configuration, routing of authentication requests, or server configuration does not match expectations. It also does not help to assume that the EAP client and server are running a particular release of 3GPP network specifications. Network vendors often provide features from future releases early or do not provide all features of the current release. And obviously, there are many EAP and even some EAP-AKA implementations that are not bundled with the 3GPP network offerings. In general, these approaches are expected to lead to hard-to-diagnose problems and increased support calls.

Appendix E. Test Vectors

Test vectors are provided below for four different cases. The test vectors may be useful for testing implementations. In the first two cases, we employ the MILENAGE algorithm and the algorithm configuration parameters (the subscriber key K and operator algorithm variant configuration value OP) from test set 19 in [TS-3GPP.35.208].

The last two cases use artificial values as the output of AKA, and is useful only for testing the computation of values within EAP-AKA', not AKA itself.

Case 1

The parameters for the AKA run are as follows:

Identity: "0555444333222111"
Network name: "WLAN"
RAND: 81e9 2b6c 0ee0 e12e bceb a8d9 2a99 dfa5
AUTN: bb52 e91c 747a c3ab 2a5c 23d1 5ee3 51d5
IK: 9744 871a d32b f9bb d1dd 5ce5 4e3e 2e5a
CK: 5349 fbe0 9864 9f94 8f5d 2e97 3a81 c00f
RES: 28d7 b0f2 a2ec 3de5

Then the derived keys are generated as follows:

CK': 0093 962d 0dd8 4aa5 684b 045c 9edf fa04
IK': ccfc 230c a74f cc96 c0a5 d611 64f5 a76c
K_encr: 766f a0a6 c317 174b 812d 52fb cd11 a179
K_aut: 0842 ea72 2ff6 835b fa20 3249 9fc3 ec23
c2f0 e388 b4f0 7543 ffc6 77f1 696d 71ea
K_re: cf83 aa8b c7e0 aced 892a cc98 e76a 9b20
95b5 58c7 795c 7094 715c b339 3aa7 d17a
MSK: 67c4 2d9a a56c 1b79 e295 e345 9fc3 d187
d42b e0bf 818d 3070 e362 c5e9 67a4 d544
e8ec fe19 358a b303 9aff 03b7 c930 588c
055b abee 58a0 2650 b067 ec4e 9347 c75a
EMSK: f861 703c d775 590e 16c7 679e a387 4ada
8663 11de 2907 64d7 60cf 76df 647e a01c
313f 6992 4bdd 7650 ca9b ac14 1ea0 75c4
ef9e 8029 c0e2 90cd bad5 638b 63bc 23fb

Case 2

The parameters for the AKA run are as follows:

Identity: "0555444333222111"
Network name: "HRPD"
RAND: 81e9 2b6c 0ee0 e12e bceb a8d9 2a99 dfa5
AUTN: bb52 e91c 747a c3ab 2a5c 23d1 5ee3 51d5
IK: 9744 871a d32b f9bb d1dd 5ce5 4e3e 2e5a
CK: 5349 fbe0 9864 9f94 8f5d 2e97 3a81 c00f
RES: 28d7 b0f2 a2ec 3de5

Then the derived keys are generated as follows:

CK': 3820 f027 7fa5 f777 32b1 fb1d 90c1 a0da
IK': db94 a0ab 557e f6c9 ab48 619c a05b 9a9f
K_encr: 05ad 73ac 915f ce89 ac77 e152 0d82 187b
K_aut: 5b4a caef 62c6 ebb8 882b 2f3d 534c 4b35
2773 37a0 0184 f20f f25d 224c 04be 2afd
K_re: 3f90 bf5c 6e5e f325 ff04 eb5e f653 9fa8
cca8 3981 94fb d00b e425 b3f4 0dba 10ac
MSK: 87b3 2157 0117 cd6c 95ab 6c43 6fb5 073f
f15c f855 05d2 bc5b b735 5fc2 1ea8 a757
57e8 f86a 2b13 8002 e057 5291 3bb4 3b82
f868 a961 17e9 1a2d 95f5 2667 7d57 2900
EMSK: c891 d5f2 0f14 8a10 0755 3e2d ea55 5c9c
b672 e967 5f4a 66b4 bafa 0273 79f9 3aee
539a 5979 d0a0 042b 9d2a e28b ed3b 17a3
1dc8 ab75 072b 80bd 0c1d a612 466e 402c

Case 3

The parameters for the AKA run are as follows:

Identity: "0555444333222111"
Network name: "WLAN"
RAND: e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0
AUTN: a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0
IK: b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0
CK: c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0
RES: d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0

Then the derived keys are generated as follows:

CK': cd4c 8e5c 68f5 7ddl d7d7 dfd0 c538 e577
IK': 3ece 6b70 5dbb f7df c459 a112 80c6 5524
K_encr: 897d 302f a284 7416 488c 28e2 0dcb 7be4
K_aut: c407 00e7 7224 83ae 3dc7 139e b0b8 8bb5
58cb 3081 eccd 057f 9207 d128 6ee7 dd53
K_re: 0a59 1a22 dd8b 5b1c f29e 3d50 8c91 dbbd
b4ae e230 5189 2c42 b6a2 de66 ea50 4473
MSK: 9f7d ca9e 37bb 2202 9ed9 86e7 cd09 d4a7
0d1a c76d 9553 5c5c ac40 a750 4699 bb89
61a2 9ef6 f3e9 0f18 3de5 861a d1be dc81
ce99 1639 1b40 1aa0 06c9 8785 a575 6df7
EMSK: 724d e00b db9e 5681 87be 3fe7 4611 4557
d501 8779 537e e37f 4d3c 6c73 8cb9 7b9d
c651 bc19 bfad c344 ffe2 b52c a78b d831
6b51 dacc 5f2b 1440 cb95 1552 1cc7 ba23

Case 4

The parameters for the AKA run are as follows:

Identity: "0555444333222111"
Network name: "HRPD"
RAND: e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0
AUTN: a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0
IK: b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0
CK: c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0
RES: d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0

Then the derived keys are generated as follows:

CK': 8310 a71c e6f7 5488 9613 da8f 64d5 fb46
IK': 5adf 1436 0ae8 3819 2db2 3f6f cb7f 8c76
K_encr: 745e 7439 ba23 8f50 fcac 4d15 d47c d1d9
K_aut: 3e1d 2aa4 e677 025c fd86 2a4b e183 61a1
3a64 5765 5714 63df 833a 9759 e809 9879
K_re: 99da 835e 2ae8 2462 576f e651 6fad 1f80
2f0f a119 1655 dd0a 273d a96d 04e0 fcd3
MSK: c6d3 a6e0 cee4 951e b20d 74f3 2c30 61d0
680a 04b0 b086 ee87 00ac e3e0 b95f a026
83c2 87be ee44 4322 94ff 98af 26d2 cc78
3bac e75c 4b0a f7fd feb5 511b a8e4 cbd0
EMSK: 7fb5 6813 838a dafa 99d1 40c2 f198 f6da
cebf b6af ee44 4961 1054 02b5 08c7 f363
352c b291 9644 b504 63e6 a693 5415 0147
ae09 cbc5 4b8a 651d 8787 a689 3ed8 536d

Contributors

The test vectors in Appendix C were provided by Yogendra Pal and Jouni Malinen, based on two independent implementations of this specification.

Jouni Malinen provided suggested text for Section 6. John Mattsson provided much of the text for Section 7.1. Karl Norrman was the source of much of the information in Section 7.2.

Acknowledgments

The authors would like to thank Guenther Horn, Joe Salowey, Mats Naslund, Adrian Escott, Brian Rosenberg, Lakshminath Dondeti, Ahmad Muhanna, Stefan Rommer, Miguel Garcia, Jan Kall, Ankur Agarwal, Jouni Malinen, John Mattsson, Jesus De Gregorio, Brian Weis, Russ Housley, Alfred Hoenes, Anand Palanigounder, Michael Richardsson, Roman Danyliw, Dan Romascanu, Kyle Rose, Benjamin Kaduk, Alissa Cooper, Erik Kline, Murray Kucherawy, Robert Wilton, Warren Kumari, Andreas Kunz, Marcus Wong, Kalle Jarvinen, Daniel Migault, and Mohit Sethi for their in-depth reviews and interesting discussions in this problem space.

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Vesa Lehtovirta
Ericsson
Jorvas 02420
Finland

Email: vesa.lehtovirta@ericsson.com

Vesa Torvinen
Ericsson
Jorvas 02420
Finland

Email: vesa.torvinen@ericsson.com

Pasi Eronen
Independent
Finland

Email: pe@iki.fi

Network Working Group
Internet-Draft
Updates: RFC7170 (if approved)
Intended status: Standards Track
Expires: 26 February 2022

E. Lear
O. Friel
N. Cam-Winget
Cisco Systems
D. Harkins
HP Enterprise
25 August 2021

TEAP Update and Extensions for Bootstrapping
draft-lear-eap-teap-brski-06

Abstract

In certain environments, in order for a device to establish any layer three communications, it is necessary for that device to be properly credentialed. This is a relatively easy problem to solve when a device is associated with a human being and has both input and display functions. It is less easy when the human, input, and display functions are not present. To address this case, this memo specifies extensions to the Tunnel Extensible Authentication Protocol (TEAP) method that leverages Bootstrapping Remote Secure Key Infrastructures (BRSKI) in order to provide a credential to a device at layer two. The basis of this work is that a manufacturer will introduce the device and the local deployment through cryptographic means. In this sense the same trust model as BRSKI is used.

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 26 February 2022.

Copyright Notice

Copyright (c) 2021 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. Introduction	3
1.1. Terminology	3
2. TEAP BRSKI Architecture	4
3. BRSKI Bootstrap and Enroll Operation	4
3.1. Discovery of Trusted MASA	5
3.2. Executing BRSKI in a TEAP Tunnel	5
4. PKI Certificate Considerations	9
4.1. TEAP Tunnel Establishment	9
4.2. BRSKI Trust Establishment	11
4.3. Certificate Expiration Times	12
4.4. LDevID Subject and Subject Alternative Names	12
4.5. PKCS#10 Retry Handling	13
5. Peer Identity	13
6. Channel and Crypto Binding	14
7. Protocol Flows	14
7.1. TEAP Server Grants Access	14
7.2. TEAP Server Instructs Client to Perform BRSKI Flow	16
7.3. TEAP Server Instructs Client to Reenroll	20
7.4. Out of Band Reenroll	22
8. TEAP TLV Formats	22
8.1. New TLVs	22
8.1.1. BRSKI-RequestVoucher TLV	23
8.1.2. BRSKI-Voucher TLV	23
8.1.3. CSR-Attributes TLV	24
8.1.4. Retry-After TLV	25
8.1.5. NAI TLV	25
8.2. Existing TEAP TLV Specifications	25
8.2.1. PKCS#10 TLV	25
8.3. TLV Rules	26
9. Fragmentation	26
10. IANA Considerations	26
11. Security Considerations	27
11.1. Issues with Provisionally Authenticated TEAP	28
11.2. Attack Against Discovery	28
11.3. TEAP Server as Registration Authority	28
11.4. Trust of Registrar	29
12. Acknowledgments	29

13. References	29
13.1. Normative References	29
13.2. Informative References	30
Appendix A. Changes from Earlier Versions	30
Authors' Addresses	30

1. Introduction

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) specifies a means to provision credentials to be used as credentials to operationally access networks. It was designed as a standalone means where some limited access to an IP network is already available. This is not always the case. For example, IEEE 802.11 networks generally require authentication prior to any form of address assignment. While it is possible to assign an IP address to a device on some form of an open network, or to accept some sort of default credential to establish initial IP connectivity, the steps that would then follow might well require that the device is placed on a new network, requiring resetting all layer three parameters.

A more natural approach in such cases is to more tightly bind the provisioning of credentials with the authentication mechanism. One such way to do this is to make use of the Extensible Authentication Protocol (EAP) [RFC3748] and the Tunnel Extensible Authentication Protocol (TEAP) method [RFC7170]. Thus we define new TEAP Type-Length-Value (TLV) objects that can be used to transport the BRSKI protocol messages within the context of a TEAP TLS tunnel.

[RFC7170] discusses the notion of provisioning peers. Several different mechanisms are available. Section 3.8 of that document acknowledges the concept of not initially authenticating the outer TLS session so that provisioning may occur. In addition, exchange of multiple TLV messages between client and EAP server permits multiple provisioning steps.

1.1. Terminology

The reader is presumed to be familiar with EAP terminology as stated in [RFC3748]. In addition, the following terms are commonly used in this document.

- * BRSKI: Bootstrapping Remote Secure Key Infrastructures, as defined in [I-D.ietf-anima-bootstrapping-keyinfra]. The term is also used to refer to the flow described in that document.
- * EST: Enrollment over Secure Transport, as defined in [RFC7030].
- * Voucher: a signed JSON object as defined in [RFC8366].

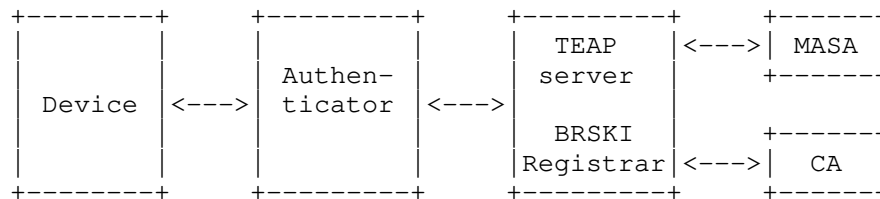
2. TEAP BRSKI Architecture

The TEAP BRSKI architecture is illustrated in Section 3. The device talks to the TEAP server via the Authenticator using any compliant transport such as [IEEE8021X]. The architecture illustrated shows an Authenticator distinct from the TEAP server. This is a deployment optimization and when so deployed the communication between Authenticator and TEAP server is a AAA protocol such as RADIUS or DIAMETER.

The architecture illustrated shows a co-located TEAP server and BRSKI registrar. Not only are these two functions co-located, they MUST be the same entity. This ensures that the entity identified in the device's voucher request (the TEAP server) is the same entity that signs the voucher request (the registrar).

The registrar communicates with the BRSKI MASA service for the purposes of getting signed vouchers.

The registrar also communicates with a Certificate Authority in order to issue LDevIDs. The architecture shows the registrar and CA as being two logically separate entities, however the CA may be integrated into the registrar. The device is not explicitly aware of whether the CA and registrar functions are integrated.



3. BRSKI Bootstrap and Enroll Operation

This section summarises the current BRSKI operation. The BRSKI flow assumes the device has an IDevID and has a manufacturer installed trust anchor that can be used to validate the BRSKI voucher. The BRSKI flow comprises several main steps from the perspective of the device:

- * Step 1: Device discovers the registrar
- * Step 2: Device establishes provisional TLS connection to registrar
- * Step 3: Device sends voucher request message and receives signed voucher response

- * Step 4: Device validates voucher and validates provisional TLS connection to registrar
- * Step 5: Device downloads additional local domain CA information
- * Step 6: Device downloads Certificate Signing Request (CSR) attributes
- * Step 7: Device does a certificate enroll to obtain an LDevID
- * Step 8: Device periodically reenrolls via EST to refresh its LDevID

Most of the operational steps require the device, and thus its internal state machine, to automatically complete the next step without being explicitly instructed to do so by the registrar. For example, the registrar does not explicitly tell the device to download additional local domain CA information, or to do an EST enroll to obtain an LDevID.

3.1. Discovery of Trusted MASA

BRSKI section 2.8 outlines how the Registrar discovers the correct MASA to connect with. BRSKI section 5.3 outlines how the Registrar can make policy decisions about which devices to trust.

Similar approaches are applicable for TEAP servers executing BRSKI. For example, the TEAP server may be configured with a list of trusted manufacturing CAs. During device bootstrap, only devices with an IDevID signed by a trusted manufacturing CA may be allowed to establish a TLS connection with the TEAP server, and the TEAP server could then extract the MASA URI from the device's IDevID.

3.2. Executing BRSKI in a TEAP Tunnel

This section outlines how the main BRSKI steps outlined above map to TEAP, and how BRSKI and enrollment can be accomplished inside a TEAP TLS tunnel. The following new TEAP TLVs are introduced:

- * BRSKI-VoucherRequest
- * BRSKI-Voucher
- * CSR-Attributes

The following steps outline how the above BRSKI flow maps to TEAP.

- * Step 1: Device discovers the registrar

When BRSKI is executed in a TEAP tunnel, the device exchanges BRSKI TLVs with the TEAP server. The discovery process for devices is therefore the standard wired or wireless LAN EAP server discovery process. The discovery processes outlined in section 4 of [I-D.ietf-anima-bootstrapping-keyinfra] are not required for initial discovery of the registrar.

* Step 2: Device establishes provisional TLS connection to registrar

The device establishes an outer TEAP tunnel with the TEAP server and does not validate the server certificate. The device presents its LDevID as its identity certificate if it has a valid LDevID, otherwise it presents its IDevID. The TEAP server validates the device's certificate using its implicit or explicit trust anchor database. If the device presents an IDevID it is verified against a database of trusted manufacturer certificates. Server policy may also be used to control which certificate the device is allowed present, as described in section {pki-certificate-authority-considerations}.

If the presented credential is sufficient to grant access, the TEAP server can return a TEAP Result TLV indicating success immediately. The device may still send a Request-Action TLV including a BRSKI-VoucherRequest TLV in response to the TEAP Result TLV if it does not have, but requires, provisioning of trust anchors for validating the TEAP server certificate. Note that no inner EAP method is required for this, only an exchange of TEAP TLVs.

[todo] Question: as the device wants the server to reply with a BRSKI-Voucher TLV, does it really send a Request-Action TLV containing a BRSKI-VoucherRequest TLV, or does it send a Request-Action TLV containing a BRSKI-Voucher TLV?? The TEAP draft is a bit ambiguous here. Normally, if one end sends a Request-Action including XXX-TLV, it means it wants the far end to send an XXX-TLV...

[todo] Question: general TEAP protocol question: does the device have to send a Request-Action w/BRSKI-VoucherRequest or can it send a BRSKI-VoucherRequest on its own? I'm not clear on this.

If the TEAP server requires that the device execute a BRSKI flow, the server sends a Request-Action TLV that includes a BRSKI-VoucherRequest TLV. For example, if the device presented its IDevID but the TEAP server requires an LDevID.

[todo] Question: to nit pick, the server should send a Request-Action TLV including a PKCS#10 TLV to tell the client to enroll. How does the server really know that the client has the correct trust

established (as previously received by a BRSKI-Voucher)? If the client sends an IDevID, does server always send a Request-Action including both BRSKI-VoucherRequest and PKCS#10 TLVs? Whats the client behaviour? I assume client can spontaneously send BRSKI-VoucherRequest and/or PCSK#10 without being explicitly instructed to. Just need to get the language correct here.

The TEAP server may also require the device to reenroll, for example, if the device presented a valid LDevID that is very closed to expiration. The server may instruct a device to reenroll by sending a Request-Action TLV that includes a zero byte length PKCS#10 TLV.

- * Step 3: Device sends voucher request message and receives signed voucher response

The device sends a BRSKI-RequestVoucher TLV to the TEAP server. The TEAP server forwards the RequestVoucher message to the MASA server, and the MASA server replies with a signed voucher. The TEAP server sends a BRSKI-Voucher TLV to the device.

If the MASA server does not issue a signed voucher, the TEAP server sends an EAP-Error TLV with a suitable error code to the device.

For wireless devices in particular, it is important that the MASA server only return a voucher for devices known to be associated with a particular registrar. In this sense, success indicates that the device is on the correct network, while failure indicates the device should try to provision itself within wireless networks (e.g, go to the next SSID).

- * Step 4: Device validates voucher and validates provisional TLS connection to registrar

The device validates the signed voucher using its manufacturer installed trust anchor, and uses the CA information in the voucher to validate the TLS connection to the TEAP server.

If the device fails to validate the voucher, then it sends a TEAP-Error TLV indicating failiure to the TEAP server.

Similarly, if the device validates the voucher, but fails to validate the provisional TLS connection, then it sends a TEAP-Error TLV indicating failure to the TEAP server. Note that the outer TLS tunnel has already been established, thus allowing the client to send a TEAP-Error TLV to the server inside that tunnel to indicate that it failed to verify the provisionally accepted outer TLS tunnel server identity.

- * Step 5: Device downloads additional local domain CA information

On completion of the BRSKI flow, the device SHOULD send a Trusted-Server-Root TLV to the TEAP server in order to discover additional local domain CAs. This is equivalent to section [todo] from [I-D.ietf-anima-bootstrapping-keyinfra].

- * Step 6: Device downloads CSR attributes

No later than the completion of step 5, server MUST send a CSR-Attributes TLV to peer server in order to discover the correct fields to include when it enrolls to get an LDevID.

- * Step 7: Device does a certificate enroll to obtain an LDevID

When executing the BRSKI flow inside a TEAP tunnel, the device does not directly leverage EST when doing its initial enroll. Instead, the device uses the existing TEAP PKCS#10 and PCKS#7 TEAP mechanisms.

Once the BRSKI flow is complete, the device can now send a PKCS#10 TLV to enroll and request an LDevID. If the TEAP server instructed the device to start the BRSKI flow via a Request-Action TLV that includes a BRSKI-RequestVoucher TLV, then the device MUST send a PKCS#10 in order to start the enroll process. The TEAP server will handle the PKCS#10 and ultimately return a PKCS#7 including an LDevID to the device.

If the TEAP server granted the device access on completion of the outer TEAP TLS tunnel in step 2 without sending a Request-Action TLV, the device does not have to send a PKCS#10 to enroll.

At this point, the device is said to be provisioned for local network access, and may authenticate in the future via 802.1X with its newly acquired credentials.

- * Step 8: Device periodically reenrolls to refresh its LDevID

When a device's LDevID is close to expiration, there are two options for re-enrollment in order to obtain a fresh LDevID. As outlined in Step 2 above, the TEAP server may instruct the device to reenroll by sending a Request-Action TLV including a PKCS#10 TLV. If the TEAP server explicitly instructs the device to reenroll via these TLV exchange, then the device MUST send a PKCS#10 to reenroll and request a fresh LDevID.

However, the device SHOULD reenroll if it determines that its LDevID is close to expiration without waiting for explicit instruction from the TEAP server. There are two options to do this.

Option 1: The device reenrolls for a new LDevID directly with the EST CA outside the context of the 802.1X TEAP flow. The device uses the registrar discovery mechanisms outlined in [I-D.ietf-anima-bootstrapping-keyinfra] to discover the registrar and the device sends the EST reenroll messages to the discovered registrar endpoint. No new TEAP TLVs are defined to facilitate discover of the registrar or EST endpoints inside the context of the TEAP tunnel.

Option 2: When the device is performing a periodic 802.1X authentication using its current LDevID, it reenrolls for a new LDevID by sending a PKCS#10 TLV inside the TEAP TLS tunnel.

4. PKI Certificate Considerations

There are multiple noteworthy PKI certificate handling considerations. These include:

- * PKI CA handling when establishing the TEAP tunnel
- * PKI CA handling establishing trust using BRSKI
- * IDevID and LDevID expiration times
- * Specifying LDevID Subject and Subject Alternative Names
- * PKCS#10 retry handling

These are described in more detail here.

4.1. TEAP Tunnel Establishment

Because this method establishes a client identity, if the peer has not been previously bootstrapped, or otherwise cannot successfully authenticate, it will use a generic identity string of teap-bootstrap@TBD1 as its network access identifier (NAI).

BRSKI section 5.3 outlines the policy decisions a Registrar may make when deciding whether to accept connections from clients. Similarly, the TEAP server operator may configure a set of trusted CAs for validating incoming TLS connections from clients. The operator may want to 'allow any device from a specific vendor', or from a set of vendors, to access the network. Network operators may do this by restricting network access to clients that have a certificate signed by one of a small set of trusted manufacturer/supplier CAs.

When the client sends its ClientHello to initiate TLS tunnel establishment, it is possible for the TEAP server to restrict the certificates that the client can use for tunnel establishment by including a list of CA distinguished names in the certificate_authorities field in the CertificateRequest message. The client should only continue with the handshake if it has a certificate signed by one of the indicated CAs.

In practice, network operators will likely want to onboard devices from a large number of device manufacturers, with each manufacturer using a different root CA when issuing IDevIDs. If the number of different manufacturer root CAs is large, this could result in very large TLS handshake messages. Therefore, the TEAP server may send a CertificateRequest message and not specify any certificate_authorities, thus allowing the client present a certificate signed by any authority in its Certificate message.

If the client has both an IDevID and an LDevID, the client should present the LDevID in preference to its IDevID, if allowed by server policy.

Once the client has sent its TLS Finished message, the TEAP server can make a policy decision, based on the CA used to sign the client's certificate, on whether to establish the outer TLS tunnel or not.

The TEAP server may delegate policy decisions to the MASA or CA function. For example, the TEAP server may declare EAP success and grant network access if the client presents a valid LDevID signed by a trusted domain CA. However, if the client presents an IDevID signed by a trusted manufacturer CA, the TEAP server may establish the TLS tunnel but not declare EAP success and grant network access until the client successfully completes a BRSKI Voucher exchange and PKCS#10/PKCS#7 exchange inside that tunnel.

It is recommended that the client validate the certificate presented by the server in the server's Certificate message, but this may not be possible for clients that have not yet provisioned appropriate trust anchors. If the client is in the provisioning phase and has not yet completed a BRSKI flow, it will not have trust anchors installed yet, and thus will not be able to validate the server's certificate. The client must however note the certificate presented by the server for (i) inclusion in the BRSKI-RequestVoucher TLV and for (ii) validation once the client has discovered the local domain trust anchors.

If the client does not present a suitable certificate to the server, the server MUST terminate the connection and fail the EAP request. If the TEAP server is unable to validate the client's certificate using its implicit or explicit trust anchor database it MUST fail the EAP request.

On establishment of the outer TLS tunnel, the TEAP server will make a policy decision on next steps. Possible policy decisions include:

- * Option 1: Server grants client full network access and returns EAP-Success. This will typically happen when the client presents a valid LDevID. Network policy may grant client network access based on LDevID without requiring the device to enroll to obtain an LDevID.
- * Option 2: Server requires that client perform a full BRSKI flow, and then enroll to get an LDevID. This will typically happen when the client presents a valid IDevID and network policy requires all clients to have LDevIDs. The server sends a Request-Action TLV that includes a BRSKI-RequestVoucher TLV to the client to instruct it to start the BRSKI flow.
- * Option 3: Server requires that the client reenroll to obtain a new LDevID. This could happen when the client presents a valid LDevID that is very close to expiration time, or the server's policy requires an LDevID update. The server sends an Action-Request TLV including a PKCS#10 TLV to the client to instruct it to reenroll.

4.2. BRSKI Trust Establishment

If the server requires that client perform a full BRSKI flow, it sends a Request-Action TLV that includes a zero byte length BRSKI-RequestVoucher TLV to the client. The client sends a new BRSKI-RequestVoucher TLV to the server, which contains all data specified in [I-D.ietf-anima-bootstrapping-keyinfra] section 5.2. The client includes the server certificate it received in the server's Certificate message during outer TLS tunnel establishment in the proximity-registrar-cert field. The client signs the request using its IDevID.

The server includes all additional information as required by [I-D.ietf-anima-bootstrapping-keyinfra] section 5.4 and signs the request prior to forwarding to the MASA.

The MASA responds as per [I-D.ietf-anima-bootstrapping-keyinfra] section 5.5. The response may indicate failure and the server should react accordingly to failures by sending a failure response to the client, and failing the TEAP method.

If the MASA replies with a signed voucher and a successful result, the server then forwards this response to the client in a BRSKI-Voucher TLV.

When the client receives the signed voucher, it validates the signature using its built in trust anchor list, and extracts the pinned-domain-cert field. The client must use the CA included in the pinned-domain-cert to validate the certificate that was presented by the server when establishing the outer TLS tunnel. If this certificate validation fails, the client must fail the TEAP request and not connect to the network.

[TBD- based on client responses, the registrar sends a status update to the MASA]

4.3. Certificate Expiration Times

[IEEE802.1AR] section 7.2.7.2 states:

notAfter: The latest time a DevID is expected to be used. Devices possessing an IDevID are expected to operate indefinitely into the future and should use the value 99991231235959Z. Solutions verifying an IDevID are expected to accept this value indefinitely.

TEAP servers SHOULD follow the 802.1AR standard when validating IDevIDs.

TEAP servers SHOULD reject LDevIDs with expired certificates and SHOULD NOT allow clients to connect with recently expired LDevIDs. If a client presents a recently expired LDevID it SHOULD be forced to authenticate using its IDevID and then reenroll to obtain a valid LDevID.

4.4. LDevID Subject and Subject Alternative Names

BRSKI section 5.9.2 specifies that the pledge MUST send a CSR Attributes request to the registrar. The registrar MAY use this mechanism to instruct the pledge about the identities it should include in the CSR request it sends as part of enrollment. The registrar may use this mechanism to tell the pledge what Subject or Subject Alternative Name identity information to include in its CSR request. This can be useful if the Subject must have a specific value in order to complete enrollment with the CA.

4.5. PKCS#10 Retry Handling

They will be scenarios where the TEAP server is willing to handle a PKCS#10 request from a client and issue a certificate via a PKCS#7 response, however, the TEAP server is unable to immediately completely the request and needs to instruct the client to retry later after a specified time interval.

A new Retry-After TLV is defined that the TEAP server uses to specify a retry interval in seconds. New error codes are defined to handle these two alternate retry scenarios.

- * The TEAP tunnel remains up: The client is instructed to resend the PKCS#10 request after a retry interval but inside the same TEAP tunnel. The TEAP server returns a Retry-After TLV to the client, and returns an Error TLV with a new code in the 1000-1999 range.
- * The TEAP tunnel is torn down: The client is instructed to establish a new TEAP connection and TEAP tunnel after a retry interval, and resend the PKCS#10 request inside the new tunnel. The TEAP server returns a Retry-After TLV to the client, and returns an Error TLV with a new code in the 2000-2999 range.

5. Peer Identity

EAP [RFC3748] recommends that "the Identity Response be used primarily for routing purposes and selecting which EAP method to use". NAI [RFC7542] recommends omitting the username part of an NAI in order to support username privacy, where appropriate.

A device that has not been bootstrapped at all SHOULD send an identity of teap-bootstrap@TBD1. Otherwise, a device SHOULD send its configured NAI.

The TEAP server may specify an NAI that it wishes the device to use. For example, the server may want a bootstrapped device to use an NAI of "abc123@example.com", or simply an NAI of "@example.com". This could be desirable in order to facilitate roaming scenarios. The server can do this by sending the device an NAI TLV inside the TEAP tunnel.

If the server specifies an NAI TLV, and the device handles the TLV, the device MAY use the specified NAI in all subsequent EAP authentication flows. If the device is not willing to handle the NAI TLV, it MUST reply with an Error TLV.

Authentication servers implementing this specification MAY reply with an Error TLV to any unrecognized NAI, or MAY attempt to bootstrap the device, regardless of the NAI. A device receiving an Error from the server MAY attempt a new session without the NAI in order to bootstrap.

6. Channel and Crypto Binding

As the TEAP BRSKI flow does not define or require an inner EAP method, there is no explicit need for exchange of Channel-Binding TLVs between the device and the TEAP server.

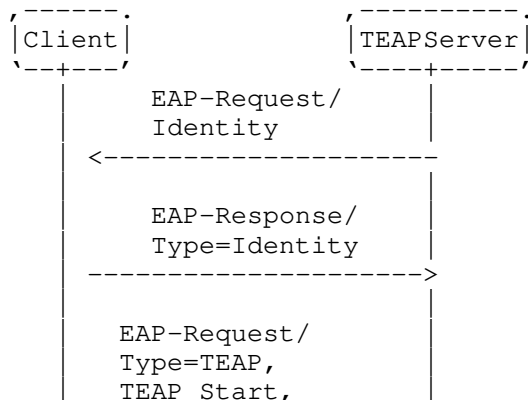
The TEAP BRSKI TLVs are expected to occur at the beginning of the TEAP Phase 2 and MUST occur before the final Crypto-Binding TLV. This draft does not exclude the possibility of having other EAP methods occur following the TEAP BRSKI TLVs and as such, the Crypto-Binding TLV process rules as defined in [RFC7170] apply.

7. Protocol Flows

This section outlines protocol flows that map to the three server policy options described in section Section 4.1. The protocol flows illustrate a TLS1.2 exchange. Pertinent notes are outlined in the protocol flows.

7.1. TEAP Server Grants Access

In this flow, the server grants access as server policy allows the client to access the network based on the identity certificate that the client presented. This means that either (i) the client has previously completed BRSKI and has presented a valid LDevID or (ii) the client presents an IDevID and network policy allows access based purely on IDevID.



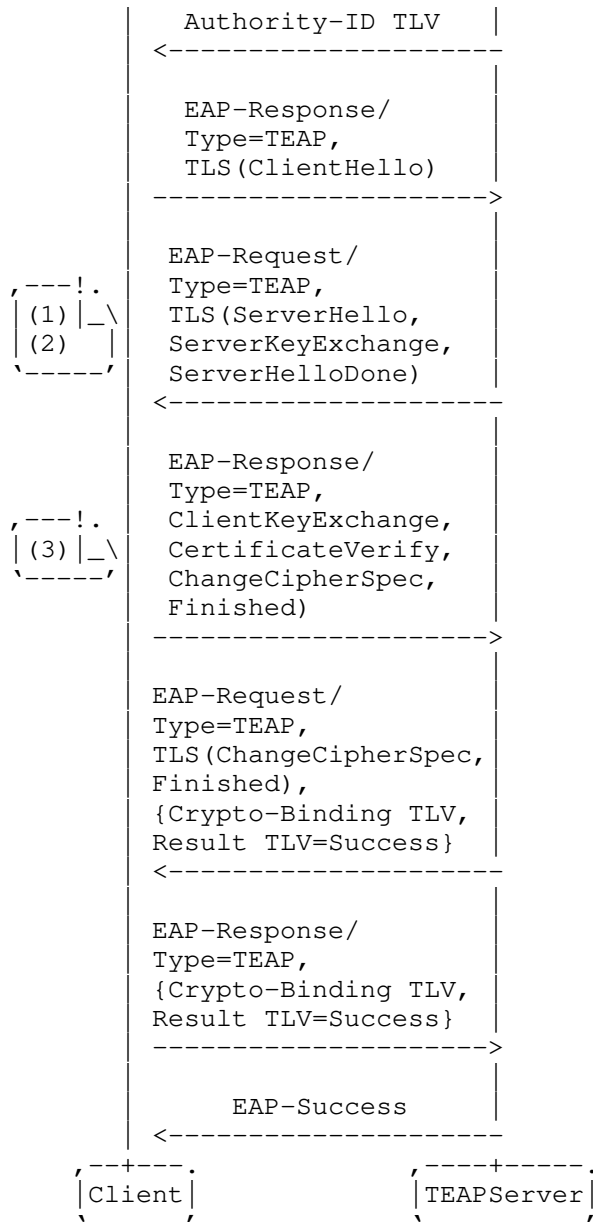


Figure 1: TEAP Server Grants Access

Notes:

(1) If the client has completed the BRSKI flow and has locally significant trust anchors, it must validate the Certificate received from the server. If the client has not yet completed the BRSKI flow, then it provisionally accepts the server Certificate and must validate it later once BRSKI is complete.

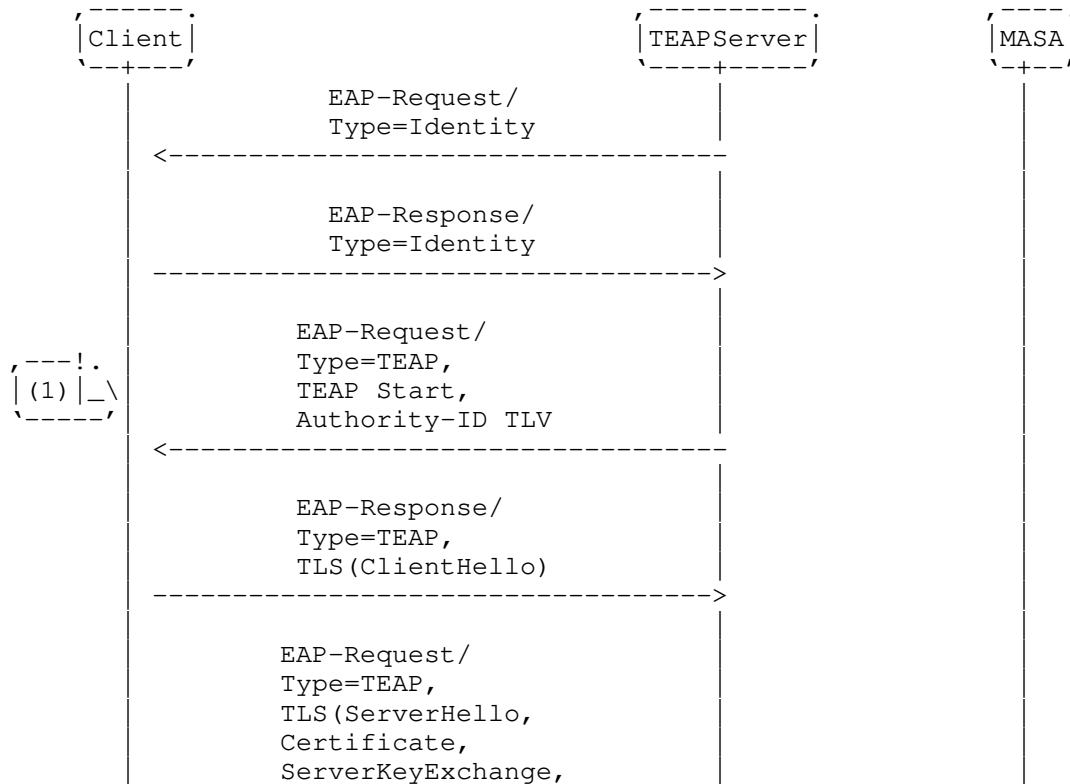
(2) The server may include `certificate_authorities` field in the `CertificateRequest` message in order to restrict the identity certificates that the device is allowed present.

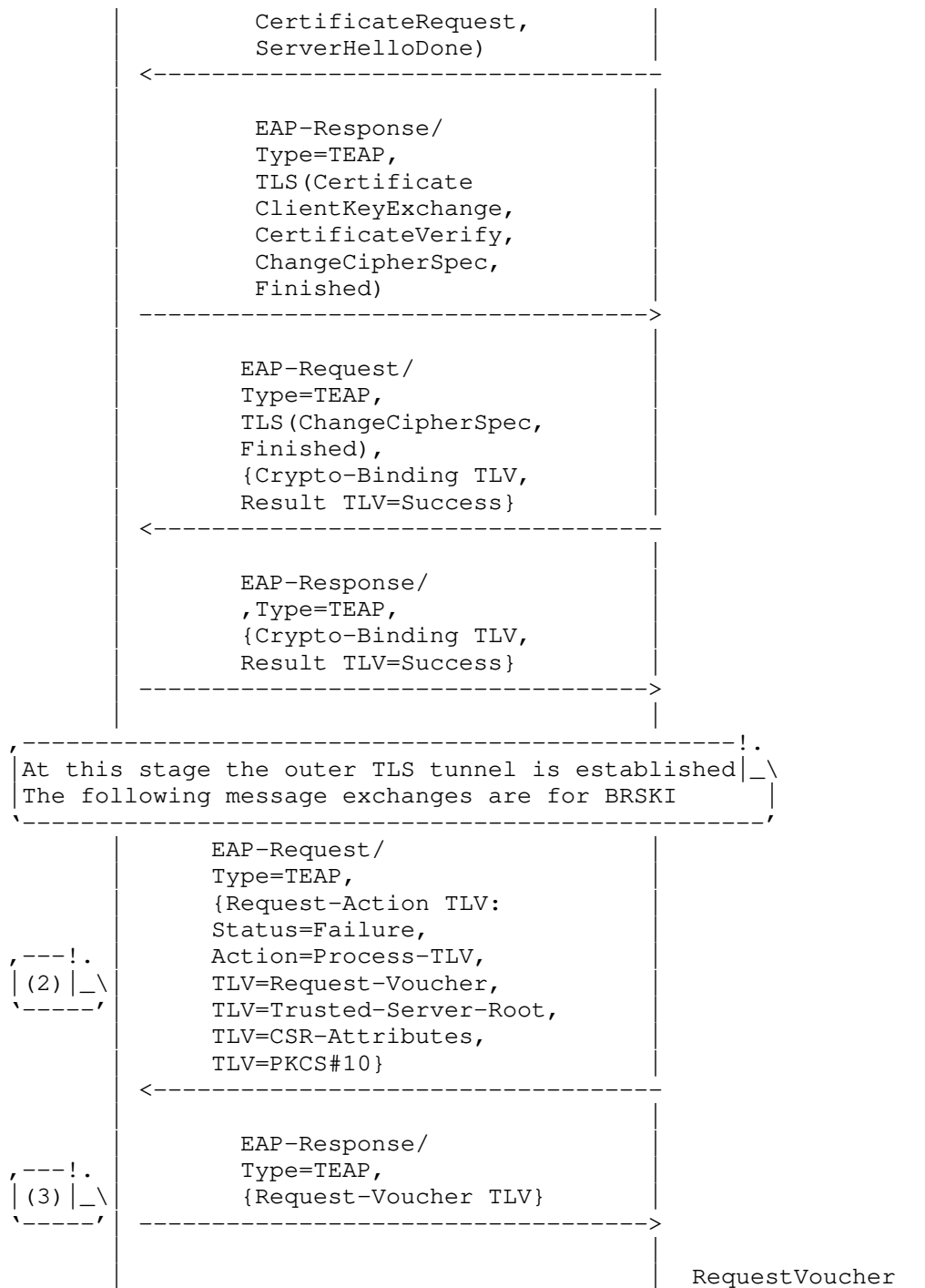
(3) The device will present its `LDevID`, if it has one, in preference to its `IDevID`, if allowed by server policy.

7.2. TEAP Server Instructs Client to Perform BRSKI Flow

In this two part flow, the server instructs the client to perform a BRSKI flow by exchanging TLVs once the outer TLS tunnel is established. After that, enrollment takes place.

In the first part of the flow, the MASA is depicted on the right.





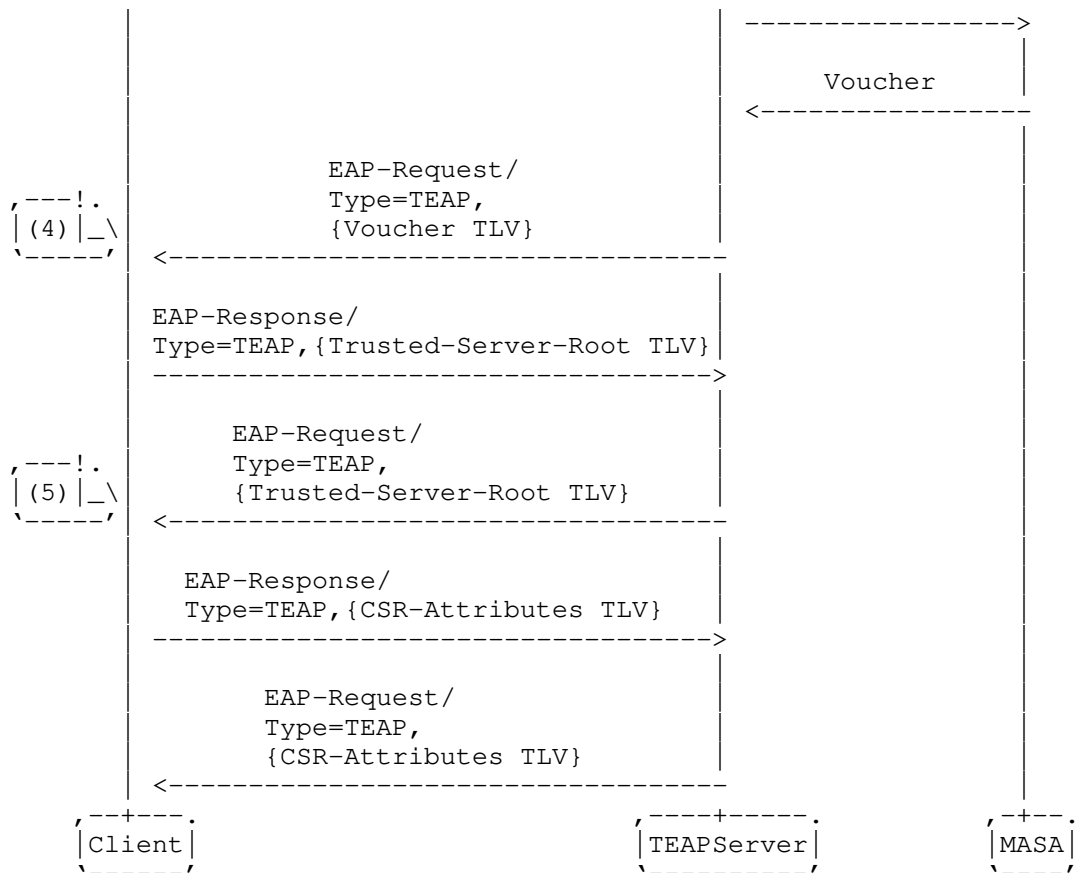


Figure 2: TEAP Server Instructs Client to Perform BRISKI Flow

The second part of the flow depicts the CA on the right.

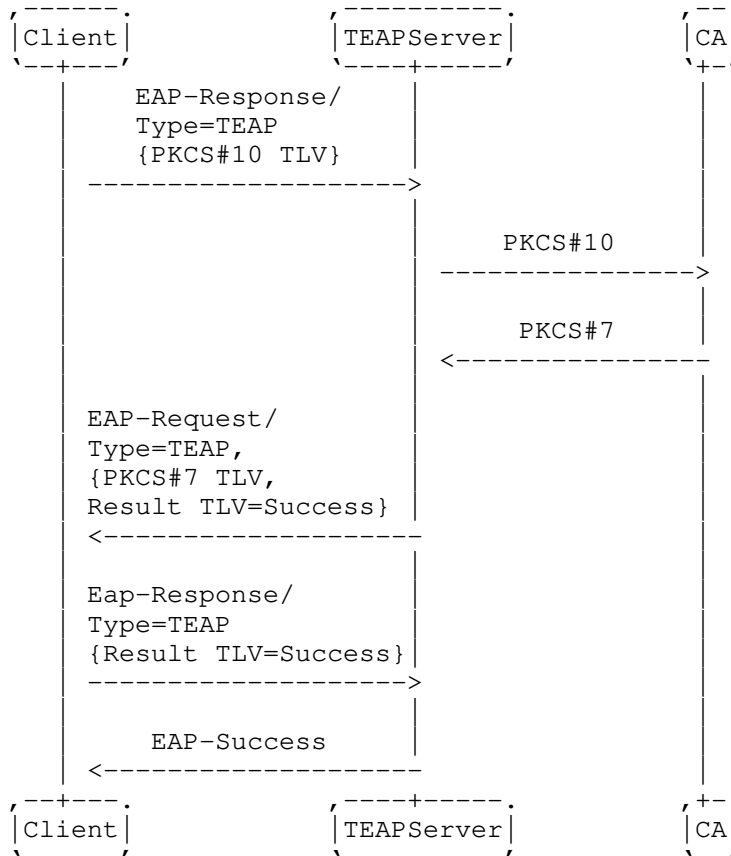


Figure 3: Enrollment after BRSKI Flow

Notes:

(1) If the client has not yet completed the BRSKI flow, then it provisionally accepts the server certificate and must validate it later once BRSKI is complete. The server validates the client certificate using its trust anchor database.

(2) The server instructs the client to start the BRSKI flow by sending a Request-Action TLV that includes a BRSKI-RequestVoucher TLV. The server also instructs the client to request trust anchors, to request CSR Attributes, and to initiate a PKCS certificate enrolment. As outlined in [RFC7170], the Request-Action TLV is sent after the Crypto-Binding TLV and Result TLV exchange.

(3) The client includes the certificate it received from the server in the RequestVoucher message.

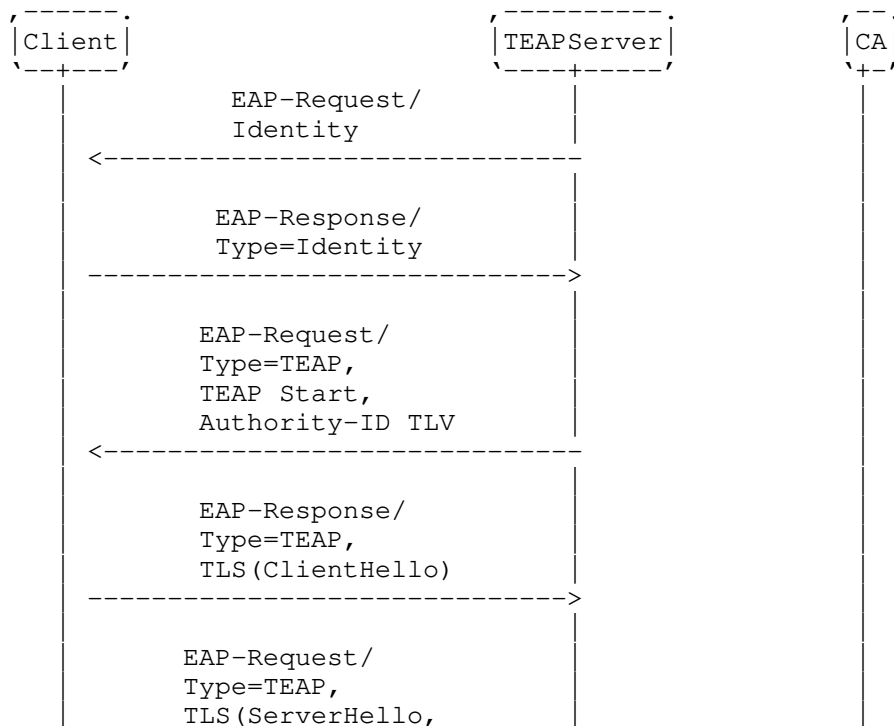
(4) Once the client receives and validates the voucher signed by the MASA, it must verify the certificate it previously received from the server.

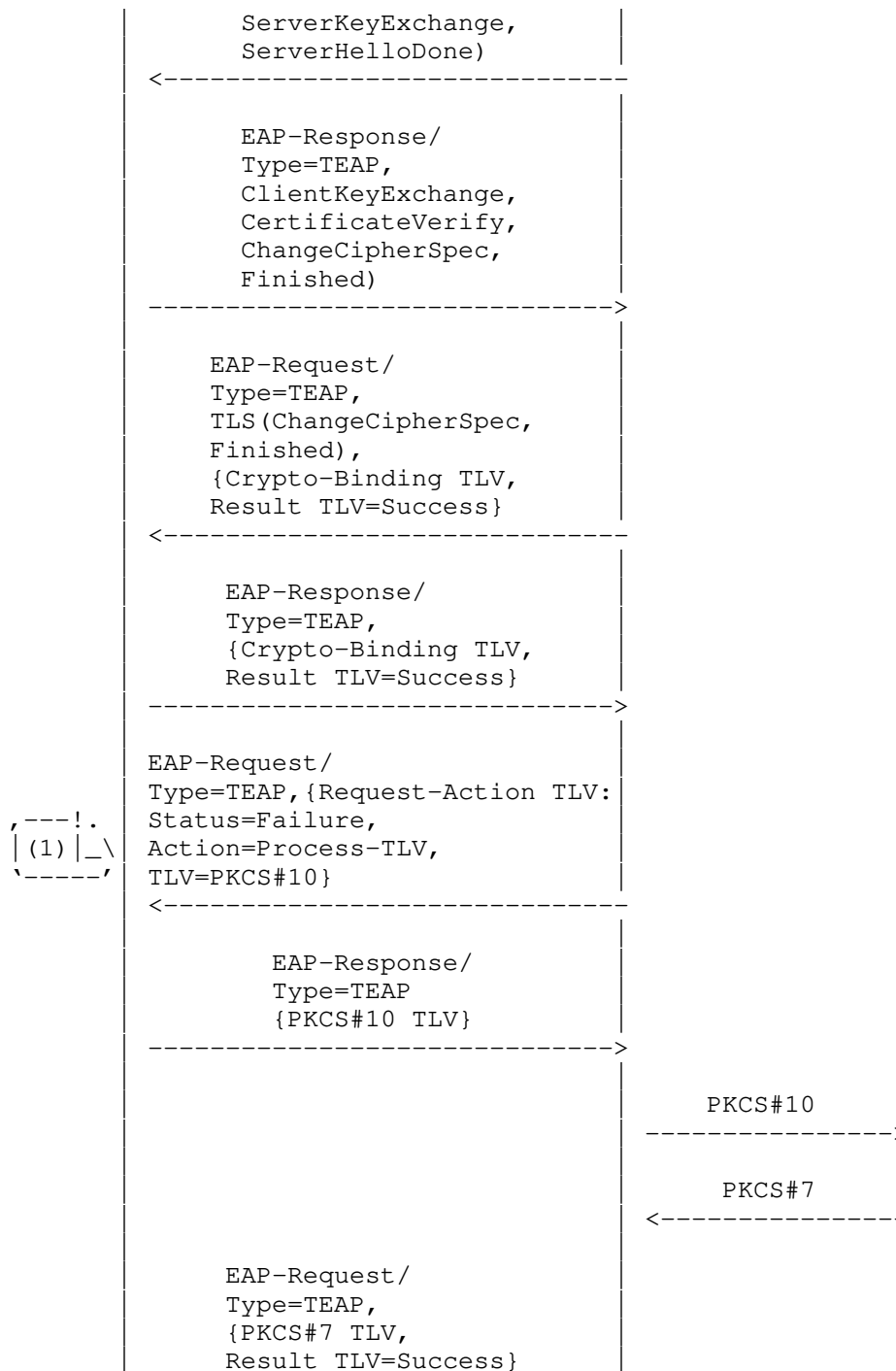
(5) As outlined in [RFC7170], the Trusted-Server-Root TLV is exchanged after the Crypto-Binding TLV exchange, and after the client has used the Voucher to authenticate the TEAP server identity. This is equivalent to section [todo] from [I-D.ietf-anima-bootstrapping-keyinfra].

(6) There is no need for an additional Crypto-Binding TLV exchange as there is no inner EAP method. All BRSKI exchanges are simply TLVs exchanged inside the outer TLS tunnel.

7.3. TEAP Server Instructs Client to Reenroll

In this flow, the server instructs the client to reenroll and get a new LDevID by exchanging TLVs once the outer TLS tunnel is established.





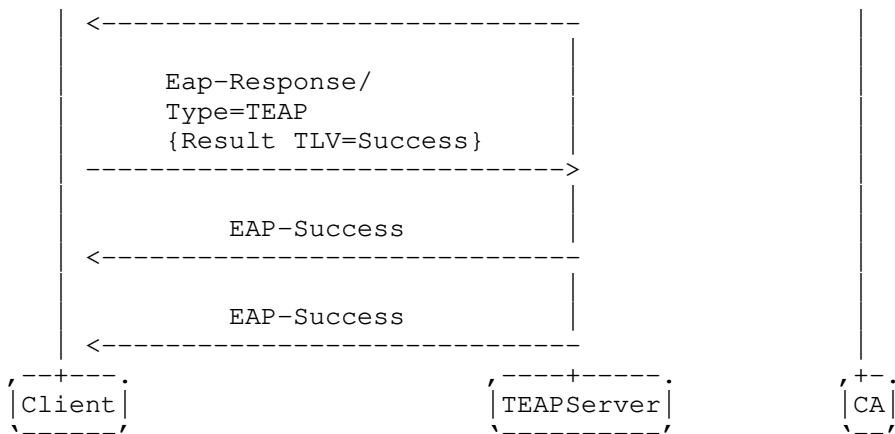


Figure 4: TEAP Server Instructs Client to Reenroll

(1) The server instructs the client to reenroll by sending a Request-Action TLV that includes a PKCS#10 TLV.

7.4. Out of Band Reenroll

This section shows how the device does a reenroll to refresh its LDEVID directly against the registrar outside the context of the TEAP tunnel.

8. TEAP TLV Formats

8.1. New TLVs

This document defines 5 new TEAP TLVs. The following table indicates whether the TLVs can be included in Request messages from TEAP server to device, or Response messages from device to TEAP server.

TLV	Message
BRSKI-VoucherRequest	Response
BRSKI-Voucher	Request
CSR-Attributes	Response
Retry-After	Response
NAI-Identity	Request

These new TLVs are detailed in this section.

8.1.1. BRSKI-RequestVoucher TLV

This TLV is used by the server as part of a Request-Action TLV to request from the peer that it initiate a voucher request. When used in this fashion, the length of this TLV will be set to zero. The Status field of the Request-Action TLV MUST be set to Failure.

It is also used by the peer to initiate the voucher request. When used in this fashion, the length of the TLV will be set to that of the voucher request, as encoded and described in Section 3.3 in [I-D.ietf-anima-bootstrapping-keyinfra].

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|M|R| TLV=TBD1-VoucherRequest | Length |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Value...
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The M and R bits are always expected to be set to 0.

The server is expected to forward the voucher request to the MASA, and then return a voucher in a BRSKI-Voucher TLV as described below. If it is unable to do so, it returns an TEAP Error TLV with one of the defined errors or the following:

```

TBD2-MASA-Notavailable  MASA unavailable
TBD3-MASA-Refused      MASA refuses to sign the voucher

```

The peer terminates the TEAP connection, but may retry at some later point. The backoff mechanism for such retries should be appropriate for the device. Retries MUST occur no more frequently than once every two (TBD) minutes.

8.1.2. BRSKI-Voucher TLV

This TLV is transmitted from the server to the peer. It contains a signed voucher, as describe in [RFC8366].

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|M|R| TLV=TBD4-Voucher | Length |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Value...
+-----+-----+-----+-----+-----+-----+-----+-----+

```


Upon receiving this TLV the peer will validate the signature of the voucher, using its pre-installed manufacturer trust anchor (LDevID). It MUST also validate the certificate used by the server to establish the TLS connection.

If successful, it installs the new trust anchor contained in the voucher.

Otherwise, the peer transmits an TEAP error TLV with one of the following error messages:

TBD5-Invalid-Signature The signature on the voucher is invalid
 TBD6-Invalid-Voucher The form or content of the voucher is invalid
 TBD7-Invalid-TLS-Signer The certificate used for the TLS connection could not be validated.

8.1.3. CSR-Attributes TLV

The server SHALL transmit this TLV to the peer, either along with the BRSKI-Voucher TLV or at any time earlier in a communication. The peer shall include attributes required by the server in any following CSR. The value of this TLV is the base64 encoding described in Section 4.5.2 of [RFC7030].

The TEAP server MAY use this TLV to specify the subject identity information to include in Subject or Subject Alternate Name fields in any following CSR.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-----+-----+-----+-----+-----+-----+-----+-----+
      |M|R| TLV=TBD8-CSR-Attributes |          length          |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | Value...
      +-----+-----+-----+-----+-----+-----+-----+-----+
  
```

Again, the M and R values are set to 0. In the case where the client is unable to provide the requested attributes, an TEAP-Error is returned as follows:

TBD9-CSR-Attribute-Fail Unable to supply the requested attributes.

8.1.4. Retry-After TLV

The server MUST transmit this TLV to the peer when replying to a PKCS#10 TLV request from the peer where the server is willing to fulfill the request and issue a certificate via a PKCS#7 response, but is unable to fulfill the request immediately. This TLV is used to tell the peer the minimum length of time it MUST wait before resending the PKCS#10 request. The value of this TLV is the time in seconds that the peer MUST wait before resending the PKCS#10 request. The peer MUST resend the exact same PKCS#10 request.

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|M|R|  TLV=TBD10-Retry-After  |          length          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value...
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Again, the M and R values are set to 0.

8.1.5. NAI TLV

The server may use this TLV to provision a realm-specific NAI on the device.

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|M|R|  TLV=TBD10-NAI          |          length          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value...
+-----+-----+-----+-----+-----+-----+-----+-----+

```

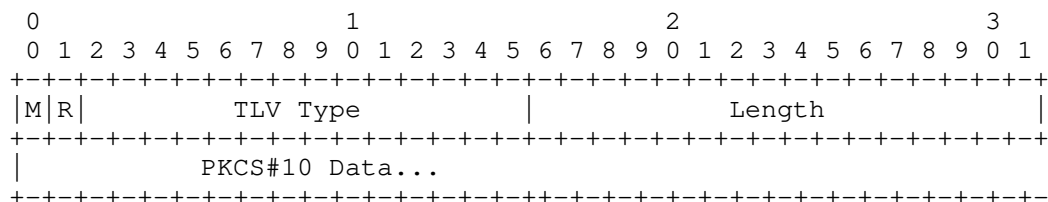
Again, the M and R values are set to 0.

8.2. Existing TEAP TLV Specifications

This section documents allowed usage of existing TEAP TLVs. The definition of the TLV is not changed, however clarifications on allowed values for the TLV fields is documented.

8.2.1. PKCS#10 TLV

[RFC7170] defines the PKCS#10 TLV as follows:



[RFC7170] does not explicitly allow a Length value of zero.

A Length value of zero is allowed for this TLV when the TEAP server sends a Request-Action TLV with a child PKCS#10 TLV to the client. In this scenario, there is no PKCS#10 Data included in the TLV. Clients MUST NOT send a zero length PKCS#10 TLV to the server.

8.3. TLV Rules

BRSKI TLVs can only be transported inside the TLS tunnel. The following table provides a guide to which TLVs may be encapsulated in which kind of packets, and in what quantity. The messages are as follows: Request is a TEAP Request, Response is a TEAP Response, Success is a message containing a successful Result TLV, and Failure is a message containing a failed Result TLV.

The following define the meaning of the table entries in the sections below:

0 This TLV MUST NOT be present in the message.

0+ Zero or more instances of this TLV MAY be present in the message.

0-1 Zero or one instance of this TLV MAY be present in the message.

1 Exactly one instance of this TLV MUST be present in the message.

Request	Response	Success	Failure	TLVs	0	0-1	0	0	BRSKI-VoucherRequest
0-1	0	0	0	BRSKI-Voucher	0	0-1	0	0	CSR-Attributes

9. Fragmentation

TEAP is expected to provide fragmentation support. Thus EAP-TEAP-BRSKI does not specifically provide any, as it is only expected to be used as an inner method to TEAP.

10. IANA Considerations

The IANA is requested to add entries into the following tables:

The following new TEAP TLVs are defined:

TBD1-VoucherRequest	Described in this document.
TBD4-Voucher	Described in this document.
TBD8-CSR-Attributes	Described in this document.
TBD10-Retry-After	Described in this document.

The following TEAP Error Codes are defined, with their meanings listed here and in previous sections:

TBD2-MASA-Notavailable	MASA unavailable
TBD3-MASA-Refused	MASA refuses to sign the voucher
TBD5-Invalid-Signature	The signature on the voucher is invalid
TBD6-Invalid-Voucher	The form or content of the voucher is invalid
TBD7-Invalid-TLS-Signer	The certificate used for the TLS connection could not be validated.
TBD9-CSR-Attribute-Fail	Unable to supply the requested attributes.
TBD11-Retry-PKCS#10	Retry PKCS#10 Request (1000 range code)
TBD12-Retry-PKCS#10	Retry PKCS#10 Request (2000 range code)
TBD13-NAI-Rejected	The device will not use the indicated NAI (1000 range code)

[[TODO: is there a registry of NAIs that map to TEAP methods? e.g. @eap-teap.net is reserved to indicate the peer wants to use TEAP method]]

11. Security Considerations

BRSKI [I-D.ietf-anima-bootstrapping-keyinfra] provides a zero touch way for devices to enroll in a certification authority (CA). It assumes the device has IP connectivity. For networks that will not grant IP connectivity before authenticating (with a local credential) this poses a Catch-22- can't get on the network without a credential and can't get a credential without getting on the network.

This protocol provides a way for BRSKI to be in an EAP method which allows the BRSKI conversation to happen as part of EAP authentication and prior to obtaining IP connectivity.

The security considerations of [I-D.ietf-anima-bootstrapping-keyinfra] apply to this protocol. Running BRSKI through EAP introduces some additional areas of concern though.

11.1. Issues with Provisionally Authenticated TEAP

This protocol establishes an unauthenticated TLS connection and passes data through it. Provided that the only messages passed in this state are self-protected BRSKI messages this does not present a problem. Passing any other messages or TLVs prior to authentication of the provisional TLS connection could potentially introduce security issues.

While the TLS connection is unauthenticated, it must still be validated to the fullest extent possible. It is critical that the device and the TEAP server perform all steps in TLS- checking the validity of the presented certificate, validating the signature using the public key of the certificate, etc- except ensuring the trust of the presented certificate.

11.2. Attack Against Discovery

The device discovery technique specified in this protocol is the standard EAP server discovery process. Since it is trivial to set up an 802.11 wireless access point and advertise any network, an attacker can impersonate a legitimate wireless network and attract unprovisioned pledges. Given that an unprovisioned device will not know the legitimate network to connect to, it will probably attempt the first network it finds, making the attack that much easier. This allows for a "rogue registrar" to provision and take control of the device.

If the MASA verifies ownership prior to issuance of a voucher, this attack can be thwarted. But if the MASA is in reduced security mode and does not verify ownership this attack cannot be prevented. Registrars SHOULD use the audit log of a MASA when deploying newly purchased equipment in order to mitigate this attack.

Another way to mitigate this attack is through normal "rogue AP" detection and prevention.

11.3. TEAP Server as Registration Authority

If the TEAP server is logically separate from the Certification Authority (CA) (see Section 2) it will be acting as a Registration Authority (RA) when it obtains the PKCS#10 TLV and replies with a PKCS#7 TLV (see [RFC7170], Sections 4.2.16 and 4.2.17, respectively). The assurance a RA makes to a CA is that the public key in the presented CSR is bound to an authenticated identity in way that will assure non-repudiation.

To make such an assurance, the TEAP server MUST authenticate the provisional TLS connection with the device by validating the voucher response received from the MASA. In addition, it is RECOMMENDED that the TEAP server indicate that proof-of-possession (see [RFC7170], Section 3.8.2) is required by including the challengePassword OID in the CSR Attributes TLV.

11.4. Trust of Registrar

The device accepts a trusted server (CA) certificate and installs it in its trust anchor database during step 5 (see Section 3.2). This can happen only after the provisional TLS connection has been authenticated using the voucher and the Crypto-Binding TLV has been validated.

12. Acknowledgments

The authors would like to thank Brian Weis for his assistance, and Alan Dakok for improving language consistency. In addition, with ruthlessly "borrowed" the concept around NAI handling from Tuomas Aura and Mohit Sethi.

13. References

13.1. Normative References

- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M. C., Eckert, T., Behringer, M. H., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-bootstrapping-keyinfra-45, 11 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-anima-bootstrapping-keyinfra-45.txt>>.
- [IEEE8021AR]
Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.
- [IEEE8021X]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", 2010.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.

- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
"Enrollment over Secure Transport", RFC 7030,
DOI 10.17487/RFC7030, October 2013,
<<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna,
"Tunnel Extensible Authentication Protocol (TEAP) Version
1", RFC 7170, DOI 10.17487/RFC7170, May 2014,
<<https://www.rfc-editor.org/info/rfc7170>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
"A Voucher Artifact for Bootstrapping Protocols",
RFC 8366, DOI 10.17487/RFC8366, May 2018,
<<https://www.rfc-editor.org/info/rfc8366>>.

13.2. Informative References

- [RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542,
DOI 10.17487/RFC7542, May 2015,
<<https://www.rfc-editor.org/info/rfc7542>>.

Appendix A. Changes from Earlier Versions

Draft -06: * nothing more than version bump

Draft -03: * Merge EAP server and Registrar * Security considerations
* References improvements * Add Dan Harkins as co-author

Draft -02: * Flow corrections

Draft -01: * Add packet descriptions, IANA considerations, smooth out
language.

Draft -00:

* Initial revision

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
CH-8304 Wallisellen
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Owen Friel
Cisco Systems
170 W. Tasman Dr.
San Jose, CA, 95134
United States

Email: ofriel@cisco.com

Nancy Cam-Winget
Cisco Systems
170 W. Tasman Dr.
San Jose, CA, 95134
United States

Email: ncamwing@cisco.com

Dan Harkins
HP Enterprise
3333 Scott Boulevard
Santa Clara, CA, 95054
United States

Email: dharkins@arubanetworks.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 27, 2019

M. Sethi
J. Mattsson
Ericsson
S. Turner
sn3rd
May 26, 2019

Handling Large Certificates and Long Certificate Chains
in TLS-based EAP Methods
draft-ms-emu-eaptlscert-03

Abstract

EAP-TLS and other TLS-based EAP methods are widely deployed and used for network access authentication. Large certificates and long certificate chains combined with authenticators that drop an EAP session after only 40 - 50 round-trips is a major deployment problem. This memo looks at the this problem in detail and describes the potential solutions available.

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 November 27, 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. Introduction	2
2. Terminology	3
3. Experience with Deployments	4
4. Handling of Large Certificates and Long Certificate Chains	4
4.1. Updating Certificates and Certificate Chains	4
4.1.1. Guidelines for certificates	5
4.2. Updating TLS and EAP-TLS Code	6
4.2.1. Pre-distributing and Omitting CA Certificates	6
4.2.2. Caching Certificates	6
4.2.3. Compressing Certificates	7
4.2.4. Suppressing Intermediate Certificates	7
4.3. Updating Authenticators	7
5. IANA Considerations	8
6. Security Considerations	8
7. References	8
7.1. Normative References	8
7.2. Informative References	9
Acknowledgements	10
Authors' Addresses	10

1. Introduction

The Extensible Authentication Protocol (EAP), defined in [RFC3748], provides a standard mechanism for support of multiple authentication methods. EAP-Transport Layer Security (EAP-TLS) [RFC5216] [I-D.ietf-emu-eap-tls13] relies on TLS [RFC8446] to provide strong mutual authentication with certificates [RFC5280] and is widely deployed and often used for network access authentication. There are also many other TLS-based EAP methods, such as FAST [RFC4851], TTLS [RFC5281], TEAP [RFC7170], and possibly many vendor specific EAP methods.

TLS certificates are often relatively large, and the certificate chains are often long. Unlike the use of TLS on the web, where typically only the TLS server is authenticated; EAP-TLS deployments typically authenticates both the EAP peer and the EAP server. Also, from deployment experience, EAP peers typically have longer certificate chains than servers. Therefore, EAP-TLS authentication usually involve significantly more bytes than when TLS is used as part of HTTPS.

As the EAP fragment size in typical deployments are just 1000 - 1500 bytes, the EAP-TLS authentication needs to be fragmented into many smaller packets for transportation over the lower layers. Such fragmentation can not only negatively affect the latency, but also results in other challenges. For example, many EAP authenticator (access point) implementations will drop an EAP session if it hasn't finished after 40 - 50 round-trips. This is a major problem and means that in many situations, the EAP peer cannot perform network access authentication even though both the sides have valid credentials for successful authentication and key derivation.

This memo looks at related work and potential tools available for overcoming the deployment challenges induced by large certificates and long certificate chains. It then discusses the solutions available to overcome these challenges.

2. Terminology

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.

Readers are expected to be familiar with the terms and concepts used in EAP-TLS [RFC5216] and TLS [RFC8446]. In particular, this document frequently uses the following terms as they have been defined in [RFC5216]:

Authenticator The entity initiating EAP authentication. Typically implemented as part of a network switch or a wireless access point.

EAP peer The entity that responds to the authenticator. In [IEEE-802.1X], this entity is known as the supplicant. In EAP-TLS, the EAP peer implements the TLS client role.

EAP server The entity that terminates the EAP authentication method with the peer. In the case where no backend authentication server is used, the EAP server is part of the authenticator. In the case where the authenticator operates in pass-through mode, the EAP server is located on the backend authentication server. In EAP-TLS, the EAP server implements the TLS server role.

3. Experience with Deployments

The EAP fragment size in typical deployments can be 1000 - 1500 bytes. Certificate sizes can be large for a number of reasons:

- o Long Subject Alternative Name field.
- o Long Public Key and Signature fields.
- o Can contain multiple object identifiers (OID) that indicate the permitted uses of the certificate. For example, Windows requires certain OID's in the certificates for EAP-TLS to work.
- o Multiple user groups in the certificate.

The certificate chain can typically include 2 - 6 certificates to the root-of-trust.

Most common access point implementations drop EAP sessions that don't complete within 50 round-trips. This means that if the chain is larger than ~ 60 kB, EAP-TLS authentication cannot complete successfully in most deployments.

4. Handling of Large Certificates and Long Certificate Chains

This section discusses some possible alternatives for overcoming the challenge of large certificates and long certificate chains in EAP-TLS authentication. In Section 4.1 we look at recommendations that require an update of the certificates or certificate chains that are used for EAP-TLS authentication without requiring changes to the existing EAP-TLS code base. We also provide some guidelines when issuing certificates for use with EAP-TLS. In Section 4.2 we look at recommendations that rely on updates to the EAP-TLS implementations which can be deployed with existing certificates. In Section 4.3 we shortly discuss the solution to update or reconfigure authenticator which can be deployed without changes to existing certificates or EAP-TLS code.

4.1. Updating Certificates and Certificate Chains

Many IETF protocols now use elliptic curve cryptography (ECC) [RFC6090] for the underlying cryptographic operations. The use of ECC can reduce the size of certificates and signatures. For example, at a 128-bit security level, the size of public keys with traditional RSA is about 384 bytes, while the size of public keys with ECC is only 32-64 bytes. Similarly, the size of digital signatures with traditional RSA is 384 bytes, while the size is only 64 bytes with elliptic curve digital signature algorithm (ECDSA) and Edwards-curve

digital signature algorithm (EdDSA) [RFC8032]. Using certificates that use ECC can reduce the number of messages in EAP-TLS authentication which can alleviate the problem of authenticators dropping an EAP session because of too many round-trips. TLS 1.3 [RFC8446] requires implementations to support ECC. New cipher suites that use ECC are also specified for TLS 1.2 [RFC5289]. Using ECC based cipher suites with existing code can significantly reduce the number of messages in a single EAP session.

4.1.1. Guidelines for certificates

This section provides some recommendations for certificates used for EAP-TLS authentication:

- o Object Identifiers (OIDs) is ASN.1 data type that defines unique identifiers for objects. The OID's ASN.1 value, which is a string of integers, is then used to name objects to which they relate. The DER length for the 1st two integers is always one byte and subsequent integers are base 128-encoded in the fewest possible bytes. OIDs are used lavishly in X.509 certificates and while not all can be avoided, e.g., OIDs for extensions or algorithms and their associate parameters, some are well within the certificate issuer's control:
 - * Each naming attribute in a DN (Directory Name) has one. DNs used in the issuer and subject fields as well as numerous extensions. A shallower naming will be smaller, e.g., C=FI, O=Example, SN=B0A123499EFC vs C=FI, O=Example, OU=Division 1, SOPN=Southern Finland, CN=Coolest IoT Gadget Ever, SN=B0A123499EFC.
 - * Every certificate policy (and qualifier) and any mappings to another policy uses identifiers. Consider carefully what policies apply.
- o DirectoryString and GeneralName types are used extensively to name things, e.g., the DN naming attribute O= (the organizational naming attribute) DirectoryString includes "Example" for the Example organization and uniformResourceIdentifier can be used to indicate the location of the CRL, e.g., "http://crl.example.com/sfig2s1-128.crl", in the CRL Distribution Point extension. For these particular examples, each character is a byte. For some non-ASCII character strings in the DN, characters can be multi-byte. Obviously, the names need to be unique, but there is more than one way to accomplish this without long strings. This is especially true if the names are not meant to be meaningful to users.

- o Extensions are necessary to comply with [RFC5280], but the vast majority are optional. Include only those that are necessary to operate.

4.2. Updating TLS and EAP-TLS Code

4.2.1. Pre-distributing and Omitting CA Certificates

The TLS Certificate message conveys the sending endpoint's certificate chain. TLS allows endpoints to reduce the sizes of the Certificate messages by omitting certificates that the other endpoint is known to possess. When using TLS 1.3, all certificates that specify a trust anchor known by the other endpoint may be omitted (see Section 4.4.2 of [RFC8446]). When using TLS 1.2 or earlier, only the self-signed certificate that specifies the root certificate authority may be omitted (see Section 7.4.2 of [RFC5246]). Therefore, updating TLS implementations to version 1.3 can help to significantly reduce the number of messages exchanged for EAP-TLS authentication. The omitted certificates need to be pre-distributed independently of TLS and the TLS implementation need to be configured to omit the pre-distributed certificates.

4.2.2. Caching Certificates

The TLS Cached Information Extension [RFC7924] specifies an extension where a server can exclude transmission of certificate information cached in an earlier TLS handshake. The client and the server would first execute the full TLS handshake. The client would then cache the certificate provided by the server. When the TLS client later connects to the same TLS server without using session resumption, it can attach the "cached_info" extension to the ClientHello message. This would allow the client to indicate that it has cached the certificate. The client would also include a fingerprint of the server certificate chain. If the server's certificate has not changed, then the server does not need to send its certificate and the corresponding certificate chain again. In case information has changed, which can be seen from the fingerprint provided by the client, the certificate payload is transmitted to the client to allow the client to update the cache. The extension however necessitates a successful full handshake before any caching. This extension can be useful when, for example, when a successful authentication between an EAP peer and EAP server has occurred in the home network. If authenticators in a roaming network are more strict at dropping long EAP sessions, an EAP peer can use the Cached Information Extension to reduce the total number of messages.

However, if all authenticators drop the EAP session for a given EAP peer and EAP server combination, a successful full handshake is not

possible. An option in such a scenario would be to cache validated certificate chains even if the EAP-TLS exchange fails, but this is currently not allowed according to [RFC7924].

4.2.3. Compressing Certificates

The TLS working group is also working on an extension for TLS 1.3 [I-D.ietf-tls-certificate-compression] that allows compression of certificates and certificate chains during full handshakes. The client can indicate support for compressed server certificates by including this extension in the ClientHello message. Similarly, the server can indicate support for compression of client certificates by including this extension in the CertificateRequest message. While such an extension can alleviate the problem of excessive fragmentation in EAP-TLS, it can only be used with TLS version 1.3 and higher. Deployments that rely on older versions of TLS cannot benefit from this extension.

4.2.4. Suppressing Intermediate Certificates

For a client that has all intermediates, having the server send intermediates in the TLS handshake increases the size of the handshake unnecessarily. The TLS working group is working on an extension for TLS 1.3 [I-D.thomson-tls-sic] that allows a TLS client that has access to the complete set of published intermediate certificates to inform servers of this fact so that the server can avoid sending intermediates, reducing the size of the TLS handshake. The mechanism is intended to be complementary with certificate compression.

4.3. Updating Authenticators

There are several legitimate reasons that Authenticators may want to limit the number of round-trips/packets/bytes that can be sent. The main reason has been to work around issues where the EAP peer and EAP server end up in an infinite loop ACKing their messages. Another second reason is that unlimited communication from an unauthenticated device as EAP could otherwise be used for bulk data transfer. A third reason is to prevent denial-of-service attacks.

Updating the millions of already deployed access points and switches is in many cases not realistic. Vendors may be out of business or do no longer support the products and admins may have lost the login information to the devices. For practical purposes the EAP infrastructure is ossified for the time being.

Vendors making new authenticators should consider increasing the number of round-trips allowed before denying the EAP authentication to complete.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

TBD

7. References

7.1. Normative References

- [I-D.ietf-emu-eap-tls13]
Mattsson, J. and M. Sethi, "Using EAP-TLS with TLS 1.3", draft-ietf-emu-eap-tls13-04 (work in progress), March 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC4851] Cam-Winget, N., McGrew, D., Salowey, J., and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, DOI 10.17487/RFC4851, May 2007, <<https://www.rfc-editor.org/info/rfc4851>>.
- [RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <<https://www.rfc-editor.org/info/rfc5216>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.

- [RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <<https://www.rfc-editor.org/info/rfc5281>>.
- [RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <<https://www.rfc-editor.org/info/rfc7170>>.
- [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>>.

7.2. Informative References

- [I-D.ietf-tls-certificate-compression] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", draft-ietf-tls-certificate-compression-05 (work in progress), April 2019.
- [I-D.thomson-tls-sic] Thomson, M., "Suppressing Intermediate Certificates in TLS", draft-thomson-tls-sic-00 (work in progress), March 2019.
- [IEEE-802.1X] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks -- Port-Based Network Access Control", IEEE Standard 802.1X-2010, February 2010.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <<https://www.rfc-editor.org/info/rfc5289>>.
- [RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <<https://www.rfc-editor.org/info/rfc6090>>.

- [RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <<https://www.rfc-editor.org/info/rfc7924>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

Acknowledgements

This draft is a result of several useful discussions with Alan DeKok, Bernard Aboba, Jari Arkko, Darshak Thakore, and Hannes Tschofenig.

Authors' Addresses

Mohit Sethi
Ericsson
Jorvas 02420
Finland

Email: mohit@piuha.net

John Mattsson
Ericsson
Kista
Sweden

Email: john.mattsson@ericsson.com

Sean Turner
sn3rd

Email: sean@sn3rd.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2020

M. Pala
CableLabs
June 3, 2020

Credentials Provisioning and Management via EAP (EAP-CREDS)
draft-pala-eap-creds-07

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 5, 2020.

Copyright Notice

Copyright (c) 2020 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	3
2.1. Overview of existing solutions	4
2.2. Scope Statement	4
2.3. EAP-CREDS as tunneled mechanism only	5
2.4. Fragmentation Support	5
2.5. Encapsulating Provisioning Protocols in EAP-CREDS	5
2.6. Algorithm Requirements	6
2.7. Notation	6
3. EAP-CREDS Protocol	6
3.1. Message Flow	6
3.2. Phase Transitioning Rules	7
3.3. Phase One: Initialization	8
3.4. Phase Two: Provisioning	10
3.5. Phase Three: Validation	12
4. EAP-CREDS Message Format	15
4.1. Message Header	15
4.2. Message Payload	17
4.3. EAP-CREDS defined TLVs	18
4.3.1. The Action TLV	18
4.3.2. The Certificate-Data TLV	19
4.3.3. The Challenge-Data TLV	20
4.3.4. The Challenge-Response TLV	21
4.3.5. The Credentials-Information TLV	21

4.3.6.	The Credentials-Data TLV	24
4.3.7.	The Error TLV	25
4.3.8.	The Network-Usage TLV	26
4.3.9.	The Profile TLV	28
4.3.10.	The Protocol TLV	29
4.3.11.	The Provisioning-Data TLV	29
4.3.12.	The Provisioning-Headers TLV	30
4.3.13.	The Provisioning-Params TLV	31
4.3.14.	The Token-Data TLV	33
4.3.15.	The Version TLV	34
5.	EAP-CREDS Messages	35
5.1.	The EAP-CREDS-Init Message	35
5.1.1.	EAP Server's Init Message	35
5.1.2.	EAP Peer's Init Message	36
5.1.2.1.	Bootstrapping Peer's Trustworthiness	36
5.1.3.	The EAP-CREDS-Provisioning Message	37
5.1.4.	The EAP-CREDS-Validate Message	38
6.	Error Handling in EAP-CREDS	39
7.	IANA Considerations	39
7.1.	Provisioning Protocols	39
7.2.	Token Types	39
7.3.	Credentials Types	40
7.4.	Credentials Algorithms	40
7.5.	Credentials Datatypes	41
7.6.	Challenge Types	41
7.7.	Network Usage Datatypes	42
7.8.	Credentials Encoding	42
7.9.	Action Types	42
7.10.	Usage Metadata Types	43
8.	Security Considerations	43
9.	Acknowledgments	44
10.	Normative References	44
	Author's Address	45

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 802.1x architecture provides network administrators with the possibility to check credentials presented by a device even before providing any connectivity or IP services 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. EAP-CREDS cannot be used as a stand-alone method, it is required that EAP-CREDS is used as an inner method of EAP-TLS, EAP-TEAP, or any other tunneling method that can provide the required secrecy and (at minimum) server-side authentication to make sure that the communication is protected and with the right server.

2.1. 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], and Automated Certificate Management Environment (ACME) . 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). 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 selected provisioning protocol.

2.2. 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.

2.3. 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, EAP-CREDS assumes that an appropriately encrypted and authenticated channel 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, or EAP-TEAP are examples of such mechanisms.s

2.4. Fragmentation Support

EAP does not directly support handling fragmented packets and it requires the outer method to 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

2.5. 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 encoding 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, the 'D' bit in the message headers is used for this purpose.

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 managed by the network. In particular, the server provides, at minimum, the ('Protocol') TLV, the ('Action') TLV, and the ('Provisioning-Params') or the ('Credentials-Info') TLV.

After checking the parameters sent by the Server, if the Peer does not support any of the proposed ones, 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 where 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 Provisioning messages until an error is detected (and the appropriate error message is sent to the other party) or until Phase Two is successfully completed.

2.6. 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.

2.7. Notation

In this document we use the following notation in the diagrams to provide information about the cardinality of the data structures (TLVs) within EAP-CREDS messages:

Symbol	Example	Usage
{ }	{TLV1}	Curly Brackets are used to indicate a set
[]	{[TLV2]}	Square Brackets are used to indicate that a field is optional
()	{TLV1(=V)}	Round Squares are used to specify a value
+	{TLV_2+}	The Plus character indicates that one or more instances are allowed

Table 1: EAP-CREDS Notation

3. EAP-CREDS Protocol

In a nutshell, EAP-CREDS provides the abstraction layer on top of which credentials provisioning/managing protocols can be deployed thus enabling their use even before provisioning IP services.

This section outlines the operation of the protocol and message flows. The format of the CREDS messages is given in Section 4.

3.1. Message Flow

EAP-CREDS message flow is logically subdivided into three different phases: Initialization, Provisioning, and Validation. EAP-CREDS

enforces the order of phases, i.e. it is not possible to move to an earlier phase.

Phase transitioning is controlled by the Server. In particular, the server, after the last message of a phase, it can decide to either (a) start the next phase by sending the first message of the next phase, or (b) continue the same phase by sending another "first" message of the phase (e.g., managing a second set of credentials) - this is allowed only in Phase Two and Phase Three but NOT in Phase One, or (c) terminate the EAP session.

Phase One (Required). Initialization. During this phase the Peer and the Server exchange the information needed to select the appropriate credentials management protocol. Phase One flow is composed by only messages. In particular, the Server sends its initial message of type ('EAP-CREDS-Init'). The Peer replies with the details about which provisioning protocols are supported, and additional information such as the list of installed credentials and, optionally, authorization data (for new credentials registration).

Phase Two (Optional). 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 Provisioning. The messages use two main TLVs. The first one is the ('Provisioning-Headers') TLV which is optional and carries information that might be normally conveyed via the transport protocol (e.g., HTTP headers). The second one is the ('Provisioning-Data'), which is required and carries the provisioning protocol's messages. The server can decide to repeat phase two again to register new credentials or to renew a separate set of credentials by issuing a new ('Provisioning') message for the new target. When no more credentials have to be managed, the Server can start phase three or simply terminate 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.

3.2. Phase Transitioning Rules

In order to keep track of starting and ending a phase, EAP-CREDS defines several bits and fields in the EAP-CREDS message headers. In particular, as described in Section 4.1, the 'S' bit is used to

indicate the beginning (or Start) of a phase, while the 'Phase' field (4 bits) is used to indicate the phase for this message.

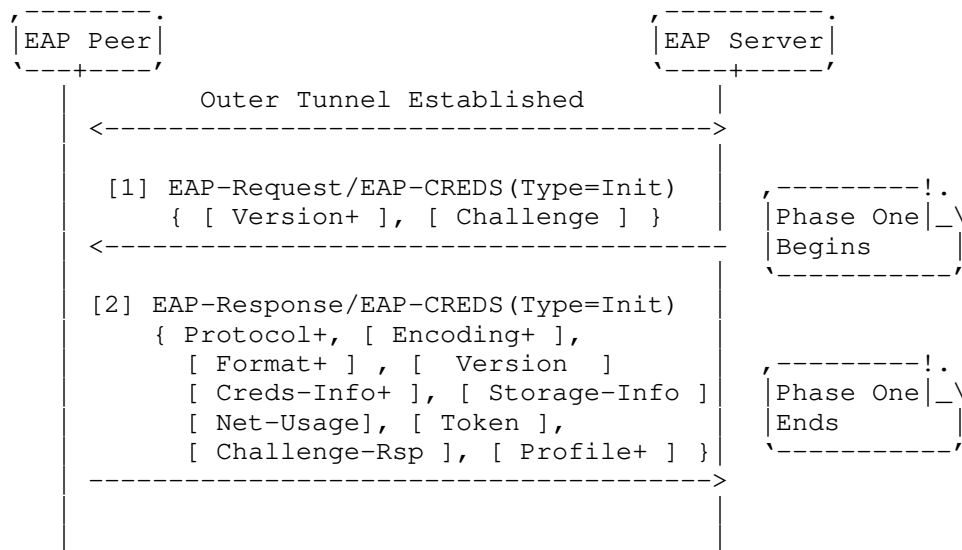
In EAP-CREDS, phase transitioning is under the sole control of the Server, therefore the value of the 'S' bit is meaningful only in messages sent by the Server. The value of the 'S' bit in Peer's messages SHALL be set to '0x0' and SHALL be ignored by the server.

When starting a new phase, the Server MUST set the 'S' bit to '1' and the 'Phase' field to the current phase number (e.g., one, two, or three).

In case the first message of a phase is to be repeated (e.g., because of processing multiple credentials), the 'S' bit SHALL be set to '0' (i.e., it should be set to '1' only on the first occurrence and set to '0' in subsequent messages).

3.3. Phase One: Initialization

The following figure provides the message flow for Phase One:



EAP-CREDS Phase One Message Flow

[1] Server sends EAP-Request/EAP-CREDS (Type=Init):

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, the Server sends an EAP-Request/EAP-CREDS(Type=Init) message to the Peer with one ('Phase-Control') TLV with the 'S' bit set to '1' and the value set to '1' (thus indicating the beginning of Phase One). Also, the Server MAY use one or more ('Version') TLVs to indicate the supported versions.

The Server MAY also specify which versions of EAP-CREDS are supported by adding one or more ('Version') TLVs. If no ('Version') TLV is added to the message, the Peer SHOULD assume the supported version is 1 ('0x1').

[2] The Peer sends EAP-Response/EAP-CREDS(Type=Init)

The Peer, sends back a message that carries one ('Version') TLV to indicate the selected version of EAP-CREDS (i.e. from the list provided by the server) (optional). If the client does not include the ('Version') TLV, the Server MUST use the most recent supported version of EAP-CREDS. Moreover, the Server includes one or more ('Protocol') TLVs to indicate the list of supported provisioning protocols, followed by one ('Credentials-Info') TLVs for each installed credentials to provide their status to the server (i.e., if multiple credentials are configured on the Peer for this Network, then the Peer MUST include one ('Credentials-Info') TLV for each of them).

The Peer also provides the list of supported Encodings and Formats by adding one or more ('Supported-Encodings') and ('Supported-Formats') TLVs. The Peer MAY also provide the Server with information about the Peer's credentials storage by using the 'Storage-Status' TLV.

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 its response 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 value: the signature SHALL be encoded in the ('Challenge-Response') TLV and it should be calculated over the concatenation of values inside the ('Challenge-Data') TLV and the ('Token-Data') TLV.

Also, the Peer MAY add one or more ('Profile') TLVs to indicate to the Server which profiles are requested/supported (e.g., a pre-configuration MAY exist on the Peer with these ecosystem-specific identifiers).

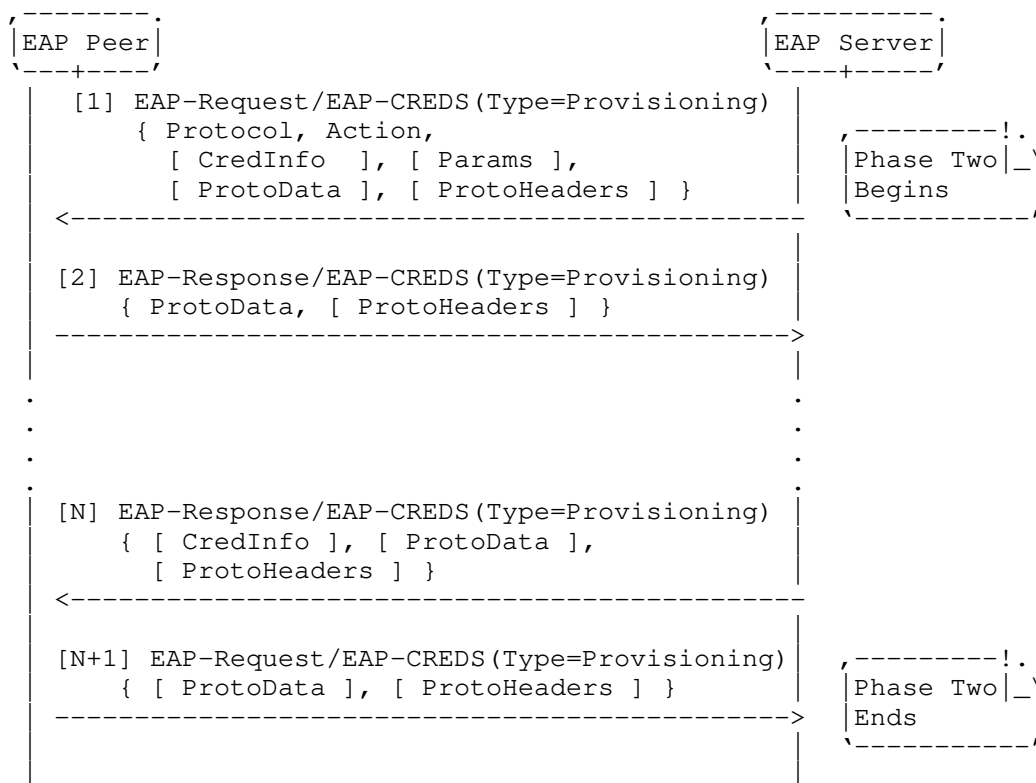
Ultimately, the Peer MAY include additional metadata regarding the status of the Peer. To this end, the Peer can use a ('Storage-Info') TLV to provide the server with additional data about the Peer's capabilities and resources. Also, the ('Network-Usage') TLV can be used to provide the Server with the indication of which network resources are needed by the Peer and what is its intended utilization pattern(s).

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. In this case, the ('Phase-Control') TLV MUST be omitted since it is not the first message of phase one. The server and the peer can repeat phase one until they reach an agreement or the session is terminated by the Server.

NOTE WELL: The determination of the need to start phase two or not is based on the contents of the ('Credentials-Info') TLV sent by the Peer (e.g., a credential is about to expire or a credential is simply missing).

3.4. Phase Two: Provisioning

The following figure provides the message flow for Phase 2:



EAP-CREDS Phase Two Message Flow

- [1] The Server sends EAP-Request/EAP-CREDS (Type=Init)

The first message of Phase Two indicates that the Server is ready to initiate the selected provisioning protocol.

- [2] The Peer sends EAP-Response/EAP-CREDS (Type=Init)

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

- [3] The Server sends EAP-Request/EAP-CREDS (Type=Init)

The Server replies to the Peer's message with EAP-Request/EAP-CREDS (Type=Provisioning) messages until the provisioning protocol reaches an end or an error condition arise (non-recoverable).

[N] The Server sends EAP-Request/EAP-CREDS (Type=Provisioning)

When the provisioning protocol has been executed for the specific set of credentials, the server sends a last message that MUST include the description of the provisioned credentials in a ('Credentials-Info') TLV and MUST set the 'D' bit in the EAP-CREDS message header to '1' to indicate that the server does not have any more ('Provisioning') messages for this credential. The final message does not need to be an empty one, i.e. other TLVs are still allowed in the same message (e.g., the 'Provisioning-Data' and the 'Provisioning-Headers' ones).

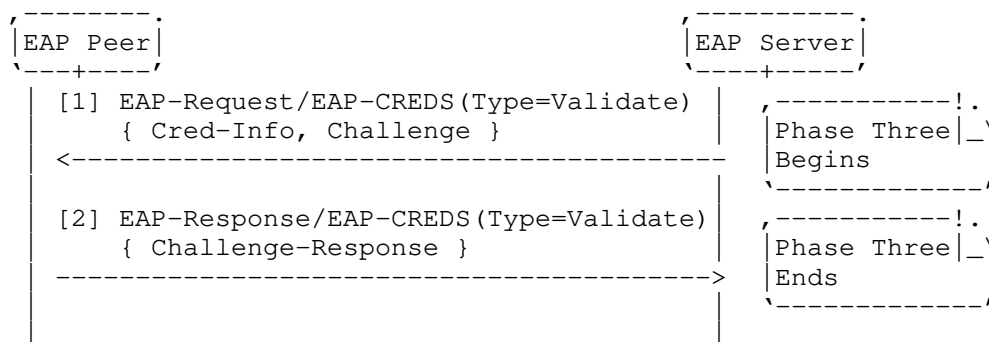
[N+1] The Peer sends EAP-Request/EAP-CREDS (Type=Provisioning)

The Peer MUST reply to the server with a ('Provisioning') message that MUST have the 'D' bit in the EAP-CREDS message header set to '1', thus indicating that the credentials have been installed correctly. In case of errors, the Peer MUST include the appropriate ('Error') TLV. Also in this case, the final message does not need to be an empty one, i.e. other TLVs are still allowed in the same message (e.g., the 'Provisioning-Data' and the 'Provisioning-Headers' ones).

At this point, the Server can decide to provision (or manage) another set of credentials by issuing a new ('Provisioning') message, or it can decide to start Phase Three by sending its first ('Validate') message, or it can terminate the EAP session.

3.5. Phase Three: Validation

The following figure provides the message flow for Phase 3:



EAP-CREDS Phase Three Message Flow

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 a dependency on the selected hashing algorithm to provide common and easy way to check the integrity and functionality of a newly installed set of credentials.

[1] The Server sends EAP-Request/EAP-CREDS (Type=Validate)

In order to start Phase Three, the Server sends an EAP-Request/EAP-CREDS (Type=Validate) message to the Peer. The Server MUST include the ('Credentials-Info') TLV to provide the indication about which set of credentials the Server intends to validate. The Server MUST also include a randomly generated challenge in the message to the client. The type of challenge determines how the ('Challenge-Response') is calculated. EAP-CREDS defines the asymmetric and symmetric challenges in Section 7.6 and others can be defined according to the specified rules.

As usual, the Server MUST set, in the headers, the 'S' bit to '1' in its first message of Phase Three and the 'Phase' value shall be set to '3' (beginning of Phase Three).

[2] The Peer sends EAP-Response/EAP-CREDS (Type=Validate)

When the client receives the Validate message from the server, it calculates the response to the challenge and sends the response back to the server in a EAP-Response/EAP-CREDS (Type=Validate) message. When the EAP-CREDS-ASYMMETRIC-CHALLENGE and EAP-CREDS-

SYMMETRIC-CHALLENGE values are used in the Challenge type, the Peer MUST calculate the response as follows:

Public-Key

For any public-key based credentials (e.g., certificates or raw key pairs), 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 2.6. The format of the signature in the ('Challenge-Response') TLV is the concatenation of:

- The signatureAlgorithm (DER encoded) which contains the identifier for the cryptographic algorithm used by the Peer to generate the signature. [RFC3279], [RFC4055], and [RFC4491] list supported signature algorithms, but other signature algorithms MAY also be supported. The definition of the signatureAlgorithm is provided in Section 4.1.1.2 of [RFC5280].
- The signatureValue (DER encoded) which contains the digital signature itself. The signature value is encoded as a BIT STRING and the details of how to generate the signatures' structures can be found in Section 4.1.1.3 of [RFC5280] and referenced material.

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 (see Section 2.6) on the concatenation of (a) the value carried in the server-provided ('Challenge-Data') TLV, and (b) the secret value itself (salted hash).

The initial values for the type of challenges are described in the Section 7.6. Other types of challenges MAY be defined according to the specified procedures.

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

4. EAP-CREDS Message Format

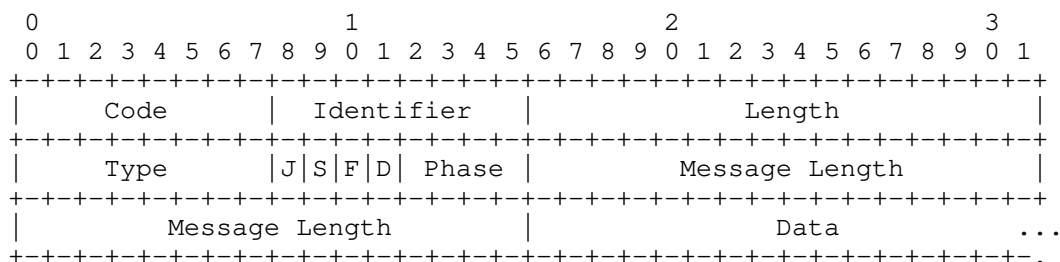
The EAP-CREDS defines the following message types:

1. EAP-CREDS/Init
2. EAP-CREDS/Provisioning
3. EAP-CREDS/Validate

Each of these message types have the basic structure as identified in Section 4.1. EAP-CREDS messages contain zero, one, or more TLVs. The internal structure of the different types of TLVs is described in Section 4.2, while a detailed description of the EAP-CREDS message types is provided in Section 5.

4.1. 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:



Where the Code, Identifier, Length, and Type fields are all part of the EAP header as defined in [RFC3748]. Since EAP-CREDS can only be used as a tunneled mechanism, the presence of these fields is only for backward compatibility with existing parsers. In particular, the 'Length' field is not used (can be ignored): the message length is carried in the 'Message Length' field instead.

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

The Flags bitfield is used to convey status information (e.g., extra long message, phase number, phase transitioning state). The transition-control bit (i.e., the 'S' bit) are set in Server's messages and are ignored in Peer's messages (the Server is the entity

that unilaterally controls the phase transition process). The meanings of the bits in the 'Flags' field are as follows:

Bit 'J' (Jumbo Message) - If set, it indicates the presence of the 'Message Length' field. This bit SHALL be used only when the size of the message exceeds the maximum value allowed in the 'Length' field. In this case, the 'Message Length' field is added to the message and set to the whole message size and the 'Length' field is used for the current fragment length. If not set, the 'Message Length' field is not present in the Message and the 'Length' field is used for the message size (and the 'F' bit MUST be set to '0').

Bit 'S' (Start) - If set, this message is the first one of a new EAP-CREDS phase. The value of the new phase is encoded in the 'Phase' field.

Bit 'F' - If set, this message is a fragment of a message. In this case, the 'Data' field is to be concatenated with all messages with the 'F' bit set to '1' until the message with the 'F' bit set to '0' that indicates the end of the message. If the message is not fragmented, the 'F' bit MUST be set to '0'. The use of this bit is required when the tunneling method does not provide support for messages up to 2^{32} bits in size.

Bit 'D' - This bit is used in Phase Two and Phase Three to indicate that the specific operation for the identified credential is over. For example, when multiple credentials exist on the Peer and the Server needs to manage and validate one of them. In its last message, when the provisioning protocol is done, the server sets the 'D' (Done) bit to indicate that it is done. The Peer, in its reply, sets the bit to indicate the end of provisioning for this credentials is also over. After that, the Server can continue Phase Two, transition to Phase Three, or terminate the EAP session.

The Phase field is a 4-bits value and identifies the EAP-CREDS phase for the current message. The version of EAP-CREDS described in this document supports three values for this field:

0x01 - Phase One

0x02 - Phase Two

0x03 - Phase Three

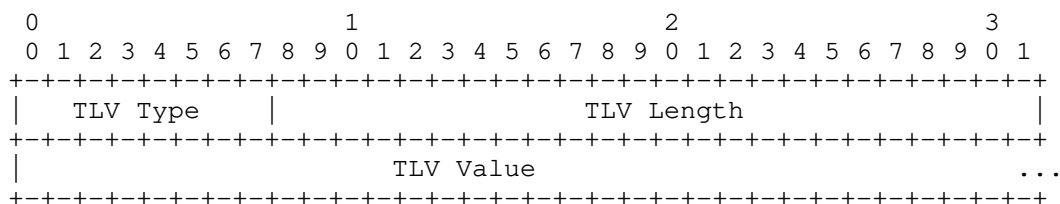
A detailed explanation of the 'Phase' and 'Flags' fields of the message headers is provided in Section 3.2.

The Data field is the message payload. The full description of this field is provided in the next section.

4.2. Message Payload

The Data part of the message is organized as zero, one, or more TLV objects whose structure is defined in this section.

Each TLV object has the same basic structure that is defined as follows:



Where:

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 fields 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).

TLV Name	TLV Type	Scope/Usage
<TBD>	Action TLV	Phase Two
<TBD>	Certificate-Data TLV	Phase Two
<TBD>	Challenge-Data TLV	Phase Two, Phase Three
<TBD>	Challenge-Response TLV	Phase Two, Phase Three
<TBD>	Credentials-Data TLV	Phase Two
<TBD>	Credentials-Info TLV	Phase Two, Phase Three
<TBD>	Error TLV	All Phases
<TBD>	Network-Usage TLV	Phase One
<TBD>	Profile TLV	Phase Two
<TBD>	Protocol TLV	Phase One, Phase Two
<TBD>	Provisioning-Data TLV	Phase Two
<TBD>	Provisioning-Headers TLV	Phase Two
<TBD>	Provisioning-Params TLV	Phase Two
<TBD>	Token-Data TLV	Phase One
<TBD>	Version TLV	Phase One

Table 2: EAP-CREDS Supported TLVs Types

TLV Value (> 1 octet)

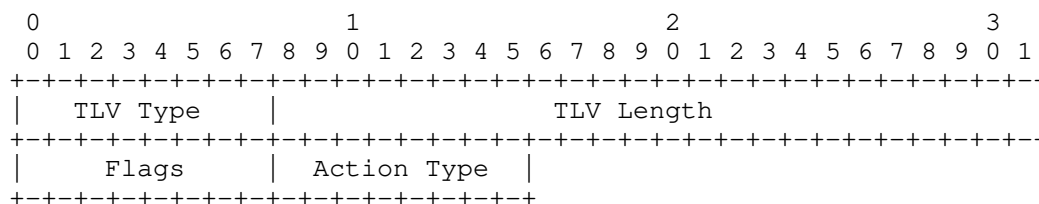
This field carries data for the identified TLV. The internal structure is determined by the TLV Type field.

The rest of this section describes the structure of the different supported TLVs and their usage in the different messages.

4.3. 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.

4.3.1. The Action TLV



TLV Type (uint8)

<TBD> - Action TLV

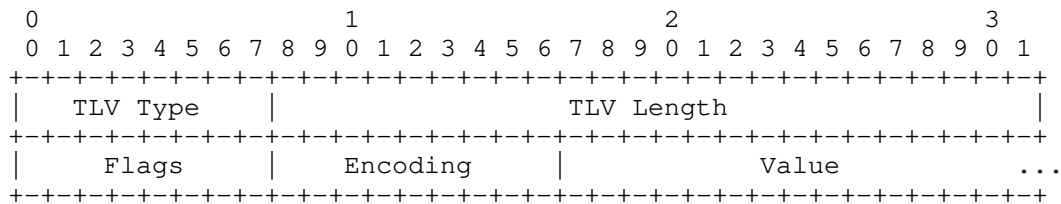
TLV Length (uint24)

Fixed value (=2)

Flags (uint8)

Reserved

4.3.2. The Certificate-Data TLV



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 attribute 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 7.5.

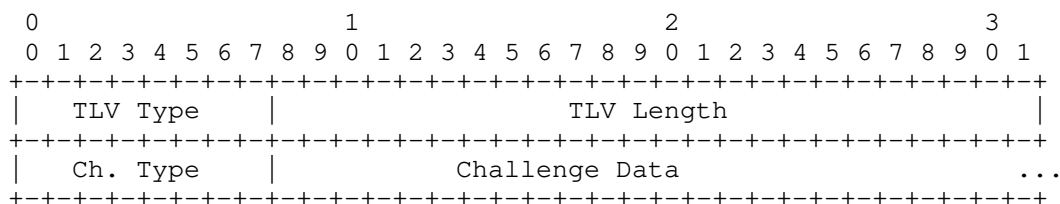
Encoding (uint8)

Provides the indication of the Encoding the certificate is in. The allowed values for this field are listed in Section 7.8.

Value (octet string)

This field carries the data for the certificate.

4.3.3. The Challenge-Data TLV



TLV Type (uint8)

<TBD> - Challenge-Data TLV

Length (uint24)

3 octets

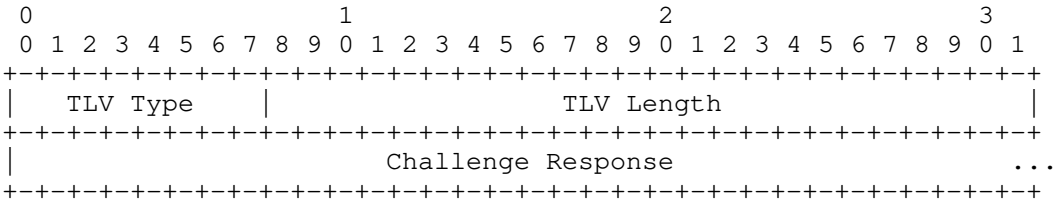
Challenge Type (uint8)

This field carries the type of Challenge. In particular, the challenge type determines how the Peer MUST calculate the ('Challenge-Response'). The initial values for this field are listed in Section 7.6. Please refer to Section 3.5 for a detailed explanation of how to calculate the response to the challenge for the challenge types defined in this document.

Challenge Data (> 1 octet)

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

4.3.4. The Challenge-Response TLV



TLV Type (uint8)

<TBD> - Challenge-Response TLV

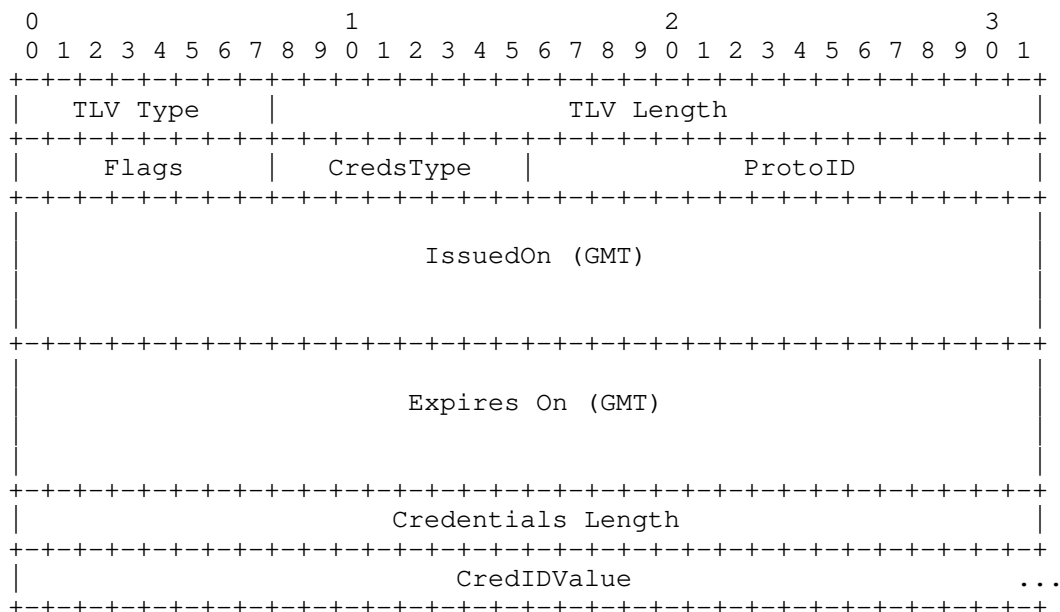
Length (uint24)

3 octets

Challenge Response (> 1 octet)

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

4.3.5. 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.

For example, when a set of credentials need to be renewed, the server checks the ('Credentials-Info') from the Peer and eventually selects the right one for renewal. 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 Section 7.3

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 is ignored. The values for this field are listed in Section 7.1.

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 }.

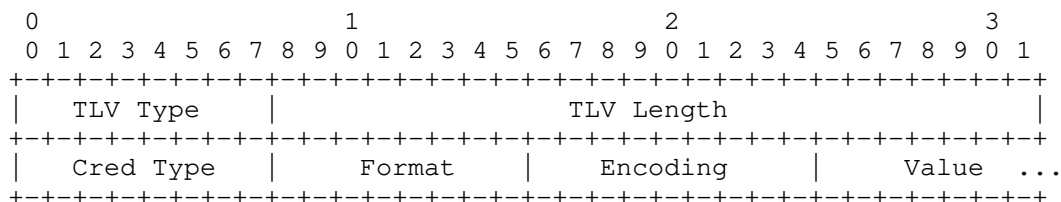
Credentials Length (uint16)

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 random handle. As long as the ID allows the peer and the server to uniquely (in its context) identify the credentials, the value of this field can be calculated in any way.

4.3.6. 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 Section 7.3.

Format (uint8)

Provides the indication of the Format the credentials are in. The allowed values for this field are listed in Section 7.5.

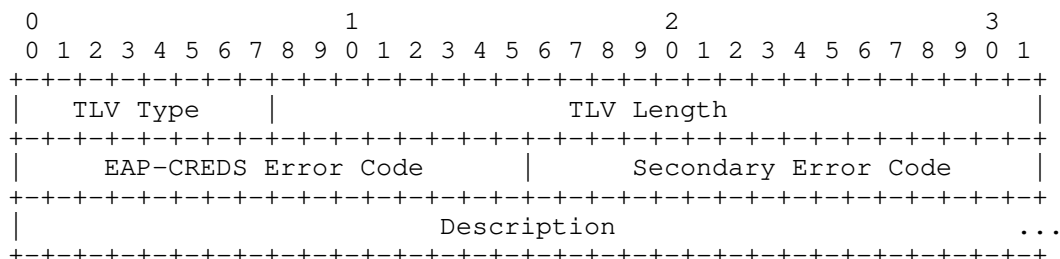
Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Section 7.8.

Value (octet string)

This field carries the data for the credentials.

4.3.7. The Error TLV



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.

The error codes supported by this specifications are listed in Section 4.3.7.

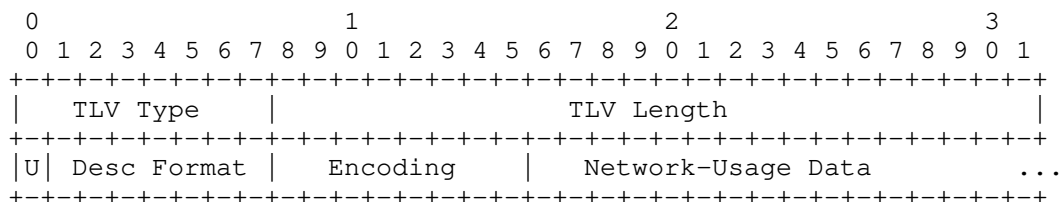
Secondary Error Code (2 octets)

This field is used to convey an error at the encapsulation layer (i.e., the provisioning protocol error). For example, this field can be used to convey a transport protocol error code (e.g., HTTP status code). 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.

4.3.8. The Network-Usage TLV



TLV Type (uint8)

<TBD> - Network-Usage TLV

Length (uint24)

Variable Length TLV (Value must be > 2)

Description Format (uint8)

The Type of data encoded in the Peer Description Data. The initial values for this field are listed in Section 7.10.

Encoding (uint8)

Provides the indication of the Encoding the network usage description data is in. The allowed values for this field are listed in Section 7.8.

The 'U' field (1 bit)

The 'URL' bit ('U') is used to indicate if the value of the Network-Usage Data field is to be interpreted as a URL or as the actual data. In particular, if the value in the 'URL' bit is '1', then the value in the Network-Usage Data field is to be interpreted as the URL where the actual data can be downloaded from. Otherwise, if the 'URL' bit is set to '0', then the value in the Network-Usage Data field is to be interpreted as the actual data (not a URL referencing it).

An example use of this bit is when the Peer wants to convey the URL of the MUD file [RFC8520]. In this case, the Peer can set the Network-Usage Data field to the Url of the MUD file related to the Peer.

Desc Format (7 bits)

This field provide the expected data format for the Network-Usage Data. For example, the value in this field could be set to 'MUD'

and have the 'U' bit set to '1' to provide the MUD-related information at credentials management time instead of at network-provisioning time (DHCP option). This possibility could help the Network controller to decide if the device shall be allowed to register its credentials or not.

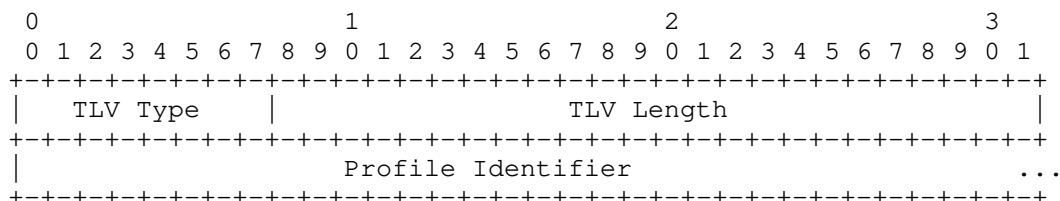
The list of initial values for this field is provided in Section 7.7.

Network-Usage Data (octet string)

This is additional information related to the device. In particular, this TLV can be used by the Peer to provide the Server with the description of the intended network usage or a URL that points to the same information.

For example, this field can be used to convey a MUD file (Manufacturer Usage Description) or the latest firmware-update manifest.

4.3.9. The Profile TLV



TLV Type (uint8)

<TBD> - Profile Identifying Data TLV

Length (uint24)

Length value should be ≥ 1

Profile Identifier (octet string)

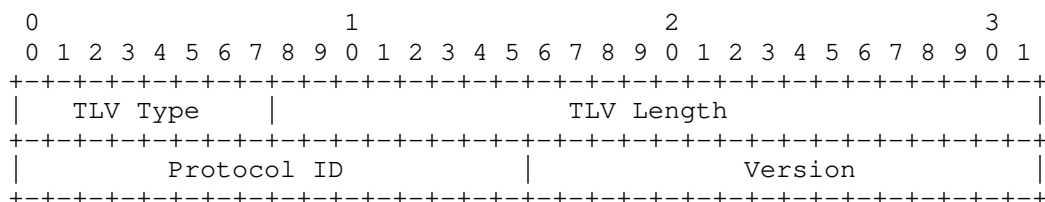
The Profile Identifier is used to provide indication to the other party about which profiles are supported when requesting credentials management.

Also in this case, the data used in this field is left to be interpreted by the end-point and it is independent from the EAP-

CREDS data types. This could be a raw byte value, a string, or a more complex structured data (e.g., an OID).

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).

4.3.10. The Protocol TLV



TLV Type (uint8)

<TBD> - Protocol TLV

TLV Length (uint24)

Fixed TLV Length value of 4.

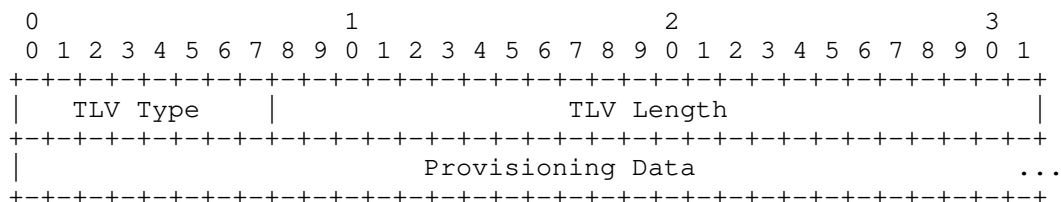
Protocol ID (uint16)

The Protocol ID value carries the id of a supported provisioning protocol. The initial list of values for the provisioning protocol identifiers can be found in Section 7.1.

Version (uint16)

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'.

4.3.11. The Provisioning-Data TLV



TLV Type (uint8)

<TBD> - Provisioning-Data TLV

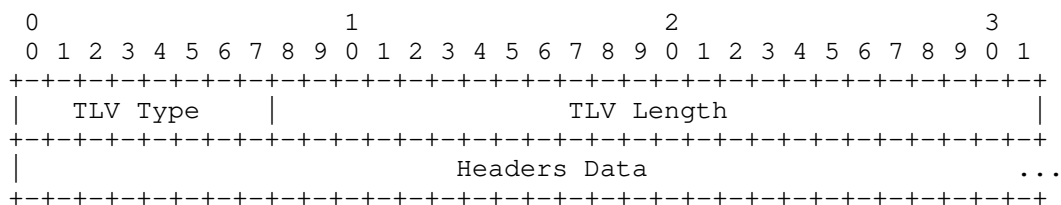
Length (uint24)

3 octets

Headers Data (> 1 octet)

This field carries the provisioning protocol's messages.

4.3.12. The Provisioning-Headers TLV



TLV Type (uint8)

<TBD> - Provisioning-Headers TLV

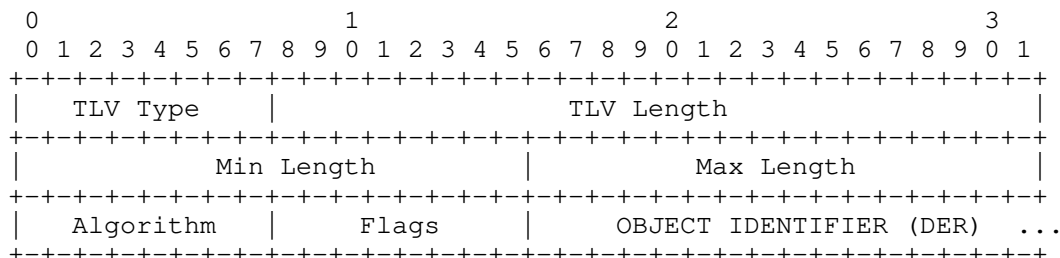
Length (uint24)

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.

4.3.13. The Provisioning-Params TLV



TLV Type (uint8)

<TBD> - Provisioning-Params TLV

Length (uint24)

Provides the length of the TLV (≥ 6 octets)

Min Length (uint16)

Provides the minimum allowed size size for 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 prime256r1 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 - Credentials are to be generated on dedicated hardware

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 Server generates the first part of the password and sends it to the Peer, while the Peer generates the second part of the password and sends it to the Server. 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.

NOTE WELL: Last but not least, since these passwords/secrets are meant to be used in a automated fashion, there is no restriction around the character set to use or their interpretation. Therefore, it is good practice to generate

random pass-phrases that use the full 8-bit character set (on client and server) to maximize the secret's search space.

Algorithm (uint8)

Provides the indication of the algorithm used for the generation of the credentials. The allowed values for this field are listed in Section 7.4.

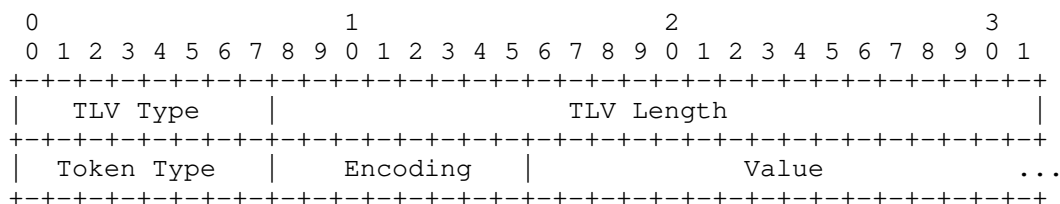
Object Identifier (binary; > 1 octet)

Provides the indication of additional parameters that are needed 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 generation algorithm to be used. We provide some useful values that can be used as reference:

OID Name	Dotted Representation	Binary Encoding
secp256r1 curve	1.2.840.10045.3.1.7	06 08 2A 86 48 CE 3D 03 01 07
secp384r1 curve	1.2.840.10045.3.1.34	06 08 2A 86 48 CE 3D 03 01 22
secp521r1 curve	1.2.840.10045.3.1.35	06 08 2A 86 48 CE 3D 03 01 23
X25519 curve	1.3.101.110	06 03 2B 65 6E
X25519 curve	1.3.101.110	06 03 2B 65 6E
X448 curve	1.3.101.111	06 03 2B 65 6F
Ed25519 curve	1.3.101.112	06 03 2B 65 70
Ed448 curve	1.3.101.113	06 03 2B 65 71

Table 3: Object Identifiers Examples

4.3.14. The Token-Data TLV



TLV Type (uint8)

<TBD> - Token-Data TLV

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 Section 7.2.

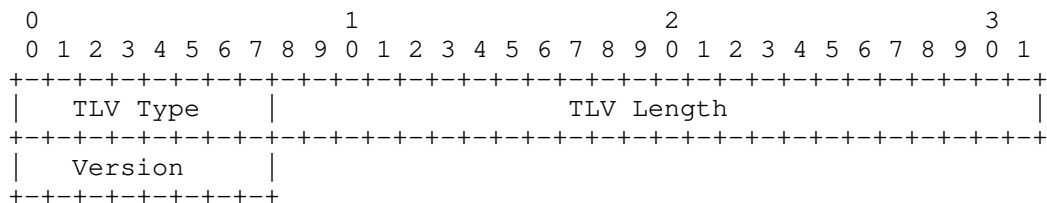
Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Section 7.8.

Value (octet string)

This field carries the data for the credentials.

4.3.15. The Version TLV



TLV Type (uint8)

<TBD> - Version TLV

TLV Length (uint24)

Provides the length of the TLV. The field has a fixed value of 1.

Version (uint8)

The Version field 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' (any version).

5. 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 Type	Name	Description
0	EAP-CREDS-Init	Initialization Phase
1	EAP-CREDS-Provisioning	Carries Provisioning Protocol Messages
2	EAP-CREDS-Validate	Validates newly installed credentials

Table 4: EAP-CREDS Message Types

5.1. 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 3.3. This message supports the following TLVs: Version, Protocol, Credentials-Info, and Error.

5.1.1. 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 first message from the server is the one that starts Phase One, therefore the Server MUST set the headers' 'S' bit to '1' (Start) and the headers' 'Phase' value to '1' (Phase One).

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 MUST include, in its Init message, a ('Challenge-Data') field that can be used by the client to provide challenge data for proof-of-possession of secrets.

5.1.2. 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 SHOULD include one ('Version') TLV in its first 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 one or more 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) via the ('Token-Data') and ('Challenge-Response') TLV.

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 and selected depending on the location and other factors (e.g., X.509 certificate and username/password combination).

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. In this case, the Peer SHOULD include authorization information via the ('Token-Data') TLV as described in Section 5.1.2.1. Additionally, the Peer can add the ('Profile') TLV to indicate a preferred profile for the credentials.

5.1.2.1. Bootstrapping Peer's Trustworthiness

When the Peer does not have any valid credentials for the Network that it is authenticating to, it does not provide any ('Credentials-Info') TLV. This indicates to the Server that new credentials MUST be registered before the Peer is allowed on the network.

The Registration process might rely on information exchanged during the Provisioning Process in Phase Two. However, if an authorization mechanism is not available from the supported provisioning protocol and no credentials are available on the Peer, EAP-CREDS provides a simple mechanism for the Peer to leverage an out-of-band token/passphrase/ott that may be already available on the Peer (e.g., a device certificate or a 'spendable' credentials token like a

kerberos ticket or a crypto-currency transaction) 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 additional authorization data) 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.

5.1.3. The EAP-CREDS-Provisioning Message

The EAP-CREDS-Provisioning message type is used in Phase Two only of EAP-CREDS. The message flow is depicted in Section 3.4. This message type supports the following TLVs: Protocol, Profile, Credentials-Info, Provisioning-Headers, Provisioning-Data, Token-Data, and Error.

After the exchange of phase one messages, the Server MAY start phase two by issuing an ('EAP-CREDS-Provisioning') 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'), and metadata to the client in a EAP-Request/EAP-CREDS(Type=Provisioning) message. The header's 'S' bit MUST be set to '1' (Start) and the 'Phase' value set to '2' (Phase Two begins).

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-Provisioning') 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.

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

5.1.4. The EAP-CREDS-Validate Message

The EAP-CREDS-Validate message type is used in Phase Three only of EAP-CREDS. The message flow is depicted in Section 3.5. This message type supports the following TLVs: Protocol, Credentials-Info, Provisioning-Headers, Provisioning-Data, Token-Data, and Error.

After Phase One (and/or Phase Two) ends, the Server MAY start phase three by issuing an ('EAP-CREDS-Validate') message for the Peer where it encodes all the required details for starting the validation process. In particular, the server sends the ('Credentials-Info'), a ('Challenge'), and the ('Phase-Control') TLVs in a EAP-Request/EAP-CREDS(Type=Validate) message. The ('Phase-Control') TLV should carry the '1' value for the 'S' bit (Start) and the number '3' for its value (Phase Three begins).

The Peer generates the answer to the Challenge and sends back a EAP-Response/EAP-CREDS(Type=Validate) message with the ('Challenge-Response') and an optional ('Challenge') field (only for server-side validation of the symmetric credentials). If the Peer requested server-side validation of the credentials, the Server MUST include (if a symmetric secret) the response to the Peer-issued ('Challenge') TLV by computing the response and adding it to the ('Challenge-Response') TLV in its reply.

Finally, in the last message, the Server (if Phase Three is to be ended) SHALL include the ('Phase-Control') TLV with the 'S' bit set to '0' (end of phase) and the value set to '3' (Phase Three ended).

At this point, EAP-CREDS has terminated all possible operations and can be terminated. The Server can now terminate the EAP session successfully. In case the Peer was not authenticated during the tunnel establishment (i.e., no credentials were already available on the Peer), the Server should terminate the EAP session with a Failure (thus requiring the device to re-attach and authenticate to the network - phase two should have provided the Peer with the credentials to use for authenticating to the Network).

6. Error Handling in EAP-CREDS

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

7. IANA Considerations

This document uses a new EAP type, EAP-CREDS, whose value (TBD) MUST be allocated by IANA from the EAP TYPEs sub-registry 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.

7.1. Provisioning Protocols

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

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].

7.2. Token Types

Token Type	Description
0	Unspecified
1	JWT
2	Kerberos
3	OAuth
4	Certificate
200..254	Vendor Specific

Table 6: Token Types

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

7.3. Credentials Types

Credentials Type	Description
0	X.509 Certificate
1	Public Key
2	Symmetric Key
3	Username and Password
4	AKA Subscriber Key
5	Bearer Token
6	One-Time Token
7	API Key

Table 7: Credentials Types

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

7.4. Credentials Algorithms

ID	Algorithm
0	None
1	RSA
2	ECDSA
3	XMMS
4	AKA Subscriber Key
5	OAuth
6	Kerberos4
7	Kerberos5
200-254	Reserved

Table 8: Credentials Algorithms

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

7.5. Credentials Datatypes

ID	Data Type
0	None (Binary)
1	PKCS#8
2	PKCS#10
3	PKCS#12
4	PublicKeyInfo
200-254	Reserved

Table 9: Credentials Datatypes

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

7.6. Challenge Types

ID	Data Type
0	Not Specified
1	EAP-CREDS-ASYMMETRIC
2	EAP-CREDS-SYMMETRIC

Table 10: Challenge Type

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

7.7. Network Usage Datatypes

ID	Data Type
0	Vendor-Specific
1	Manufacturer Usage Description [RFC8520]
2	Network Access Granting System
3	Firmware Manifest
4..127	Reserved for Future Use

Table 11: Network Usage Datatypes

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

7.8. Credentials Encoding

ID	Encoding
0	None (Raw)
1	DER
2	PEM
3	Base64
4	JSON
5	XML
6	ASCII
7	UTF-8
200-254	Reserved

Table 12: Credentials Encoding

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

7.9. Action Types

ID	Data Type	Description
0	Registration	Registers New Credentials
1	Renewal	Renew an Existing Credential
2	Remove	Removes an Existing Credential
200-254	n/a	Reserved

Table 13: Action Types

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

7.10. Usage Metadata Types

Type	Description
0	Binary (Unspecified)
1	MUD File
2	TEEP Manifest

Table 14: Usage Metadata Types

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

8. 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.

9. Acknowledgments

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

10. Normative References

- [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>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <<https://www.rfc-editor.org/info/rfc5272>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, DOI 10.17487/RFC6402, November 2011, <<https://www.rfc-editor.org/info/rfc6402>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.

- [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>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

Author's Address

Massimiliano Pala
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: m.pala@openca.org
URI: <http://www.linkedin.com/in/mpala>