

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 26, 2019

J. Arkko
K. Norrman
V. Torvinen
Ericsson
October 23, 2018

Perfect-Forward Secrecy for the Extensible Authentication Protocol
Method for Authentication and Key Agreement (EAP-AKA' PFS)
draft-arkko-eap-aka-pfs-03

Abstract

Many different attacks have been reported as part of revelations associated with pervasive surveillance. Some of the reported attacks involved compromising smart cards, such as attacking SIM card manufacturers and operators in an effort to compromise shared secrets stored on these cards. Since the publication of those reports, manufacturing and provisioning processes have gained much scrutiny and have improved. However, the danger of resourceful attackers for these systems is still a concern.

This specification is an optional extension to the EAP-AKA' authentication method which was defined in RFC 5448 (to be superseded by draft-ietf-emu-rfc5448bis). The extension, when negotiated, provides Perfect Forward Secrecy for the session key generated as a part of the authentication run in EAP-AKA'. This prevents an attacker who has gained access to the long-term pre-shared secret in a SIM card from being able to decrypt all past communications. In addition, if the attacker stays merely a passive eavesdropper, the extension prevents attacks against future sessions. This forces attackers to use active attacks instead.

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 April 26, 2019.

Copyright Notice

Copyright (c) 2018 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. Protocol Design and Deployment Objectives	4
3. Background	5
3.1. AKA	5
3.2. EAP-AKA' Protocol	6
3.3. Attacks Against Long-Term Shared Secrets in Smart Cards	8
4. Requirements Language	8
5. Protocol Overview	8
6. Extensions to EAP-AKA'	11
6.1. AT_PUB_ECDHE	11
6.2. AT_KDF_PFS	11
6.3. New Key Derivation Function	14
6.4. ECDHE Groups	15
6.5. Message Processing	15
6.5.1. EAP-Request/AKA'-Identity	15
6.5.2. EAP-Response/AKA'-Identity	16
6.5.3. EAP-Request/AKA'-Challenge	16
6.5.4. EAP-Response/AKA'-Challenge	16
6.5.5. EAP-Request/AKA'-Reauthentication	17
6.5.6. EAP-Response/AKA'-Reauthentication	17
6.5.7. EAP-Response/AKA'-Synchronization-Failure	17
6.5.8. EAP-Response/AKA'-Authentication-Reject	17
6.5.9. EAP-Response/AKA'-Client-Error	18
6.5.10. EAP-Request/AKA'-Notification	18
6.5.11. EAP-Response/AKA'-Notification	18
7. Security Considerations	18
8. IANA Considerations	22
9. References	22
9.1. Normative References	22

9.2. Informative References	23
Appendix A. Change Log	24
Appendix B. Acknowledgments	25
Authors' Addresses	25

1. Introduction

Many different attacks have been reported as part of revelations associated with pervasive surveillance. Some of the reported attacks involved compromising smart cards, such as attacking SIM card manufacturers and operators in an effort to compromise shared secrets stored on these cards. Such attacks are conceivable, for instance, during the manufacturing process of cards, or during the transfer of cards and associated information to the operator. Since the publication of reports about such attacks, manufacturing and provisioning processes have gained much scrutiny and have improved.

However, the danger of resourceful attackers attempting to gain information about SIM cards is still a concern. They are a high-value target and concern a large number of people. Note that the attacks are largely independent of the used authentication technology; the issue is not vulnerabilities in algorithms or protocols, but rather the possibility of someone gaining unlawful access to key material. While the better protection of manufacturing and other processes is essential in protecting against this, there is one question that we as protocol designers can ask. Is there something that we can do to limit the consequences of attacks, should they occur?

The authors want to provide a public specification of an extension that helps defend against one aspect of pervasive surveillance. This is important, given the large number of users such practices may affect. It is also a stated goal of the IETF to ensure that we understand the surveillance concerns related to IETF protocols and take appropriate countermeasures [RFC7258]. This document does that for EAP-AKA'.

This specification is an optional extension to the EAP-AKA' authentication method [RFC5448] (to be superseded by [I-D.ietf-emu-rfc5448bis]). The extension, when negotiated, provides Perfect Forward Secrecy for the session key generated as a part of the authentication run in EAP-AKA'. This prevents an attacker who has gained access to the long-term pre-shared secret in a SIM card from being able to decrypt all past communications. In addition, if the attacker stays merely a passive eavesdropper, the extension prevents attacks against future sessions. This forces attackers to use active attacks instead. As with other protocols, an active attacker with access to the long-term key material will of course be

able to attack all future communications, but risks detection, particularly if done at scale.

Attacks against AKA authentication via compromising the long-term secrets in the SIM cards have been an active discussion topic in many contexts. Perfect forward secrecy is on the list of features for the next release of 3GPP (5G Phase 2), and this document provides a basis for providing this feature in a particular fashion.

It should also be noted that 5G network architecture includes the use of the EAP framework for authentication. While any methods can be run, the default authentication method within that context will be EAP-AKA'. As a result, improvements in EAP-AKA' security have a potential to improve security for large number of users.

2. Protocol Design and Deployment Objectives

This extension specified here re-uses large portions of the current structure of 3GPP interfaces and functions, with the rationale that this will make the construction more easily adopted. In particular, the construction maintains the interface between the Universal Subscriber Identification Module (USIM) and the mobile terminal intact. As a consequence, there is no need to roll out new credentials to existing subscribers. The work is based on an earlier paper [TrustCom2015], and uses much of the same material, but applied to EAP rather than the underlying AKA method.

It has been a goal to implement this change as an extension of the widely supported EAP-AKA' method, rather than a completely new authentication method. The extension is implemented as a set of new, optional attributes, that are provided alongside the base attributes in EAP-AKA'. Old implementations can ignore these attributes, but their presence will nevertheless be verified as part of base EAP-AKA' integrity verification process, helping protect against bidding down attacks. This extension does not increase the number of rounds necessary to complete the protocol.

The use of this extension is at the discretion of the authenticating parties. It should be noted that PFS and defenses against passive attacks are by no means a panacea, but they can provide a partial defense that increases the cost and risk associated with pervasive surveillance.

While adding perfect forward secrecy to the existing mobile network infrastructure can be done in multiple different ways, the authors believe that the approach chosen here is relatively easily deployable. In particular:

- o As noted above, no new credentials are needed; there is no change to SIM cards.
- o PFS property can be incorporated into any current or future system that supports EAP, without changing any network functions beyond the EAP endpoints.
- o Key generation happens at the endpoints, enabling highest grade key material to be used both by the endpoints and the intermediate systems (such as access points that are given access to specific keys).
- o While EAP-AKA' is just one EAP method, for practical purposes perfect forward secrecy being available for both EAP-TLS [RFC5216] [I-D.mattsson-eap-tls13] and EAP-AKA' ensures that for many practical systems perfect forward secrecy can be enabled for either all or significant fraction of users.

3. Background

3.1. AKA

AKA is based on challenge-response mechanisms and symmetric cryptography. AKA typically runs in a UMTS Subscriber Identity Module (USIM) or a CDMA2000 (Removable) User Identity Module ((R)UIM). In contrast with its earlier GSM counterparts, 3rd generation AKA provides long key lengths and mutual authentication.

AKA works in the following manner:

- o The identity module and the home environment have agreed on a secret key beforehand.
- o The actual authentication process starts by having the home environment produce an authentication vector, based on the secret key and a sequence number. The authentication vector contains a random part RAND, an authenticator part AUTN used for authenticating the network to the identity module, an expected result part XRES, a 128-bit session key for integrity check IK, and a 128-bit session key for encryption CK.
- o The authentication vector is passed to the serving network, which uses it to authenticate the device.
- o The RAND and the AUTN are delivered to the identity module.
- o The identity module verifies the AUTN, again based on the secret key and the sequence number. If this process is successful (the

AUTN is valid and the sequence number used to generate AUTN is within the correct range), the identity module produces an authentication result RES and sends it to the serving network.

- o The serving network verifies the correct result from the identity module. If the result is correct, IK and CK can be used to protect further communications between the identity module and the home environment.

3.2. EAP-AKA' Protocol

When AKA (and AKA') are embedded into EAP, the authentication on the network side is moved to the home environment; the serving network performs the role of a pass-through authenticator. Figure 1 describes the basic flow in the EAP-AKA' authentication process. The definition of the full protocol behaviour, along with the definition of attributes AT_RAND, AT_AUTN, AT_MAC, and AT_RES can be found in [I-D.ietf-emu-rfc5448bis] and [RFC4187].

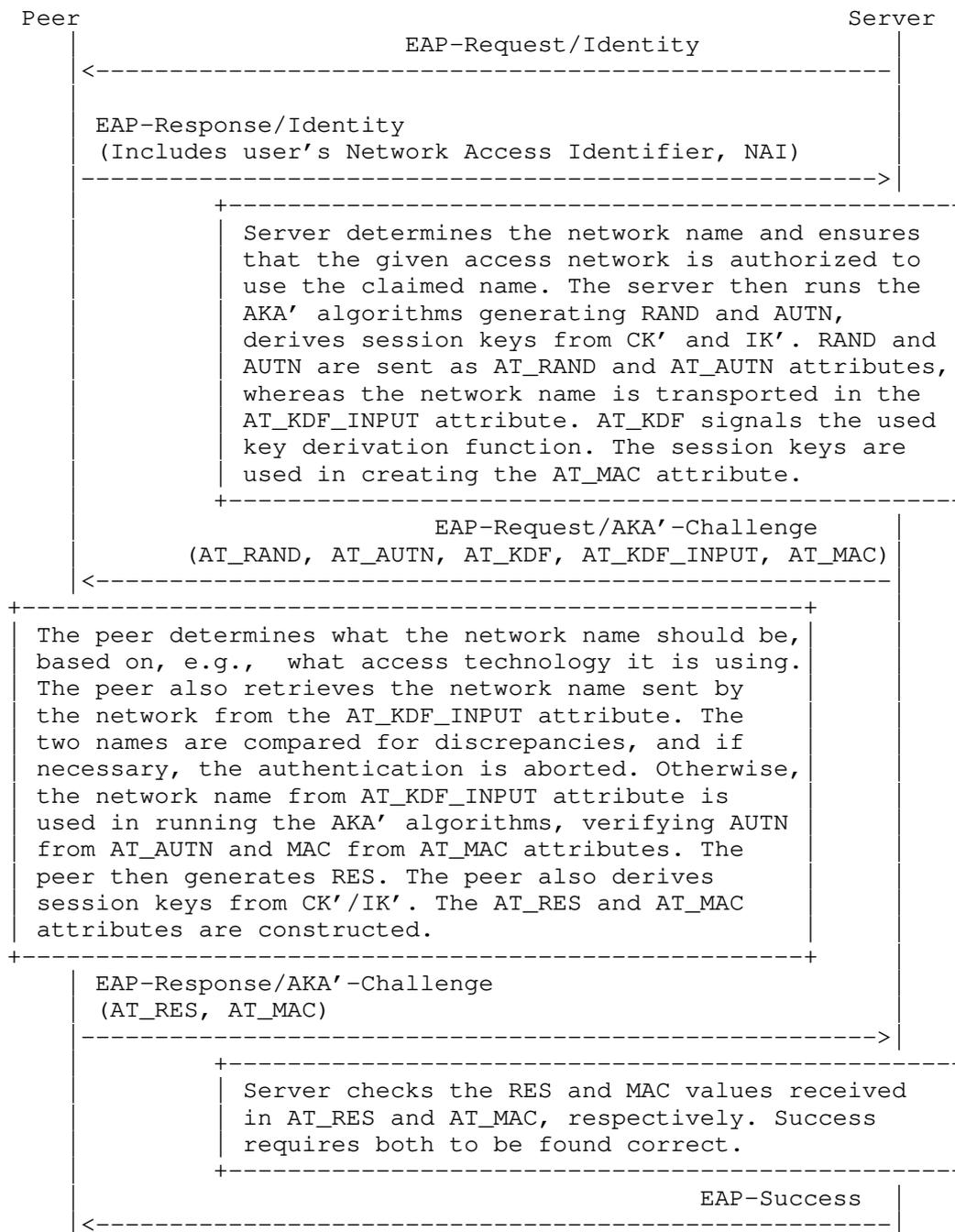


Figure 1: EAP-AKA' Authentication Process

3.3. Attacks Against Long-Term Shared Secrets in Smart Cards

Current 3GPP systems use (U)SIM pre-shared key based protocols and Authentication and Key Agreement (AKA) to authenticate subscribers. The general security properties and potential vulnerabilities of AKA and EAP-AKA' are discussed in [I-D.ietf-emu-rfc5448bis].

An important vulnerability in that discussion relates to the recent reports of compromised long term pre-shared keys used in AKA [Heist2015]. These attacks are not specific to AKA or EAP-AKA', as all security systems fail at least to some extent if key material is stolen. However, the reports indicate a need to look into solutions that can operate at least to an extent under these types of attacks. It is noted in [Heist2015] that some security can be retained even in the face of the attacks by providing Perfect Forward Security (PFS) [DOW1992] for the session key. If AKA would have provided PFS, compromising the pre-shared key would not be sufficient to perform passive attacks; the attacker is, in addition, forced to be a Man-In-The-Middle (MITM) during the AKA run.

4. 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.

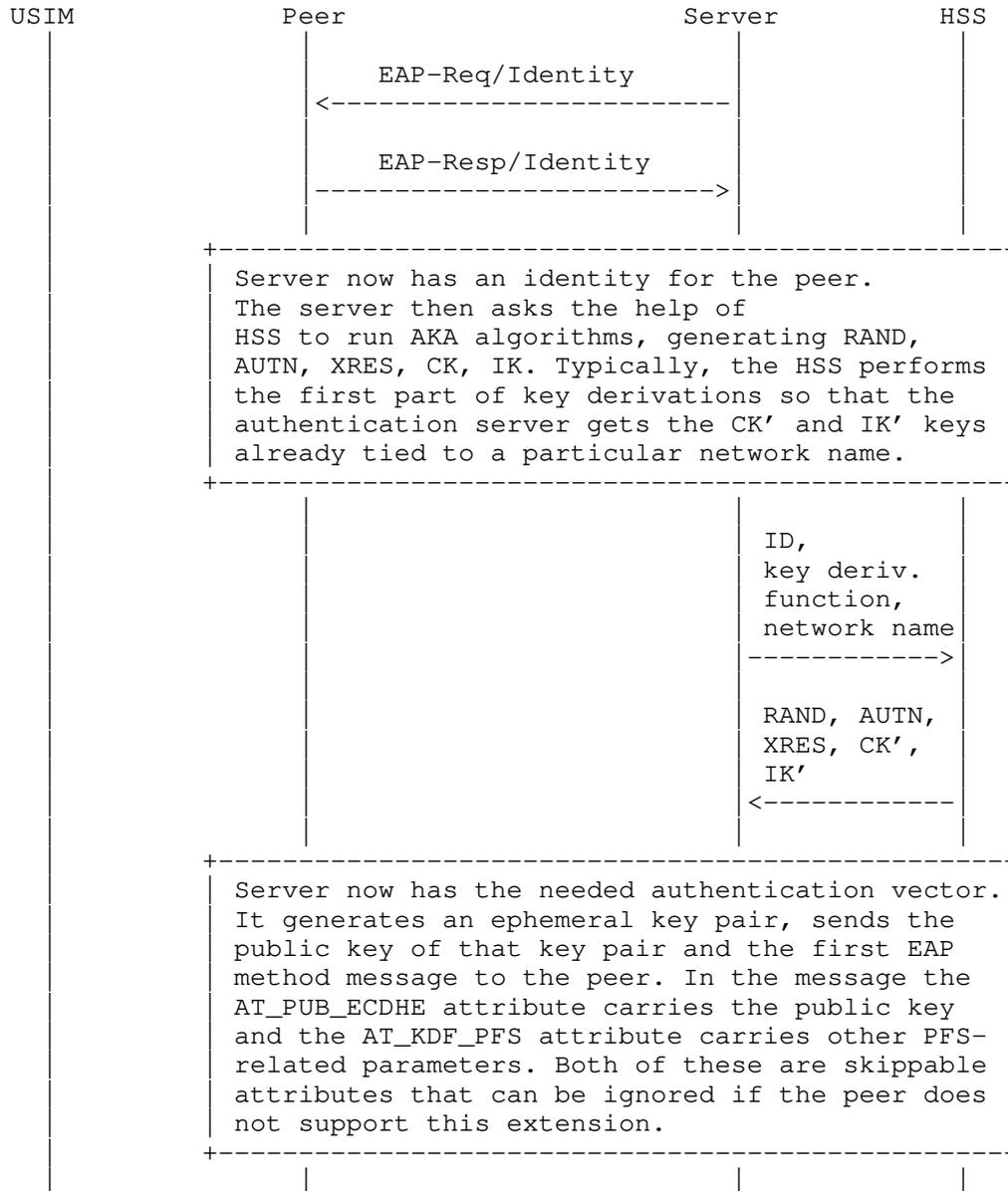
5. Protocol Overview

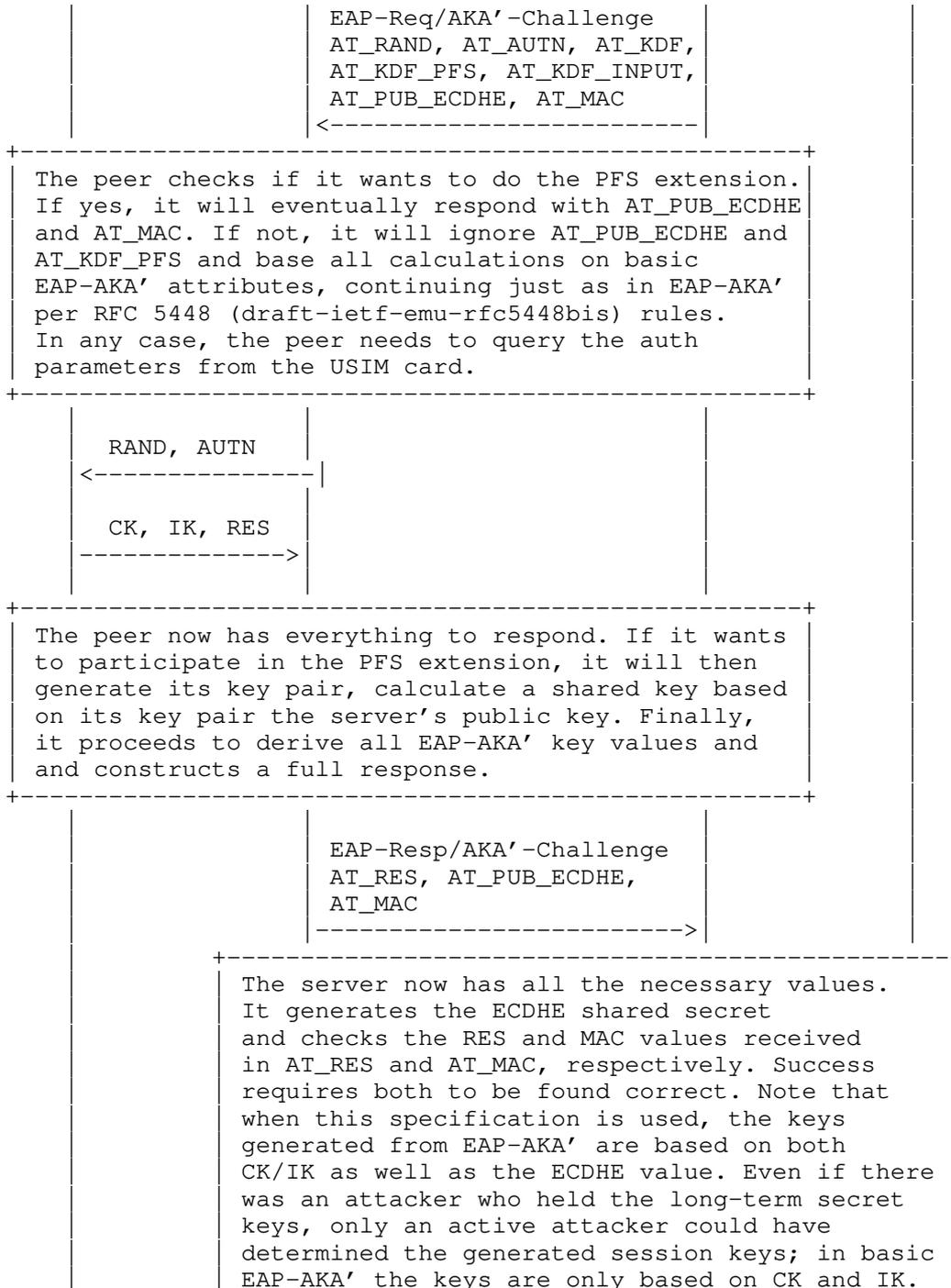
Introducing PFS for EAP-AKA' can be achieved by using an Elliptic Curve Diffie-Hellman (ECDH) exchange [RFC7748]. In EAP-AKA' PFS this exchange is run in an ephemeral manner, i.e., using temporary keys as specified in [RFC8031] Section 2. This method is referred to as ECDHE, where the last 'E' stands for Ephemeral.

The enhancements in the EAP-AKA' PFS protocol are compatible with the signaling flow and other basic structures of both AKA and EAP-AKA'. The intent is to implement the enhancement as optional attributes that legacy implementations can ignore.

The purpose of the protocol is to achieve mutual authentication between the EAP server and peer, and to establish keying material for secure communication between the two. This document specifies the calculation of key material, providing new properties that are not present in key material provided by EAP-AKA' in its original form.

Figure 2 below describes the overall process. Since our goal has been to not require new infrastructure or credentials, the flow diagrams also show the conceptual interaction with the USIM card and the 3GPP authentication server (HSS). The details of those interactions are outside the scope of this document, however, and the reader is referred to the 3GPP specifications .





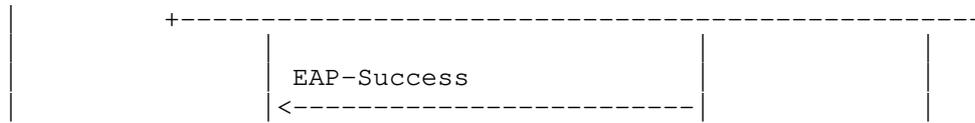


Figure 2: EAP-AKA' PFS Authentication Process

6. Extensions to EAP-AKA'

6.1. AT_PUB_ECDHE

The AT_PUB_ECDHE carries an ECDHE value.

The format of the AT_PUB_ECDHE attribute is shown below.

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				
AT_PUB_ECDHE									Length									Value ...																	

The fields are as follows:

AT_PUB_ECDHE

This is set to TBA1 BY IANA.

Length

The length of the attribute, set as other attributes in EAP-AKA [RFC4187].

Value

This value is the sender's ECDHE public value. For Curve25519, the length of this value is 32 bytes, encoded in binary as specified [RFC7748] Section 6.1.

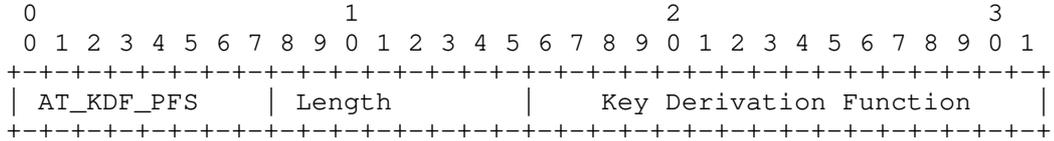
To retain the security of the keys, the sender SHALL generate a fresh value for each run of the protocol.

6.2. AT_KDF_PFS

The AT_KDF_PFS indicates the used or desired key generation function, if the Perfect Forward Secrecy extension is taken into use. It will also at the same time indicate the used or desired ECDHE group. A new attribute is needed to carry this information, as AT_KDF carries

the legacy KDF value for those EAP peers that cannot or do not want to use this extension.

The format of the AT_KDF_PFS attribute is shown below.



The fields are as follows:

AT_KDF_PFS

This is set to TBA2 BY IANA.

Length

The length of the attribute, MUST be set to 1.

Key Derivation Function

An enumerated value representing the key derivation function that the server (or peer) wishes to use. See Section 6.3 for the functions specified in this document. Note: This field has a different name space than the similar field in the AT_KDF attribute Key Derivation Function defined in [I-D.ietf-emu-rfc5448bis].

Servers MUST send one or more AT_KDF_PFS 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. The most preferred function is the only one that the server includes a public key value for, however. So for a set of AT_KDF_PFS attributes, there is always only one AT_PUB_ECDHE attribute.

Upon receiving a set of these attributes:

- o If the peer supports and is willing to use the key derivation function indicated by the first AT_KDF_PFS attribute, and is willing and able to use the extension defined in this specification, the function is taken into use without any further negotiation.

- o If the peer does not support this function or is unwilling to use it, it responds to the server with an indication that a different function is needed. Similarly with the negotiation process defined in [I-D.ietf-emu-rfc5448bis] for AT_KDF, the peer sends EAP-Response/AKA'-Challenge message that contains only one attribute, AT_KDF_PFS with the value set to the desired alternative function from among the ones suggested by the server earlier. If there is no suitable alternative, the peer has a choice of either falling back to EAP-AKA' or behaving as if AUTN had been incorrect and failing authentication (see Figure 3 of [RFC4187]). The peer MUST fail the authentication if there are any duplicate values within the list of AT_KDF_PFS attributes (except where the duplication is due to a request to change the key derivation function; see below for further information).
- o If the peer does not recognize the extension defined in this specification or is unwilling to use it, it ignores the AT_KDF_PFS attribute.

Upon receiving an EAP-Response/AKA'-Challenge with AT_KDF_PFS from the peer, the server checks that the suggested AT_KDF_PFS value was one of the alternatives in its offer. The first AT_KDF_PFS value in the message from the server is not a valid alternative. If the peer has replied with the first AT_KDF_PFS 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 in [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_PFS 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_PFS attributes. If yes, it continues. 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_PFS attributes without having requested negotiation, the peer MUST behave as if AT_MAC had been incorrect and fail the authentication.

6.3. New Key Derivation Function

A new Key Derivation Function type is defined for "EAP-AKA' with ECDHE and Curve25519", represented by value 1. It represents a particular choice of key derivation function and at the same time selects an ECDHE group to be used.

The Key Derivation Function type value is only used in the AT_KDF_PFS attribute, and should not be confused with the different range of key derivation functions that can be represented in the AT_KDF attribute as defined in [I-D.ietf-emu-rfc5448bis].

Key derivation in this extension produces exactly the same keys for internal use within one authentication run as [I-D.ietf-emu-rfc5448bis] EAP-AKA' does. For instance, K_aut that is used in AT_MAC is still exactly as it was in EAP-AKA'. The only change to key derivation is in re-authentication keys and keys exported out of the EAP method, MSK and EMSK. As a result, EAP-AKA' attributes such as AT_MAC continue to be usable even when this extension is in use.

When the Key Derivation Function field in the AT_KDF_PFS attribute is set to 1 and the Key Derivation Function field in the AT_KDF attribute is also set to 1, the Master Key (MK) is derived and as follows below.

```

MK           = PRF'(IK' | CK', "EAP-AKA'" | Identity)
MK_ECDHE    = PRF'(IK' | CK' | SHARED_SECRET, "EAP-AKA' PFS" | Identity)
K_encr      = MK[0..127]
K_aut       = MK[128..383]
K_re        = MK_ECDHE[0..255]
MSK         = MK_ECDHE[256..767]
EMSK        = MK_ECDHE[768..1279]

```

Where SHARED_SECRET is the shared secret computed via ECDHE, as specified in Section 2 of [RFC8031] and Section 6.1 of [RFC7748].

Both the peer and the server MAY check for zero-value shared secret as specified in Section 6.1 of [RFC7748]. If such checking is performed and the SHARED_SECRET has a zero value, both parties MUST behave as if the current EAP-AKA' authentication process starts again from the beginning.

Note: The way that shared secret is tested for zero can, if performed inappropriately, provide an ability for attackers to listen to CPU power usage side channels. Refer to [RFC7748] for a description of how to perform this check in a way that it does not become a problem.

The rest of computation proceeds as defined in Section 3.3 of [I-D.ietf-emu-rfc5448bis].

For readability, an explanation of the notation used above is copied here: [n..m] denotes the substring from bit n to m. PRF' is a new pseudo-random function specified in [I-D.ietf-emu-rfc5448bis]. K_encr is the encryption key, 128 bits, K_aut is the authentication key, 256 bits, K_re is the re-authentication key, 256 bits, MSK is the Master Session Key, 512 bits, and EMSK is the Extended Master Session Key, 512 bits. MSK and EMSK are outputs from a successful EAP method run [RFC3748].

CK and IK are produced by the AKA algorithm. IK' and CK' are derived as specified in [I-D.ietf-emu-rfc5448bis] from IK and CK.

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

Identity is the peer identity as specified in Section 7 of [RFC4187].

6.4. ECDHE Groups

The selection of suitable groups for the elliptic curve computation is necessary. The choice of a group is made at the same time as deciding to use of particular key derivation function in AT_KDF_PFS. For "EAP-AKA' with ECDHE and Curve25519" the group is the Curve25519 group specified in [RFC8031].

6.5. Message Processing

This section specifies the changes related to message processing when this extension is used in EAP-AKA'. It specifies when a message may be transmitted or accepted, which attributes are allowed in a message, which attributes are required in a message, and other message-specific details, where those details are different for this extension than the base EAP-AKA' or EAP-AKA protocol. Unless otherwise specified here, the rules from [I-D.ietf-emu-rfc5448bis] or [RFC4187] apply.

6.5.1. EAP-Request/AKA'-Identity

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.2. EAP-Response/AKA'-Identity

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.3. EAP-Request/AKA'-Challenge

The server sends the EAP-Request/AKA'-Challenge on full authentication as specified by [RFC4187] and [I-D.ietf-emu-rfc5448bis]. The attributes AT_RAND, AT_AUTN, and AT_MAC MUST be included and checked on reception as specified in [RFC4187]. They are also necessary for backwards compatibility.

In EAP-Request/AKA'-Challenge, there is no message-specific data covered by the MAC for the AT_MAC attribute. The AT_KDF_PFS and AT_PUB_ECDHE attributes MUST be included. The AT_PUB_ECDHE attribute carries the server's public Diffie-Hellman key. If either AT_KDF_PFS or AT_PUB_ECDHE is missing on reception, the peer MUST treat them as if neither one was sent, and the assume that the extension defined in this specification is not in use.

The AT_RESULT_IND, AT_CHECKCODE, AT_IV, AT_ENCR_DATA, AT_PADDING, AT_NEXT_PSEUDONYM, AT_NEXT_REAUTH_ID and other attributes may be included as specified in Section 9.3 of [RFC4187].

When processing this message, the peer MUST process AT_RAND, AT_AUTN, AT_KDF_PFS, AT_PUB_ECDHE before processing other attributes. Only if these attributes are verified to be valid, the peer derives keys and verifies AT_MAC. If the peer is unable or unwilling to perform the extension specified in this document, it proceeds as defined in [I-D.ietf-emu-rfc5448bis]. Finally, the operation in case an error occurs is specified in Section 6.3.1. of [RFC4187].

6.5.4. EAP-Response/AKA'-Challenge

The peer sends EAP-Response/AKA'-Challenge in response to a valid EAP-Request/AKA'-Challenge message, as specified by [RFC4187] and [I-D.ietf-emu-rfc5448bis]. If the peer supports and is willing to perform the extension specified in this protocol, and the server had made a valid request involving the attributes specified in Section 6.5.3, the peer responds per the rules specified below. Otherwise, the peer responds as specified in [RFC4187] and [I-D.ietf-emu-rfc5448bis] and ignores the attributes related to this extension. If the peer has not received attributes related to this extension from the Server, and has a policy that requires it to always use this extension, it behaves as if AUTN had been incorrect and fails the authentication.

The AT_MAC attribute MUST be included and checked as specified in [I-D.ietf-emu-rfc5448bis]. In EAP-Response/AKA'-Challenge, there is no message-specific data covered by the MAC. The AT_PUB_ECDHE attribute MUST be included, and carries the peer's public Diffie-Hellman key.

The AT_RES attribute MUST be included and checked as specified in [RFC4187]. When processing this message, the Server MUST process AT_RES before processing other attributes. Only if these attribute is verified to be valid, the Server derives keys and verifies AT_MAC.

If the Server has proposed the use of the extension specified in this protocol, but the peer ignores and continues the basic EAP-AKA' authentication, the Server makes policy decision of whether this is allowed. If this is allowed, it continues the EAP-AKA' authentication to completion. If it is not allowed, the Server MUST behave as if authentication failed.

The AT_CHECKCODE, AT_RESULT_IND, AT_IV, AT_ENCR_DATA and other attributes may be included as specified in Section 9.4 of [RFC4187].

6.5.5. EAP-Request/AKA'-Reauthentication

No changes, but note that the re-authentication process uses the keys generated in the original EAP-AKA' authentication, which, if the extension specified in this documents is in use, employs key material from the Diffie-Hellman procedure.

6.5.6. EAP-Response/AKA'-Reauthentication

No changes, but as discussed in Section 6.5.5, re-authentication is based on the key material generated by EAP-AKA' and the extension defined in this document.

6.5.7. EAP-Response/AKA'-Synchronization-Failure

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.8. EAP-Response/AKA'-Authentication-Reject

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.9. EAP-Response/AKA'-Client-Error

No changes, except that the `AT_KDF_PFS` or `AT_PUB_ECDHE` attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.10. EAP-Request/AKA'-Notification

No changes.

6.5.11. EAP-Response/AKA'-Notification

No changes.

7. Security Considerations

This section deals only with the changes to security considerations as they differ from EAP-AKA', or as new information has been gathered since the publication of [I-D.ietf-emu-rfc5448bis].

The possibility of attacks against key storage offered in SIM or other smart cards has been a known threat. But as the discussion in Section 3.3 shows, the likelihood of practically feasible attacks has increased. Many of these attacks can be best dealt with improved processes, e.g., limiting the access to the key material within the factory or personnel, etc. But not all attacks can be entirely ruled out for well-resourced adversaries, irrespective of what the technical algorithms and protection measures are.

This extension can provide assistance in situations where there is a danger of attacks against the key material on SIM cards by adversaries that can not or who are unwilling to mount active attacks against large number of sessions. This extension is most useful when used in a context where EAP keys are used without further mixing that can provide Perfect Forward Secrecy. For instance, when used with IKEv2 [RFC7296], the session keys produced by IKEv2 have this property, so better characteristics of EAP keys is not that useful. However, typical link layer usage of EAP does not involve running Diffie-Hellman, so using EAP to authenticate access to a network is one situation where the extension defined in this document can be helpful.

This extension generates keying material using the ECDHE exchange in order to gain the PFS property. This means that once an EAP-AKA' authentication run ends, the session that it was used to protect is closed, and the corresponding keys are forgotten, even someone who has recorded all of the data from the authentication run and session and gets access to all of the AKA long-term keys cannot reconstruct

the keys used to protect the session or any previous session, without doing a brute force search of the session key space.

Even if a compromise of the long-term keys has occurred, PFS is still provided for all future sessions, as long as the attacker does not become an active attacker. Of course, as with other protocols, if the attacker has learned the keys and does become an active attacker, there is no protection that that can be provided for future sessions. Among other things, such an active attacker can impersonate any legitimate endpoint in EAP-AKA', become a MITM in EAP-AKA' or the extension defined in this document, retrieve all keys, or turn off PFS. Still, past sessions where PFS was in use remain protected.

Achieving PFS requires that when a connection is closed, each endpoint MUST forget not only the ephemeral keys used by the connection but also any information that could be used to recompute those keys.

The following security properties of EAP-AKA' are impacted through this extension:

Protected ciphersuite negotiation

EAP-AKA' has a negotiation mechanism for selecting the key derivation functions, and this mechanism has been extended by the extension specified in this document. The resulting mechanism continues to be secure against bidding down attacks.

There are two specific needs in the negotiation mechanism:

Negotiating key derivation function within the extension

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.

Negotiating the use of this extension

This extension is offered by the server through presenting the AT_KDF_PFS and AT_PUB_ECDHE attributes in the EAP-Request/AKA'-Challenge message. These attributes are protected by AT_MAC, so attempts to change or omit them by an adversary will be detected.

Except of course, if the adversary holds the long-term shared secret and is willing to engage in an active attack. Such an attack can, for instance, forge the negotiation process so that no PFS will be provided. However, as noted above, an attacker with these capabilities will in any case be able to impersonate any party in the protocol and perform MITM attacks. That is not a situation that can be improved by a technical solution. However, as discussed in the introduction, even an attacker with access to the long-term keys is required to be a MITM on each AKA run, which makes mass surveillance more laborous.

The security properties of the extension also depend on a policy choice. As discussed in Section 6.5.4, both the peer and the server make a policy decision of what to do when it was willing to perform the extension specified in this protocol, but the other side does not wish to use the extension. Allowing this has the benefit of allowing backwards compatibility to equipment that did not yet support the extension. When the extension is not supported or negotiated by the parties, no PFS can obviously be provided.

If turning off the extension specified in this protocol is not allowed by policy, the use of legacy equipment that does not support this protocol is no longer possible. This may be appropriate when, for instance, support for the extension is sufficiently widespread, or required in a particular version of a mobile network.

Key derivation

This extension provides key material that is based on the Diffie-Hellman keys, yet bound to the authentication through the (U)SIM card. This means that subsequent payload communications between the parties are protected with keys that are not solely based on information in the clear (such as the RAND) and information derivable from the long-term shared secrets on the (U)SIM card. As a result, if anyone successfully recovers shared secret information, they are unable to decrypt communications protected by the keys generated through this extension. Note that the recovery of shared secret information could occur either before or after the time that the protected communications are used. When this extension is used, communications at time t_0 can be protected if at some later time t_1 an adversary learns of long-term shared secret and has access to a recording of the encrypted communications.

Obviously, this extension is still vulnerable to attackers that are willing to perform an active attack and who at the time of the attack have access to the long-term shared secret.

This extension does not change the properties of related to re-authentication. No new Diffie-Hellman run is performed during the re-authentication allowed by EAP-AKA'. However, if this extension was in use when the original EAP-AKA' authentication was performed, the keys used for re-authentication (K_{re}) are based on the Diffie-Hellman keys, and hence continue to be equally safe against expose of the long-term secrets as the original authentication.

In addition, it is worthwhile to discuss Denial-of-Service attacks and their impact on this protocol. The calculations involved in public key cryptography require computing power, which could be used in an attack to overpower either the peer or the server. While some forms of Denial-of-Service attacks are always possible, the following factors help mitigate the concerns relating to public key cryptography and EAP-AKA' PFS.

- o In 5G context, other parts of the connection setup involve public key cryptography, so while performing additional operations in EAP-AKA' is an additional concern, it does not change the overall situation. As a result, the relevant system components need to be dimensioned appropriately, and detection and management mechanisms to reduce the effect of attacks need to be in place.
- o This specification is constructed so that a separation between the USIM and Peer on client side and the Server and HSS on network side is possible. This ensures that the most sensitive (or legacy) system components can not be the target of the attack. For instance, EAP-AKA' and public key cryptography takes place in the phone and not the low-power SIM card.
- o EAP-AKA' has been designed so that the first actual message in the authentication process comes from the Server, and that this message will not be sent unless the user has been identified as an active subscriber of the operator in question. While the initial identity can be spoofed before authentication has succeeded, this reduces the efficiency of an attack.
- o Finally, this memo specifies an order in which computations and checks must occur. When processing the EAP-Request/AKA'-Challenge message, for instance, the AKA authentication must be checked and succeed before the peer proceeds to calculating or processing the PFS related parameters (see Section 6.5.4). The same is true of EAP-Response/AKA'-Challenge (see Section 6.5.4). This ensures that

the parties need to show possession of the long-term secret in some way, and only then will the PFS calculations become active. This limits the Denial-of-Service to specific, identified subscribers. While botnets and other forms of malicious parties could take advantage of actual subscribers and their key material, at least such attacks are (a) limited in terms of subscribers they control, and (b) identifiable for the purposes of blocking the affected subscribers.

8. IANA Considerations

This extension of EAP-AKA' shares its attribute space and subtypes with EAP-SIM [RFC4186], EAP-AKA [RFC4186], and EAP-AKA' [I-D.ietf-emu-rfc5448bis].

Two new Attribute Type value (TBA1, TBA2) in the skippable range need to be assigned for AT_PUB_ECDHE (Section 6.1) and AT_KDF_PFS (Section 6.2 in the EAP-AKA and EAP-SIM Parameters registry under Attribute Types.

Also, a new registry should be created to represent Diffie-Hellman Key Derivation Function types. The "EAP-AKA' with ECDHE and Curve25519" type (1, see Section 6.3) needs to be assigned, along with one reserved value. The initial contents of this namespace are therefore as below; new values can be created through the Specification Required policy [RFC8126].

Value	Description	Reference
0	Reserved	[TBD BY IANA: THIS RFC]
1	EAP-AKA' with ECDHE and Curve25519	[TBD BY IANA: THIS RFC]
2-65535	Unassigned	

9. References

9.1. 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. Levkowitz, 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>>.
- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/info/rfc7748>>.
- [RFC8031] Nir, Y. and S. Josefsson, "Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement", RFC 8031, DOI 10.17487/RFC8031, December 2016, <<https://www.rfc-editor.org/info/rfc8031>>.
- [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>>.
- [I-D.ietf-emu-rfc5448bis]
Arkko, J., Lehtovirta, V., Torvinen, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", draft-ietf-emu-rfc5448bis-03 (work in progress), October 2018.

9.2. Informative References

- [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>>.
- [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>>.
- [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>>.

- [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.mattsson-eap-tls13] Mattsson, J. and M. Sethi, "Using EAP-TLS with TLS 1.3", draft-mattsson-eap-tls13-02 (work in progress), March 2018.
- [TrustCom2015] Arkko, J., Norrman, K., Naslund, M., and B. Sahlin, "A USIM compatible 5G AKA protocol with perfect forward secrecy", August 2015 in Proceedings of the TrustCom 2015, IEEE.
- [Heist2015] Scahill, J. and J. Begley, "The great SIM heist", February 2015, in <https://firstlook.org/theintercept/2015/02/19/great-sim-heist/> .
- [DOW1992] Diffie, W., vanOorschot, P., and M. Wiener, "Authentication and Authenticated Key Exchanges", June 1992, in Designs, Codes and Cryptography 2 (2): pp. 107-125.

Appendix A. Change Log

The -03 version of this draft changed the naming of various protocol components, values, and notation to match with the use of ECDH in ephemeral mode. The AT_KDF_PFS negotiation process was clarified in that exactly one key is ever sent in AT_KDF_ECDHE. The option of checking for zero key values IN ECDHE was added. The format of the actual key in AT_PUB_ECDHE was specified. Denial-of-service considerations for the PFS process have been updated. Bidding down attacks against this extension itself are discussed extensively. This version also addressed comments from reviewers, including the August review from Mohit Sethi, and comments made during IETF-102 discussion.

Appendix B. Acknowledgments

The authors would like to note that the technical solution in this document came out of the TrustCom paper [TrustCom2015], whose authors were J. Arkko, K. Norrman, M. Naslund, and B. Sahlin. This document uses also a lot of material from [RFC4187] by J. Arkko and H. Haverinen as well as [RFC5448] by J. Arkko, V. Lehtovirta, and P. Eronen.

The authors would also like to thank Tero Kivinen, John Mattson, Mohit Sethi, Vesa Lehtovirta, Joseph Salowey, Kathleen Moriarty, Zhang Fu, Bengt Sahlin, Ben Campbell, Prajwol Kumar Nakarmi, Goran Rune, Tim Evans, Helena Vahidi Mazinani, Anand R. Prasad, and many other people at the GSMA and 3GPP groups for interesting discussions in this problem space.

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Karl Norrman
Ericsson
Stockholm 16483
Sweden

Email: karl.norrman@ericsson.com

Vesa Torvinen
Ericsson
Jorvas 02420
Finland

Email: vesa.torvinen@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 25, 2019

J. Arkko
K. Norrman
V. Torvinen
Ericsson
January 21, 2019

Perfect-Forward Secrecy for the Extensible Authentication Protocol
Method for Authentication and Key Agreement (EAP-AKA' PFS)
draft-arkko-eap-aka-pfs-04

Abstract

Many different attacks have been reported as part of revelations associated with pervasive surveillance. Some of the reported attacks involved compromising smart cards, such as attacking SIM card manufacturers and operators in an effort to compromise shared secrets stored on these cards. Since the publication of those reports, manufacturing and provisioning processes have gained much scrutiny and have improved. However, the danger of resourceful attackers for these systems is still a concern.

This specification is an optional extension to the EAP-AKA' authentication method which was defined in RFC 5448 (to be superseded by draft-ietf-emu-rfc5448bis). The extension, when negotiated, provides Perfect Forward Secrecy for the session key generated as a part of the authentication run in EAP-AKA'. This prevents an attacker who has gained access to the long-term pre-shared secret in a SIM card from being able to decrypt all past communications. In addition, if the attacker stays merely a passive eavesdropper, the extension prevents attacks against future sessions. This forces attackers to use active attacks instead.

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 July 25, 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	3
2. Protocol Design and Deployment Objectives	4
3. Background	5
3.1. AKA	5
3.2. EAP-AKA' Protocol	6
3.3. Attacks Against Long-Term Shared Secrets in Smart Cards	8
4. Requirements Language	8
5. Protocol Overview	8
6. Extensions to EAP-AKA'	11
6.1. AT_PUB_ECDHE	11
6.2. AT_KDF_PFS	11
6.3. New Key Derivation Function	14
6.4. ECDHE Groups	15
6.5. Message Processing	15
6.5.1. EAP-Request/AKA'-Identity	15
6.5.2. EAP-Response/AKA'-Identity	16
6.5.3. EAP-Request/AKA'-Challenge	16
6.5.4. EAP-Response/AKA'-Challenge	16
6.5.5. EAP-Request/AKA'-Reauthentication	17
6.5.6. EAP-Response/AKA'-Reauthentication	17
6.5.7. EAP-Response/AKA'-Synchronization-Failure	17
6.5.8. EAP-Response/AKA'-Authentication-Reject	17
6.5.9. EAP-Response/AKA'-Client-Error	18
6.5.10. EAP-Request/AKA'-Notification	18
6.5.11. EAP-Response/AKA'-Notification	18
7. Security Considerations	18
8. IANA Considerations	22
9. References	22
9.1. Normative References	22

9.2. Informative References	23
Appendix A. Change Log	24
Appendix B. Acknowledgments	25
Authors' Addresses	25

1. Introduction

Many different attacks have been reported as part of revelations associated with pervasive surveillance. Some of the reported attacks involved compromising smart cards, such as attacking SIM card manufacturers and operators in an effort to compromise shared secrets stored on these cards. Such attacks are conceivable, for instance, during the manufacturing process of cards, or during the transfer of cards and associated information to the operator. Since the publication of reports about such attacks, manufacturing and provisioning processes have gained much scrutiny and have improved.

However, the danger of resourceful attackers attempting to gain information about SIM cards is still a concern. They are a high-value target and concern a large number of people. Note that the attacks are largely independent of the used authentication technology; the issue is not vulnerabilities in algorithms or protocols, but rather the possibility of someone gaining unlawful access to key material. While the better protection of manufacturing and other processes is essential in protecting against this, there is one question that we as protocol designers can ask. Is there something that we can do to limit the consequences of attacks, should they occur?

The authors want to provide a public specification of an extension that helps defend against one aspect of pervasive surveillance. This is important, given the large number of users such practices may affect. It is also a stated goal of the IETF to ensure that we understand the surveillance concerns related to IETF protocols and take appropriate countermeasures [RFC7258]. This document does that for EAP-AKA'.

This specification is an optional extension to the EAP-AKA' authentication method [RFC5448] (to be superseded by [I-D.ietf-emu-rfc5448bis]). The extension, when negotiated, provides Perfect Forward Secrecy for the session key generated as a part of the authentication run in EAP-AKA'. This prevents an attacker who has gained access to the long-term pre-shared secret in a SIM card from being able to decrypt all past communications. In addition, if the attacker stays merely a passive eavesdropper, the extension prevents attacks against future sessions. This forces attackers to use active attacks instead. As with other protocols, an active attacker with access to the long-term key material will of course be

able to attack all future communications, but risks detection, particularly if done at scale.

Attacks against AKA authentication via compromising the long-term secrets in the SIM cards have been an active discussion topic in many contexts. Perfect forward secrecy is on the list of features for the next release of 3GPP (5G Phase 2), and this document provides a basis for providing this feature in a particular fashion.

It should also be noted that 5G network architecture includes the use of the EAP framework for authentication. While any methods can be run, the default authentication method within that context will be EAP-AKA'. As a result, improvements in EAP-AKA' security have a potential to improve security for large number of users.

2. Protocol Design and Deployment Objectives

This extension specified here re-uses large portions of the current structure of 3GPP interfaces and functions, with the rationale that this will make the construction more easily adopted. In particular, the construction maintains the interface between the Universal Subscriber Identification Module (USIM) and the mobile terminal intact. As a consequence, there is no need to roll out new credentials to existing subscribers. The work is based on an earlier paper [TrustCom2015], and uses much of the same material, but applied to EAP rather than the underlying AKA method.

It has been a goal to implement this change as an extension of the widely supported EAP-AKA' method, rather than a completely new authentication method. The extension is implemented as a set of new, optional attributes, that are provided alongside the base attributes in EAP-AKA'. Old implementations can ignore these attributes, but their presence will nevertheless be verified as part of base EAP-AKA' integrity verification process, helping protect against bidding down attacks. This extension does not increase the number of rounds necessary to complete the protocol.

The use of this extension is at the discretion of the authenticating parties. It should be noted that PFS and defenses against passive attacks are by no means a panacea, but they can provide a partial defense that increases the cost and risk associated with pervasive surveillance.

While adding perfect forward secrecy to the existing mobile network infrastructure can be done in multiple different ways, the authors believe that the approach chosen here is relatively easily deployable. In particular:

- o As noted above, no new credentials are needed; there is no change to SIM cards.
- o PFS property can be incorporated into any current or future system that supports EAP, without changing any network functions beyond the EAP endpoints.
- o Key generation happens at the endpoints, enabling highest grade key material to be used both by the endpoints and the intermediate systems (such as access points that are given access to specific keys).
- o While EAP-AKA' is just one EAP method, for practical purposes perfect forward secrecy being available for both EAP-TLS [RFC5216] [I-D.mattsson-eap-tls13] and EAP-AKA' ensures that for many practical systems perfect forward secrecy can be enabled for either all or significant fraction of users.

3. Background

3.1. AKA

AKA is based on challenge-response mechanisms and symmetric cryptography. AKA typically runs in a UMTS Subscriber Identity Module (USIM) or a CDMA2000 (Removable) User Identity Module ((R)UIM). In contrast with its earlier GSM counterparts, 3rd generation AKA provides long key lengths and mutual authentication.

AKA works in the following manner:

- o The identity module and the home environment have agreed on a secret key beforehand.
- o The actual authentication process starts by having the home environment produce an authentication vector, based on the secret key and a sequence number. The authentication vector contains a random part RAND, an authenticator part AUTN used for authenticating the network to the identity module, an expected result part XRES, a 128-bit session key for integrity check IK, and a 128-bit session key for encryption CK.
- o The authentication vector is passed to the serving network, which uses it to authenticate the device.
- o The RAND and the AUTN are delivered to the identity module.
- o The identity module verifies the AUTN, again based on the secret key and the sequence number. If this process is successful (the

AUTN is valid and the sequence number used to generate AUTN is within the correct range), the identity module produces an authentication result RES and sends it to the serving network.

- o The serving network verifies the correct result from the identity module. If the result is correct, IK and CK can be used to protect further communications between the identity module and the home environment.

3.2. EAP-AKA' Protocol

When AKA (and AKA') are embedded into EAP, the authentication on the network side is moved to the home environment; the serving network performs the role of a pass-through authenticator. Figure 1 describes the basic flow in the EAP-AKA' authentication process. The definition of the full protocol behaviour, along with the definition of attributes AT_RAND, AT_AUTN, AT_MAC, and AT_RES can be found in [I-D.ietf-emu-rfc5448bis] and [RFC4187].

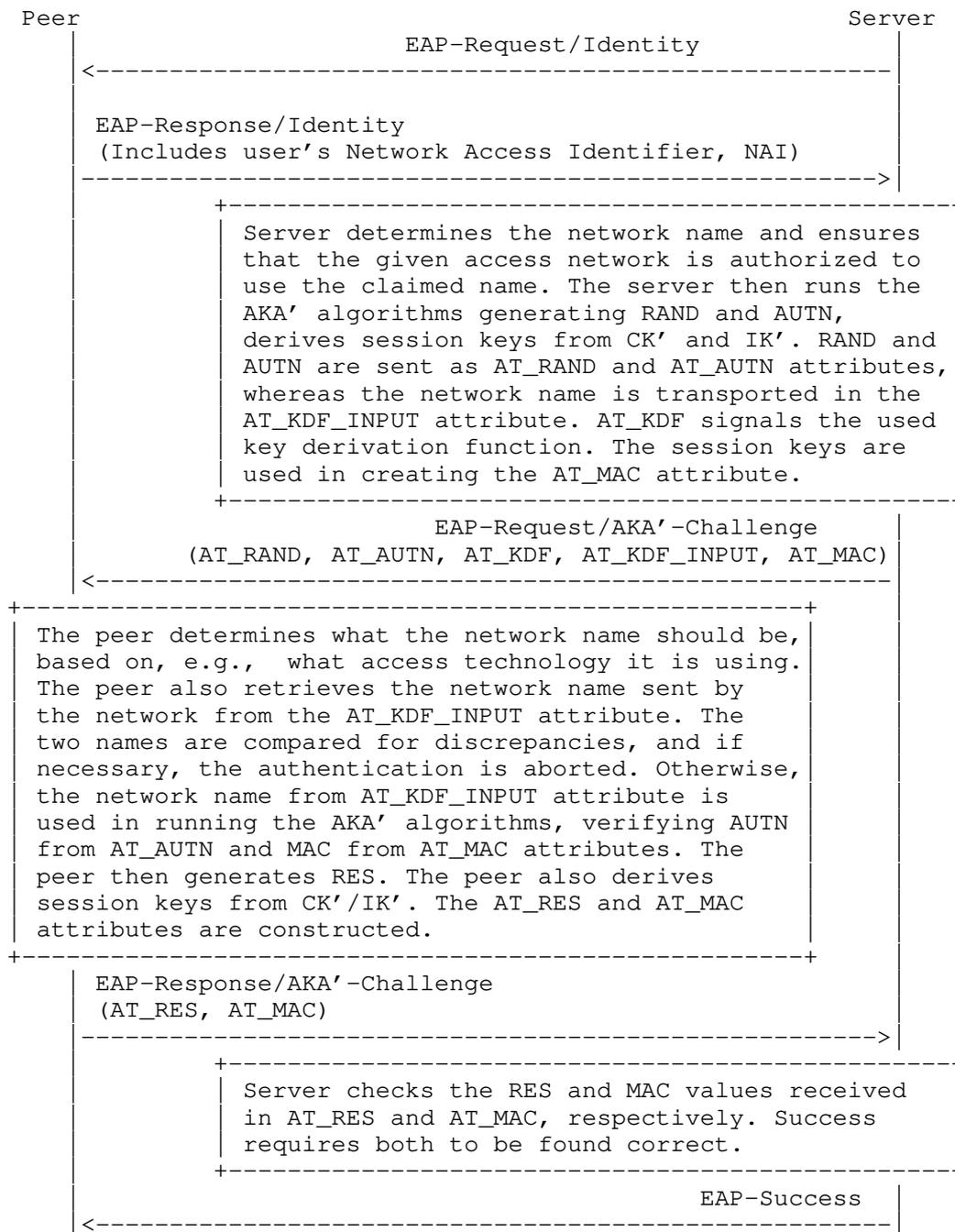


Figure 1: EAP-AKA' Authentication Process

3.3. Attacks Against Long-Term Shared Secrets in Smart Cards

Current 3GPP systems use (U)SIM pre-shared key based protocols and Authentication and Key Agreement (AKA) to authenticate subscribers. The general security properties and potential vulnerabilities of AKA and EAP-AKA' are discussed in [I-D.ietf-emu-rfc5448bis].

An important vulnerability in that discussion relates to the recent reports of compromised long term pre-shared keys used in AKA [Heist2015]. These attacks are not specific to AKA or EAP-AKA', as all security systems fail at least to some extent if key material is stolen. However, the reports indicate a need to look into solutions that can operate at least to an extent under these types of attacks. It is noted in [Heist2015] that some security can be retained even in the face of the attacks by providing Perfect Forward Security (PFS) [DOW1992] for the session key. If AKA would have provided PFS, compromising the pre-shared key would not be sufficient to perform passive attacks; the attacker is, in addition, forced to be a Man-In-The-Middle (MITM) during the AKA run and subsequent communication between the parties.

4. 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.

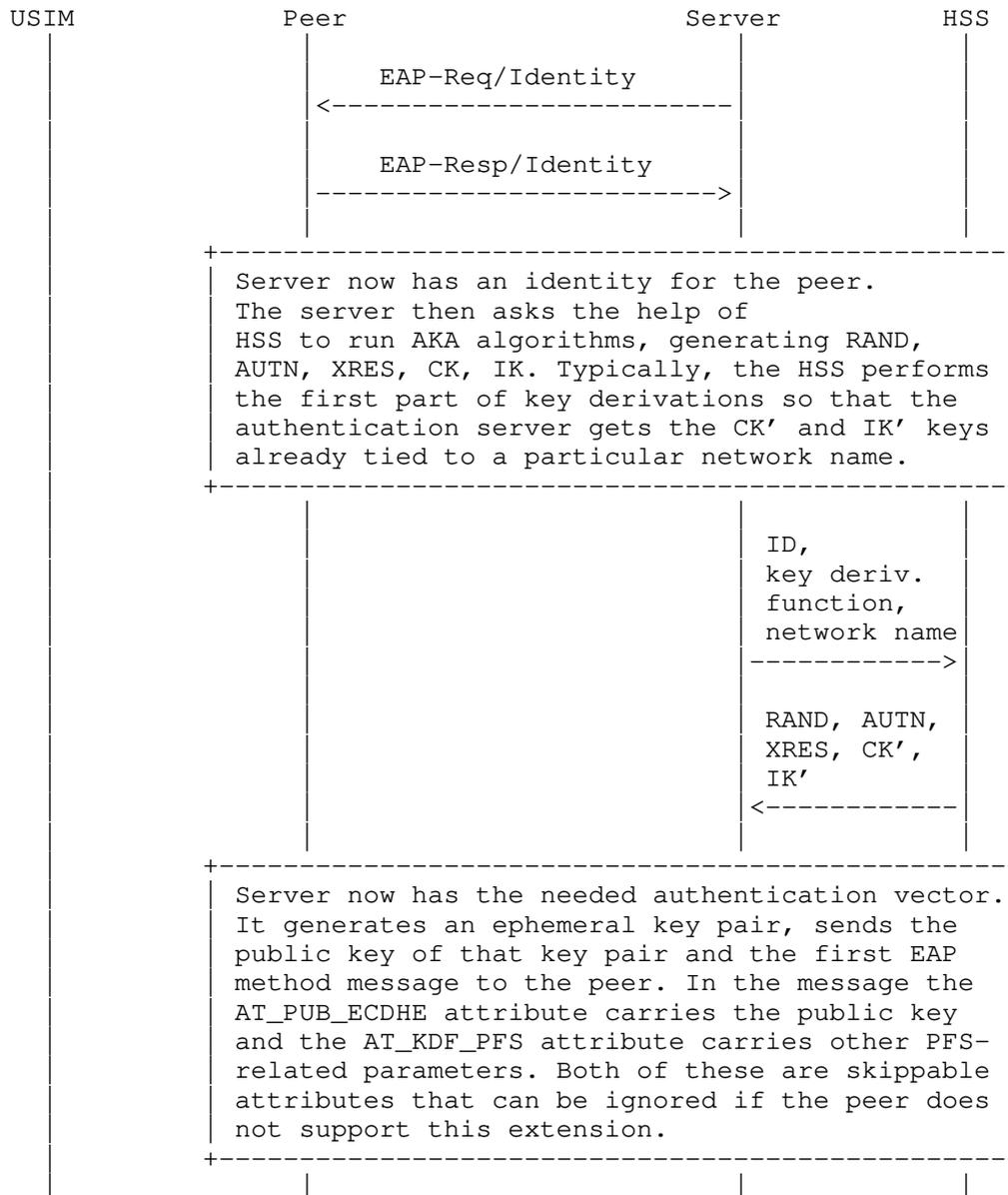
5. Protocol Overview

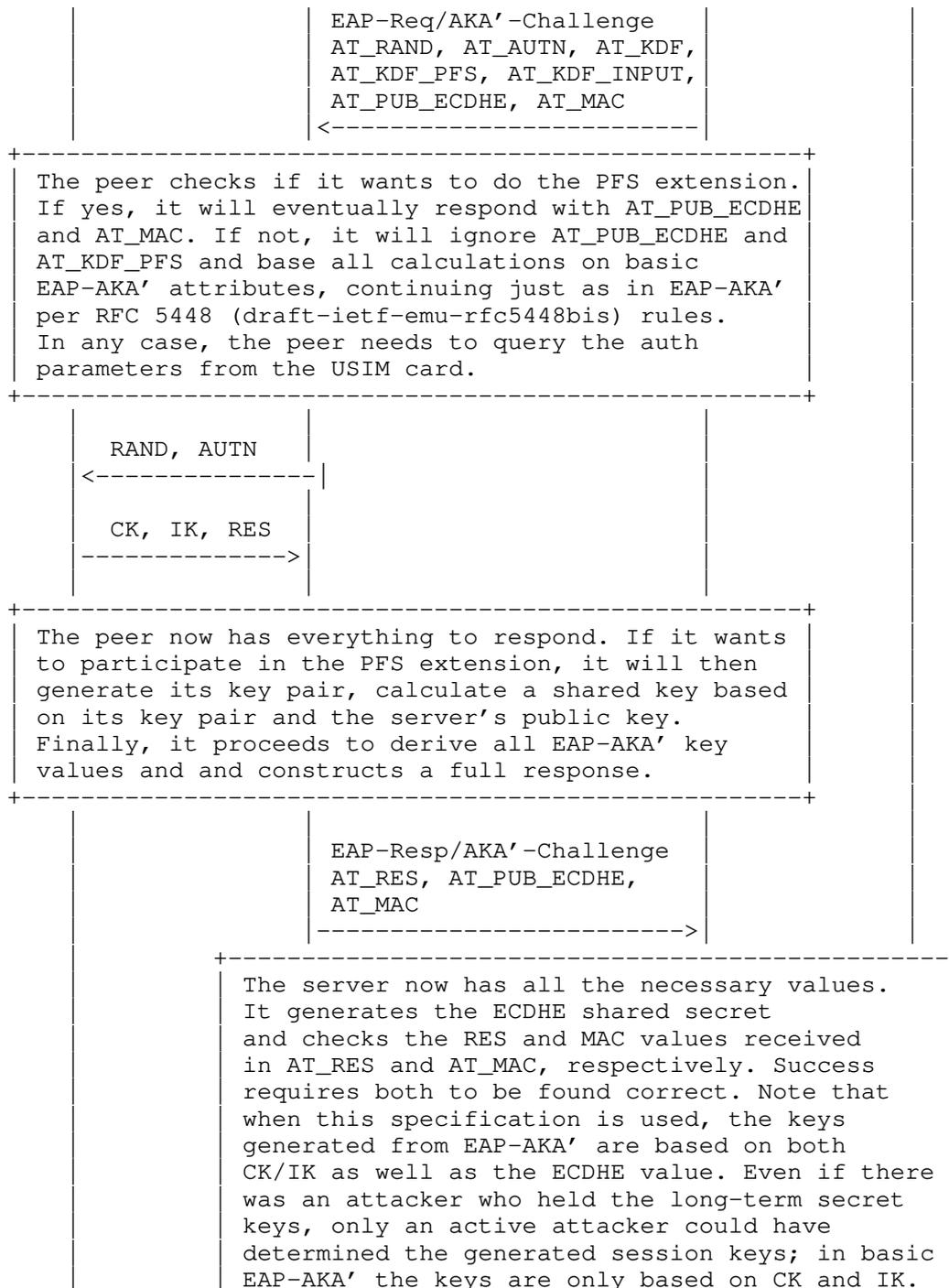
Introducing PFS for EAP-AKA' can be achieved by using an Elliptic Curve Diffie-Hellman (ECDH) exchange [RFC7748]. In EAP-AKA' PFS this exchange is run in an ephemeral manner, i.e., using temporary keys as specified in [RFC8031] Section 2. This method is referred to as ECDHE, where the last 'E' stands for Ephemeral.

The enhancements in the EAP-AKA' PFS protocol are compatible with the signaling flow and other basic structures of both AKA and EAP-AKA'. The intent is to implement the enhancement as optional attributes that legacy implementations can ignore.

The purpose of the protocol is to achieve mutual authentication between the EAP server and peer, and to establish keying material for secure communication between the two. This document specifies the calculation of key material, providing new properties that are not present in key material provided by EAP-AKA' in its original form.

Figure 2 below describes the overall process. Since our goal has been to not require new infrastructure or credentials, the flow diagrams also show the conceptual interaction with the USIM card and the 3GPP authentication server (HSS). The details of those interactions are outside the scope of this document, however, and the reader is referred to the 3GPP specifications .





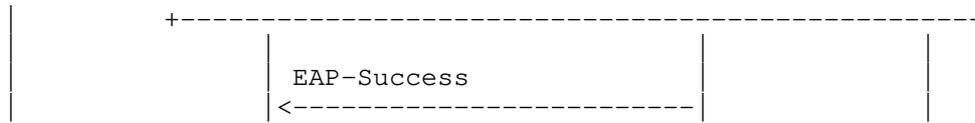


Figure 2: EAP-AKA' PFS Authentication Process

6. Extensions to EAP-AKA'

6.1. AT_PUB_ECDHE

The AT_PUB_ECDHE carries an ECDHE value.

The format of the AT_PUB_ECDHE attribute is shown below.

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				
AT_PUB_ECDHE									Length									Value ...																	

The fields are as follows:

AT_PUB_ECDHE

This is set to TBA1 BY IANA.

Length

The length of the attribute, set as other attributes in EAP-AKA [RFC4187].

Value

This value is the sender's ECDHE public value. For Curve25519, the length of this value is 32 bytes, encoded in binary as specified [RFC7748] Section 6.1.

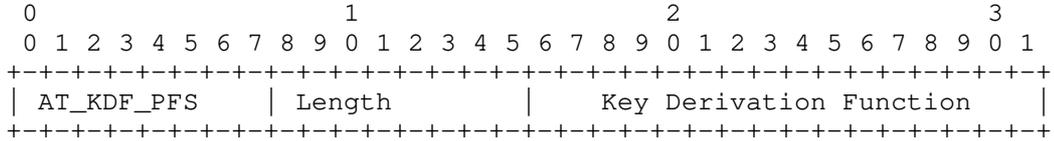
To retain the security of the keys, the sender SHALL generate a fresh value for each run of the protocol.

6.2. AT_KDF_PFS

The AT_KDF_PFS indicates the used or desired key generation function, if the Perfect Forward Secrecy extension is taken into use. It will also at the same time indicate the used or desired ECDHE group. A new attribute is needed to carry this information, as AT_KDF carries

the legacy KDF value for those EAP peers that cannot or do not want to use this extension.

The format of the AT_KDF_PFS attribute is shown below.



The fields are as follows:

AT_KDF_PFS

This is set to TBA2 BY IANA.

Length

The length of the attribute, MUST be set to 1.

Key Derivation Function

An enumerated value representing the key derivation function that the server (or peer) wishes to use. See Section 6.3 for the functions specified in this document. Note: This field has a different name space than the similar field in the AT_KDF attribute Key Derivation Function defined in [I-D.ietf-emu-rfc5448bis].

Servers MUST send one or more AT_KDF_PFS 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. The most preferred function is the only one that the server includes a public key value for, however. So for a set of AT_KDF_PFS attributes, there is always only one AT_PUB_ECDHE attribute.

Upon receiving a set of these attributes:

- o If the peer supports and is willing to use the key derivation function indicated by the first AT_KDF_PFS attribute, and is willing and able to use the extension defined in this specification, the function is taken into use without any further negotiation.

- o If the peer does not support this function or is unwilling to use it, it responds to the server with an indication that a different function is needed. Similarly with the negotiation process defined in [I-D.ietf-emu-rfc5448bis] for AT_KDF, the peer sends EAP-Response/AKA'-Challenge message that contains only one attribute, AT_KDF_PFS with the value set to the desired alternative function from among the ones suggested by the server earlier. If there is no suitable alternative, the peer has a choice of either falling back to EAP-AKA' or behaving as if AUTN had been incorrect and failing authentication (see Figure 3 of [RFC4187]). The peer MUST fail the authentication if there are any duplicate values within the list of AT_KDF_PFS attributes (except where the duplication is due to a request to change the key derivation function; see below for further information).
- o If the peer does not recognize the extension defined in this specification or is unwilling to use it, it ignores the AT_KDF_PFS attribute.

Upon receiving an EAP-Response/AKA'-Challenge with AT_KDF_PFS from the peer, the server checks that the suggested AT_KDF_PFS value was one of the alternatives in its offer. The first AT_KDF_PFS value in the message from the server is not a valid alternative. If the peer has replied with the first AT_KDF_PFS 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 in [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_PFS 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_PFS attributes. If yes, it continues. 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_PFS attributes without having requested negotiation, the peer MUST behave as if AT_MAC had been incorrect and fail the authentication.

6.3. New Key Derivation Function

A new Key Derivation Function type is defined for "EAP-AKA' with ECDHE and Curve25519", represented by value 1. It represents a particular choice of key derivation function and at the same time selects an ECDHE group to be used.

The Key Derivation Function type value is only used in the AT_KDF_PFS attribute, and should not be confused with the different range of key derivation functions that can be represented in the AT_KDF attribute as defined in [I-D.ietf-emu-rfc5448bis].

Key derivation in this extension produces exactly the same keys for internal use within one authentication run as [I-D.ietf-emu-rfc5448bis] EAP-AKA' does. For instance, K_aut that is used in AT_MAC is still exactly as it was in EAP-AKA'. The only change to key derivation is in re-authentication keys and keys exported out of the EAP method, MSK and EMSK. As a result, EAP-AKA' attributes such as AT_MAC continue to be usable even when this extension is in use.

When the Key Derivation Function field in the AT_KDF_PFS attribute is set to 1 and the Key Derivation Function field in the AT_KDF attribute is also set to 1, the Master Key (MK) is derived as follows below.

```

MK           = PRF'(IK' | CK', "EAP-AKA'" | Identity)
MK_ECDHE    = PRF'(IK' | CK' | SHARED_SECRET, "EAP-AKA' PFS" | Identity)
K_encr      = MK[0..127]
K_aut       = MK[128..383]
K_re        = MK_ECDHE[0..255]
MSK         = MK_ECDHE[256..767]
EMSK        = MK_ECDHE[768..1279]

```

Where SHARED_SECRET is the shared secret computed via ECDHE, as specified in Section 2 of [RFC8031] and Section 6.1 of [RFC7748].

Both the peer and the server MAY check for zero-value shared secret as specified in Section 6.1 of [RFC7748]. If such checking is performed and the SHARED_SECRET has a zero value, both parties MUST behave as if the current EAP-AKA' authentication process starts again from the beginning.

Note: The way that shared secret is tested for zero can, if performed inappropriately, provide an ability for attackers to listen to CPU power usage side channels. Refer to [RFC7748] for a description of how to perform this check in a way that it does not become a problem.

The rest of computation proceeds as defined in Section 3.3 of [I-D.ietf-emu-rfc5448bis].

For readability, an explanation of the notation used above is copied here: [n..m] denotes the substring from bit n to m. PRF' is a new pseudo-random function specified in [I-D.ietf-emu-rfc5448bis]. K_encr is the encryption key, 128 bits, K_aut is the authentication key, 256 bits, K_re is the re-authentication key, 256 bits, MSK is the Master Session Key, 512 bits, and EMSK is the Extended Master Session Key, 512 bits. MSK and EMSK are outputs from a successful EAP method run [RFC3748].

CK and IK are produced by the AKA algorithm. IK' and CK' are derived as specified in [I-D.ietf-emu-rfc5448bis] from IK and CK.

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

Identity is the peer identity as specified in Section 7 of [RFC4187].

6.4. ECDHE Groups

The selection of suitable groups for the elliptic curve computation is necessary. The choice of a group is made at the same time as deciding to use of particular key derivation function in AT_KDF_PFS. For "EAP-AKA' with ECDHE and Curve25519" the group is the Curve25519 group specified in [RFC8031].

6.5. Message Processing

This section specifies the changes related to message processing when this extension is used in EAP-AKA'. It specifies when a message may be transmitted or accepted, which attributes are allowed in a message, which attributes are required in a message, and other message-specific details, where those details are different for this extension than the base EAP-AKA' or EAP-AKA protocol. Unless otherwise specified here, the rules from [I-D.ietf-emu-rfc5448bis] or [RFC4187] apply.

6.5.1. EAP-Request/AKA'-Identity

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.2. EAP-Response/AKA'-Identity

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.3. EAP-Request/AKA'-Challenge

The server sends the EAP-Request/AKA'-Challenge on full authentication as specified by [RFC4187] and [I-D.ietf-emu-rfc5448bis]. The attributes AT_RAND, AT_AUTN, and AT_MAC MUST be included and checked on reception as specified in [RFC4187]. They are also necessary for backwards compatibility.

In EAP-Request/AKA'-Challenge, there is no message-specific data covered by the MAC for the AT_MAC attribute. The AT_KDF_PFS and AT_PUB_ECDHE attributes MUST be included. The AT_PUB_ECDHE attribute carries the server's public Diffie-Hellman key. If either AT_KDF_PFS or AT_PUB_ECDHE is missing on reception, the peer MUST treat them as if neither one was sent, and the assume that the extension defined in this specification is not in use.

The AT_RESULT_IND, AT_CHECKCODE, AT_IV, AT_ENCR_DATA, AT_PADDING, AT_NEXT_PSEUDONYM, AT_NEXT_REAUTH_ID and other attributes may be included as specified in Section 9.3 of [RFC4187].

When processing this message, the peer MUST process AT_RAND, AT_AUTN, AT_KDF_PFS, AT_PUB_ECDHE before processing other attributes. Only if these attributes are verified to be valid, the peer derives keys and verifies AT_MAC. If the peer is unable or unwilling to perform the extension specified in this document, it proceeds as defined in [I-D.ietf-emu-rfc5448bis]. Finally, the operation in case an error occurs is specified in Section 6.3.1. of [RFC4187].

6.5.4. EAP-Response/AKA'-Challenge

The peer sends EAP-Response/AKA'-Challenge in response to a valid EAP-Request/AKA'-Challenge message, as specified by [RFC4187] and [I-D.ietf-emu-rfc5448bis]. If the peer supports and is willing to perform the extension specified in this protocol, and the server had made a valid request involving the attributes specified in Section 6.5.3, the peer responds per the rules specified below. Otherwise, the peer responds as specified in [RFC4187] and [I-D.ietf-emu-rfc5448bis] and ignores the attributes related to this extension. If the peer has not received attributes related to this extension from the Server, and has a policy that requires it to always use this extension, it behaves as if AUTN had been incorrect and fails the authentication.

The AT_MAC attribute MUST be included and checked as specified in [I-D.ietf-emu-rfc5448bis]. In EAP-Response/AKA'-Challenge, there is no message-specific data covered by the MAC. The AT_PUB_ECDHE attribute MUST be included, and carries the peer's public Diffie-Hellman key.

The AT_RES attribute MUST be included and checked as specified in [RFC4187]. When processing this message, the Server MUST process AT_RES before processing other attributes. Only if these attribute is verified to be valid, the Server derives keys and verifies AT_MAC.

If the Server has proposed the use of the extension specified in this protocol, but the peer ignores and continues the basic EAP-AKA' authentication, the Server makes policy decision of whether this is allowed. If this is allowed, it continues the EAP-AKA' authentication to completion. If it is not allowed, the Server MUST behave as if authentication failed.

The AT_CHECKCODE, AT_RESULT_IND, AT_IV, AT_ENCR_DATA and other attributes may be included as specified in Section 9.4 of [RFC4187].

6.5.5. EAP-Request/AKA'-Reauthentication

No changes, but note that the re-authentication process uses the keys generated in the original EAP-AKA' authentication, which, if the extension specified in this documents is in use, employs key material from the Diffie-Hellman procedure.

6.5.6. EAP-Response/AKA'-Reauthentication

No changes, but as discussed in Section 6.5.5, re-authentication is based on the key material generated by EAP-AKA' and the extension defined in this document.

6.5.7. EAP-Response/AKA'-Synchronization-Failure

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.8. EAP-Response/AKA'-Authentication-Reject

No changes, except that the AT_KDF_PFS or AT_PUB_ECDHE attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.9. EAP-Response/AKA'-Client-Error

No changes, except that the `AT_KDF_PFS` or `AT_PUB_ECDHE` attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

6.5.10. EAP-Request/AKA'-Notification

No changes.

6.5.11. EAP-Response/AKA'-Notification

No changes.

7. Security Considerations

This section deals only with the changes to security considerations as they differ from EAP-AKA', or as new information has been gathered since the publication of [I-D.ietf-emu-rfc5448bis].

The possibility of attacks against key storage offered in SIM or other smart cards has been a known threat. But as the discussion in Section 3.3 shows, the likelihood of practically feasible attacks has increased. Many of these attacks can be best dealt with improved processes, e.g., limiting the access to the key material within the factory or personnel, etc. But not all attacks can be entirely ruled out for well-resourced adversaries, irrespective of what the technical algorithms and protection measures are.

This extension can provide assistance in situations where there is a danger of attacks against the key material on SIM cards by adversaries that can not or who are unwilling to mount active attacks against large number of sessions. This extension is most useful when used in a context where EAP keys are used without further mixing that can provide Perfect Forward Secrecy. For instance, when used with IKEv2 [RFC7296], the session keys produced by IKEv2 have this property, so better characteristics of EAP keys is not that useful. However, typical link layer usage of EAP does not involve running Diffie-Hellman, so using EAP to authenticate access to a network is one situation where the extension defined in this document can be helpful.

This extension generates keying material using the ECDHE exchange in order to gain the PFS property. This means that once an EAP-AKA' authentication run ends, the session that it was used to protect is closed, and the corresponding keys are forgotten, even someone who has recorded all of the data from the authentication run and session and gets access to all of the AKA long-term keys cannot reconstruct

the keys used to protect the session or any previous session, without doing a brute force search of the session key space.

Even if a compromise of the long-term keys has occurred, PFS is still provided for all future sessions, as long as the attacker does not become an active attacker. Of course, as with other protocols, if the attacker has learned the keys and does become an active attacker, there is no protection that that can be provided for future sessions. Among other things, such an active attacker can impersonate any legitimate endpoint in EAP-AKA', become a MITM in EAP-AKA' or the extension defined in this document, retrieve all keys, or turn off PFS. Still, past sessions where PFS was in use remain protected.

Achieving PFS requires that when a connection is closed, each endpoint MUST forget not only the ephemeral keys used by the connection but also any information that could be used to recompute those keys.

The following security properties of EAP-AKA' are impacted through this extension:

Protected ciphersuite negotiation

EAP-AKA' has a negotiation mechanism for selecting the key derivation functions, and this mechanism has been extended by the extension specified in this document. The resulting mechanism continues to be secure against bidding down attacks.

There are two specific needs in the negotiation mechanism:

Negotiating key derivation function within the extension

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.

Negotiating the use of this extension

This extension is offered by the server through presenting the AT_KDF_PFS and AT_PUB_ECDHE attributes in the EAP-Request/AKA'-Challenge message. These attributes are protected by AT_MAC, so attempts to change or omit them by an adversary will be detected.

Except of course, if the adversary holds the long-term shared secret and is willing to engage in an active attack. Such an attack can, for instance, forge the negotiation process so that no PFS will be provided. However, as noted above, an attacker with these capabilities will in any case be able to impersonate any party in the protocol and perform MITM attacks. That is not a situation that can be improved by a technical solution. However, as discussed in the introduction, even an attacker with access to the long-term keys is required to be a MITM on each AKA run and subsequent communication, which makes mass surveillance more laborous.

The security properties of the extension also depend on a policy choice. As discussed in Section 6.5.4, both the peer and the server make a policy decision of what to do when it was willing to perform the extension specified in this protocol, but the other side does not wish to use the extension. Allowing this has the benefit of allowing backwards compatibility to equipment that did not yet support the extension. When the extension is not supported or negotiated by the parties, no PFS can obviously be provided.

If turning off the extension specified in this protocol is not allowed by policy, the use of legacy equipment that does not support this protocol is no longer possible. This may be appropriate when, for instance, support for the extension is sufficiently widespread, or required in a particular version of a mobile network.

Key derivation

This extension provides key material that is based on the Diffie-Hellman keys, yet bound to the authentication through the (U)SIM card. This means that subsequent payload communications between the parties are protected with keys that are not solely based on information in the clear (such as the RAND) and information derivable from the long-term shared secrets on the (U)SIM card. As a result, if anyone successfully recovers shared secret information, they are unable to decrypt communications protected by the keys generated through this extension. Note that the recovery of shared secret information could occur either before or after the time that the protected communications are used. When this extension is used, communications at time t_0 can be protected if at some later time t_1 an adversary learns of long-term shared secret and has access to a recording of the encrypted communications.

Obviously, this extension is still vulnerable to attackers that are willing to perform an active attack and who at the time of the attack have access to the long-term shared secret.

This extension does not change the properties related to re-authentication. No new Diffie-Hellman run is performed during the re-authentication allowed by EAP-AKA'. However, if this extension was in use when the original EAP-AKA' authentication was performed, the keys used for re-authentication (K_{re}) are based on the Diffie-Hellman keys, and hence continue to be equally safe against exposure of the long-term secrets as the original authentication.

In addition, it is worthwhile to discuss Denial-of-Service attacks and their impact on this protocol. The calculations involved in public key cryptography require computing power, which could be used in an attack to overpower either the peer or the server. While some forms of Denial-of-Service attacks are always possible, the following factors help mitigate the concerns relating to public key cryptography and EAP-AKA' PFS.

- o In 5G context, other parts of the connection setup involve public key cryptography, so while performing additional operations in EAP-AKA' is an additional concern, it does not change the overall situation. As a result, the relevant system components need to be dimensioned appropriately, and detection and management mechanisms to reduce the effect of attacks need to be in place.
- o This specification is constructed so that a separation between the USIM and Peer on client side and the Server and HSS on network side is possible. This ensures that the most sensitive (or legacy) system components can not be the target of the attack. For instance, EAP-AKA' and public key cryptography takes place in the phone and not the low-power SIM card.
- o EAP-AKA' has been designed so that the first actual message in the authentication process comes from the Server, and that this message will not be sent unless the user has been identified as an active subscriber of the operator in question. While the initial identity can be spoofed before authentication has succeeded, this reduces the efficiency of an attack.
- o Finally, this memo specifies an order in which computations and checks must occur. When processing the EAP-Request/AKA'-Challenge message, for instance, the AKA authentication must be checked and succeed before the peer proceeds to calculating or processing the PFS related parameters (see Section 6.5.4). The same is true of EAP-Response/AKA'-Challenge (see Section 6.5.4). This ensures

that the parties need to show possession of the long-term secret in some way, and only then will the PFS calculations become active. This limits the Denial-of-Service to specific, identified subscribers. While botnets and other forms of malicious parties could take advantage of actual subscribers and their key material, at least such attacks are (a) limited in terms of subscribers they control, and (b) identifiable for the purposes of blocking the affected subscribers.

8. IANA Considerations

This extension of EAP-AKA' shares its attribute space and subtypes with EAP-SIM [RFC4186], EAP-AKA [RFC4186], and EAP-AKA' [I-D.ietf-emu-rfc5448bis].

Two new Attribute Type value (TBA1, TBA2) in the skippable range need to be assigned for AT_PUB_ECDHE (Section 6.1) and AT_KDF_PFS (Section 6.2 in the EAP-AKA and EAP-SIM Parameters registry under Attribute Types.

Also, a new registry should be created to represent Diffie-Hellman Key Derivation Function types. The "EAP-AKA' with ECDHE and Curve25519" type (1, see Section 6.3) needs to be assigned, along with one reserved value. The initial contents of this namespace are therefore as below; new values can be created through the Specification Required policy [RFC8126].

Value	Description	Reference
0	Reserved	[TBD BY IANA: THIS RFC]
1	EAP-AKA' with ECDHE and Curve25519	[TBD BY IANA: THIS RFC]
2-65535	Unassigned	

9. References

9.1. 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. Levkowitz, 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>>.
- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/info/rfc7748>>.
- [RFC8031] Nir, Y. and S. Josefsson, "Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement", RFC 8031, DOI 10.17487/RFC8031, December 2016, <<https://www.rfc-editor.org/info/rfc8031>>.
- [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>>.
- [I-D.ietf-emu-rfc5448bis]
Arkko, J., Lehtovirta, V., Torvinen, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", draft-ietf-emu-rfc5448bis-04 (work in progress), January 2019.

9.2. Informative References

- [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>>.
- [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>>.
- [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>>.

- [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.mattsson-eap-tls13]
Mattsson, J. and M. Sethi, "Using EAP-TLS with TLS 1.3", draft-mattsson-eap-tls13-02 (work in progress), March 2018.
- [TrustCom2015]
Arkko, J., Norrman, K., Naslund, M., and B. Sahlin, "A USIM compatible 5G AKA protocol with perfect forward secrecy", August 2015 in Proceedings of the TrustCom 2015, IEEE.
- [Heist2015]
Scahill, J. and J. Begley, "The great SIM heist", February 2015, in <https://firstlook.org/theintercept/2015/02/19/great-sim-heist/> .
- [DOW1992] Diffie, W., vanOorschot, P., and M. Wiener, "Authentication and Authenticated Key Exchanges", June 1992, in Designs, Codes and Cryptography 2 (2): pp. 107-125.

Appendix A. Change Log

The -04 version of this draft made only editorial changes.

The -03 version of this draft changed the naming of various protocol components, values, and notation to match with the use of ECDH in ephemeral mode. The AT_KDF_PFS negotiation process was clarified in that exactly one key is ever sent in AT_KDF_ECDHE. The option of checking for zero key values IN ECDHE was added. The format of the actual key in AT_PUB_ECDHE was specified. Denial-of-service considerations for the PFS process have been updated. Bidding down attacks against this extension itself are discussed extensively. This version also addressed comments from reviewers, including the August review from Mohit Sethi, and comments made during IETF-102 discussion.

Appendix B. Acknowledgments

The authors would like to note that the technical solution in this document came out of the TrustCom paper [TrustCom2015], whose authors were J. Arkko, K. Norrman, M. Naslund, and B. Sahlin. This document uses also a lot of material from [RFC4187] by J. Arkko and H. Haverinen as well as [RFC5448] by J. Arkko, V. Lehtovirta, and P. Eronen.

The authors would also like to thank Tero Kivinen, John Mattson, Mohit Sethi, Vesa Lehtovirta, Joseph Salowey, Kathleen Moriarty, Zhang Fu, Bengt Sahlin, Ben Campbell, Prajwol Kumar Nakarmi, Goran Rune, Tim Evans, Helena Vahidi Mazinani, Anand R. Prasad, and many other people at the GSMA and 3GPP groups for interesting discussions in this problem space.

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Karl Norrman
Ericsson
Stockholm 16483
Sweden

Email: karl.norrman@ericsson.com

Vesa Torvinen
Ericsson
Jorvas 02420
Finland

Email: vesa.torvinen@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

T. Aura
Aalto University
M. Sethi
Ericsson
October 22, 2018

Nimble out-of-band authentication for EAP (EAP-NOOB)
draft-aura-eap-noob-04

Abstract

Extensible Authentication Protocol (EAP) provides support for multiple authentication methods. This document defines the EAP-NOOB authentication method for nimble out-of-band (OOB) authentication and key derivation. This EAP method is intended for bootstrapping all kinds of Internet-of-Things (IoT) devices that have a minimal user interface and no pre-configured authentication credentials. The method makes use of a user-assisted one-directional OOB channel between the peer device and authentication server.

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 April 25, 2019.

Copyright Notice

Copyright (c) 2018 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
2. Terminology	4
3. EAP-NOOB protocol	4
3.1. Protocol overview	5
3.2. Protocol messages and sequences	8
3.2.1. Initial Exchange	8
3.2.2. OOB Step	10
3.2.3. Completion Exchange	11
3.2.4. Waiting Exchange	13
3.3. Protocol data fields	15
3.3.1. Peer identifier, realm and NAI	15
3.3.2. Message data fields	16
3.4. Fast reconnect and rekeying	21
3.4.1. Persistent EAP-NOOB association	21
3.4.2. Reconnect Exchange	22
3.4.3. User reset	25
3.5. Key derivation	26
3.6. Error handling	29
3.6.1. Invalid messages	31
3.6.2. Unwanted peer	31
3.6.3. State mismatch	31
3.6.4. Negotiation failure	31
3.6.5. Cryptographic verification failure	32
3.6.6. Application-specific failure	32
4. IANA Considerations	33
4.1. Cryptosuites	33
4.2. Message Types	33
4.3. Error codes	34
4.4. Domain name reservation considerations	35
5. Implementation Status	36
5.1. Implementation with wpa_supplicant and hostapd	36
5.2. Protocol modeling	37
6. Security considerations	37
6.1. Authentication principle	37
6.2. Identifying correct endpoints	39
6.3. Trusted path issues	39
6.4. Peer identifiers and attributes	40
6.5. Identity protection	40
6.6. Downgrading threats	41
6.7. Recovery from loss of last message	42
6.8. Other denial-of-service threats	42

6.9. EAP security claims	43
7. References	45
7.1. Normative references	45
7.2. Informative references	46
Appendix A. Exchanges and events per state	48
Appendix B. Application-specific parameters	49
Appendix C. ServerInfo and PeerInfo contents	50
Appendix D. EAP-NOOB roaming	52
Appendix E. OOB message as URL	53
Appendix F. Example messages	54
Appendix G. TODO list	56
Appendix H. Version history	56
Appendix I. Acknowledgments	58
Authors' Addresses	58

1. Introduction

This document describes a method for registration, authentication and key derivation for network-connected ubiquitous computing devices, such as consumer and enterprise appliances that are part of the Internet of Things (IoT). These devices may be off-the-shelf hardware that is sold and distributed without any prior registration or credential-provisioning process. Thus, the device registration in a server database, ownership of the device, and the authentication credentials for both network access and application-level security must all be established at the time of the device deployment. Furthermore, many such devices have only limited user interfaces that could be used for their configuration. Often, the interfaces are limited to either only input (e.g. camera) or output (e.g. display screen). The device configuration is made more challenging by the fact that the devices may exist in large numbers and may have to be deployed or re-configured nimbly based on user needs.

More specifically, the devices may have the following characteristics:

- o no pre-established relation with a specific server or user,
- o no pre-provisioned device identifier or authentication credentials,
- o limited user interface and configuration capabilities.

Many proprietary OOB configuration methods exists for specific IoT devices. The goal of this specification is to provide an open standard and a generic protocol for bootstrapping the security of network-connected appliances, such as displays, printers, speakers, and cameras. The security bootstrapping in this specification makes

use of a user-assisted out-of-band (OOB) channel. The security is based on the assumption that attackers are not able to observe or modify the messages conveyed through the OOB channel. We follow the common approach of performing a Diffie-Hellman key exchange over the insecure network and authenticating the established key with the help of the OOB channel in order to prevent impersonation and man-in-the-middle (MitM) attacks.

The solution presented here is intended for devices that have either an input or output interface, such as a camera or display screen, which is able to send or receive dynamically generated messages of tens of bytes in length. Naturally, this solution may not be appropriate for very small sensors or actuators that have no user interface at all. We also assume that the OOB channel is at least partly automated (e.g. camera scanning a bar code) and, thus, there is no need to absolutely minimize the length of the data transferred through the OOB channel. This differs, for example, from Bluetooth simple pairing [BluetoothPairing], where it is critical to minimize the length of the manually transferred or compared codes. Since the OOB messages are dynamically generated, we do not support static printed registration codes. This also prevents attacks where a static secret code would be leaked.

2. Terminology

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].

In addition, this document frequently uses the following terms as they have been defined in [RFC5216]:

authenticator The entity initiating EAP authentication.

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

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.

3. EAP-NOOB protocol

This section defines the EAP-NOOB protocol. The protocol is a generalized version of the original idea presented by Sethi et al. [Sethi14].

3.1. Protocol overview

One EAP-NOOB protocol execution spans multiple EAP conversations, called Exchanges. This is necessary to leave time for the OOB message to be delivered, as will be explained below.

The overall protocol starts with the Initial Exchange, in which the server allocates an identifier to the peer, and the server and peer negotiate the protocol version and cryptosuite (i.e. cryptographic algorithm suite), exchange nonces and perform an Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key exchange. The user-assisted OOB Step then takes place. This step requires only one out-of-band message either from the peer to the server or from the server to the peer. While waiting for the OOB Step action, the peer MAY probe the server by reconnecting to it with EAP-NOOB. If the OOB Step has already taken place, the probe leads to the Completion Exchange, which completes the mutual authentication and key confirmation. On the other hand, if the OOB Step has not yet taken place, the probe leads to the Waiting Exchange, and the peer will perform another probe after a server-defined minimum waiting time. The Initial Exchange and Waiting Exchange always end in EAP-Failure, while the Completion Exchange may result in EAP-Success. Once the peer and server have performed a successful Completion Exchange, both endpoints store the created association in persistent storage, and the OOB Step is not repeated. Thereafter, creation of new temporal keys, ECDHE rekeying, and updates of cryptographic algorithms can be achieved with the Reconnect Exchange.

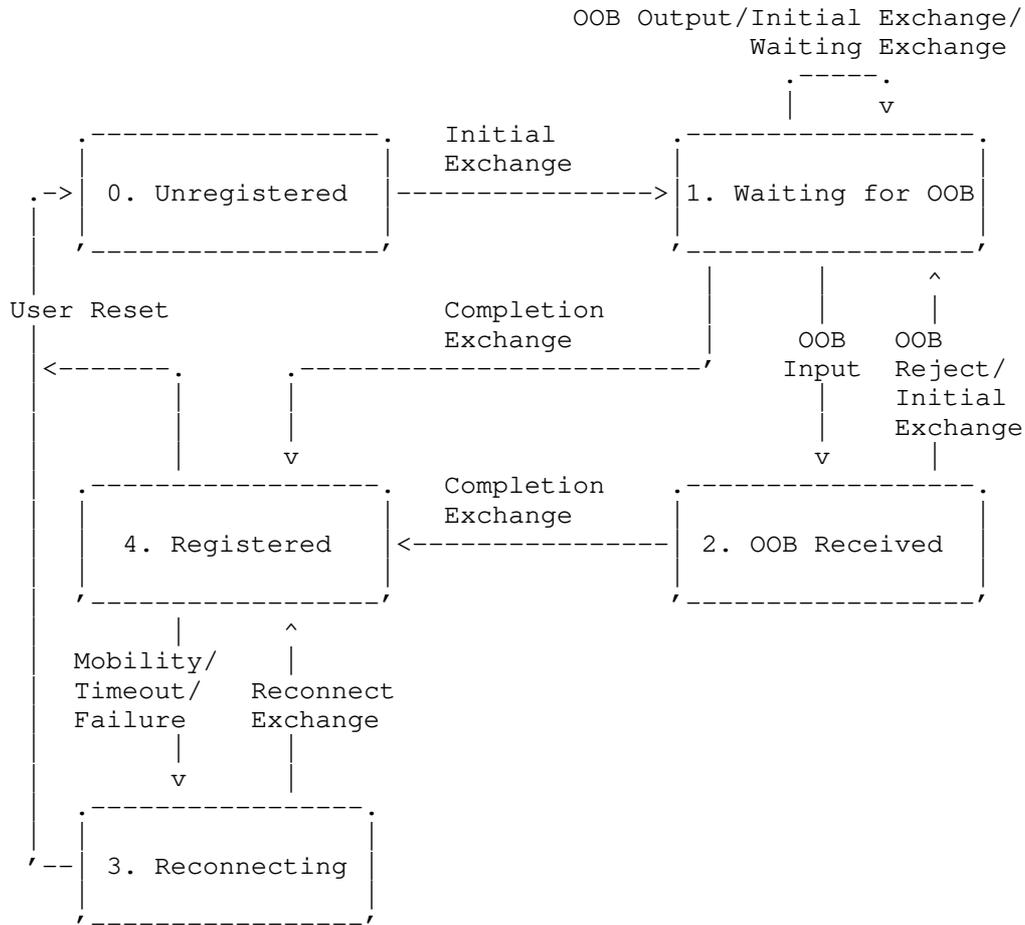


Figure 1: EAP-NOOB server-peer association state machine

Figure 1 shows the association state machine, which is the same for the server and for the peer. (For readability, only the main state transitions are shown. The complete table of transitions can be found in Appendix A.) When the peer initiates the EAP-NOOB method, the server chooses the ensuing message exchange based on the combination of the server and peer states. The EAP server and peer are initially in the Unregistered state, in which no state information needs to be stored. Before a successful Completion Exchange, the server-peer association state is ephemeral in both the server and peer (ephemeral states 0..2), and either endpoint may cause the protocol to fall back to the Initial Exchange. After the Completion Exchange has resulted in EAP-Success, the association

state becomes persistent (persistent states 3..4). Only user reset or memory failure can cause the return of the server or the peer from the persistent states to the ephemeral states and to the Initial Exchange.

The server MUST NOT repeat a successful OOB Step with the same peer except if the association with the peer is explicitly reset by the user or lost due to failure of the persistent storage in the server. More specifically, once the association has entered the Registered state, the server MUST NOT delete the association or go back to states 0..2 without explicit user approval. Similarly, the peer MUST NOT repeat the OOB Step unless the user explicitly deletes the association with the server or resets the peer to the Unregistered state. The server and peer MAY implement user reset of the association by deleting the state data from that endpoint. If an endpoint continues to store data about the association after the user reset, its behavior SHOULD be equivalent to having deleted the association data.

It can happen that the peer accidentally or through user reset loses its persistent state and reconnects to the server without a previously allocated peer identifier. In that case, the server MUST treat the peer as a new peer. The server MAY use auxiliary information, such as the PeerInfo field received in the Initial Exchange, to detect multiple associations with the same peer. However, it MUST NOT delete or merge redundant associations without user or application approval because EAP-NOOB internally has no secure way of verifying that the two peers are the same physical device. Similarly, the server might lose the association state because of a memory failure or user reset. In that case, the only way to recover is that the user resets also the peer.

A special feature of the EAP-NOOB method is that the server is not assumed to have any a-priori knowledge of the peer. Therefore, the peer initially uses the generic identity string "noob@eap-noob.net" as its network access identifier (NAI). The server then allocates a server-specific identifier to the peer. The generic NAI serves two purposes: firstly, it tells the server that the peer supports and expects the EAP-NOOB method and, secondly, it allows routing of the EAP-NOOB sessions to a specific authentication server in the AAA architecture.

EAP-NOOB is an unusual EAP method in that the peer has to have multiple EAP conversations with the server before it can receive EAP-Success. The reason is that, while EAP allows delays between the request-response pairs, e.g. for repeated password entry, the user delays in OOB authentication can be much longer than in password trials. In particular, EAP-NOOB supports also peers with no input

capability in the user interface. Since these output-only devices cannot be told to perform the protocol at the right moment, they have to perform the initial exchange opportunistically and hope for the OOB Step to take place within a timeout period (NoobTimeout), which is why the timeout needs to be several minutes rather than seconds. For example, consider a printer (peer) that outputs the OOB message on paper, which is then scanned for the server. To support such high-latency OOB channels, the peer and server perform the Initial Exchange in one EAP conversation, then allow time for the OOB message to be delivered, and later perform the Waiting and Completion Exchanges in different EAP conversations.

3.2. Protocol messages and sequences

This section defines the EAP-NOOB exchanges, which correspond to EAP conversations. The protocol messages in each exchange and the data members in each message are listed in the diagrams below.

Each EAP-NOOB exchange begins with the authenticator sending an EAP-Request/Identity packet to the peer. From this point on, the EAP conversation occurs between the server and the peer, and the authenticator acts as a pass-through device. The peer responds to the authenticator with an EAP-Response/Identity packet, containing the network access identifier (NAI), which conveys both the peer identity and the current peer state as defined in Section 3.3.1.

After receiving the NAI, the server chooses the EAP-NOOB exchange, i.e. the ensuing message sequence, based on the combination of the peer and server states. The peer recognizes the exchange based on the message type field (Type) of the EAP-NOOB request received from the server. The available exchanges are defined in the following subsections. Each exchange comprises one or more EAP request-response pairs and ends in either EAP-Failure, indicating that authentication is not (yet) successful, or in EAP-Success.

3.2.1. Initial Exchange

Upon receiving the EAP-Response/Identity from the peer, if either the peer or the server is in the Unregistered (0) state and the other is in one of the ephemeral states (0..2), the server chooses the Initial Exchange.

The Initial Exchange comprises two EAP-NOOB request-response pairs, one for version, cryptosuite and parameter negotiation and the other for the ECDHE key exchange. The first EAP-NOOB request (Type=1) from the server contains a newly allocated PeerId for the peer and an optional Realm. The server allocates a new PeerId in the Initial Exchange regardless of any old PeerId in the username part of the

received NAI. The server also sends in the request a list of the protocol versions (Vers) and cryptosuites (Cryptosuites) it supports, an indicator of the OOB channel directions it supports (Dirs), and a ServerInfo object. The peer chooses one of the versions and cryptosuites. The peer sends a response (Type=1) with the selected protocol version (Verp), the received PeerId, the selected cryptosuite (Cryptosuitep), an indicator of the OOB channel directions selected by the peer (Dirp), and a PeerInfo object. In the second EAP-NOOB request and response (Type=2), the server and peer exchange the public components of their ECDHE keys and nonces (PKs, Ns, PKp, Np). The ECDHE keys MUST be based on the negotiated cryptosuite. The Initial Exchange ends with EAP-Failure from the server because the authentication cannot yet be completed.

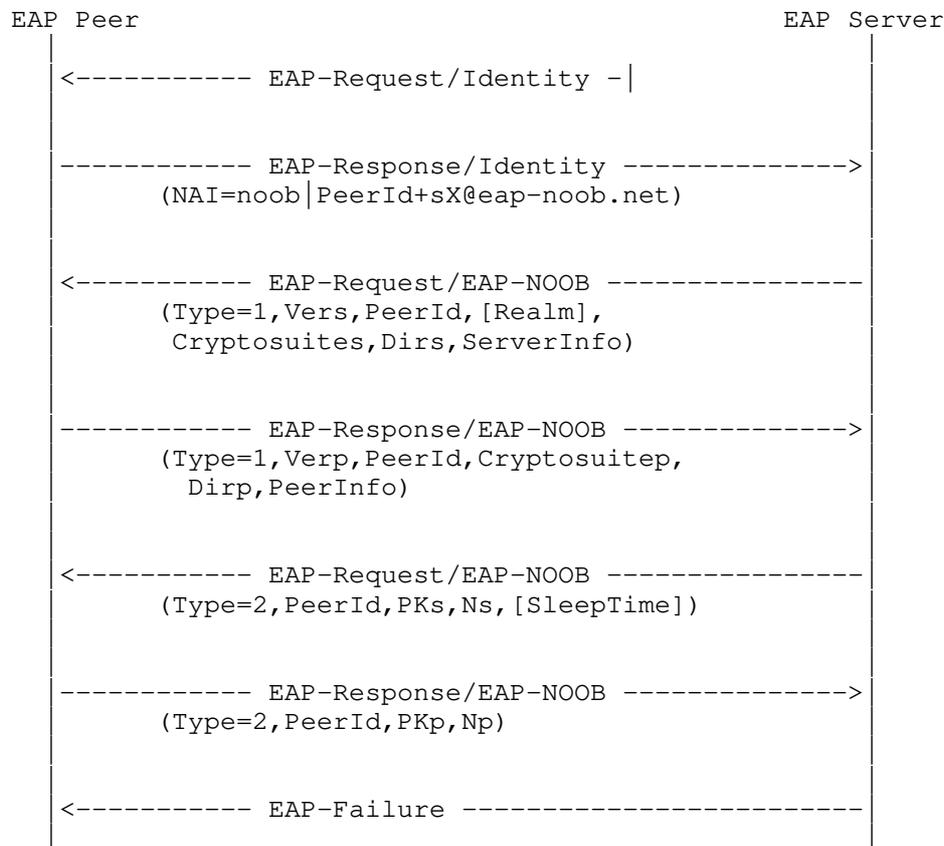


Figure 2: Initial Exchange

At the conclusion of the Initial Exchange, both the server and the peer move to the Waiting for OOB (1) state.

3.2.2. OOB Step

The OOB Step, labeled as OOB Output and OOB Input in Figure 1, takes place after the Initial Exchange. Depending on the direction negotiated, the peer or the server outputs the OOB message shown in Figure 3 or Figure 4. The data fields are the PeerId, the secret nonce Noob, and the cryptographic fingerprint Hoob. The contents of the data fields are defined in Section 3.3.2. The OOB message is delivered to the other endpoint via a user-assisted OOB channel.

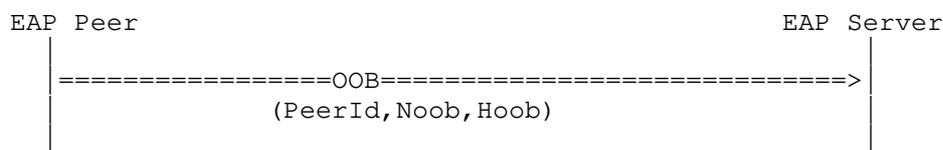


Figure 3: OOB Step, from peer to EAP server

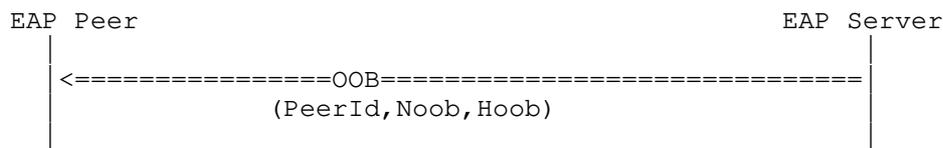


Figure 4: OOB Step, from EAP server to peer

The receiver of the OOB message MUST compare the received value of the fingerprint Hoob with a value that it computes locally. If the values are equal, the receiver moves to the OOB Received (2) state. Otherwise, the receiver MUST reject OOB message. For usability reasons, the receiver SHOULD indicate the acceptance or rejection of the OOB message to the user. The receiver SHOULD reject invalid OOB messages without changing its state, until an application-specific number of invalid messages (OobRetries) has been reached, after which the receiver SHOULD consider it an error and go back to the Unregistered (0) state.

The server or peer MAY send multiple OOB messages with different Noob values while in the Waiting for OOB (1) state. The OOB sender SHOULD remember the Noob values until they expire and accept any one of them in the following Completion Exchange. The Noob values sent by the

server expire after an application-dependent timeout (NoobTimeout), and the server MUST NOT accept Noob values older than that in the Completion Exchange. The RECOMMENDED value for NoobTimeout is 3600 seconds if there are no application-specific reasons for making it shorter or longer. The Noob values sent by the peer expire as defined in Section 3.2.4.

The OOB receiver does not accept further OOB messages after it has accepted one and moved to the OOB Received (2) state. However, the receiver MAY buffer redundant OOB messages in case OOB message expiry or similar error detected in the Completion Exchange causes it to return to the Waiting for OOB (1) state. It is RECOMMENDED that the OOB receiver notifies the user about redundant OOB messages, but it MAY also discard them silently.

The sender will typically generate a new Noob, and therefore a new OOB message, at constant intervals (NoobInterval). The RECOMMENDED interval is $\text{NoobInterval} = \text{NoobTimeout} / 2$, so that the two latest values are always accepted. However, the timing of the Noob generation may also be based on user interaction or on implementation considerations.

Even though not recommended (see Section 3.3), this specification allows both directions to be negotiated (Dirp=3) for the OOB channel. In that case, both sides SHOULD output the OOB message, and it is up to the user to deliver one of them.

The details of the OOB channel implementation including the message encoding are defined by the application. Appendix E gives an example of how the OOB message can be encoded as a URL that may be embedded in a QR code and NFC tag.

3.2.3. Completion Exchange

After the Initial Exchange, if both the server and the peer support the peer-to-server direction for the OOB channel, the peer SHOULD initiate the EAP-NOOB method again after an applications-specific waiting time in order to probe for completion of the OOB Step. Also, if both sides support the server-to-peer direction of the OOB exchange and the peer receives the OOB message, it SHOULD initiate the EAP-NOOB method immediately. Once the server receives the EAP-Response/Identity, if one of the server and peer is in the OOB Received (2) state and the other is either in the Waiting for OOB (1) or OOB Received (2) state, the OOB Step has taken place and the server SHOULD continue with the Completion Exchange.

The Completion Exchange comprises one or two EAP-NOOB request-response pairs. If the peer is in the Waiting for OOB (1) state, the

OOB message has been sent in the peer-to-server direction. In that case, only one request-response pair (Type=4) takes place. In the request, the server sends the NoobId value, which the peer uses to identify the exact OOB message received by the server. On the other hand, if the peer is in the OOB Received (2) state, the direction of the OOB message is from server to peer. In that case, two request-response pairs (Type=8 and Type=4) are needed. With the first request, the server discovers NoobId, which identifies the exact OOB message received by the peer. The server returns the same NoobId to the peer in the latter request.

In the last and sometimes only request-response pair (Type=4) of the Completion Exchange, the server and peer exchange message authentication codes. Both sides MUST compute the keys Kms and Kmp as defined in Section 3.5 and the message authentication codes MACs and MACp as defined in Section 3.3.2. Both sides MUST compare the received message authentication code with a locally computed value. If the peer finds that it has received the correct value of MACs and the server finds that it has received the correct value of MACp, the Completion Exchange ends in EAP-Success. Otherwise, the endpoint where the comparison fails indicates this with an error message (error code 4001, see Section 3.6.1) and the Completion Exchange ends in EAP-Failure.

After successful Completion Exchange, both the server and the peer move to the Registered (4) state. They also derive the output key material and store the persistent association state as defined in Section 3.4 and Section 3.5.

It is possible that the OOB message expires before it is received. In that case, the sender of the OOB message no longer recognizes the NoobId that it receives in the Completion Exchange. Another reason why the OOB sender might not recognize the NoobId is if the received OOB message was spoofed and contained an attacker-generated Noob value. The recipient of an unrecognized NoobId indicates this with an error message (error code 1006, see Section 3.6.1) and the Completion Exchange ends in EAP-Failure. The recipient of the error message 1006 moves back to the Waiting for OOB (1) state. This state transition is shown as OOB Reject in Figure 1 (even though it really is a specific type of failed Completion Exchange). The sender of the error message, on the other hand, stays in its previous state.

Although it is not expected to occur in practice, poor user interface design could lead to two OOB messages delivered simultaneously, one from the peer to the server and the other from the server to the peer. The server detects this event by observing that both the server and peer are in the OOB Received state (2). In that case, as

a tiebreaker, the server MUST behave as if only the server-to-peer message was delivered.

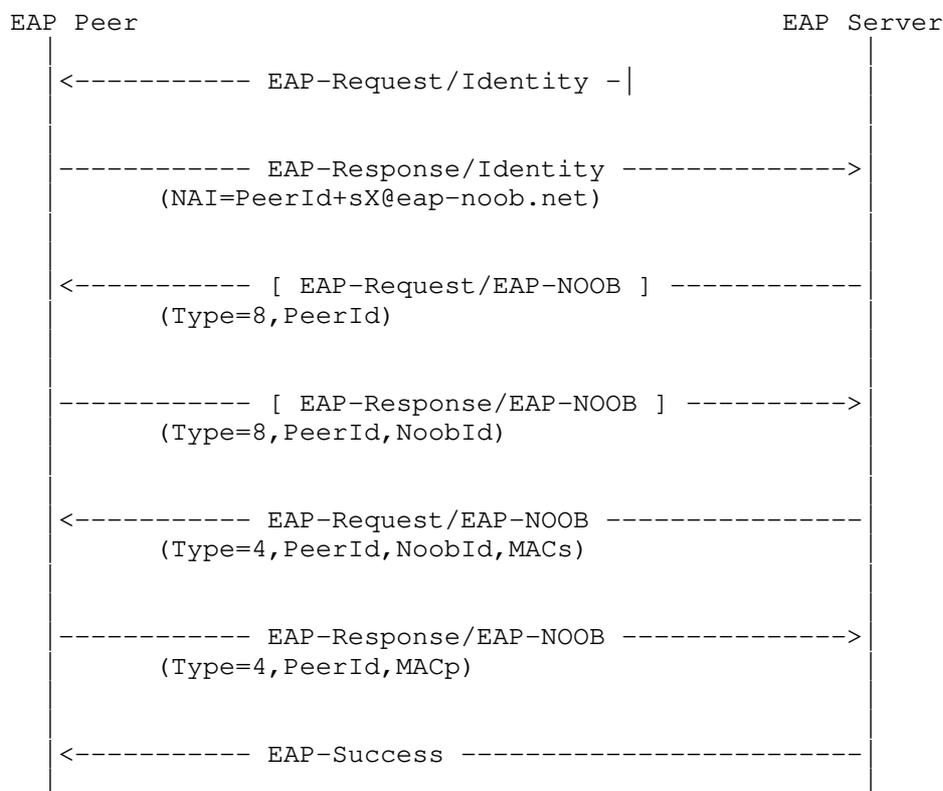


Figure 5: Completion Exchange

3.2.4. Waiting Exchange

As explained in Section 3.2.3, the peer SHOULD probe the server for completion of the OOB Step. If both the server and peer states are Waiting for OOB (1), the server will continue with the Waiting Exchange (Type=3). The main purpose of this exchange is to indicate to the peer that the server has not yet received a peer-to-server OOB message.

In order to limit the rate at which peers probe the server, the server MAY send to the peer either in the Initial Exchange or Waiting Exchange a minimum time to wait before probing the server again. A peer that has not received an OOB message MUST wait at least the

server-specified minimum waiting time in seconds (SleepTime) before initiating EAP again with the same server. The peer uses the latest SleepTime value that it has received in or after the Initial Exchange. If the server has not sent any SleepTime value, the peer SHOULD wait for an application-specified minimum time.

After the Waiting Exchange, the peer MUST discard (from its local ephemeral storage) Noob values that it has sent to the server in OOB messages that are older than the application-defined timeout NoobTimeout (see Section 3.2.2). The peer SHOULD discard such expired Noob values even if the probing failed, e.g. because of failure to connect to the EAP server or incorrect MAC. The timeout of peer-generated Noob values is defined like this in order to allow the peer to probe the server once after it has waited for the server-specified SleepTime.

If the server and peer have negotiated to use only the server-to-peer direction for the OOB channel (Dirp=2), the peer SHOULD nevertheless probe the server. The purpose of this is to keep the server informed about the peers that are still waiting for OOB messages. The server MAY set SleepTime to a high number (3600) to prevent the peer from probing the server frequently.

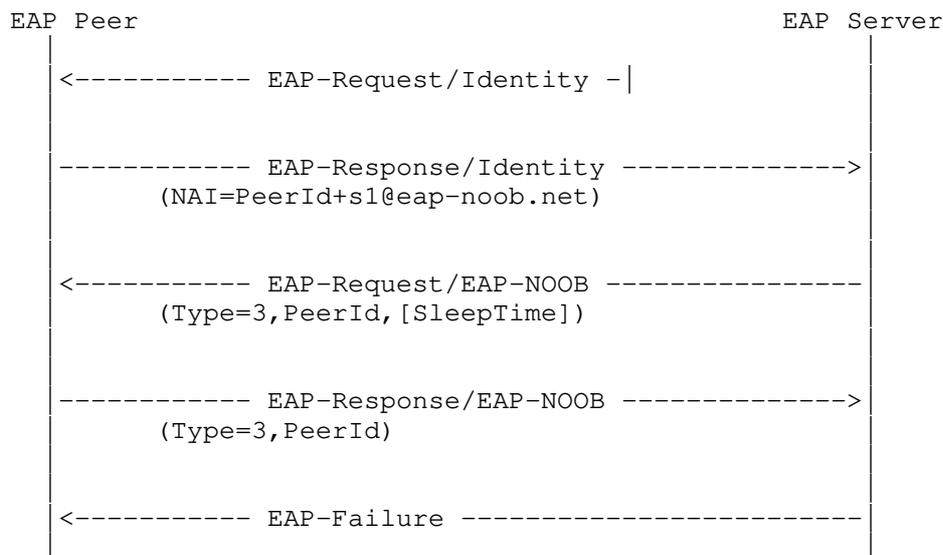


Figure 6: Waiting Exchange

3.3. Protocol data fields

This section defines the various identifiers and data fields used in the EAP-NOOB protocol.

3.3.1. Peer identifier, realm and NAI

The server allocates a new peer identifier (PeerId) for the peer in the Initial Exchange. The peer identifier MUST follow the syntax of the utf8-username specified in [RFC7542]; however, it MUST NOT exceed 60 bytes in length and MUST NOT contain the character '+'. The server MUST generate the identifiers in such a way that they do not repeat and cannot be guessed by the peer or third parties beforehand. One way to generate the identifiers is to choose a random 16-byte identifier and to base64url encode it without padding [RFC4648] into a 22-character string. Another way to generate the identifiers is to choose a random 22-character alphanumeric string. It should be noted that longer the peer identifiers result in longer OOB messages.

When the peer is in one of the states 1..2, the peer MUST use the PeerId that the server assigned to it in the latest Initial Exchange. When the peer is in one of the persistent states 3..4, it MUST use the PeerId from its persistent EAP-NOOB association. When the peer is in the Unregistered (0) state, it MUST use the default value "noob" as its PeerId.

The server MAY also assign a realm (Realm) to the peer by in Initial Exchange or Reconnect Exchange. The realm value MUST follow the syntax of the utf8-realm specified in [RFC7542]. When the peer is in one of the states 1..2, the peer MUST use the Realm that the server assigned to it the latest Initial Exchange, if one was assigned. When the peer is in one of the persistent states 3..4, it MUST use the Realm from its persistent EAP-NOOB association, if one is stored in the association. When the peer is in the Unregistered (0) state, or when the peer is in one of the other states 1..4 and the server has not assigned it a realm, the peer SHOULD use the default realm "eap-noob.net". However, the user or application MAY provide a different default realm to the peer.

To compose its NAI [RFC7542], the peer uses the PeerId in the user part and the Realm as the realm part. When no server-assigned PeerId or Realm is available, the default value is used instead. In the Unregistered (0) state, the peer must create the NAI as the concatenation of the PeerId, the symbol "@", and the Realm. In the other states 1..4, the peer MUST create the NAI as the concatenation of the PeerId, the string "+s", a single digit indicating the state of the peer, and the Realm.

Note that a new peer typically sends the NAI "noob@eap-noob.net" in its first EAP-Response/Identity message, and this default NAI is always acceptable when the peer is in the Unregistered (0) state. The purpose of the state indicator "+sX" is to inform the server about the peer state without the cost of an additional round trip. The server uses this information together with the server state to decide on the type of the exchange and, thus, the type of the next EAP-Request. The lack of a state indicator means that that peer is in state 0.

The purpose of the server-assigned realm is to enable more flexible routing of the EAP sessions over the AAA infrastructure, including roaming scenarios (see Appendix D). The possibility to configure a different default realm further enables registration of new devices while roaming. It also enables manufacturers to set up their own AAA servers for bootstrapping of new peer devices. Moreover, some Authenticators or AAA servers use the assigned Realm to determine peer-specific connection parameters, such as isolating the peer to a specific VLAN.

The peer's PeerId and Realm are ephemeral until a successful Completion Exchange takes place. Thereafter, the values become parts of the persistent EAP-NOOB association, until the user resets the peer and the server or until a new Realm is assigned in the Reconnect Exchange. Resetting the server means deleting the association for the peer from the server database.

3.3.2. Message data fields

Table 1 defines the data fields in the protocol messages. The in-band messages are formatted as JSON objects [RFC8259] in UTF-8 encoding. The JSON member names are in the left-hand column of the table.

Data field	Description
Vers,Verp	EAP-NOOB protocol versions supported by the EAP server, and the protocol version chosen by the peer. Vers is a JSON array of unsigned integers, and Verp is an unsigned integer. Currently, the only defined values are "[1]" and "1", respectively.
PeerId	Peer identifier as defined in Section 3.3.1.
Realm	Peer identifier as defined in Section 3.3.1.

Peer State "+sX"	Peer state indicator as defined in Section 3.3.1.
Type	EAP-NOOB message type. The type is an integer in the range 0..8. EAP-NOOB requests and the corresponding responses share the same type value.
PKs, PKp	The public components of the ECDHE keys of the server and peer. PKs and PKp are sent in the JSON Web Key (JWK) format [RFC7517]. Detailed format of the JWK object is defined by the cryptosuite.
Cryptosuites, Cryptosuitep	The identifiers of cryptosuites supported by the server and of the cryptosuite selected by the peer. The server-supported cryptosuites in Cryptosuites are formatted as a JSON array of the identifier integers. The server MUST send an nonempty array with no repeating elements, ordered by decreasing priority. The peer MUST respond with exactly one suite in the Cryptosuitep value, formatted as an identifier integer. The registration of cryptosuites is specified in Section 4.1.
Dirs, Dirp	The OOB channel directions supported by the server and ones selected by the peer. The possible values are 1=peer-to-server, 2=server-to-peer, 3=both directions. Endpoints that are general-purpose computers or online services SHOULD support both directions. IoT devices with a limited user interface will mostly support only one direction. If the negotiated value is 3, the user may be presented with two OOB messages, one for each direction, even though the user needs to deliver only one of them. Since this can be confusing to the user, it is RECOMMENDED that the peer selects Dirp value 1 or 2. The EAP-NOOB protocol itself is designed to cope also with selected value 3, in which case it uses the first delivered OOB message. In the unlikely case of simultaneously delivered OOB messages, the protocol prioritizes the server-to-peer direction.
Dir	The actual direction of the OOB message (1

	=peer-to-server, 2=server-to-peer). This value is not sent over any communication channel but it is included in the computation of the cryptographic fingerprint Hoob.
Ns, Np	32-byte nonces for the Initial Exchange.
ServerInfo	This field contains information about the server to be passed from the EAP method to the application layer in the peer. The information is specific to the application or to the OOB channel and it is encoded as a JSON object of at most 500 bytes. It could include, for example, the access-network name and server name or a Uniform Resource Locator (URL) [RFC4266] or some other information that helps the user to deliver the OOB message to the server through the out-of-band channel.
PeerInfo	This field contains information about the peer to be passed from the EAP method to the application layer in the server. The information is specific to the application or to the OOB channel and it is encoded as a JSON object of at most 500 bytes. It could include, for example, the peer brand, model and serial number, which help the user to distinguish between devices and to deliver the OOB message to the correct peer through the out-of-band channel.
SleepTime	The number of seconds for which peer MUST NOT start a new execution of the EAP-NOOB method with the authenticator, unless the peer receives the OOB message or the peer is reset by the user. The server can use this field to limit the rate at which peers probe it. SleepTime is an unsigned integer in the range 0..3600.
Noob	16-byte secret nonce sent through the OOB channel and used for the session key derivation. The endpoint that received the OOB message uses this secret in the Completion Exchange to authenticate the exchanged key to the endpoint that sent the OOB message.
Hoob	16-byte cryptographic fingerprint (i.e. hash

	value) computed from all the parameters exchanged in the Initial Exchange and in the OOB message. Receiving this fingerprint over the OOB channel guarantees the integrity of the key exchange and parameter negotiation. Hence, it authenticates the exchanged key to the endpoint that receives the OOB message.
NoobId	16-byte identifier for the OOB message, computed with a one-way function from the nonce Noob.
MACs, MACp	Message authentication codes for mutual authentication, key confirmation, and integrity check on the exchanged information. The input to the HMAC is defined below, and the key for the HMAC is defined in Section 3.5.
Ns2, Np2	32-byte Nonces for the Reconnect Exchange.
KeyingMode	Integer indicating the key derivation method. 0 in the Completion Exchange, and 1..3 in the Reconnect Exchange.
PKs2, PKp2	The public components of the ECDHE keys of the server and peer for the Reconnect Exchange. PKp2 and PKs2 are sent in the JSON Web Key (JWK) format [RFC7517]. Detailed format of the JWK object is defined by the cryptosuite.
MACs2, MACp2	Message authentication codes for mutual authentication, key confirmation, and integrity check on the Reconnect Exchange. The input to the HMAC is defined below, and the key for the HMAC is defined in Section 3.5.

Table 1: Message data fields

The nonces in the in-band messages (N_s , N_p , N_{s2} , N_{p2}) are 32-byte fresh random byte strings, and the secret nonce $Noob$ is a 16-byte fresh random byte string. All the nonces are generated by the endpoint that sends the message.

The fingerprint $Hoob$ and the identifier $NoobId$ are computed with the hash function specified in the negotiated cryptosuite and truncated

to the 16 leftmost bytes of the output. The message authentication codes (MACs, MACp, MACs2, MACp2) are computed with the HMAC function [RFC2104] based on the same cryptographic hash function and truncated to the 32 leftmost bytes of the output.

The inputs to the hash function for computing the fingerprint Hoob and to the HMAC for computing MACs, MACp, MACs2 and MACp2 are JSON arrays containing a fixed number (16) of elements. The array elements MUST be copied to the array verbatim from the sent and received in-band messages. When the element is a JSON object, its members MUST NOT be reordered or re-encoded, and whitespace MUST NOT be added anywhere in the JSON structure. Implementers should check that their JSON library copies the elements as UTF-8 strings and does not modify them in any way.

The inputs for computing the fingerprint and message authentication codes are the following:

```
Hoob = H(Dir, Vers, Verp, PeerId, Cryptosuites, Dirs, ServerInfo, Cryptosuitep, Dirp, [Realm], PeerInfo, 0, PKs, Ns, PKp, Np, Noob) .
```

```
NoobId = H("NoobId", Noob) .
```

```
MACs = HMAC(Kms; 2, Vers, Verp, PeerId, Cryptosuites, Dirs, ServerInfo, Cryptosuitep, Dirp, [Realm], PeerInfo, 0, PKs, Ns, PKp, Np, Noob) .
```

```
MACp = HMAC(Kmp; 1, Vers, Verp, PeerId, Cryptosuites, Dirs, ServerInfo, Cryptosuitep, Dirp, [Realm], PeerInfo, 0, PKs, Ns, PKp, Np, Noob) .
```

```
MACs2 = HMAC(Kms2; 2, Vers, Verp, PeerId, Cryptosuites, "", [ServerInfo], Cryptosuitep, "", [Realm], [PeerInfo], KeyingMode, [PKs2], Ns2, [PKp2], Np2, "")
```

```
MACp2 = HMAC(Kmp2; 1, Vers, Verp, PeerId, Cryptosuites, "", [ServerInfo], Cryptosuitep, "", [Realm], [PeerInfo], KeyingMode, [PKs2], Ns2, [PKp2], Np2, "")
```

Missing input values are represented by empty strings "" in the array. The values indicated with "" above are always empty strings. Realm is included in the computation of Macs and Macp if it was sent or received in the preceding Initial Exchange. Each of the values in brackets for the computation of Macs2 and Macp2 MUST be included if it was sent or received in the same Reconnect Exchange; otherwise the value is replaced by an empty string "".

The parameter Dir indicates the direction in which the OOB message containing the Noob value is being sent (1=peer-to-server, 2=server-to-peer). This field is needed to prevent the user from accidentally

delivering the OOB message back to its originator in the rare cases where both OOB directions have been negotiated. The keys for the HMACs are defined in Section 3.5.

The nonces (N_s , N_p , N_{s2} , N_{p2} , $Noob$), hash value ($NoobId$) and message authentication codes ($MACs$, MAC_p , MAC_{s2} , MAC_{p2}) in the in-band messages and when used as inputs to the hash and message authentication code MUST be base64url encoded [RFC4648]. The values $Noob$ and $Hoob$ in the OOB channel MAY also be base64url encoded, if that is appropriate for the application and the OOB channel. All base64url encoding is done without padding. The base64url encoded values will naturally consume more space than the number of bytes specified above (22-character string for a 16-byte nonce and 43-character string for a 32-byte nonce or message authentication code). In the key derivation in Section 3.5, on the other hand, the unencoded nonces (raw bytes) are used as input to the key derivation function.

The $ServerInfo$ and $PeerInfo$ are JSON objects with UTF-8 encoding. The length of either encoded object as a byte array MUST NOT exceed 500 bytes. The format and semantics of these objects MUST be defined by the application that uses the EAP-NOOB method.

3.4. Fast reconnect and rekeying

EAP-NOOB implements Fast Reconnect ([RFC3748], section 7.2.1) that avoids repeated use of the user-assisted OOB channel.

The rekeying and the Reconnect Exchange may be needed for several reasons. New EAP output values MSK and EMSK may be needed because of mobility or timeout of session keys. Software or hardware failure or user action may also cause the authenticator, EAP server or peer to lose its non-persistent state data. The failure would typically be detected by the peer or authenticator when session keys no longer are accepted by the other endpoint. Change in the supported cryptosuites in the EAP server or peer may also cause the need for a new key exchange. When the EAP server or peer detects any one of these events, it MUST change from the Registered to Reconnecting state. These state transitions are labeled Mobility/Timeout/Failure in Figure 1. The EAP-NOOB method will then perform the Reconnect Exchange next time when EAP is triggered.

3.4.1. Persistent EAP-NOOB association

To enable rekeying, the EAP server and peer store the session state in persistent memory after a successful Completion Exchange. This state data, called "persistent EAP-NOOB association", MUST include at least the data fields shown in Table 2. They are used for

identifying and authenticating the peer in the Reconnect Exchange. When a persistent EAP-NOOB association exists, the EAP server and peer are in the Registered state (4) or Reconnecting state (3), as shown in Figure 1.

Data field	Value	Type
PeerId	Peer identifier allocated by server	UTF-8 string (typically 22 bytes)
Verp	Negotiated protocol version	integer
Cryptosuitep	Negotiated cryptosuite	integer
CryptosuitepPrev (peer only)	Previous cryptosuite	integer
Realm	Optional realm assigned by server (default value is "eap-noob.net")	UTF-8 string
Kz	Persistent key material	32 bytes
KzPrev (peer only)	Previous Kz value	32 bytes

Table 2: Persistent EAP-NOOB association

3.4.2. Reconnect Exchange

The server chooses the Reconnect Exchange when peer is in the Reconnecting (3) state and the server itself is in the Registered (4) or Reconnecting (3) state. The peer MUST NOT initiate EAP-NOOB when the peer is in Registered state.

The Reconnect Exchange comprises three EAP-NOOB request-response pairs, one for cryptosuite and parameter negotiation, another for the nonce and ECDHE key exchange, and the last one for exchanging message authentication codes. In the first request and response (Type=5) the server and peer negotiate a protocol version and cryptosuite in the same way as in the Initial Exchange. The server SHOULD NOT offer and the peer MUST NOT accept protocol versions or cryptosuites that it knows to be weaker than the one currently in the Cryptosuitep field of the persistent EAP-NOOB association. The server SHOULD NOT needlessly change the cryptosuites it offers to the same peer because

peer devices may have limited ability to update their persistent storage. However, if the peer has different values in the `Cryptosuitep` and `CryptosuitepPrev` fields, it SHOULD also accept offers that are not weaker than `CryptosuitepPrev`. Note that `Cryptosuitep` and `CryptosuitepPrev` from the persistent EAP-NOOB association are only used to support the negotiation as described above; all actual cryptographic operations use the negotiated cryptosuite. The request and response (Type=5) MAY additionally contain `PeerInfo` and `ServerInfo` objects.

The server then determines the `KeyingMode` (defined in Section 3.5) based on changes in the negotiated cryptosuite and whether it desires to achieve forward secrecy or not. The server SHOULD only select `KeyingMode` 3 when the negotiated cryptosuite differs from the `Cryptosuitep` in the server's persistent EAP-NOOB association, although it is technically possible to select this values without changing the cryptosuite. In the second request and response (Type=6), the server informs the peer about the `KeyingMode`, and the server and peer exchange nonces (`Ns2`, `Np2`). When `KeyingMode` is 2 or 3 (rekeying with ECDHE), they also exchange public components of ECDHE keys (`PKs2`, `PKp2`). The server ECDHE key MUST be fresh, i.e. not previously used with the same peer, and the peer ECDHE key SHOULD be fresh, i.e. not previously used.

In the third and final request and response (Type=7), the server and peer exchange message authentication codes. Both sides MUST compute the keys `Kms2` and `Kmp2` as defined in Section 3.5 and the message authentication codes `MACs2` and `MACp2` as defined in Section 3.3.2. Both sides MUST compare the received message authentication code with a locally computed value.

The rules by which the peer compares the received `MACs2` are non-trivial because, in addition to authenticating the current exchange, `MACs2` may confirm the success or failure of a recent cryptosuite upgrade. The peer processes the final request (Type=7) as follows:

1. The peer first compares the received `MACs2` value with one it computed using the `Kz` stored in the persistent EAP-NOOB association. If the received and computed values match, the peer deletes any data stored in the `CryptosuitepPrev` and `KzPrev` fields of the persistent EAP-NOOB association. It does this because the received `MACs2` confirms that the peer and server share the same `Cryptosuitep` and `Kz`, and any previous values must no longer be accepted.
2. If, on the other hand, the peer finds that the received `MACs2` value does not match the one it computed locally with `Kz`, the peer checks whether the `KzPrev` field in the persistent EAP-NOOB

association stores a key. If it does, the peer repeats the key derivation (Section 3.5) and local MACs2 computation (Section 3.3.2) using KzPrev in place of Kz. If this second computed MACs2 matches the received value, it indicates synchronization failure caused by the loss of the last response (Type=7) in a previously attempted cryptosuite upgrade. In this case, the peer rolls back that upgrade by overwriting Cryptosuitep with CryptosuitepPrev and Kz with KzPrev in the persistent EAP-NOOB association. It also clears the CryptosuitepPrev and KzPrev fields.

3. If the received MACs2 matched one of the locally computed values, the peer proceeds to send the final response (Type=7). The peer also moves to the Registered (4) state. When KeyingMode is 1 or 2, the peer stops here. When KeyingMode is 3, the peer also updates the persistent EAP-NOOB association with the negotiated Cryptosuitep and the newly-derived Kz value. To prepare for possible synchronization failure caused by the loss of the final response (Type=7) during cryptosuite upgrade, the peer copies the old Cryptosuitep and Kz values in the persistent EAP-NOOB association to the CryptosuitepPrev and KzPrev fields.
4. Finally, if the peer finds that the received MACs2 does not match either of the two values that it computed locally (or one if no KzPrev was stored), the peer sends an error message (error code 4001, see Section 3.6.1), which causes the the Reconnect Exchange to end in EAP-Failure.

The server rules for processing the final message are simpler than the peer rules because the server does not store previous keys and it never rolls back a cryptosuite upgrade. Upon receiving the final response (Type=7), the server compares the received value of MACp2 with one it computes locally. If the values match, the Reconnect Exchange ends in EAP-Success. When KeyingMode is 3, the server also updates Cryptosuitep and Kz in the persistent EAP-NOOB association. On the other hand, if the server finds that the values do not match, it sends an error message (error code 4001), and the Reconnect Exchange ends in EAP-Failure.

The endpoints MAY send updated Realm, ServerInfo and PeerInfo objects in the Reconnect Exchange. When there is no update to the values, they SHOULD omit this information from the messages. If the Realm was sent, each side updates Realm in the persistent EAP-NOOB association when moving to the Registered (4) state.

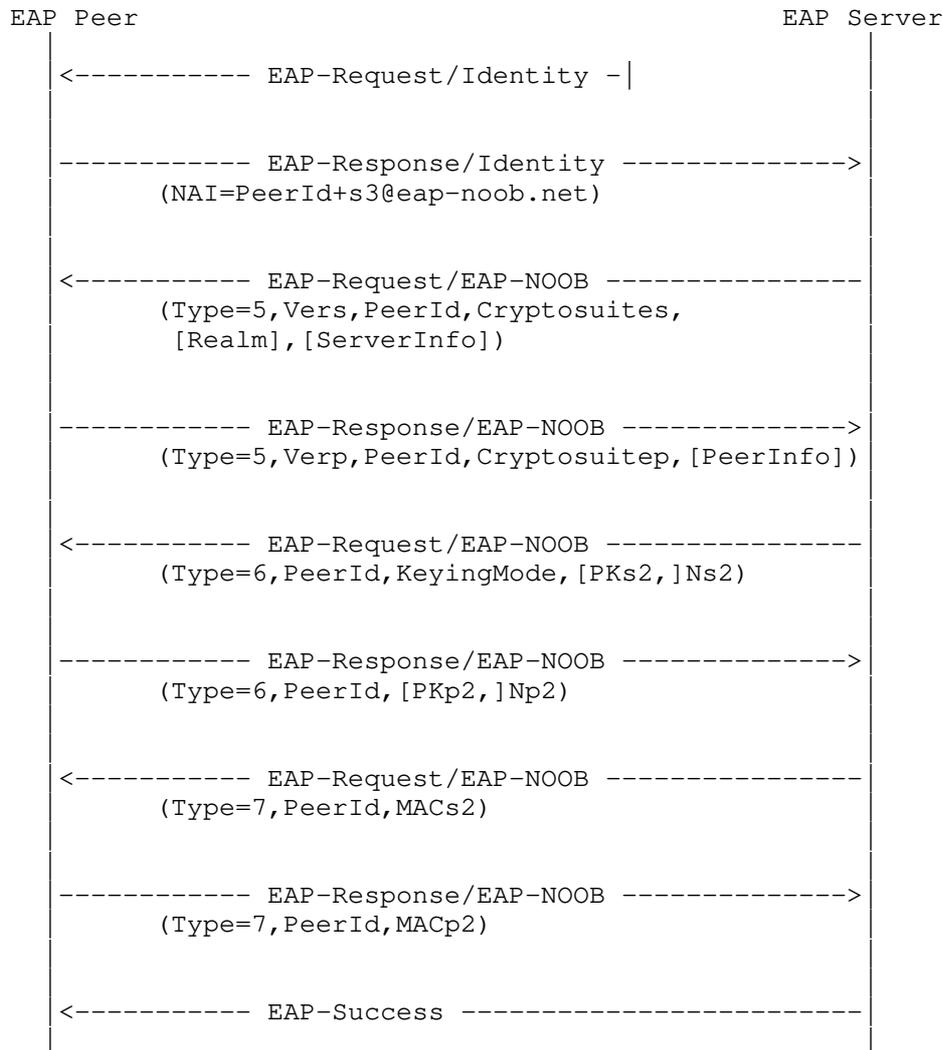


Figure 7: Reconnect Exchange

3.4.3. User reset

As shown in the association state machine in Figure 1, the only specified way for the association to return from the Registered state (4) to the Unregistered state (0) is through user-initiated reset. After the reset, a new OOB message will be needed to establish a new association between the EAP server and peer. Typical situations in which the user reset is required are when the other side has

accidentally lost the persistent EAP-NOOB association data, or when the peer device is decommissioned.

The server could detect that the peer is in the Registered or Reconnecting state but the server itself is in one of the ephemeral states 0..2 (including situations where the server does not recognize the PeerId). In this case, effort should be made to recover the persistent server state, for example, from a backup storage - especially if many peer devices are similarly affected. If that is not possible, the EAP server SHOULD log the error or notify an administrator. The only way to continue from such a situation is by having the user reset the peer device.

On the other hand, if the peer is in any of the ephemeral states 0..2, including the Unregistered state, the server will treat the peer as a new peer device and allocate a new PeerId to it. The PeerInfo can be used by the administrator as a clue to which physical device has lost its state. However, there is no secure way of matching the "new" peer with the old PeerId without repeating the OOB Step. This situation will be resolved when the user performs the OOB Step and, thus, identifies the physical peer device. The server user interface SHOULD support situations where the "new" peer is actually a previously registered peer that has been reset by a user or has otherwise lost the persistent EAP-NOOB association data and needs to be merged with the old peer data in the server.

3.5. Key derivation

EAP-NOOB derives the EAP output values MSK and EMSK and other secret keying material from the output of an Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) algorithm following the NIST specification [NIST-DH]. In NIST terminology, we use a $C(2, 0, \text{ECC CDH})$ scheme, i.e. two ephemeral keys and no static keys. In the Initial and Reconnect Exchanges, the server and peer compute the ECDHE shared secret Z as defined in section 6.1.2.2 of the NIST specification [NIST-DH]. In the Completion and Reconnect Exchanges, the server and peer compute the secret keying material from Z with the single-step key derivation function (KDF) defined in section 5.8.1 of the NIST specification. The hash function H for KDF is taken from the negotiated cryptosuite.

KeyingMode	Description
0	Completion Exchange (always with ECDHE)
1	Reconnect Exchange, rekeying without ECDHE
2	Reconnect Exchange, rekeying with ECHDE, no change in cryptosuite
3	Reconnect Exchange, rekeying with ECDHE, new cryptosuite negotiated

Table 3: Keying modes

The key derivation has three different modes (`KeyingMode`), which are specified in Table 3. Table 4 defines the inputs to KDF in each `KeyingMode`.

In the Completion Exchange (`KeyingMode=0`), the input `Z` comes from the preceding Initial exchange. KDF takes some additional inputs (`OtherInfo`), for which we use the concatenation format defined in section 5.8.1.2.1 of the NIST specification [NIST-DH]. `OtherInfo` consists of the `AlgorithmId`, `PartyUInfo`, `PartyVInfo`, and `SuppPrivInfo` fields. The first three fields are fixed-length bit strings, and `SuppPrivInfo` is a variable-length string with a one-byte `Datalength` counter. `AlgorithmId` is the fixed-length 8-byte ASCII string "EAP-NOOB". The other input values are the server's and peer's nonces. In the Completion Exchange, the inputs also include the secret nonce `Noob` from the OOB message.

In the simplest form of the Reconnect Exchange (`KeyingMode=1`), fresh nonces are exchanged but no ECDHE keys are sent. In this case, input `Z` to the KDF is replaced with the shared key `Kz` from the persistent EAP-NOOB association. The result is rekeying without the computational cost of the ECDHE exchange, but also without forward secrecy.

When forward secrecy is desired in the Reconnect Exchange (`KeyingMode=2` or `KeyingMode=3`), both nonces and ECDHE keys are exchanged. Input `Z` is the fresh shared secret from the ECDHE exchange with `PKs2` and `PKp2`. The inputs also include the shared secret `Kz` from the persistent EAP-NOOB association.

KeyingMode	KDF input field	Value	Length (bytes)
0 Completion	Z	ECDHE shared secret from PKs and PKp	variable
	AlgorithmId	"EAP-NOOB"	8
	PartyUInfo	Np	32
	PartyVInfo	Ns	32
	SuppPubInfo SuppPrivInfo	(not allowed) Noob	16
1 Reconnect, rekeying without ECDHE	Z	Kz	32
	AlgorithmId	"EAP-NOOB"	8
	PartyUInfo	Np2	32
	PartyVInfo	Ns2	32
	SuppPubInfo SuppPrivInfo	(not allowed) (null)	0
2 or 3 Reconnect, rekeying, with ECDHE, same or new cryptosuite	Z	ECDHE shared secret from PKs2 and PKp2	variable
	AlgorithmId	"EAP-NOOB"	8
	PartyUInfo	Np2	32
	PartyVInfo	Ns2	32
	SuppPubInfo SuppPrivInfo	(not allowed) Kz	32

Table 4: Key derivation input

Table 5 defines how the output bytes of KDF are used. In addition to the EAP output values MSK and EMSK, the server and peer derive another shared secret key AMSK, which MAY be used for application-layer security. Further output bytes are used internally by EAP-NOOB for the message authentication keys (Kms, Kmp, Kms2, Kmp2).

The Completion Exchange (KeyingMode=0) produces the shared secret Kz, which the server and peer store in the persistent EAP-NOOB association. When a new cryptosuite is negotiated in the Reconnect Exchange (KeyingMode=3), it similarly produces a new Kz. In that case, the server and peer update both the cryptosuite and Kz in the persistent EAP-NOOB association. Additionally, the peer stores the previous cryptosuite and Kz values in the CryptosuitePrev and KzPrev fields of the persistent EAP-NOOB association.

KeyingMode	KDF output bytes	Used as	Length (bytes)
0 Completion	0..63	MSK	64
	64..127	EMSK	64
	128..191	AMSK	64
	192..223	MethodId	32
	224..255	Kms	32
	256..287	Kmp	32
	288..319	Kz	32
1 or 2 Reconnect, rekeying without ECDHE, or with ECDHE and unchanged cryptosuite	0..63	MSK	64
	64..127	EMSK	64
	128..191	AMSK	64
	192..223	MethodId	32
	224..255	Kms2	32
	256..287	Kmp2	32
3 Reconnect, rekeying with ECDHE, new cryptosuite	0..63	MSK	64
	64..127	EMSK	64
	128..191	AMSK	64
	192..223	MethodId	32
	224..255	Kms2	32
	256..287	Kmp2	32
288..319	Kz	32	

Table 5: Key derivation output

Finally, every EAP method must export a Server-Id, Peer-Id and Session-Id [RFC5247]. In EAP-NOOB, the exported Peer-Id is the PeerId which the server has assigned to the peer. The exported Server-Id is a zero-length string (i.e. null string) because EAP-NOOB neither knows nor assigns any server identifier. The exported Session-Id is created by concatenating the Type-Code xxx (TBA) with the MethodId, which is obtained from the KDF output as shown in Table 5.

3.6. Error handling

Various error conditions in EAP-NOOB are handled by sending an error notification message (Type=0) instead of the expected next EAP request or response message. Both the EAP server and the peer may send the error notification, as shown in Figure 8 and Figure 9. After sending or receiving an error notification, the server MUST send an EAP-Failure (as required by [RFC3748] section 4.2). The

notification MAY contain an ErrorInfo field, which is a UTF-8 encoded text string with a maximum length of 500 bytes. It is used for sending descriptive information about the error for logging and debugging purposes.

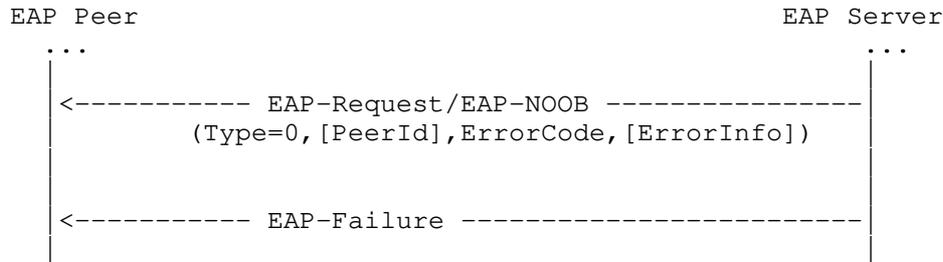


Figure 8: Error notification from server to peer

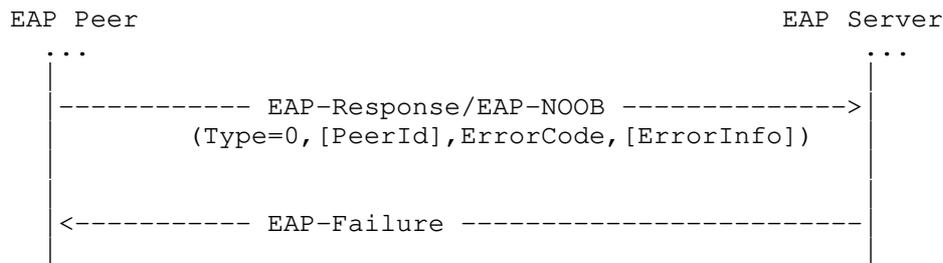


Figure 9: Error notification from peer to server

After the exchange fails due to an error notification, the server and peer set the association state as follows. In the Initial Exchange, both the sender and recipient of the error notification MUST set the association state to the Unregistered (0) state. In the Waiting and Completion Exchanges, each side MUST remain in its old state as if the failed exchange did not take place, with the exception that the recipient of error code 1006 processes it as specified in Section 3.2.3. In the Reconnect Exchange, both sides MUST set the association state to the Reconnecting (3) state.

Errors that occur in the OOB channel are not explicitly notified in-band.

3.6.1. Invalid messages

If the NAI structure is invalid, the server SHOULD send the error code 1001 to the peer. The recipient of an EAP-NOOB request or response SHOULD send the following error codes back to the sender: 1002 if it cannot parse the message as a JSON object or the top-level JSON object has missing or unrecognized members; 1003 if a data field has an invalid value, such as an integer out of range, and there is no more specific error code available; 1004 if the received message type was unexpected; 1005 if the PeerId has an unexpected value; 1006 if the NoobId is not recognized; and 1007 if the ECDHE key is invalid.

3.6.2. Unwanted peer

The preferred way for the EAP server to rate limit EAP-NOOB connections from a peer is to use the SleepTime parameter in the Waiting Exchange. However, if the EAP server receives repeated EAP-NOOB connections from a peer which apparently should not connect to this server, the server MAY indicate that the connections are unwanted by sending the error code 2001. After receiving this error message, the peer MAY refrain from reconnecting to the same EAP server and, if possible, both the EAP server and peer SHOULD indicate this error condition to the user. However, in order to avoid persistent denial of service, the peer is not required to stop entirely from reconnecting to the server.

3.6.3. State mismatch

In the states indicated by "-" in Figure 10 in Appendix A, user action is required to reset the association state or to recover it, for example, from backup storage. In those cases, the server sends the error code 2002 to the peer. If possible, both the EAP server and peer SHOULD indicate this error condition to the user.

3.6.4. Negotiation failure

If there is no matching protocol version, the peer sends the error code 3001 to the server. If there is no matching cryptosuite, the peer sends the error code 3002 to the server. If there is no matching OOB direction, the peer sends the error code 3003 to the server.

In practice, there is no way of recovering from these errors without software or hardware changes. If possible, both the EAP server and peer SHOULD indicate these error conditions to the user. In particular, user action is needed for resetting the association from a persistent state to the Unregistered (0) state.

3.6.5. Cryptographic verification failure

If the receiver of the OOB message detects an unrecognized PeerId or incorrect fingerprint (Hoob) in the OOB message, the receiver MUST remain in the Waiting for OOB state (1) as if no OOB message was received. The receiver SHOULD indicate the failure to accept the OOB message to the user.

Note that if the OOB message was delivered from the server to the peer and the peer does not recognize the PeerId, the likely cause is that the user has unintentionally delivered the OOB message to the wrong destination. If possible, the peer SHOULD indicate this to the user; however, the peer device may not have the capability for many different error indications to the user and it MAY use the same indication as in the case of an incorrect fingerprint.

The rationale for the above is that the invalid OOB message could have been presented to the receiver by mistake or intentionally by a malicious party and, thus, it should be ignored in the hope that the honest user will soon deliver a correct OOB message.

If the EAP server or peer detects an incorrect message authentication code (MACs, MACp, MACs2, MACp2), it sends the error code 4001 to the other side. As specified in the beginning of Section 3.6, the failed Completion Exchange will not result in server or peer state changes while error in the Reconnect Exchange will put both sides to the Reconnecting (3) state and thus lead to another reconnect attempt.

The rationale for this is that the invalid cryptographic message may have been spoofed by a malicious party and, thus, it should be ignored. In particular, a spoofed message on the in-band channel should not force the honest user to perform the OOB Step again. In practice, however, the error may be caused by other failures, such as a software bug. For this reason, the EAP server MAY limit the rate of peer connections with SleepTime after the above error. Also, there MUST be a way for the user to reset the EAP server and peer to the Unregistered state (0), so that the OOB Step can be repeated.

3.6.6. Application-specific failure

Applications MAY define new error messages for failures that are specific to the application or to one type of OOB channel. They MAY also use the generic application-specific error code 5001, or the error codes 5002 and 5003, which have been reserved for indicating invalid data in the ServerInfo and PeerInfo fields, respectively. Additionally, anticipating OOB channels that make use of a URL, the error code 5003 has been reserved for indicating invalid server URL.

4. IANA Considerations

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

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

This memo also requires IANA to create new registries as defined in the following subsections.

4.1. Cryptosuites

Cryptosuites are identified by an integer. Each cryptosuite MUST specify an ECDHE curve for the key exchange, encoding of the ECDHE public key as a JWK object, and a cryptographic hash function for the fingerprint and MAC computation and key derivation. The hash value output by the cryptographic hash function MUST be at least 32 bytes in length. The following suites are defined by EAP-NOOB:

Cryptosuite	Algorithms
1	ECDHE curve Curve25519 [RFC7748], public-key format [RFC7518] Section 6.2.1, hash function SHA-256 [RFC6234]

Table 6: EAP-NOOB cryptosuites

An example of Cryptosuite 1 public-key encoded as a JWK object is given below (line breaks are for readability only).

```
"jwk":{"kty":"EC","crv":"Curve25519","x":"3p7bfXt9wbTTW2HC7OQ1Nz-DQ8hbeGdNrfx-FG-IK08"}
```

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

4.2. Message Types

EAP-NOOB Request and Response pairs are identified by an integer Message Type. The following Message Types are defined by EAP-NOOB:

Message Type	Used in Exchange	Purpose
1	Initial	Version, cryptosuite and parameter negotiation
2	Initial	Exchange of ECDHE keys and nonces
3	Waiting	Indication to peer that the server has not yet received an OOB message
4	Completion	Authentication and key confirmation with MAC
5	Reconnect	Version, cryptosuite, and parameter negotiation
6	Reconnect	Exchange of ECDHE keys and nonces
7	Reconnect	Authentication and key confirmation with MAC
8	Completion	NoobId discovery
0	Error	Error notification

Table 7: EAP-NOOB

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

4.3. Error codes

The error codes defined by EAP-NOOB are listed in Table 8.

Error code	Purpose
1001	Invalid NAI
1002	Invalid message structure
1003	Invalid data
1004	Unexpected message type
1005	Unexpected peer identifier
1006	Unrecognized OOB message identifier
1007	Invalid ECDHE key
2001	Unwanted peer
2002	State mismatch, user action required
3001	No mutually supported protocol version
3002	No mutually supported cryptosuite
3003	No mutually supported OOB direction
4001	MAC verification failure
5001	Application-specific error
5002	Invalid server info
5003	Invalid server URL
5004	Invalid peer info
6001-6999	Private and experimental use

Table 8: EAP-NOOB error codes

Assignment of new error codes MUST be done through IANA with "Specification Required" and "IESG Approval" as defined in [RFC8126], with the exception of the range 6001-6999, which is reserved for "Private Use" and "Experimental Use".

4.4. Domain name reservation considerations

"eap-noob.net" should be registered as a special-use domain. The considerations required by [RFC6761] for registering this special use domain name are as follows:

- o **Users:** Non-admin users are not expected to encounter this name or recognize it as special. AAA administrators may need to recognize the name.
- o **Application Software:** Application software is not expected to recognize this domain name as special.
- o **Name Resolution APIs and Libraries:** Name resolution APIs and libraries are not expected to recognize this domain name as special.

- o Caching DNS Servers: Caching servers are not expected to recognize this domain name as special.
- o Authoritative DNS Servers: Authoritative DNS servers MUST respond to queries for eap-noob.net with NXDOMAIN.
- o DNS Server Operators: Except for the authoritative DNS server, there are no special requirements for the operators.
- o DNS Registries/Registrars: There are no special requirements for DNS registrars.

5. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

5.1. Implementation with wpa_supplicant and hostapd

- o Responsible Organization: Aalto University
- o Location: <<https://github.com/tuomaura/eap-noob>>
- o Coverage: This implementation includes all of the features described in the current specification. The implementation supports two dimensional QR codes and NFC as example out-of-band (OOB) channels
- o Level of Maturity: Alpha
- o Version compatibility: Version 03 of the draft implemented
- o Licensing: BSD
- o Contact Information: Tuomas Aura, tuomas.aura@aalto.fi

5.2. Protocol modeling

The current EAP-NOOB specification has been modeled with the mCRL2 formal specification language [mcr12]. The model <<https://github.com/tuomaura/eap-noob/tree/master/protocolmodel/mcrl2>> was used mainly for simulating the protocol behavior and for verifying basic safety and liveness properties as part of the specification process. For example, the model was used to verify the correctness of the tiebreaking mechanism when two OOB messages are received simultaneously, one in each direction. The mCRL2 model is not intended for verifying security of the authenticated key-exchange; a different modeling method is needed for that. It was, however, used to verify that a man-in-the-middle attacker cannot cause persistent failure by spoofing a finite number of messages.

6. Security considerations

EAP-NOOB is an authentication and key derivation protocol and, thus, security considerations can be found in most sections of this specification. In the following, we explain the protocol design and highlight some other special considerations.

6.1. Authentication principle

EAP-NOOB establishes a shared secret with an authenticated ECDHE key exchange. The mutual authentication in EAP-NOOB is based on two separate features, both conveyed in the OOB message. The first authentication feature is the secret nonce Noob. The peer and server use this secret in the Completion Exchange to mutually authenticate the session key previously created with ECDHE. The message authentication codes computed with the secret nonce Noob are alone sufficient for authenticating the key exchange. The second authentication feature is the integrity-protecting fingerprint Hoob. Its purpose is to prevent impersonation and man-in-the-middle attacks even in situations where the attacker is able to eavesdrop the OOB channel and the nonce Noob is compromised. In some human-assisted OOB channels, such as sound burst or user-transferred URL, it may be easier to detect tampering than eavesdropping of the OOB message, and such applications benefit from the second authentication feature.

The additional security provided by the cryptographic fingerprint Hoob is somewhat intricate to understand. The endpoint that receives the OOB message uses Hoob to verify the integrity of the ECDHE exchange. Thus, the OOB receiver can detect impersonation and man-in-the-middle attacks on the in-band channel. The other endpoint, however, is not equally protected because the OOB message and fingerprint are sent only in one direction. Some protection to the OOB sender is afforded by the fact that the user may notice the

failure of the association at the OOB receiver and therefore reset the OOB sender. Other device-pairing protocols have solved similar situations by requiring the user to confirm to the OOB sender that the association was accepted by the OOB receiver, e.g. by pressing an "accept" button on the sender side. Applications MAY implement EAP-NOOB in this way. Nevertheless, since EAP-NOOB was designed to work with strictly one-directional OOB communication and the fingerprint is only the second authentication feature, the EAP-NOOB specification does not mandate such explicit confirmation to the OOB sender.

To summarize, EAP-NOOB uses the combined protection of the secret nonce Noob and the cryptographic fingerprint Hoob, both conveyed in the OOB message. The secret nonce Noob alone is sufficient for mutual authentication, unless the attacker can eavesdrop it from the OOB channel. Even if an attacker is able to eavesdrop the secret nonce Noob, it nevertheless cannot perform a full man-in-the-middle attack on the in-band channel because the mismatching fingerprint would alert the OOB receiver, which would reject the OOB message. That attacker that eavesdropped the secret nonce can impersonate the OOB receiver to the OOB sender. In this case, the association will appear to be complete only on the OOB sender side, and such situations have to be resolved by the user by resetting the sender to the initial state.

The expected use cases for EAP-NOOB are ones where it replaces a user-entered access credentials in IoT appliances. In wireless network access without EAP, the user-entered credential is often a passphrase that is shared by all the network stations. The advantage of an EAP-based solution, including EAP-NOOB, is that it establishes a different master secret for each peer device, which makes the system more resilient against device compromise than if there were a common master secret. Additionally, it is possible to revoke the security association for an individual device on the server side.

Forward secrecy in EAP-NOOB is optional. The Reconnect Exchange in EAP-NOOB provides forward secrecy only if both the server and peer send their fresh ECDHE keys. This allows both the server and the peer to limit the frequency of the costly computation that is required for forward secrecy. The server MAY adjust the frequency of its attempts at ECDHE rekeying based on what it knows about the peer's computational capabilities.

The users delivering the OOB messages will often authenticate themselves to the EAP server, e.g. by logging into a secure web page. In this case, the server can reliably associate the peer device with the user account. Applications that make use of EAP-NOOB can use this information for configuring the initial owner of the freshly-registered device.

6.2. Identifying correct endpoints

One remaining threat against EAP-NOOB is that the attacker first impersonates the peer on the in-band channel in the Initial Exchange and, during the OOB step, tricks the user to deliver the OOB message to or from the wrong peer device. The server will now be associated with that wrong peer. Similarly, the attacker can try to trick the user to deliver the OOB message to the wrong server, so that the peer device becomes associated with the wrong server. This reliance on user in identifying the correct endpoints is an inherent property of user-assisted out-of-band authentication.

One mechanism that can mitigate user mistakes is certification of peer devices. The certificate can convey to the server authentic identifiers and attributes of the peer device. Compared to a fully certificate-based authentication, however, EAP-NOOB can be used without trusted third parties and does not require the user to know any identifier of the peer device; physical access to the device is sufficient.

The user could accidentally deliver the OOB message to a wrong peer device in addition to the correct one. Such accidents in EAP-NOOB will not enable the wrong device to compute the master key of the correct device or to eavesdrop the communication protected by that key. This is because the wrong device would need to have performed a man-in-the-middle attack on the in-band Initial Exchange before the accident occurred. In comparison, simpler solutions where the master key is transferred to the device via the OOB channel are vulnerable to opportunistic attacks if the user mistakenly delivers the master key to more than one device.

6.3. Trusted path issues

Another remaining threat is spoofed user input or output on the peer device. When the user is delivering the OOB message to or from a peer device, a trusted path between the user and the peer device is needed. That is, the user must communicate directly with an authentic operating system and EAP-NOOB implementation in the peer device and not with a spoofed user interface. Otherwise, a Registered device that is under the control of the attacker could emulate the behavior of an Unregistered device. The secure path can be implemented, for example, by indicating the peer device's status (Unregistered or Registered/Reconnecting) when the device is powered up or by requiring the user to press a reset button to trigger the Initial Exchange.

6.4. Peer identifiers and attributes

The PeerId value in the protocol is a server-allocated identifier for its association with the peer and SHOULD NOT be shown to the user because its value is initially ephemeral. Since the PeerId is allocated by the server and the scope of the identifier is the single server, the so-called identifier squatting attacks, where a malicious peer could reserve another peer's identifier, are not possible in EAP-NOOB. The server SHOULD assign a random or pseudo-random PeerId to each new peer. It SHOULD NOT select the PeerId based on any peer characteristics that it may know, such as the peer's link-layer network address.

User reset or failure in the OOB Step can cause the peer to perform many Initial Exchanges with the server and to allocate many PeerIds and to store the ephemeral protocol state for them. The peer will typically only remember the latest one. EAP-NOOB leaves it to the implementation to decide when to delete these ephemeral associations. There is no security reason to delete them early, and the server does not have any way to verify that the peers are actually the same one. Thus, it is safest to store the ephemeral states for at least one day. If the OOB messages are sent only in the server-to-peer direction, the server SHOULD NOT delete the ephemeral state before all the related Noob values have expired.

After completion of EAP-NOOB, the server may store the PeerInfo data, and the user may use it to identify the peer and its properties, such as the make and model or serial number. A compromised peer could lie in the PeerInfo that it sends to the server. If the server stores any information about the peer, it is important that this information is approved by the user during or after the OOB Step. Without verification by the user or authentication with vendor certificates on the application level, the PeerInfo is not authenticated information and should not be relied on.

One possible use for the PeerInfo field is EAP channel binding ([RFC3748] Section 7.15). That is, the PeerInfo may include data items that bind the EAP-NOOB association and exported keys to properties of the authenticator or the access link, such as the SSID and BSSID of the wireless network (see Appendix C).

6.5. Identity protection

The PeerInfo field contains identifiers and other information about the peer device (see Appendix C), and the peer sends this information in plaintext to the EAP server before the server authentication in EAP-NOOB has been completed. While the information refers to the peer device and not directly to the user, it may be better for user

privacy to avoid sending unnecessary information. In the Reconnect Exchange, the optional PeerInfo SHOULD be omitted unless some critical data has changed.

Peer devices that randomize their layer-2 address to prevent tracking can do this whenever the user resets the EAP-NOOB association. During the lifetime of the association, the PeerId is a unique identifier that can be used to track the peer in the access network. Later versions of this specification may consider updating the PeerId at each Reconnect Exchange. In that case, it is necessary to consider how the authenticator and access-network administrators can recognize and blacklist misbehaving peer devices and how to avoid loss of synchronization between the server and the peer if messages are lost during the identifier update.

6.6. Downgrading threats

The fingerprint Hoob protects all the information exchanged in the Initial Exchange, including the cryptosuite negotiation. The message authentication codes MACs and MACp also protect the same information. The message authentication codes MACs2 and MACp2 protect information exchanged during key renegotiation in the Reconnect Exchange. This prevents downgrading attacks to weaker cryptosuites as long as the possible attacks take more time than the maximum time allowed for the EAP-NOOB completion. This is typically the case for recently discovered cryptanalytic attacks.

As an additional precaution, the EAP server and peer SHOULD check for downgrading attacks in the Reconnect Exchange. As long as the server or peer saves any information about the other endpoint, it MUST also remember the previously negotiated cryptosuite and MUST NOT accept renegotiation of any cryptosuite that is known to be weaker than the previous one, such as a deprecated cryptosuite.

Integrity of the direction negotiation cannot be verified in the same way as the integrity of the cryptosuite negotiation. That is, if the OOB channel used in an application is critically insecure in one direction, a man-in-the-middle attacker could modify the negotiation messages and thereby cause that direction to be used. Applications that support OOB messages in both directions SHOULD therefore ensure that the OOB channel has sufficiently strong security in both directions. While this is a theoretical vulnerability, it could arise in practice if EAP-NOOB is deployed in unexpected applications. However, most devices acting as the peer are likely to support only one direction of exchange, in which case interfering with the direction negotiation can only prevent the completion of the protocol.

The long-term shared key material K_z in the persistent EAP-NOOB association is established with an ECDHE key exchange when the peer and server are first associated. It is a weaker secret than a manually configured random shared key because advances in cryptanalysis against the used ECDHE curve could eventually enable the attacker to recover K_z . EAP-NOOB protects against such attacks by allowing cryptosuite upgrades in the Reconnect Exchange and by updating shared key material K_z whenever the cryptosuite is upgraded. We do not expect the cryptosuite upgrades to be frequent, but if one becomes necessary, the upgrade can be made without manual resetting and reassociation of the peer devices.

6.7. Recovery from loss of last message

EAP methods do not receive notification of an EAP-Success message from the parent EAP state machine [RFC4137] after successful authentication. As stated in [RFC3748], EAP-Success messages are not protected, may be lost, and are not re-transmitted. Therefore, in the Completion Exchange (shown in Figure 5), after the EAP peer sends the last EAP-Response (Type=4) containing the MACp, it exports the keying material such as MSK, EMSK and Session-Id to the parent EAP state machine. A loss of this message in the absence of an attacker would not have any consequences as EAP Retransmission Behavior defined in Section 4.3 of [RFC3748] suggests 3-5 retransmissions.

However, a determined attacker can drop the final EAP-Response message (and all subsequent retransmissions) leaving the peer in registered state (4) while the server remains in state 1 or 2 (depending if the OOB message direction was server-to-peer or peer-to-server).

An attacker can try to use a similar strategy for leaving the peer and the server in discordant states during the Reconnect Exchange. In this scenario, after the EAP peer successfully verifies the server MACs2 received, it sends the last EAP-Response (Type=7) containing the MACp2. The peer would move state 4 and export all the keying material to the parent EAP state machine. However, an attacker can drop the last EAP-Response (and all subsequent retransmissions) leaving the server in state 3.

6.8. Other denial-of-service threats

To be written.

6.9. EAP security claims

EAP security claims are defined in section 7.2.1 of [RFC3748]. The security claims for EAP-NOOB are listed in Table 9.

Security property	EAP-NOOB claim
Authentication mechanism	ECDHE key exchange with out-of-band authentication
Protected cryptosuite negotiation	yes
Mutual authentication	yes
Integrity protection	yes
Replay protection	yes
Key derivation	yes
Key strength	The specified cryptosuites provide key strength of at least 128 bits.
Dictionary attack protection	yes
Fast reconnect	yes
Cryptographic binding	not applicable
Session independence	yes
Fragmentation	no
Channel binding	yes (The ServerInfo and PeerInfo can be used to convey integrity-protected channel properties such as network SSID or peer MAC address.)

Table 9: EAP security claims

7. References

7.1. Normative references

- [NIST-DH] Barker, E., Chen, L., Roginsky, A., and M. Smid, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A Revision 2, May 2013, <<http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.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>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [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>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<https://www.rfc-editor.org/info/rfc6761>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.

- [RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <<https://www.rfc-editor.org/info/rfc7518>>.
- [RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <<https://www.rfc-editor.org/info/rfc7542>>.
- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/info/rfc7748>>.
- [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>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.

7.2. Informative references

- [BluetoothPairing] Bluetooth, SIG, "Simple pairing whitepaper", Technical report , 2007.
- [EUI-48] Institute of Electrical and Electronics Engineers, "802-2014 IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.", IEEE Standard 802-2014. , June 2014.
- [IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004. , December 2004.
- [mcrl2] Groote, J. and M. Mousavi, "Modeling and analysis of communicating systems", The MIT press , 2014, <<https://mitpress.mit.edu/books/modeling-and-analysis-communicating-systems>>.
- [RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, DOI 10.17487/RFC2904, August 2000, <<https://www.rfc-editor.org/info/rfc2904>>.

- [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>>.
- [RFC4266] Hoffman, P., "The gopher URI Scheme", RFC 4266, DOI 10.17487/RFC4266, November 2005, <<https://www.rfc-editor.org/info/rfc4266>>.
- [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>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [Sethi14] Sethi, M., Oat, E., Di Francesco, M., and T. Aura, "Secure Bootstrapping of Cloud-Managed Ubiquitous Displays", Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2014), pp. 739-750, Seattle, USA , September 2014, <<http://dx.doi.org/10.1145/2632048.2632049>>.

Appendix A. Exchanges and events per state

Figure 10 shows how the EAP server chooses the exchange type depending on the server and peer states. In the state combinations marked with hyphen "-", there is no possible exchange and user action is required to make progress. Note that peer state 4 is omitted from the table because the peer never connects to the server when the peer is in that state. The table also indicates the handling of errors in each exchange. A notable detail is that the recipient of error code 1006 moves to state 1.

peer states	exchange chosen by server	next peer and server states
server state: Unregistered (0)		
0..2 3	Initial Exchange -	both 1 (0 on error) no change, notify user
server state: Waiting for OOB (1)		
0 1 2 3	Initial Exchange Waiting Exchange Completion Exchange Completion Exchange	both 1 (0 on error) both 1 (no change on error) both 4 (A) both 4
server state: OOB Received (2)		
0 1 2 3	Initial Exchange Completion Exchange Completion Exchange Completion Exchange	both 1 (0 on error) both 4 (B) both 4 (A) both 4
server state: Reconnecting (3) or Registered (4)		
0..2 3	- Reconnect Exchange	no change, notify user both 4 (3 on error)

(A) peer to 1 on error 1006, no other changes on error

(B) server to 1 on error 1006, no other changes on error

Figure 10: How server chooses the exchange type

Figure 11 lists the local events that can take place in the server or peer. Both the server and peer output and accept OOB messages in

association state 1, leading the receiver to state 2. Communication errors and timeouts in states 0..2 lead back to state 0, while similar errors in states 3..4 lead to state 3. Application request for rekeying (e.g. to refresh session keys or to upgrade cryptosuite) also takes the association from state 3..4 to state 3. User can always reset the association state to 0. Recovering association data, e.g. from a backup, leads to state 3.

server/ peer state	possible local events on server and peer	next state
1	OOB Output*	1
1	OOB Input*	2 (1 on error)
0..2	Timeout/network failure	0
3..4	Timeout/network failure	3
3..4	Rekeying request	3
0..4	User resets peer state	0
0..4	Association state recovery	3

Figure 11: Local events on server and peer

Appendix B. Application-specific parameters

Table 10 lists OOB channel parameters that need to be specified in each application that makes use of EAP-NOOB. The list is not exhaustive and is included for the convenience of implementors only.

Parameter	Description
OobDirs	Allowed directions of the OOB channel
OobMessageEncoding	How the OOB message data fields are encoded for the OOB channel
SleepTimeDefault	Default minimum time in seconds that the peer should sleep before the next Waiting Exchange
OobRetries	Number of received OOB messages with invalid Hoob after which the receiver moves to Unregistered (0) state
NoobTimeout	How many seconds the sender of the OOB message remembers the sent Noob value. The RECOMMENDED value is 3600 seconds.
ServerInfoMembers	Required members in ServerInfo
PeerInfoMembers	Required members in PeerInfo

Table 10: OOB channel characteristics

Appendix C. ServerInfo and PeerInfo contents

The ServerInfo and PeerInfo fields in the Initial Exchange and Reconnect Exchange enable the server and peer, respectively, send information about themselves to the other endpoint. They contain JSON objects whose structure may be specified separately for each application and each type of OOB channel. ServerInfo and PeerInfo MAY contain auxiliary data needed for the OOB channel messaging and for EAP channel binding. Table 11 lists some suggested data fields for ServerInfo.

Data field	Description
ServerName	String that may be used to aid human identification of the server.
ServerURL	Prefix string when the OOB message is formatted as URL, as suggested in Appendix E.
SSIDList	List of wireless network identifier (SSID) strings used for roaming support, as suggested in Appendix D. The SSIDs are UTF-8 encoded.
Base64SSIDList	List of wireless network identifier (SSID) strings used for roaming support, as suggested in Appendix D. The SSIDs are base64url encoded. Peer SHOULD send at most one of the fields SSIDList and Base64SSIDList in PeerInfo, and the server SHOULD ignore SSIDList if Base64SSIDList is included.

Table 11: Suggested ServerInfo data fields

PeerInfo typically contains auxiliary information for identifying and managing peers on the application level at the server end. Table 12 lists some suggested data fields for PeerInfo.

Data field	Description
PeerName	String that may be used to aid human identification of the peer.
Manufacturer	Manufacturer or brand string.
Model	Manufacturer-specified model string.
SerialNumber	Manufacturer-assigned serial number.
MACAddress	Peer link-layer identifier (EUI-48) in the 12-digit base-16 form [EUI-48]. The string MAY include additional colon ':' or dash '-' characters that MUST be ignored by the server.
SSID	Wireless network SSID for channel binding. The SSID is a UTF-8 string.
Base64SSID	Wireless network SSID for channel binding. The SSID is base64url encoded. Peer SHOULD send at most one of the fields SSID and Base64SSID in PeerInfo, and the server SHOULD ignore SSID if Base64SSID is included.
BSSID	Wireless network BSSID (EUI-48) in the 12-digit base-16 form [EUI-48]. The string MAY include additional colon ':' or dash '-' characters that MUST be ignored by the server.

Table 12: Suggested PeerInfo data fields

Appendix D. EAP-NOOB roaming

AAA architectures [RFC2904] allow for roaming of network-connected appliances that are authenticated over EAP. While the peer is roaming in a visited network, authentication still takes place between the peer and an authentication server at its home network. EAP-NOOB supports such roaming by assigning a Realm to the peer. After the Realm has been assigned, the peer's NAI enables the visited network to route the EAP session to the peer's home AAA server.

A peer device that is new or has gone through a hard reset should be connected first to the home network and establish an EAP-NOOB association with its home AAA server before it is able to roam.

After that, it can perform the Reconnect Exchange from the visited network.

Alternatively, the device may provide some method for the user to configure the Realm of the home network. In that case, the EAP-NOOB association can be created while roaming. The device will use the user-assigned Realm in the Initial Exchange, which enables the EAP messages to be routed correctly to the home AAA server.

While roaming, the device needs to identify the networks where the EAP-NOOB association can be used to gain network access. For 802.11 access networks, the server MAY send a list of SSIDs in the ServerInfo JSON object in a member called SSIDList or Base64SSIDList. This member contains a JSON array where each element is an SSID that is base64url encoded without padding. If present, the peer MAY use this list as a hint to determine the networks where the EAP-NOOB association can be used for access authorization, in addition to the access network where the Initial Exchange took place.

Appendix E. OOB message as URL

While EAP-NOOB does not mandate any particular OOB communication channel, typical OOB channels include graphical displays and emulated NFC tags. In the peer-to-server direction, it may be convenient to encode the OOB message as a URL, which is then encoded as a QR code for displays and printers or as an NDEF record for NFC tags. A user can then simply scan the QR code or NFC tag and open the URL, which causes the OOB message to be delivered to the authentication server. The URL MUST specify the https protocol i.e. secure connection to the server, so that the man-in-the-middle attacker cannot read or modify the OOB message.

The ServerInfo in this case includes a JSON member called ServerUrl of the following format with maximum length of 60 characters:

```
https://<host>[:<port>]/[<path>]
```

To this, the peer appends the OOB message fields (PeerId, Noob, Hoob) as a query string. PeerId is provided to the peer by the server and might be a 22-character string. The peer base64url encodes, without padding, the 16-byte values Noob and Hoob into 22-character strings. The query parameters MAY be in any order. The resulting URL is of the following format:

```
https://<host>[:<port>]/[<path>]?P=<PeerId>&N=<Noob>&H=<Hoob>
```

The following is an example of a well-formed URL encoding the OOB message (without line breaks):

https://example.com/Noob?P=ZrD7qkczNoHGbGcN2bN0&N=rMinS0-F4EfCU8D9ljxX_A&H=QvnMp4UGxuQVFaxPW_14UW

Appendix F. Example messages

The message examples in this section are generated with Curve25519 ECDHE test vectors specified in section 6.1 of [RFC7748] (server=Alice, peer=Bob). The direction of the OOB channel negotiated is 1 (peer-to-server). The JSON messages are as follows (line breaks are for readability only).

=====
Initial Exchange
=====

Identity response:
noob@eap-noob.net

EAP request (type 1):
{ "Type":1, "Vers":[1], "PeerId":"07KRU6OgqX0HIerFl دنب SW", "Realm":"noob.example.com", "Cryptosuites":[1], "Dirs":3, "ServerInfo":{"Name":"Example", "Url":"https://noob.example.com/sendOOB"} }

EAP response (type 1):
{ "Type":1, "Verp":1, "PeerId":"07KRU6OgqX0HIerFl دنب SW", "Cryptosuitep":1, "Dirp":1, "PeerInfo":{"Make":"Acme", "Type":"None", "Serial":"DU-9999", "SSID":"Noob1", "BSSID":"6c:19:8f:83:c2:80"} }

EAP request (type 2):
{ "Type":2, "PeerId":"07KRU6OgqX0HIerFl دنب SW", "PKs":{"kty":"EC", "crv":"Curve25519", "x":"hSDwCYkwp1R0i33ctD73Wg2_Og0mOBr066SpjqqbTmo"}, "Ns":"PYO7NVd9Af3BxEri1MI6hL8Ck49YxwCjSRPqlC1SPbw", "SleepTime":60} }

EAP response (type 2):
{ "Type":2, "PeerId":"07KRU6OgqX0HIerFl دنب SW", "PKp":{"kty":"EC", "crv":"Curve25519", "x":"3p7bfXt9wbTTW2HC7OQ1Nz-DQ8hbeGdNrfx-FG-IK08"}, "Np":"HIvB6g0n2btpxEcU7YXnWB-451ED6L6veQQd6ugiPFU"} }

=====
Waiting Exchange
=====

Identity response:
07KRU6OgqX0HIerFl دنب SW+s1@noob.example.com

EAP request (type 3):
{ "Type":3, "PeerId":"07KRU6OgqX0HIerFl دنب SW", "SleepTime":60} }

EAP response (type 3):
{ "Type":3, "PeerId":"07KRU6OgqX0HIerFl دنب SW"} }

=====
OOB Step
=====

Identity response:

```
P=07KRU6OgqX0HIeRFldnbSW&N=x3JlolaPciK4Wa6XlMJxtQ&H=faqWz68trUrBTK
AnioZMQA
```

=====
Completion Exchange
=====

Identity response:

```
07KRU6OgqX0HIeRFldnbSW+s2@noob.example.com
```

EAP request (type 8):

```
{"Type":8,"PeerId":"07KRU6OgqX0HIeRFldnbSW"}
```

EAP response (type 8):

```
{"Type":8,"PeerId":"07KRU6OgqX0HIeRFldnbSW","NoobId":"U0OHwYGCS4nE
kzk2TPIE6g"}
```

EAP request (type 4):

```
{"Type":4,"PeerId":"07KRU6OgqX0HIeRFldnbSW","NoobId":"U0OHwYGCS4nE
kzk2TPIE6g","MACs":"Y5NfKQkZTbRW3sEFhWy0Bv0ic2wsMnaA6xGqtUmQqmc"}
```

EAP response (type 4):

```
{"Type":4,"PeerId":"07KRU6OgqX0HIeRFldnbSW","MACp":"ddY225rN31Yzo7
qZNPStbVO1HRdNnTx0Rit6_8xEh7A"}
```

=====
Reconnect Exchange
=====

Identity response:

```
07KRU6OgqX0HIeRFldnbSW+s3@noob.example.com
```

EAP request (type 5):

```
{"Type":5,"Vers":[1],"PeerId":"07KRU6OgqX0HIeRFldnbSW","Cryptosuit
es":[1],"Realm":"noob.example.com","ServerInfo":{"Name":"Example",
"Url":"https://noob.example.com/sendOOB"}}
```

EAP response (type 5):

```
{"Type":5,"Verp":1,"PeerId":"07KRU6OgqX0HIeRFldnbSW","Cryptosuitep
":1,"PeerInfo":{"Make":"Acme","Type":"None","Serial":"DU-
9999","SSID":"Noob1","BSSID":"6c:19:8f:83:c2:80"}}
```

EAP request (type 6):

```
{"Type":6,"PeerId":"07KRU6OgqX0HIeRFldnbSW","PKs2":{"kty":"EC","cr
v":"Curve25519","x":"hSDwCYkwp1R0i33ctD73Wg2_Og0mOBr066SpjqqbTmo"}
,"Ns2":"RDLahHB1IgmL_F_xcynrHurLPkCsrp3G3B_S82WUF4"}
```

EAP response (type 6):

```
{"Type":6,"PeerId":"07KRU6OgqX0HIeRFldnbSW","PKp2":{"kty":"EC","cr
v":"Curve25519","x":"3p7bfXt9wbTTW2HC7OQ1Nz-DQ8hbeGdNrfx-FG-
IK08"},"Np2":"jN0_V4P0JoTqwI9VHHQKd9ozUh7tQdc9ABd-j6oTy_4"}
```

EAP request (type 7):

```
{"Type":7,"PeerId":"07KRU6OgqX0HIeRF1dnbSW","MACs2":"_pXDF4-7uBKXKqVKKKB6U-GP9EDnGCNOMdkyFEQp_iwA"}
```

EAP response (type 7):

```
{"Type":7,"PeerId":"07KRU6OgqX0HIeRF1dnbSW","MACp2":"qSUH4zA0VzMqU2O1U-JJTqwGRXGB8i3bgasYL6o1uU"}
```

Appendix G. TODO list

- o Write security considerations on persistent denial-of-service conditions and avoiding them. Explain how the Completion Exchange is protected against DoS by dropped messages.

Appendix H. Version history

- o Version 01:
 - * Fixed Reconnection Exchange.
 - * URL examples.
 - * Message examples.
 - * Improved state transition (event) tables.
- o Version 02:
 - * Reworked the rekeying and key derivation.
 - * Increased internal key lengths and in-band nonce and MAC lengths to 32 bytes.
 - * Less data in the persistent EAP-NOOB association.
 - * Updated reference [NIST-DH] to Revision 2 (2013).
 - * Shorter suggested PeerId format.
 - * Optimized the example of encoding OOB message as URL.
 - * NoobId in Completion Exchange to differentiate between multiple valid Noob values.
 - * List of application-specific parameters in appendix.
 - * Clarified the equivalence of Unregistered state and no state.

- * Peer SHOULD probe the server regardless of the OOB channel direction.
 - * Added new error messages.
 - * Realm is part of the persistent association and can be updated.
 - * Clarified error handling.
 - * Updated message examples.
 - * Explained roaming in appendix.
 - * More accurate definition of timeout for the Noob nonce.
 - * Additions to security considerations.
- o Version 03:
- * Clarified reasons for going to Reconnecting state.
 - * Included Verp in persistent state.
 - * Added appendix on suggested ServerInfo and PeerInfo fields.
 - * Exporting PeerId and SessionId.
 - * Explicitly specified next state after OOB Step.
 - * Clarified the processing of an expired OOB message and unrecognized NoobId.
 - * Enabled protocol version upgrade in Reconnect Exchange.
 - * Explained handling of redundant received OOB messages.
 - * Clarified where raw and base64url encoded values are used.
 - * Cryptosuite must specify the detailed format of the JWK object.
 - * Base64url encoding in JSON strings is done without padding.
 - * Simplified explanation of PeerId, Realm and NAI.
 - * Added error codes for private and experimental use.
 - * Updated the security considerations.

- o Version 04:

- * Recovery from synchronization failure due to lost last response.

Appendix I. Acknowledgments

Alexsi Peltonen modeled the protocol specification with the mCRL2 formal specification language. Shiva Prasad TP and Raghavendra MS implemented parts of the protocol with wpa_supplicant and hostapd. Their inputs helped us in improving the specification.

The authors would also like to thank Rhys Smith and Josh Howlett for providing valuable feedback as well as new use cases and requirements for the protocol. Thanks to Darshak Thakore, Stefan Winter and Hannes Tschofenig for interesting discussions and arguments in this problem space.

Authors' Addresses

Tuomas Aura
Aalto University
Aalto 00076
Finland

EMail: tuomas.aura@aalto.fi

Mohit Sethi
Ericsson
Jorvas 02420
Finland

EMail: mohit@piuha.net

Network Working Group
Internet-Draft
Updates: 5216 (if approved)
Intended status: Standards Track
Expires: April 19, 2019

J. Mattsson
M. Sethi
Ericsson
October 16, 2018

Using EAP-TLS with TLS 1.3
draft-ietf-emu-eap-tls13-02

Abstract

This document specifies the use of EAP-TLS with TLS 1.3 while remaining backwards compatible with existing implementations of EAP-TLS. TLS 1.3 provides significantly improved security, privacy, and reduced latency when compared to earlier versions of TLS. 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 April 19, 2019.

Copyright Notice

Copyright (c) 2018 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
1.1.	Requirements and Terminology	3
2.	Protocol Overview	3
2.1.	Overview of the EAP-TLS Conversation	3
2.1.1.	Base Case	3
2.1.2.	Resumption	6
2.1.3.	Termination	8
2.1.4.	Privacy	10
2.1.5.	Fragmentation	12
2.2.	Identity Verification	12
2.3.	Key Hierarchy	12
2.4.	Parameter Negotiation and Compliance Requirements	13
2.5.	EAP State Machines	13
3.	Detailed Description of the EAP-TLS Protocol	14
4.	IANA considerations	14
5.	Security Considerations	14
5.1.	Security Claims	14
5.2.	Peer and Server Identities	15
5.3.	Certificate Validation	15
5.4.	Certificate Revocation	15
5.5.	Packet Modification Attacks	15
5.6.	Privacy Considerations	15
5.7.	Pervasive Monitoring	16
6.	References	16
6.1.	Normative References	16
6.2.	Informative references	18
	Appendix A. Updated references	19
	Acknowledgments	20
	Authors' Addresses	20

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 and key derivation utilizing the TLS handshake protocol for cryptographic algorithms and protocol version negotiation, mutual authentication, and establishment of shared secret keying material. EAP-TLS is widely supported for authentication in IEEE 802.11 [IEEE-802.11] networks (Wi-Fi) using IEEE 802.1X [IEEE-802.1X] and it's the default mechanism for certificate based authentication in MulteFire [MulteFire] and 3GPP 5G

[TS.33.501] networks. EAP-TLS [RFC5216] references TLS 1.0 [RFC2246] and TLS 1.1 [RFC4346], but works perfectly also with TLS 1.2 [RFC5246].

Weaknesses found in previous versions of TLS, as well as new requirements for security, privacy, and reduced latency has led to the development of TLS 1.3 [RFC8446], which in large parts is a complete remodeling of the TLS handshake protocol including a different message flow, different handshake messages, different key schedule, different cipher suites, different resumption, and different privacy protection. 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 (or higher). Therefore, aspects such as resumption, privacy handling, and key derivation need to be appropriately addressed for EAP-TLS with TLS 1.3 (or higher).

This document defines how to use EAP-TLS with TLS 1.3 (or higher) and does not change how EAP-TLS is used with older versions of TLS. While this document updates EAP-TLS [RFC5216], it remains backwards compatible with it and existing implementations of EAP-TLS. This document only describes differences compared to [RFC5216].

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. When EAP-TLS is used with support for privacy, TLS 1.3 requires two fewer round-trips. TLS 1.3 also 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 1.3 [RFC8446].

2. Protocol Overview

2.1. Overview of the EAP-TLS Conversation

2.1.1. Base Case

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 [RFC5216] does not apply for TLS 1.3 (or higher).

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 or higher, the formatting and processing of the TLS handshake SHALL be done as specified in that version of TLS. This document only lists additional and different requirements, restrictions, and processing compared to [RFC8446] and [RFC5216].

The EAP server MUST authenticate with a certificate and SHOULD require the EAP 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. SessionID is deprecated in TLS 1.3 and the EAP server SHALL ignore the `legacy_session_id` field if TLS 1.3 is negotiated. Resumption is handled as described in Section 2.1.2. After the TLS handshake has completed, the EAP server sends EAP-Success.

As stated in [RFC5216], the TLS cipher suite shall not be used to protect application data. This applies also for early application data. When EAP-TLS is used with TLS 1.3, early application data SHALL NOT be used.

In the case where EAP-TLS with mutual authentication is successful, the conversation will appear as shown in Figure 1. The EAP server commits to not send any more handshake messages by sending an empty TLS record, see Section 2.5.

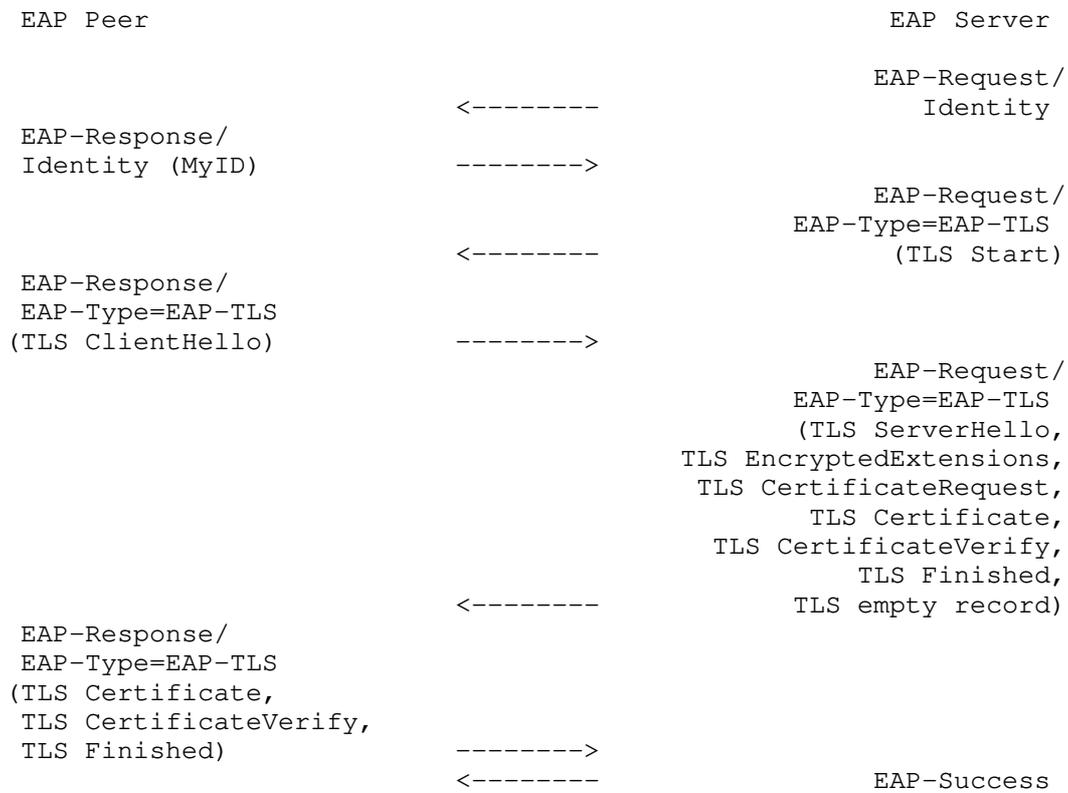


Figure 1: EAP-TLS mutual authentication

When using EAP-TLS with TLS 1.3, the EAP server MUST indicate support of resumption in the initial authentication. To indicate support of resumption, the EAP server sends a `NewSessionTicket` message (containing a PSK and other parameters) after it has received the `Finished` message.

In the case where EAP-TLS with mutual authentication and ticket establishment is successful, the conversation will appear as shown in Figure 2.

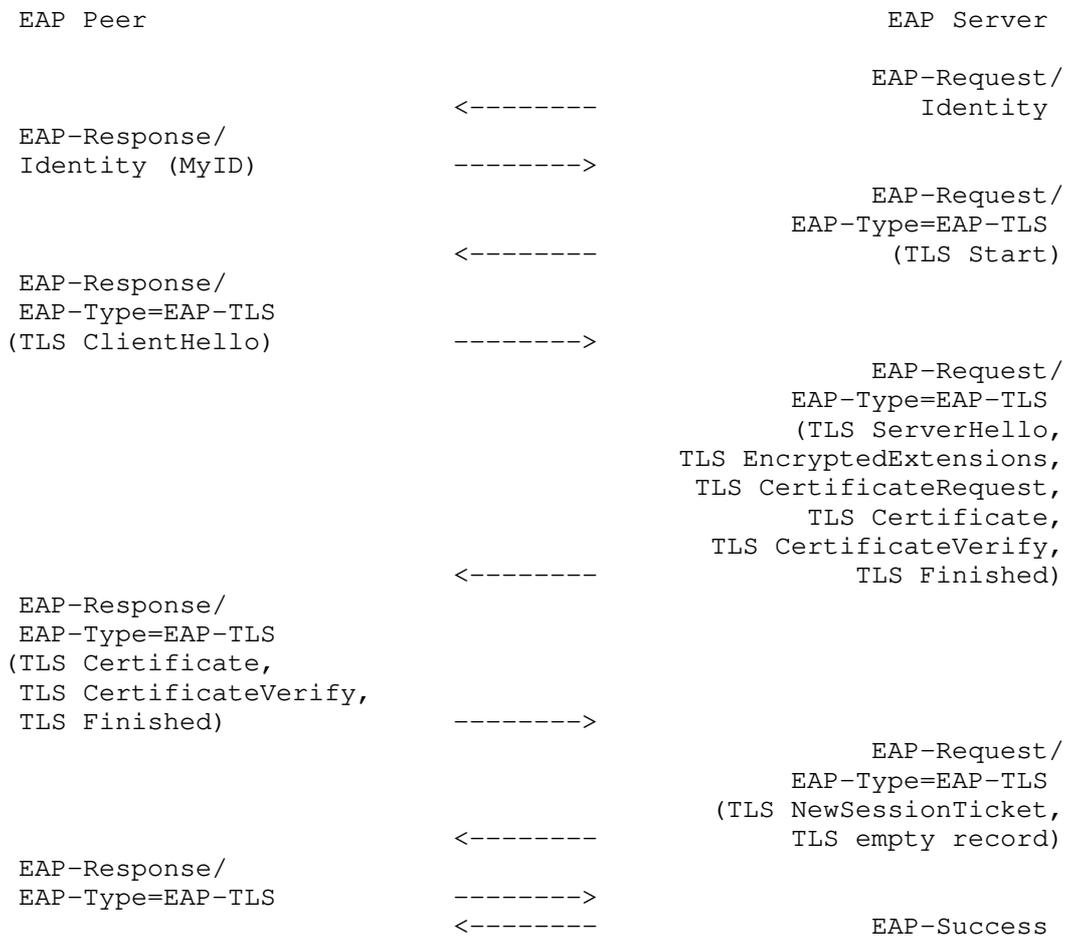


Figure 2: EAP-TLS ticket establishment

2.1.2. Resumption

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 or higher, EAP-TLS SHALL use a resumption mechanism compatible with that version 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 server, the client can use the PSK identity received in the ticket to negotiate the use of the associated PSK. If the server accepts it, then the security context of the new connection is tied to the original connection and the key derived from the initial handshake is

used to bootstrap the cryptographic state instead of a full handshake. It is left up to the EAP peer whether to use resumption, but a EAP peer SHOULD use resumption as long as it has a valid ticket cached. It is RECOMMENDED that the EAP server accept resumption as long as the ticket is valid. However, the server MAY choose to require a full authentication.

A subsequent authentication using resumption, where both sides authenticate successfully is shown in Figure 3.

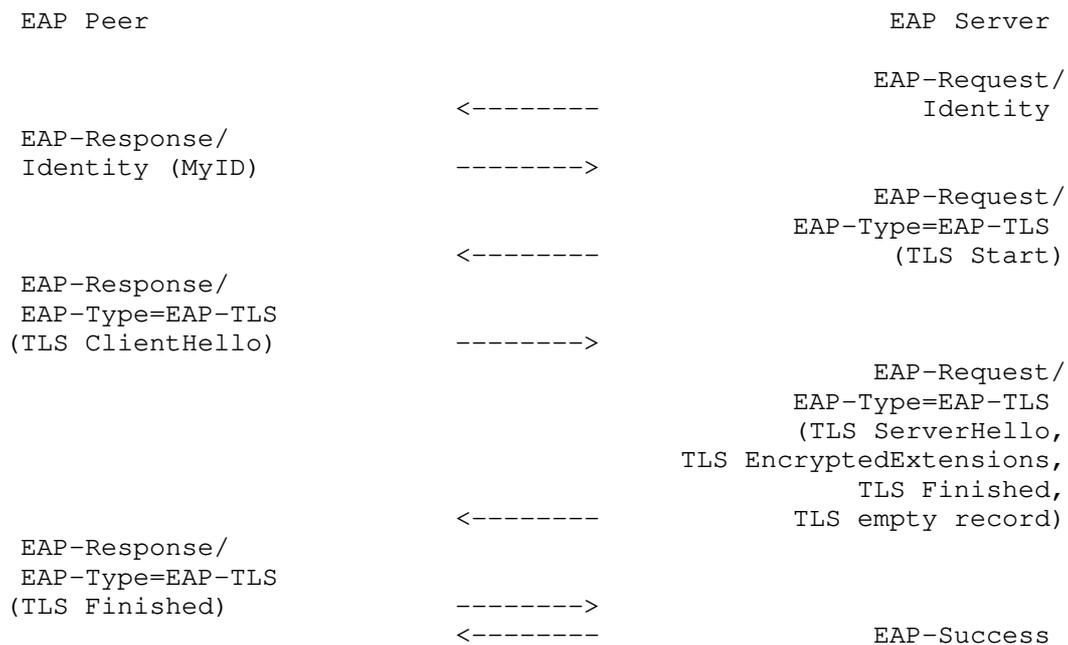


Figure 3: EAP-TLS resumption

As specified in Section 2.2 of [RFC8446], the EAP peer SHOULD supply a "key_share" extension when offering resumption, which allows the EAP server to decline resumption and continue the handshake as a full handshake. The message flow in this case is given by Figure 1 or Figure 2. If the EAP peer did not supply a "key_share" extension when offering resumption, the EAP server needs to reject the ClientHello and the EAP peer needs to restart a full handshake. The message flow in this case is given by Figure 4 followed by Figure 1 or Figure 2.

2.1.3. Termination

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 RFC 5216 [RFC5216] does not apply for TLS 1.3 or higher. The two paragraphs below replaces the corresponding paragraphs in Section 2.1.3 of RFC 5216 [RFC5216] when EAP-TLS is used with TLS 1.3 or higher. The other paragraphs in Section 2.1.3 of RFC 5216 [RFC5216] still apply with the exception that SessionID is deprecated.

If the EAP server authenticates successfully the EAP peer MUST send an EAP-Response message with EAP-Type=EAP-TLS containing TLS records confirming the processing in the version of TLS used.

If the EAP peer authenticates successfully the EAP server MUST send an EAP-Request packet with EAP-Type=EAP-TLS containing TLS records confirming to the processing in the version of TLS used. The message flow ends with the EAP server sending a EAP-Success message.

In the case where the server rejects the ClientHello, the conversation will appear as shown in Figure 4.



Figure 4: EAP-TLS server rejection of ClientHello

In the case where server authentication is unsuccessful, the conversation will appear as shown in Figure 5.

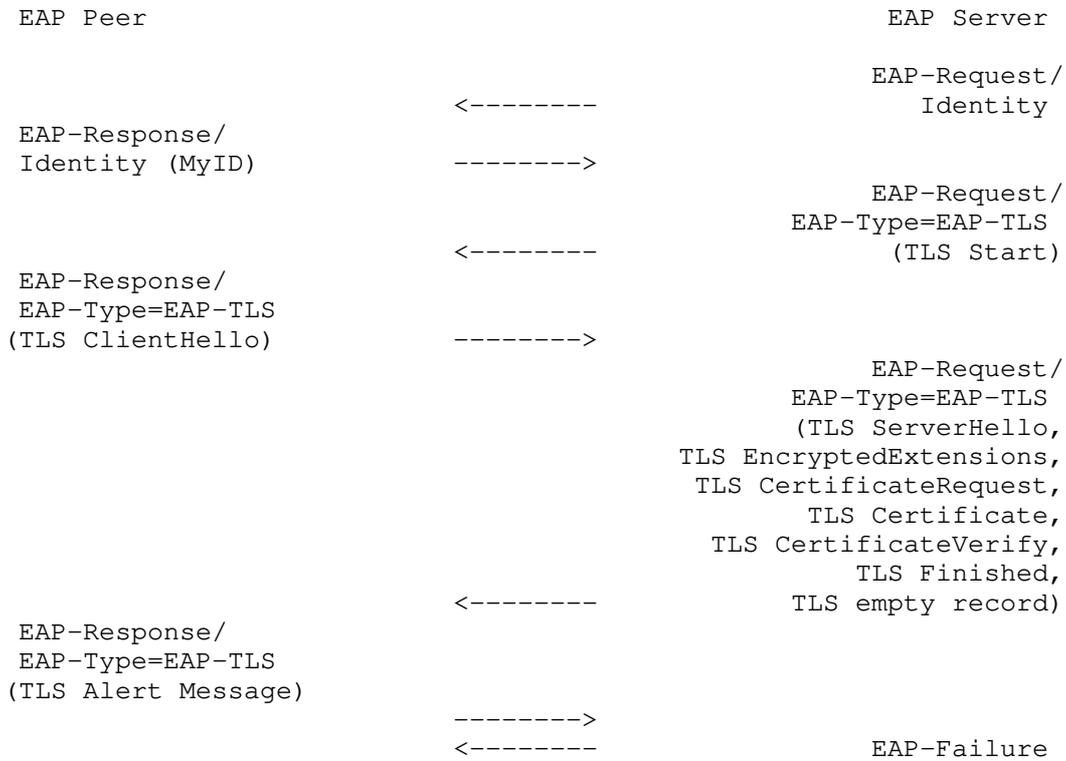


Figure 5: EAP-TLS unsuccessful server authentication

In the case where the server authenticates to the peer successfully, but the peer fails to authenticate to the server, the conversation will appear as shown in Figure 6.

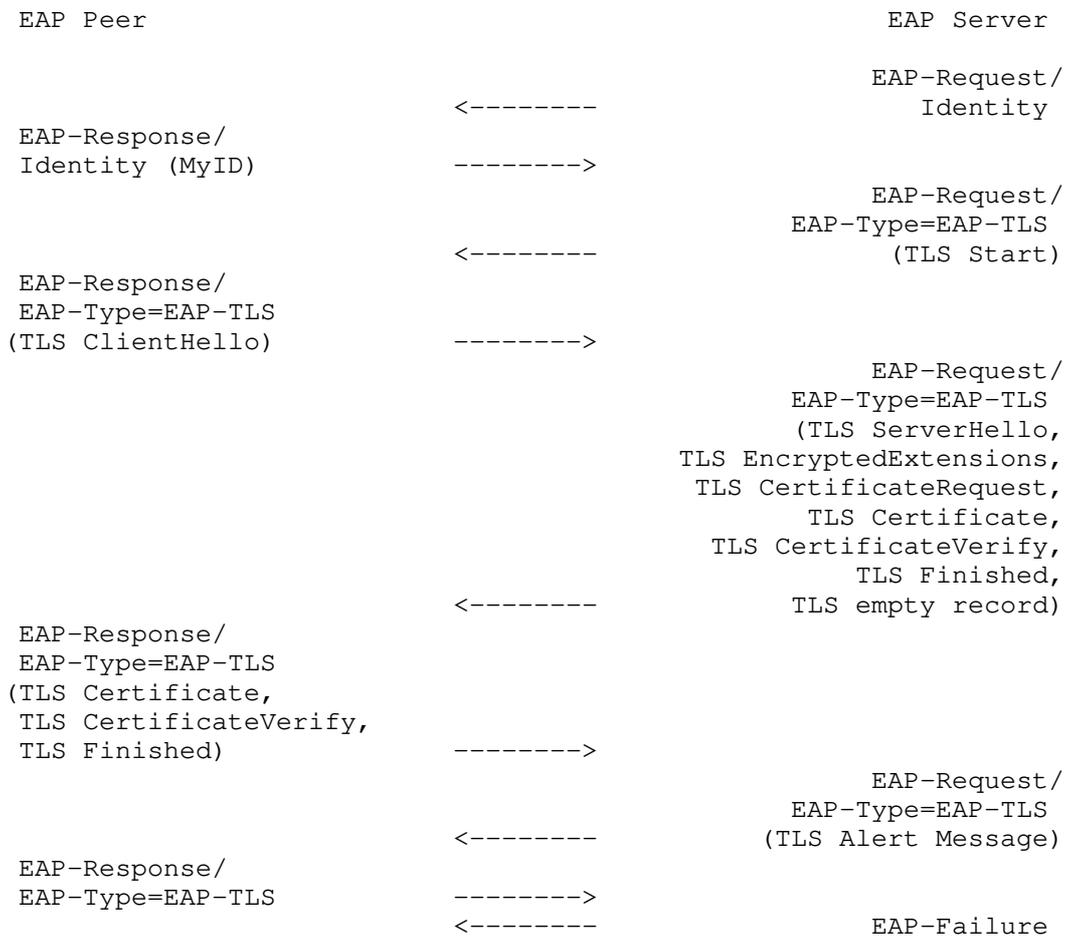


Figure 6: EAP-TLS unsuccessful client authentication

2.1.4. Privacy

TLS 1.3 significantly improves privacy when compared to earlier versions of TLS by forbidding cipher suites without confidentiality and encrypting large parts of the TLS handshake including the certificate messages.

EAP-TLS peer and server implementations supporting TLS 1.3 or higher MUST support anonymous NAIs (Network Access Identifiers) (Section 2.4 in [RFC7542]) and the client MUST confidentiality protect its identity (e.g. using Anonymous NAIs) when the EAP-TLS server is known to support TLS 1.3 or higher.

As the certificate messages in TLS 1.3 are encrypted, there is no need to send an empty `certificate_list` or perform a second handshake (as needed by EAP-TLS with earlier versions of TLS). When EAP-TLS is used with TLS version 1.3 or higher the EAP-TLS peer and EAP-TLS server SHALL follow the processing specified by the used version of TLS. For TLS 1.3 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.

When EAP-TLS is used with TLS 1.3 and privacy, no extra round-trips are added and the message flow looks just like a normal message flow with the only difference that an anonymous NAI is used. In the case where EAP-TLS with mutual authentication and privacy is successful, the conversation will appear as shown in Figure 7.

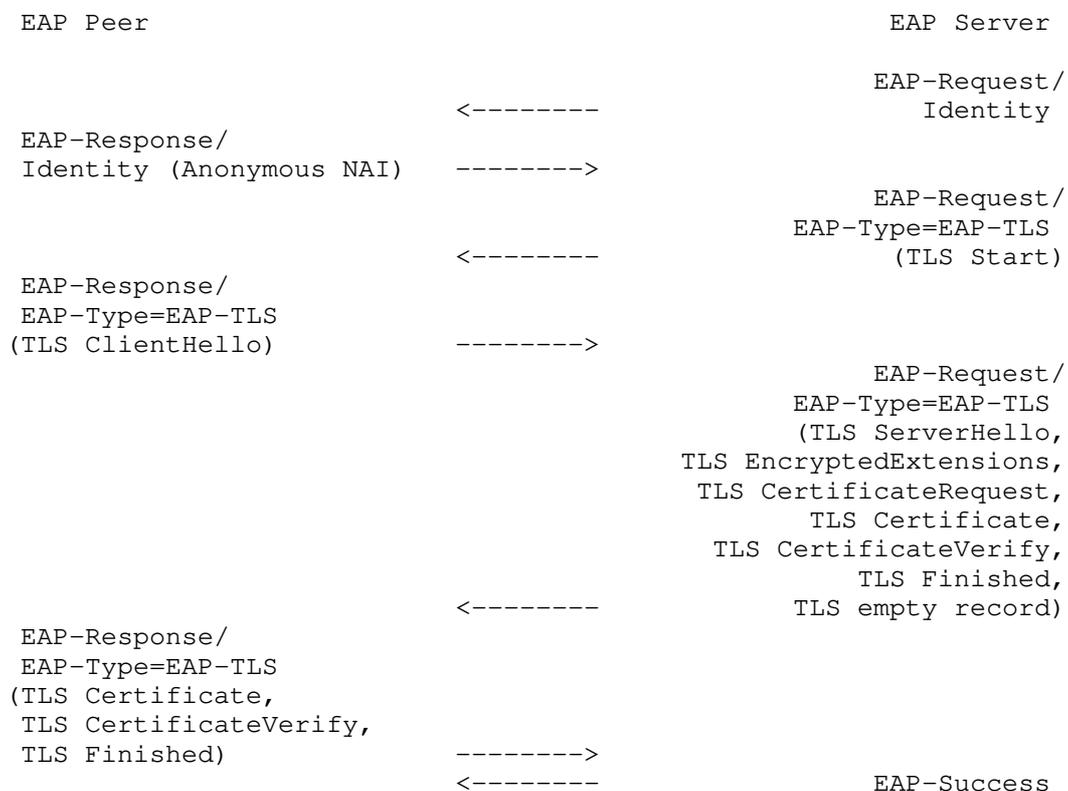


Figure 7: EAP-TLS privacy

2.1.5. Fragmentation

Including ContentType and ProtocolVersion a single TLS record may be up to 16387 octets in length. 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 client, 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.

While Elliptic Curve Cryptography (ECC) was optional for earlier version of TLS, TLS 1.3 mandates support of ECC (see Section 9 of [RFC8446]). To avoid fragmentation, the use of ECC in certificates, signature algorithms, and groups are RECOMMENDED when using EAP-TLS with TLS 1.3 or higher. At a 128-bit security level, this reduces public key sizes from 384 bytes (RSA and DHE) to 32 bytes (ECDHE) and signatures from 384 bytes (RSA) to 64 bytes (ECDSA and EdDSA). An EAP-TLS deployment MAY further reduce the certificate sizes by limiting the number of Subject Alternative Names.

Endpoints SHOULD reduce the sizes of Certificate messages by omitting certificates that the other endpoint is known to possess. When using TLS 1.3, all certificates that specifies a trust anchor 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]). EAP-TLS peers and servers SHOULD support and use the Cached Information Extension as specified in [RFC7924]. EAP-TLS peers and servers MAY use other extensions for reducing the sizes of Certificate messages, e.g. certificate compression [I-D.ietf-tls-certificate-compression].

2.2. Identity Verification

No updates to [RFC5216].

2.3. Key Hierarchy

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 or higher. 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 or higher the Key_Material, IV, and Method-Id SHALL be derived from the exporter_master_secret using the TLS exporter interface [RFC5705] (for TLS 1.3 this is defined in Section 7.5 of [RFC8446]).

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

By using the TLS exporter, EAP-TLS can use any TLS 1.3 implementation without having to extract the Master Secret, ClientHello.random, and ServerHello.random in a non-standard way.

All other parameters such as MSK and EMSK are derived as specified in EAP-TLS [RFC5216], Section 2.3. The use of these keys is specific to the lower layer, as described [RFC5247].

2.4. Parameter Negotiation and Compliance Requirements

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 or higher. The requirements on protocol version and compression given in Section 2.4 of [RFC5216] still apply.

When EAP-TLS is used with TLS version 1.3 or higher, the EAP-TLS peers and servers MUST comply with the requirements for the TLS version used. For TLS 1.3 the compliance requirements are defined in Section 9 of [RFC8446].

2.5. EAP State Machines

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. One such Post-Handshake message is NewSessionTicket. The NewSessionTicket can be used for resumption. After sending TLS Finished, the EAP server may send any number of Post-Handshake messages in separate EAP-Requests. To decrease the uncertainty for the EAP peer, the following procedure MUST be followed:

When an EAP server has sent its last handshake message (Finished or a Post-Handshake), it commits to not sending any more handshake messages by appending an empty application data record (i.e. a TLS record with TLSPlaintext.type = application_data and TLSPlaintext.length = 0) to the last handshake record. After sending an empty application data record, the EAP server may only send an EAP-Success, an EAP-Failure, or an EAP-Request with a TLS Alert Message.

Note that the use of an empty application data record does not violate the requirement that the TLS cipher suite shall not be used to protect application data, as the application data is the empty string, no application data is protected.

3. Detailed Description of the EAP-TLS Protocol

No updates to [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 memo 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, IV and Method-Id as defined in Section 2.3:

- o "EXPORTER_EAP_TLS_Key_Material"
- o "EXPORTER_EAP_TLS_IV"
- o "EXPORTER_EAP_TLS_Method-Id"

5. Security Considerations

5.1. Security Claims

Using EAP-TLS with TLS 1.3 does not change the security claims for EAP-TLS as given in Section 4.1 of [RFC5216]. However, it strengthens several of the claims as described in the following updates to the notes given in Section 4.1 of [RFC5216].

[2] Confidentiality: The TLS 1.3 handshake offers much better confidentiality than earlier versions of TLS by mandating cipher suites with confidentiality and encrypting certificates and some of the extensions, see [RFC8446]. When using EAP-TLS with TLS 1.3, the use of privacy does not cause any additional round-trips.

[3] Key strength: TLS 1.3 forbids all algorithms with known weaknesses including 3DES, CBC mode, RC4, SHA-1, and MD5. TLS 1.3 only supports cryptographic algorithms offering at least 112-bit security, see [RFC8446].

[4] Cryptographic Negotiation: TLS 1.3 increases the number of cryptographic parameters that are negotiated in the handshake. 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 [RFC5216].

5.3. Certificate Validation

No updates to [RFC5216].

5.4. Certificate Revocation

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 [RFC4366]. This enables sending OCSP information for all certificates in the certificate chain.

EAP-TLS peers and servers supporting TLS 1.3 SHOULD support Certificate Status Requests (OCSP stapling) as specified in [RFC6066] and Section 4.4.2.1 of [RFC8446]. The use of Certificate Status Requests to determine the current status of the EAP server's certificate is RECOMMENDED.

5.5. Packet Modification Attacks

No updates to [RFC5216].

5.6. Privacy Considerations

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

TLS 1.3 offers much better privacy than earlier versions of TLS as discussed in Section 2.1.4. 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 peer sends an identity in the first EAP-Response. The fields in the EAP-TLS Request and the EAP-TLS Response packets do not contain any cleartext privacy sensitive information.

It is strongly RECOMMENDED to confidentiality protect the identity (e.g. using Anonymous NAIs), as the username part of the NAI may otherwise enable identification and tracking of the user. However, 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 peers have the same domain, no additional information is leaked. If a domain name is used by a small subset of the EAP peers, it may aid an attacker in tracking or identifying the user.

An EAP peer with a policy allowing communication with EAP servers supporting only TLS 1.2 (or lower) without privacy and with RSA key exchange is vulnerable to disclosure of the peer username. An active attacker can in this case make the EAP peer believe that an EAP server supporting TLS 1.3 does not support privacy. The attacker can simply impersonate the EAP server and negotiate TLS 1.2 (or lower) with RSA key exchange and send an TLS alert message when the EAP peer tries to use privacy by sending an empty certificate message. Since the attacker (impersonating the EAP server) does not provide a proof-of-possession of the private key until the Finished message when RSA key exchange is used, an EAP peer may inadvertently disclose its identity (username) to an attacker. Therefore, it is RECOMMENDED for EAP peers to not use EAP-TLS with TLS 1.2 (or lower) and RSA based ciphersuites without privacy.

5.7. Pervasive Monitoring

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

Pervasive Monitoring is widespread surveillance of users. By encrypting more information, 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 wider 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.

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. Levkowitz, 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>>.
- [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>>.
- [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>>.
- [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

- [I-D.ietf-tls-certificate-compression]
Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", draft-ietf-tls-certificate-compression-04 (work in progress), October 2018.
- [IEEE-802.11]
Institute of Electrical and Electronics Engineers, "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", IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012) , December 2016.
- [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.
- [MulleFire]
MulleFire, "MulleFire Release 1.0.1 specification", 2017.
- [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>>.
- [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>>.

- [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>>.
- [RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, DOI 10.17487/RFC4366, April 2006, <<https://www.rfc-editor.org/info/rfc4366>>.
- [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>>.
- [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>>.
- [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>>.
- [TS.33.501] 3GPP, "Security architecture and procedures for 5G System", 3GPP TS 33.501 0.7.1, February 2018.

Appendix A. Updated references

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

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

All references to [RFC3280] are updated with [RFC5280].

All references to [RFC4282] are updated with [RFC7542].

Acknowledgments

The authors want to thank Alan DeKok, Ari Keraenen, Bernard Aboba, Eric Rescorla, Jari Arkko, Jim Schaad, Jouni Malinen, and Vesa Torvinen for comments and suggestions on the draft.

Authors' Addresses

John Mattsson
Ericsson
Stockholm 164 40
Sweden

Email: john.mattsson@ericsson.com

Mohit Sethi
Ericsson
Jorvas 02420
Finland

Email: mohit@piuha.net

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).

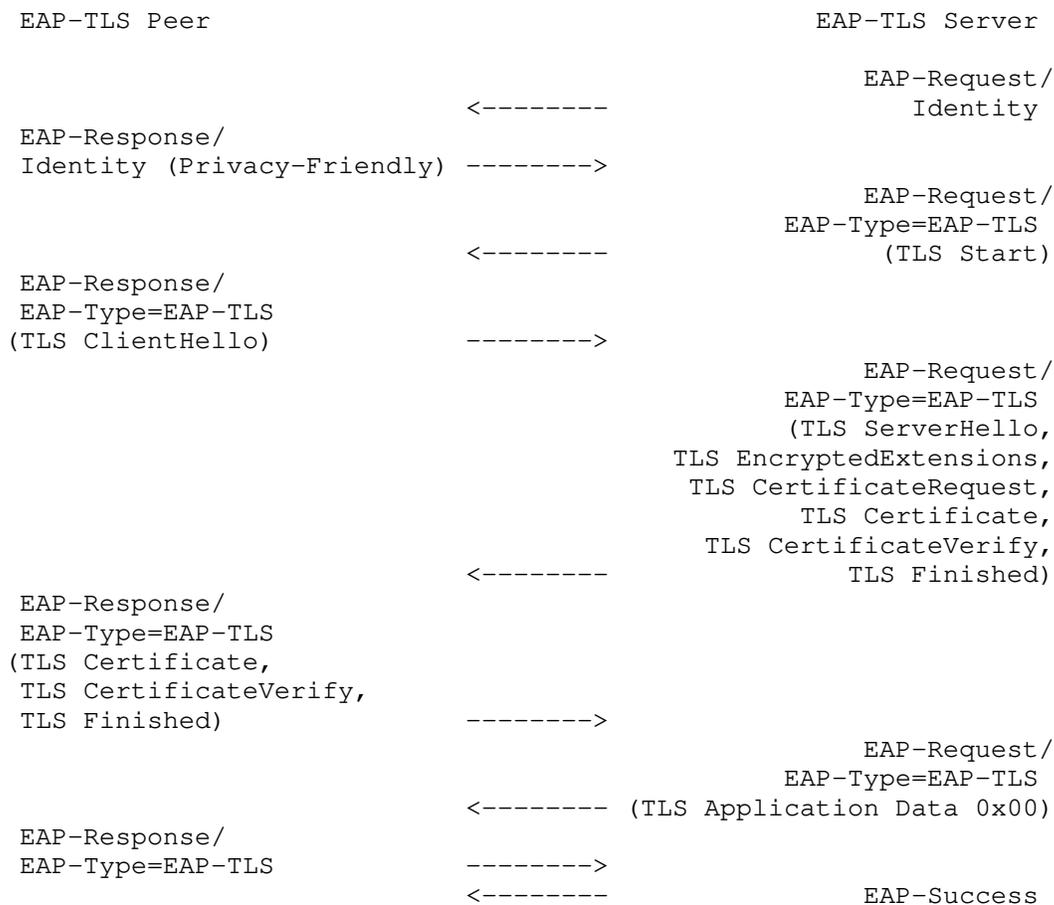


Figure 1: EAP-TLS mutual authentication

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.

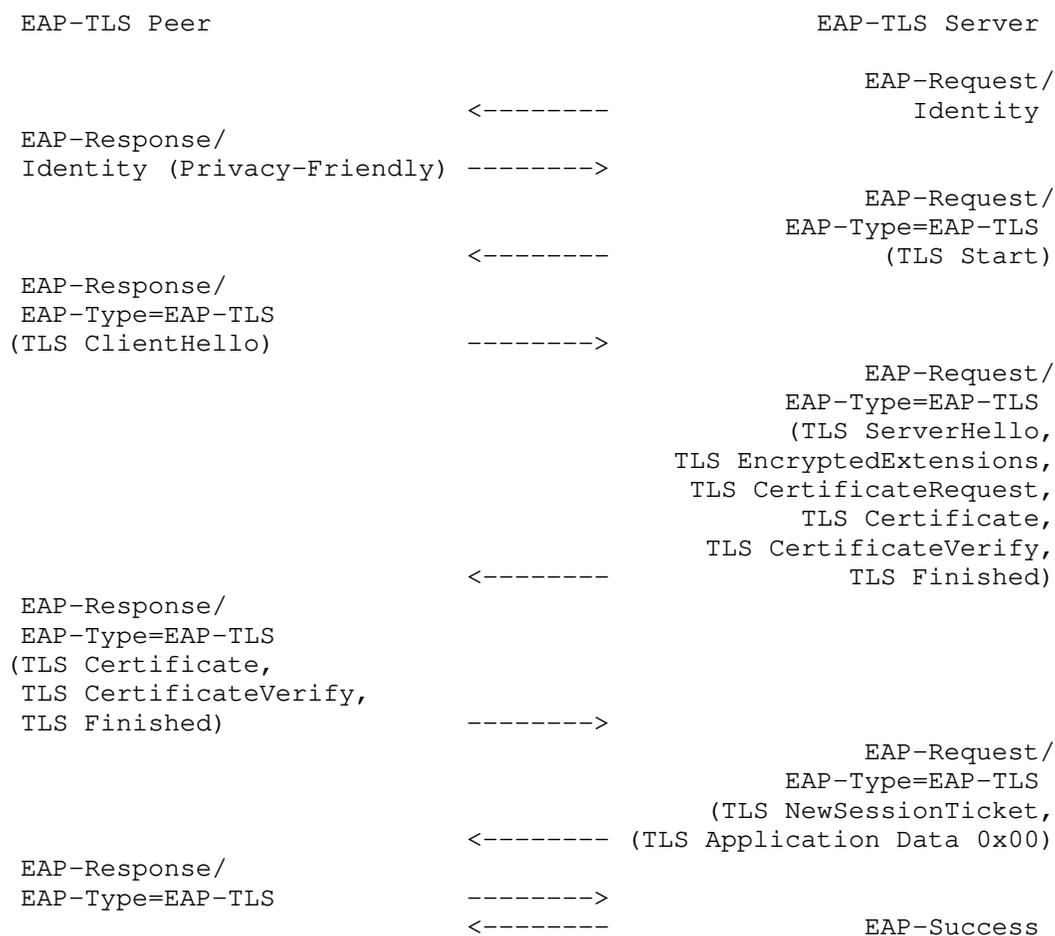


Figure 2: EAP-TLS ticket establishment

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.

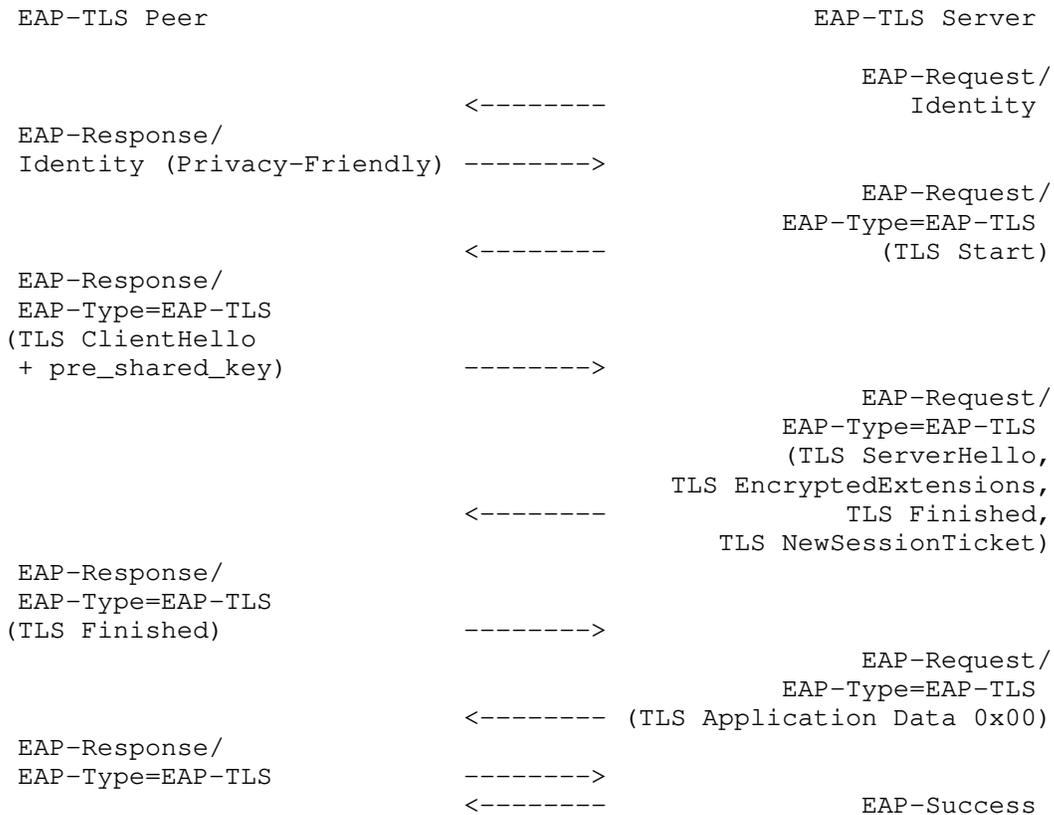


Figure 3: EAP-TLS resumption

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.

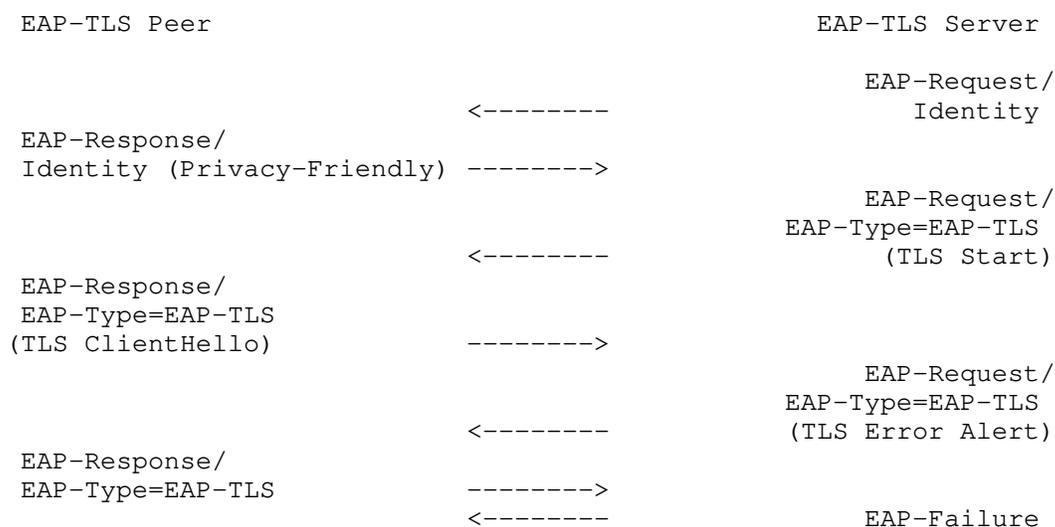


Figure 4: EAP-TLS server rejection of ClientHello

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

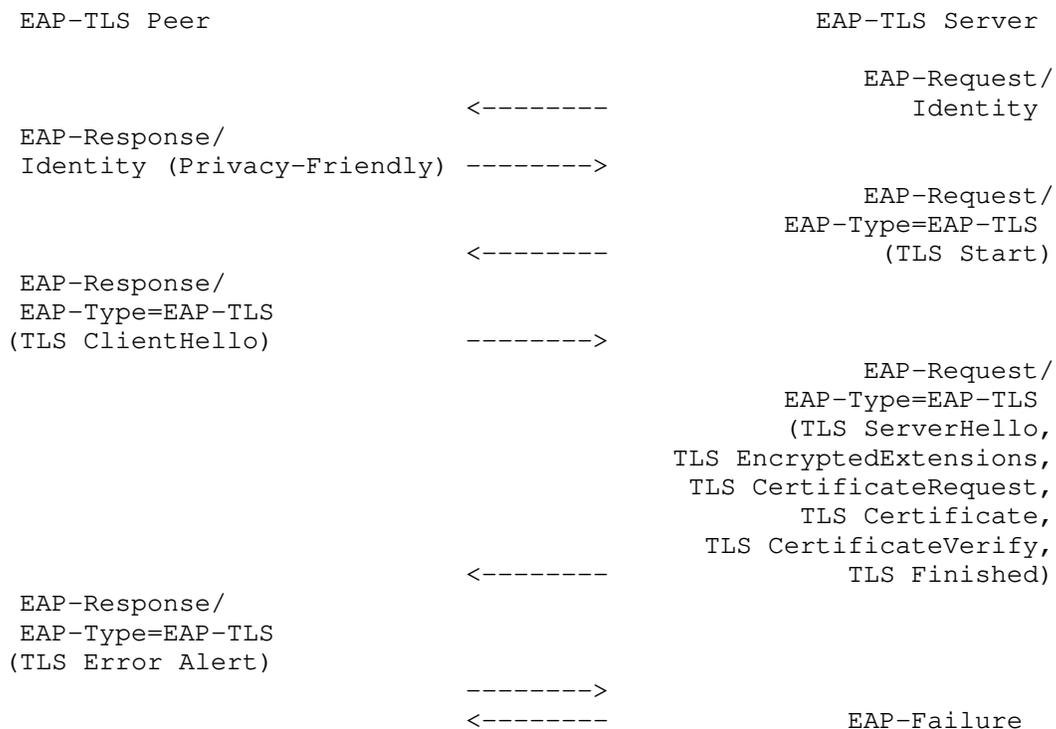


Figure 5: EAP-TLS unsuccessful EAP-TLS server authentication

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.

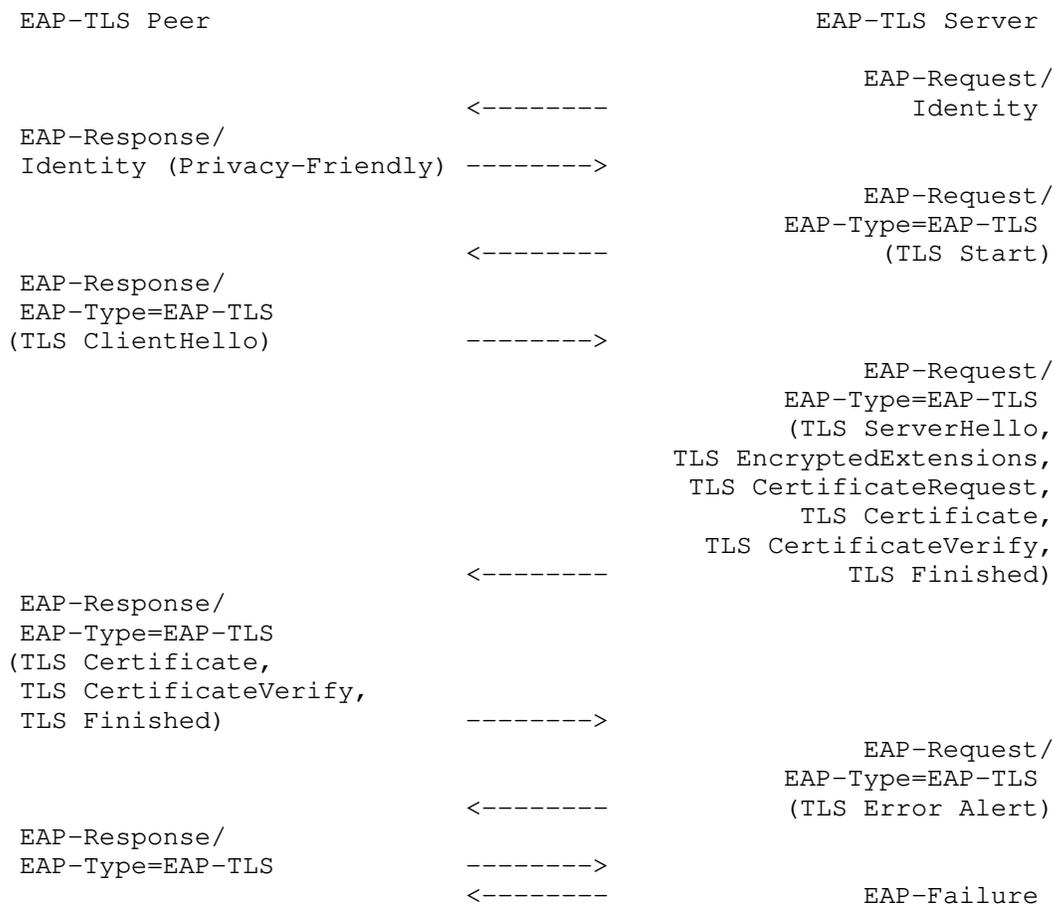


Figure 6: EAP-TLS unsuccessful client authentication

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]).

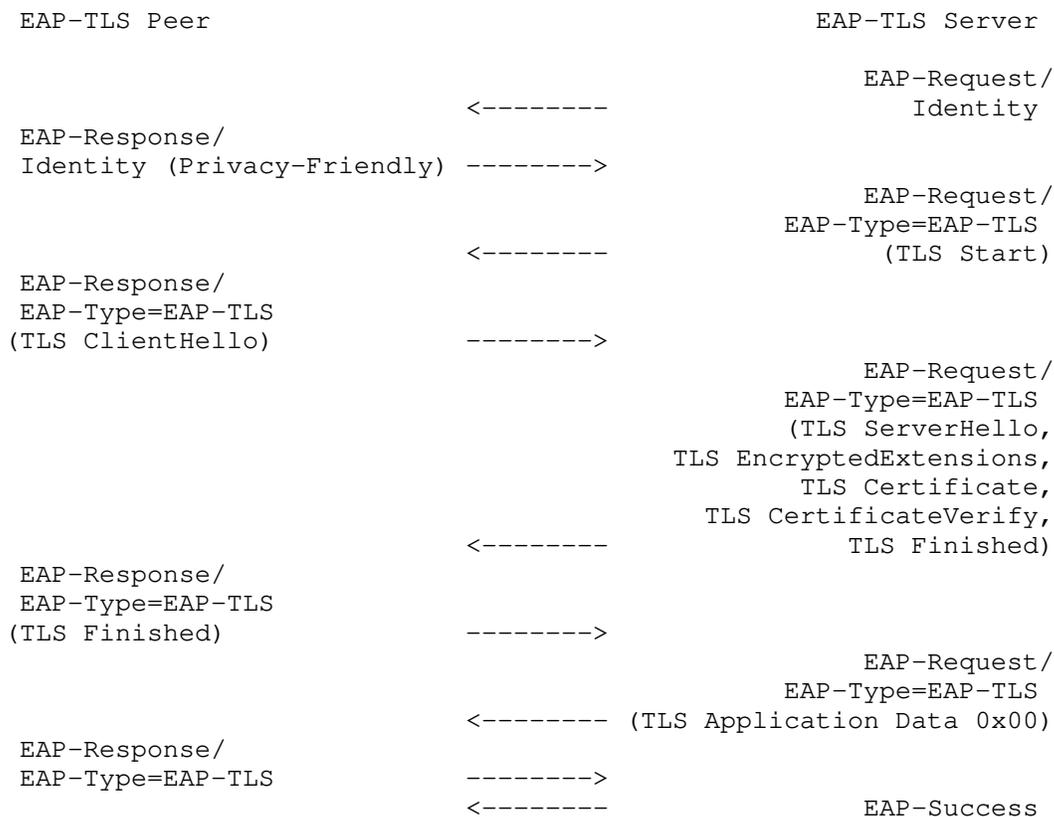


Figure 7: EAP-TLS without peer authentication

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.

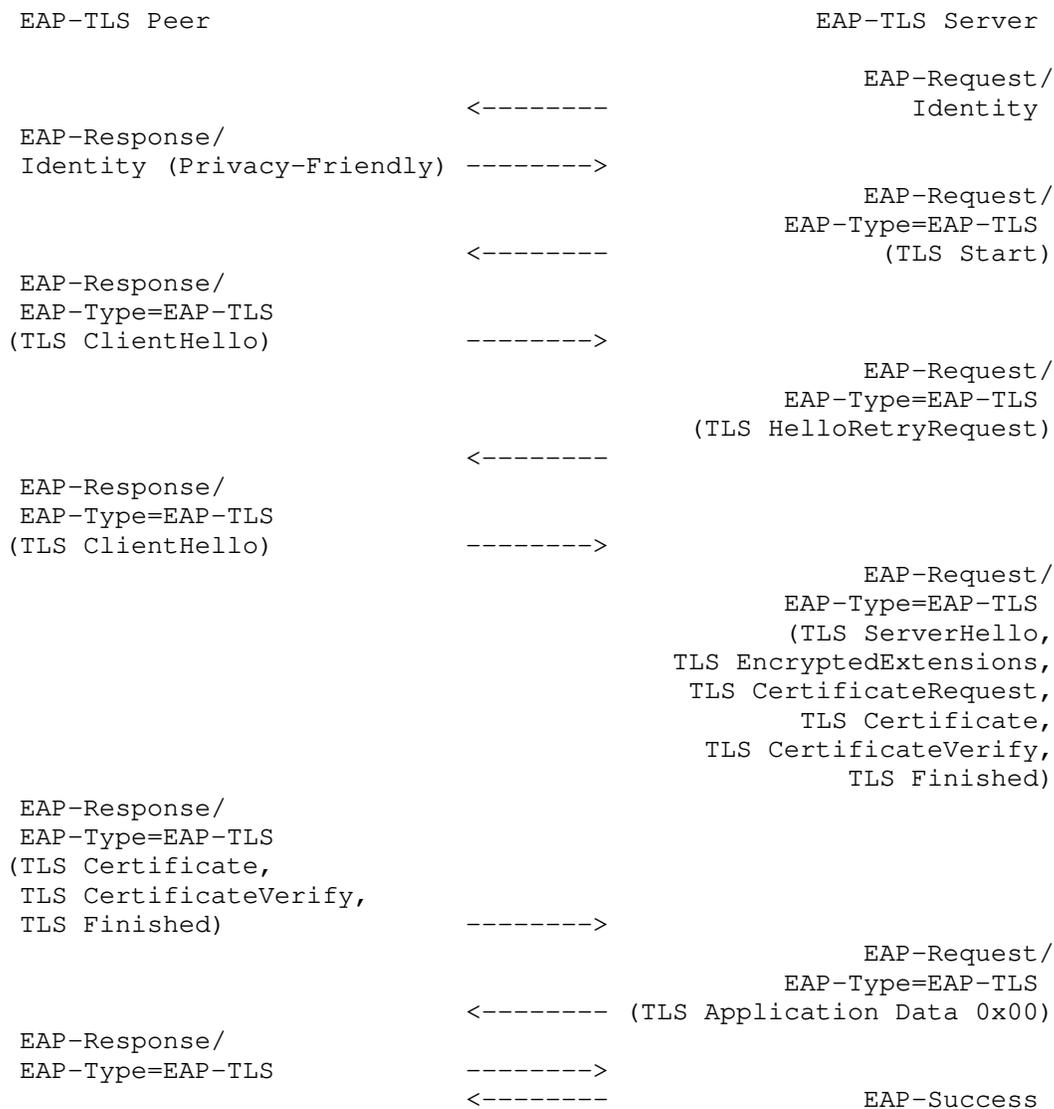


Figure 8: EAP-TLS with Hello Retry Request

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. Levkowitz, 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-TLSv0)", 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. Kroeselberg, "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-eaptlscert]
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-eaptlscert-08, 20 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-emu-eaptlscert-08.txt>>.
- [I-D.ietf-tls-ticketrequests]
Pauly, T., Schinazi, D., and C. A. Wood, "TLS Ticket Requests", Work in Progress, Internet-Draft, draft-ietf-tls-ticketrequests-07, 3 December 2020, <<https://www.ietf.org/archive/id/draft-ietf-tls-ticketrequests-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
Obsoletes: 5448 (if approved)
Updates: 4187 (if approved)
Intended status: Informational
Expires: April 22, 2019

J. Arkko
V. Lehtovirta
V. Torvinen
Ericsson
P. Eronen
Nokia
October 19, 2018

Improved Extensible Authentication Protocol Method for 3rd Generation
Authentication and Key Agreement (EAP-AKA')
draft-ietf-emu-rfc5448bis-03

Abstract

This specification defines an EAP method, EAP-AKA', a small revision of the EAP-AKA method. EAP-AKA' provides a key derivation function that binds the keys derived within the method to the name of the access network. The key derivation mechanism has been defined in the 3rd Generation Partnership Project (3GPP). This specification allows its use in EAP in an interoperable manner. In addition, EAP-AKA' employs SHA-256 instead of SHA-1.

This specification also updates RFC 4187 EAP-AKA to prevent bidding down attacks from EAP-AKA'.

This version of the EAP-AKA' specification provides updates to specify the protocol behaviour for 5G deployments as well.

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 April 22, 2019.

Copyright Notice

Copyright (c) 2018 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 Generation	13
3.4. Hash Functions	15
3.4.1. PRF'	15
3.4.2. AT_MAC	15
3.4.3. AT_CHECKCODE	15
4. Bidding Down Prevention for EAP-AKA	16
5. Peer Identities	17
5.1. Username Types in EAP-AKA' Identities	18
5.2. Generating Pseudonyms and Fast Re-Authentication Identities	18
5.3. Identifier Usage in 5G	19
5.3.1. Key Derivation	20
5.3.2. EAP Identity Response and EAP-AKA' AT_IDENTITY Attribute	21
6. Exported Parameters	23
7. Security Considerations	23
7.1. Privacy	26
7.2. Discovered Vulnerabilities	28
7.3. Pervasive Monitoring	29
7.4. Security Properties of Binding Network Names	30
8. IANA Considerations	31
8.1. Type Value	31
8.2. Attribute Type Values	31
8.3. Key Derivation Function Namespace	32
9. References	32
9.1. Normative References	32

9.2. Informative References	34
Appendix A. Changes from RFC 5448	36
Appendix B. Changes from RFC 4187 to RFC 5448	37
Appendix C. Changes from Previous Version of This Draft	37
Appendix D. Importance of Explicit Negotiation	38
Appendix E. Test Vectors	39
Appendix F. Contributors	43
Appendix G. Acknowledgments	44
Authors' Addresses	44

1. Introduction

This specification defines an Extensible Authentication Protocol (EAP) [RFC3748] method, EAP-AKA', a small revision of the EAP-AKA method originally defined in [RFC4187]. EAP-AKA' provides a new key derivation function, specified in [TS-3GPP.33.402]. 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. This specification defines the EAP encapsulation for AKA when the new key derivation mechanism is in use.

3GPP has defined a number of applications for the revised AKA mechanism, some based on native encapsulation of AKA over 3GPP radio access networks and others based on the use of EAP.

For making the new key derivation mechanisms usable in EAP-AKA, additional protocol mechanisms are necessary. Given that RFC 4187 calls for the use of CK (the encryption key) and IK (the integrity key) from AKA, existing implementations continue to use these. 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. The change must also be secure against bidding down attacks that attempt to force the participants to use the least secure mechanism.

This specification therefore introduces a variant of the EAP-AKA method, called EAP-AKA'. This method can employ the derived keys CK' and IK' from the 3GPP specification and updates the used hash function to SHA-256 [FIPS.180-4]. But it is otherwise equivalent to RFC 4187. 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: Appendix D explains why it is important to be explicit about the change of semantics for the keys, and why other approaches would lead to severe interoperability problems.

This version of the EAP-AKA' specification obsoletes RFC 5448. The changes are as follows:

- o Update the reference on how the Network Name field is constructed in the protocol. The 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 generation. 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'.

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: This specification refers only to the 5G specifications. Any further update that affects, for instance, key generation is something that EAP-AKA' implementations should take into account. Upon such updates there will be a need to both update the 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 EAP-AKA' identifiers are used in 5G 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.

Editor's Note: The publication of this RFC depends on its normative references [TS-3GPP.24.302] and [TS-3GPP.33.501] reaching a stable status for Release 15, as indicated by 3GPP. This is expected to happen shortly. The RFC Editor should check with the 3GPP liaisons that this has happened. RFC Editor: Please delete this note upon publication of this specification as an RFC.

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 a new EAP method that follows the EAP-AKA specification [RFC4187] in all respects except the following:

- o It uses the Type code 50, not 23 (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, not SHA-1 [FIPS.180-4] (Section 3.4).

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_RAND, AT_AUTN, AT_MAC, and AT_RES can be found in [RFC4187].

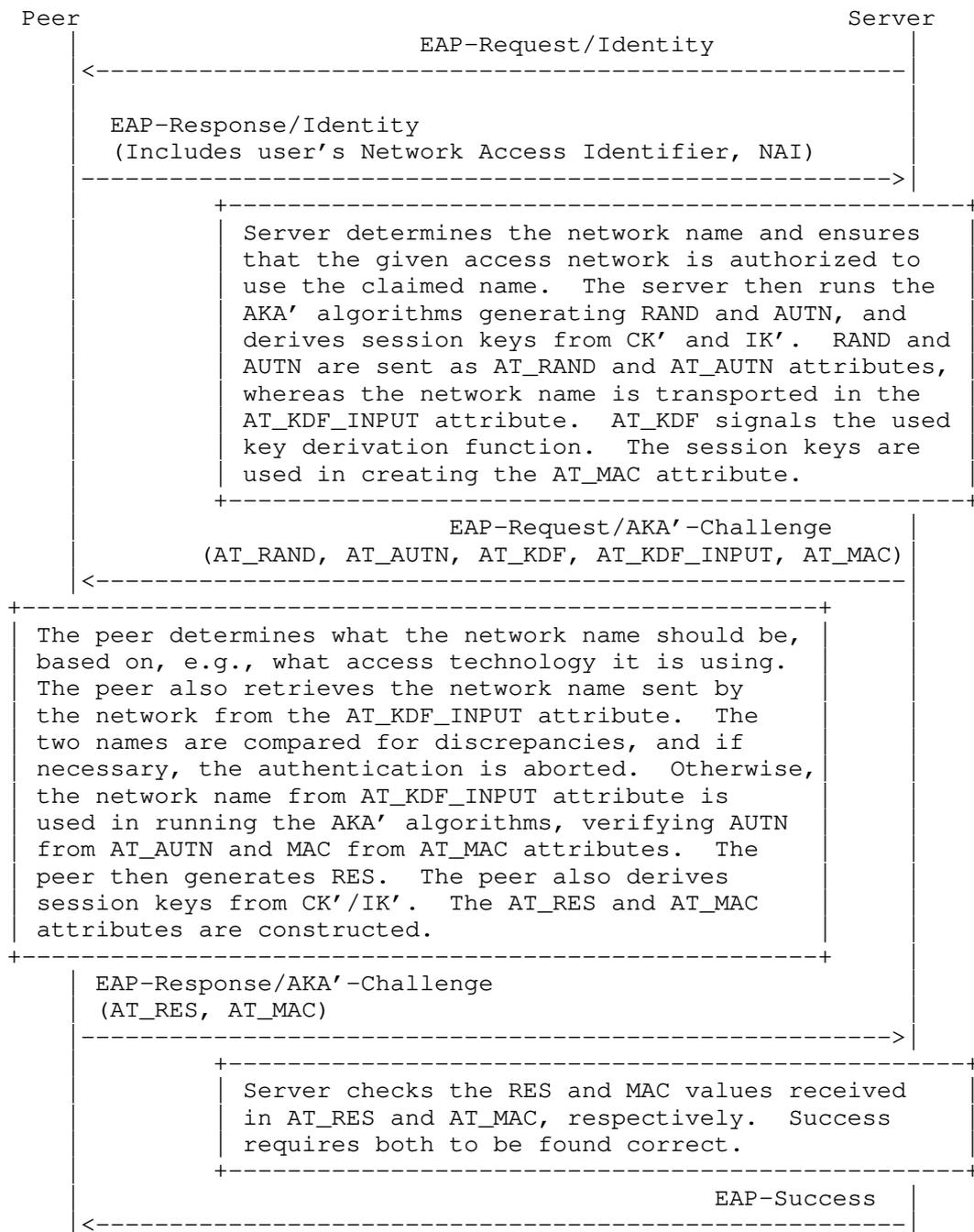


Figure 1: EAP-AKA' Authentication Process

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 non-3GPP access networks, and as specified in [TS-3GPP.33.501] 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. If it 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.

Note: Currently, [TS-3GPP.24.302] or [TS-3GPP.33.501] specify separate values. The former specifies what is called "Access Network ID" and the latter specifies what is called "Serving Network Name". However, from an EAP-AKA' perspective both occupy the same field, and need to be distinguishable from each other. Currently specified values are distinguishable, but it would be useful that this be specified explicitly in the 3GPP specifications.

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.

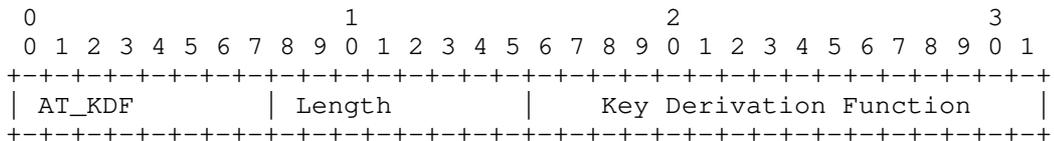
The AT_KDF_INPUT attribute MUST be sent and processed as explained above when AT_KDF attribute has the value 1. Future definitions of

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, 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. 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. An AKA'-Synchronization-Failure message is sent as a response to the newly received EAP-Request/AKA'-Challenge (the last message of the AT_KDF negotiation). 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 Generation

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

AT_KDF set to 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. 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].

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, not SHA-1 (see [FIPS.180-4]) 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 [RFC4306]). The function takes two arguments. K is a 256-bit value and S is an octet string of arbitrary length. PRF' is defined as follows:

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

where:

$$\begin{aligned} T1 &= \text{HMAC-SHA-256}(K, S \mid 0x01) \\ T2 &= \text{HMAC-SHA-256}(K, T1 \mid S \mid 0x02) \\ T3 &= \text{HMAC-SHA-256}(K, T2 \mid S \mid 0x03) \\ T4 &= \text{HMAC-SHA-256}(K, T3 \mid S \mid 0x04) \\ &\dots \end{aligned}$$

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

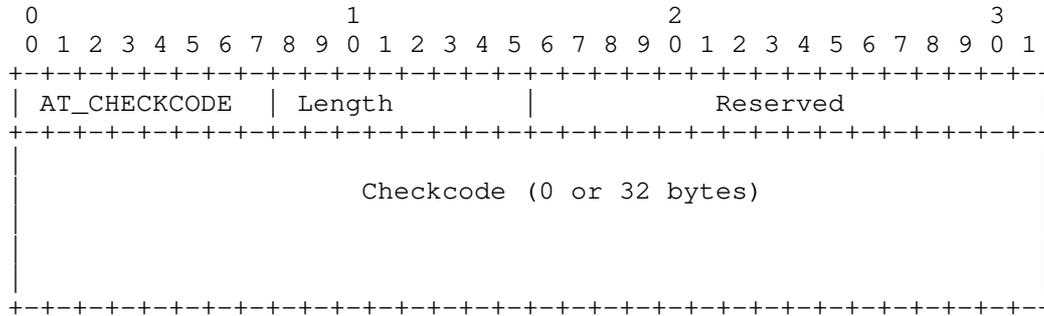
3.4.2. AT_MAC

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:



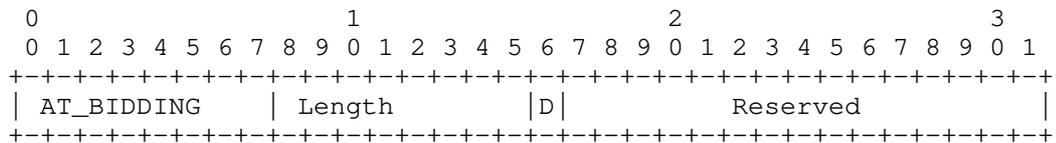
Second, the checkcode is a hash value, calculated with SHA-256 [FIPS.180-4], over the data specified in Section 10.13 of [RFC4187].

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 as defined in this RFC. It is only included in EAP-AKA messages. This 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, 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, there is no need to prevent bidding "down" attacks in the other direction, i.e., attackers forcing the endpoints to use EAP-AKA'.

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, which may further help the tracking, because the same identifier may be used in other contexts as well. 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'. 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 privacy identifiers (see Section 5.3.2 and Section 5.3.2.1.

(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

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 ensure 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.

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 this RFC.

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, two areas need further specification in EAP-AKA' to ensure that different implementations understand each other and stay interoperable:

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

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.

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 the 5G SUPI for the peer. This rule applies to all full EAP-AKA' authentication processes, even if the peer sent some other identifier at a lower layer or as a response to an EAP Identity Request or if no identity was sent.

The identity MUST also be represented in the exact correct format for the key derivation formula to produce correct results. For the SUPI, this format is as defined Section 5.3.1.1.

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. Again, the identity MUST be used exactly as sent.

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.

5.3.1.1. Format of the SUPI

A SUPI is either an IMSI or a Network Access Identifier [RFC4282].

When used in EAP-AKA', the format of the SUPI MUST be as specified in [TS-3GPP.23.003] Sections 28.7.2, 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 entire IMSI.

For instance, if the IMSI is 234150999999999 (MCC = 234, MNC = 15), the NAI format for the SUPI takes the form:

```
234150999999999@nai.5gc.mnc015.mcc234.3gppnetwork.org
```

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 peer is connecting to a 5G access network and uses the 5G Non-Access Stratum (NAS) protocol [TS-3GPP.24.501], the EAP server is in a 5G network. The EAP identity exchanges are generally not used in this case, as the identity is already made available on previous link layer exchanges.

In this situation, the EAP server SHOULD NOT request an additional identity from the peer. If the peer for some reason receives EAP-Request/Identity or EAP-Request/AKA-Identity messages, the peer should behave as follows.

Receive EAP-Request/Identity

In this case, the peer SHOULD respond with a EAP-Response/Identity containing the privacy-friendly 5G identifier, the SUCI. The SUCI SHOULD be represented as specified in Section 5.3.2.1.

EAP-Request/AKA-Identity with AT_PERMANENT_REQ

For privacy reasons, the peer should follow a "conservative" policy and terminate the authentication exchange rather than risk revealing its permanent identity.

The peer SHOULD respond with EAP-Response/AKA-Client-Error with the client error code 0, "unable to process packet".

EAP-Request/AKA-Identity with AT_FULLAUTH_REQ

In this case, the peer SHOULD respond with a EAP-Response/AKA-Identity containing the SUCI. The SUCI SHOULD be represented as specified in Section 5.3.2.1.

EAP-Request/AKA-Identity with AT_ANY_ID_REQ

If the peer supports fast re-authentication and has a fast re-authentication identity available, the peer SHOULD respond with EAP-Response/AKA-Identity containing the fast re-authentication identity. Otherwise the peer SHOULD respond with a EAP-Response/AKA-Identity containing the SUCI, and SHOULD represent the SUCI as specified in Section 5.3.2.1.

Similarly, if the peer is communicating over a non-3GPP network but carrying EAP inside 5G NAS protocol, it MUST assume that the EAP server is in a 5G network, and again employ the SUCI within EAP.

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

5.3.2.1. Format of the SUCI

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 instance, assuming the IMSI 234150999999999, where MCC=234, MNC=15 and MSISN=0999999999, the Routing Indicator 678, and a Home Network Public Key Identifier of 27, the NAI format for the SUCI takes the form:

For the null-scheme:

```
type0.rid678.schid0.userid0999999999@nai.5gc.mnc015.  
mcc234.3gppnetwork.org
```

For the Profile <A> protection scheme:

```
type0.rid678.schid1.hnkey27.ecckey<ECC ephemeral public key>.  
cip<encryption of 0999999999>.mac<MAC tag value>@nai.5gc.  
mnc015.mcc234.3gppnetwork.org
```

6. Exported Parameters

The EAP-AKA' Session-Id is the concatenation of the EAP Type Code (50, one octet) 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:

$$\text{Session-Id} = 50 \parallel \text{RAND} \parallel \text{AUTN}$$

When using fast re-authentication, the EAP-AKA' Session-Id is the concatenation of the EAP Type Code (50) 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:

$$\text{Session-Id} = 50 \parallel \text{NONCE_S} \parallel \text{MAC}$$

The Peer-Id is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length octets 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 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 SHA-256 is at least as secure as SHA-1. 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. 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.

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, SHA-256, is used instead of 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 [RFC4306] 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 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 5G SUCI identifier should be used. The use of pseudonyms in this situation is at best limited. 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 employ permanent identifier, SUPI, as an input to key generation. However, this use of the SUPI is only internal and the SUPI need not be communicated in EAP messages. SUCI 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.23.501] 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, there is a full choice to use 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 also not be used for other purposes (e.g. in other protocols).

If the peer and server cannot guarantee that 5G SUCI can be used or pseudonyms will available, generated properly, and maintained reliably, and identity privacy is required then additional protection

from an external security mechanism such as tunneled EAP methods 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 has 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 where, for instance, it is not required to encrypt all payload traffic after encryption. 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.

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 has 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.

Finally, while this is not a problem with the protocol itself, bad implementations may not produce pseudonym usernames or fast re-authentication identities in a manner that is sufficiently secure. Recommendations from Section 5.2 need to be followed to avoid this.

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 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 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 an backwards-compatible extension of EAP-AKA', and is pursued separately from this specification [I-D.arkko-eap-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

8.1. Type Value

EAP-AKA' has the EAP Type value 50 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 15)", 3GPP Draft Technical Specification 23.003, September 2018.
- [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 15)", 3GPP Technical Specification 23.501, September 2018.
- [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 15)", 3GPP Draft Technical Specification 24.302, September 2018.

- [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 15)", 3GPP Draft Technical Specification 24.501, September 2018.
- [TS-3GPP.33.102]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture (Release 15)", 3GPP Draft Technical Specification 33.102, June 2018.
- [TS-3GPP.33.402]
3GPP, "3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses (Release 15)", 3GPP Draft Technical Specification 33.402, June 2018.
- [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 15)", 3GPP Draft Technical Specification 33.501, September 2018.
- [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. Levkowitz, 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>>.
- [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>>.
- [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 f1, f1*, f2, f3, f4, f5 and f5*; Document 4: Design Conformance Test Data (Release 14)", 3GPP Technical Specification 35.208, March 2017.
- [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>>.

- [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>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [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>>.
- [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>>.
- [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>>.

- [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>>.
- [I-D.arkko-eap-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-arkko-eap-aka-pfs-02 (work in progress), July 2018.
- [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.

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.

Finally, the references to [RFC2119], [RFC5226], [FIPS.180-1] and [FIPS.180-2] have been updated to their most recent versions and language in this document changed accordingly. Similarly, references to all 3GPP technical specifications have been updated to their 5G (Release 15) versions or otherwise most recent version when there has not been a 5G-related update.

Appendix B. Changes from RFC 4187 to RFC 5448

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 in RFC 4187 (it uses CK and IK, not CK' and IK'); neither is any processing of the AMF bit added to RFC 4187.

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).

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 mechanism. 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 7dd1 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 cee a 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

```

Appendix F. 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.

Appendix G. Acknowledgments

The authors would like to thank Guenther Horn, Joe Salowey, Mats Naslund, Adrian Escott, Brian Rosenberg, Laksminath 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, 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
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

Email: pasi.eronen@nokia.com

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_RAND, AT_AUTN, AT_MAC, and AT_RES can be found in [RFC4187].

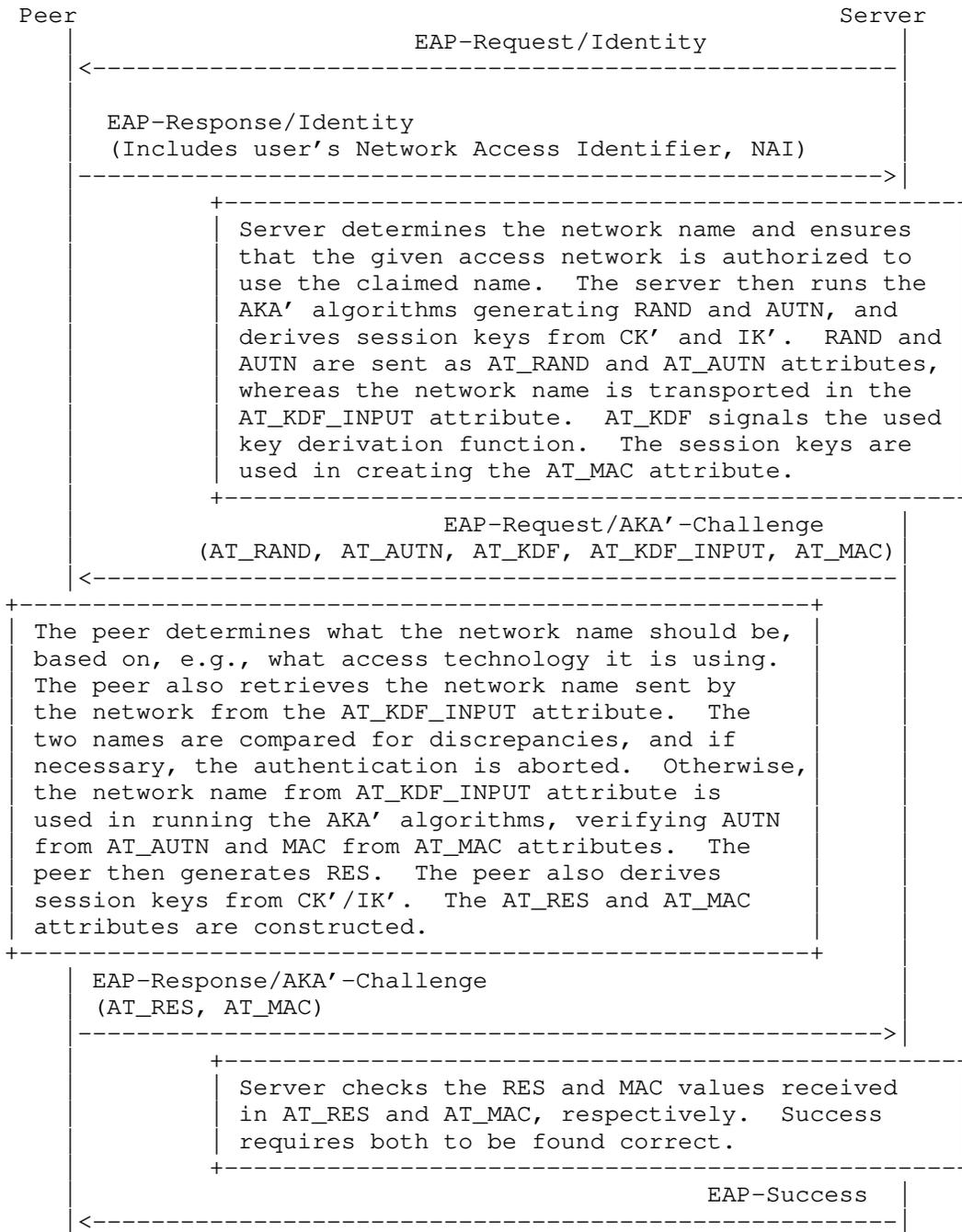


Figure 1: EAP-AKA' Authentication Process

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.

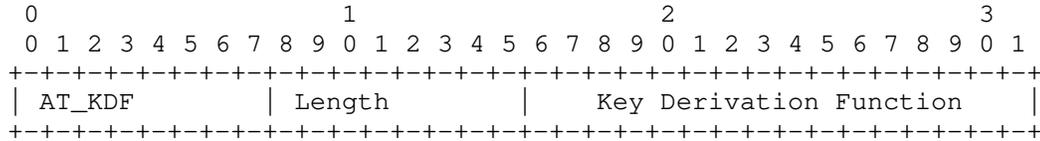
The allocation procedures in [TS-3GPP.24.302] ensure that conflicts potentially arising from using the same name in different types of networks are avoided. The specification also has detailed rules about how a client can determine these based on information available to the client, such as the type 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:

$$\begin{aligned} T1 &= \text{HMAC-SHA-256}(K, S \mid 0x01) \\ T2 &= \text{HMAC-SHA-256}(K, T1 \mid S \mid 0x02) \\ T3 &= \text{HMAC-SHA-256}(K, T2 \mid S \mid 0x03) \\ T4 &= \text{HMAC-SHA-256}(K, T3 \mid S \mid 0x04) \\ &\dots \end{aligned}$$

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

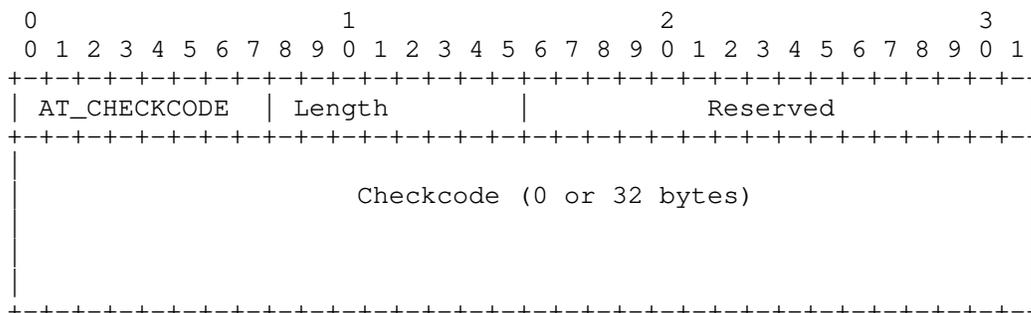
3.4.2. AT_MAC

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

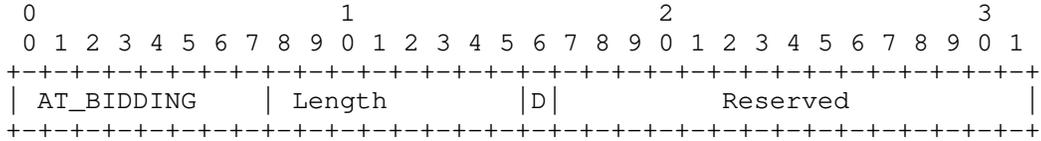
Table 1: The attribute table

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_FULLLAUTH_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:

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

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. Levkowitz, 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 f1, f1*, 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-TLSv0)", 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 led 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 7dd1 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, Laksminath 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
Intended status: Standards Track
Expires: April 21, 2019

E. Lear
O. Friel
N. Cam-Winget
Cisco Systems
October 18, 2018

Bootstrapping Key Infrastructure over EAP
draft-lear-eap-teap-brski-01

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 April 21, 2019.

Copyright Notice

Copyright (c) 2018 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
1.1.	Terminology	3
2.	TEAP BRSKI Architecture	3
3.	BRSKI Bootstrap and Enroll Operation	4
3.1.	Executing BRSKI in a TEAP Tunnel	5
4.	PKI Certificate Authority Considerations	8
4.1.	TEAP Tunnel Establishment	8
4.2.	BRSKI Trust Establishment	9
5.	Channel and Crypto Binding	10
6.	Protocol Flows	10
6.1.	TEAP Server Grants Access	10
6.2.	TEAP Server Instructs Client to Perform BRSKI Flow	12
6.3.	TEAP Server Instructs Client to Reenroll	13
6.4.	Out of Band Reenroll	14
7.	TEAP TLV Formats	14
7.1.	BRSKI TLVs	14
7.1.1.	BRSKI-RequestVoucher TLV	15
7.1.2.	BRSKI-Voucher TLV	16
7.1.3.	CSR-Attributes TLV	16
7.2.	Existing TEAP TLV Specifications	17
7.2.1.	PKCS#10 TLV	17
8.	Fragmentation	17
9.	IANA Considerations	17
10.	Security Considerations	18
11.	Acknowledgments	18
12.	Informative References	18
	Appendix A. Changes from Earlier Versions	19
	Authors' Addresses	19

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.

- o 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.
- o EST: Enrollment over Secure Transport, as defined in [RFC7030].
- o Voucher: a signed JSON object as defined in [RFC8366].

2. TEAP BRSKI Architecture

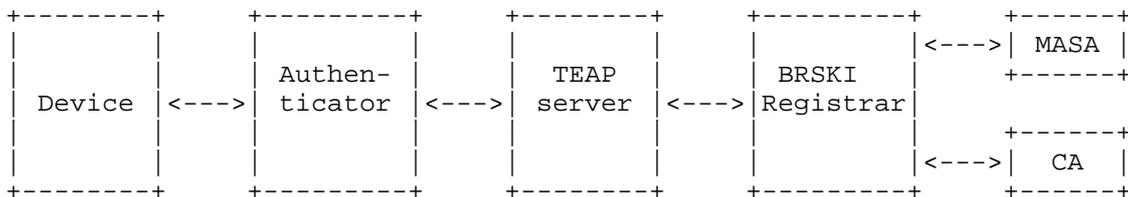
The TEAP BRSKI architecture is illustrated in Section 3. The device talks to the TEAP server via the Authenticator as per any normal EAP exchange. There is no need for an inner EAP method server, and there is no explicit EAP method type defined for BRSKI.

The architecture illustrated shows the TEAP server and registrar function as being two logically separate entities, however the BRSKI registrar functionality may be integrated into the TEAP server. The device is not explicitly aware of where the registrar functionality is deployed when executing BRSKI inside a TEAP tunnel. Note that the

device may connect directly to the registrar for the purposes of certificate reenrollment, but this happens outside the context to 801.1X and TEAP authentication.

The registrar in turn communicates with the BRSKI MASA service for the purposes of getting signed vouchers. [[TODO: I guess we should mention TEAP server talking to vendor default registrar in the cloud]]

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:

- o Step 1: Device discovers the registrar
- o Step 2: Device establishes provisional TLS connection to registrar
- o Step 3: Device sends voucher request message and receives signed voucher response
- o Step 4: Device validates voucher and validates provisional TLS connection to registrar
- o Step 5: Device downloads additional local domain CA information
- o Step 6: Device downloads Certificate Signing Request (CSR) attributes
- o Step 7: Device does an EST enroll to obtain an LDevID

- o 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. 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:

- o BRSKI-VoucherRequest
- o BRSKI-Voucher
- o CSR-Attributes

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

- o 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.

- o 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. Similarly, at this provisioning stage, the server does not validate the certificate of the device. The device presents its LDevID as its identity certificate if it has a valid LDevID, otherwise it presents its IDevID. Server policy may also be used to control which certificate the device is allowed present, as described in section Section 4.

If the presented credential is sufficient to grant access, the TEAP server can return an EAP-Success immediately. The device may still send a BRSKI-RequestVoucher TLV in response to the EAP-Success if it does not have, but requires, trust anchors for validating the TEAP server certificate.

If the TEAP server requires that the device execute a BRSKI flow, it 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.

The TEAP server may also require the device to reenroll, for example, if the device presented a valid LDevID that is very close 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.

- o 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).

- o 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 outer TEAP TLS connection to the TEAP server.

If the device fails to validate the voucher, or fails to validate the outer TEAP TLS connection, then it sends a TEAP-Error TLV indicating failure to the TEAP server.

- o 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.

- o 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.

- o Step 7: Device does an EST 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 PKCS#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.

- o 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 Authority Considerations

Careful consideration must be given to PKI certificate authority handling when:

- o Establishing the TEAP tunnel
- o Establishing trust using BRSKI

These are described in more detail here.

4.1. TEAP Tunnel Establishment

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. Network operators may want to do this in order to restrict network access to clients that have a certificate signed by one of a small set of trusted manufacturer/supplier CAs. 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.

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. Operators may prefer to include no CAs in the certificate_authorities field thus allowing devices to present IDevIDs signed by any CA when establishing the TEAP tunnel, and instead enforce policy at LDevID enrollment time.

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.

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

- o 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 IDevID without requiring the device to enroll to obtain an LDevID.
- o 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.
- o 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]

5. 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.

6. Protocol Flows

This section outlines protocol flows that map to the 3 server policy options described in section Section 4.1. The protocol flows illustrate a TLS1.2 exchange.

[[TODO]] MISSING EAP Start/identity/Crypto binding, etc. Flows illustrative, but not 100% accurate.

6.1. TEAP Server Grants Access

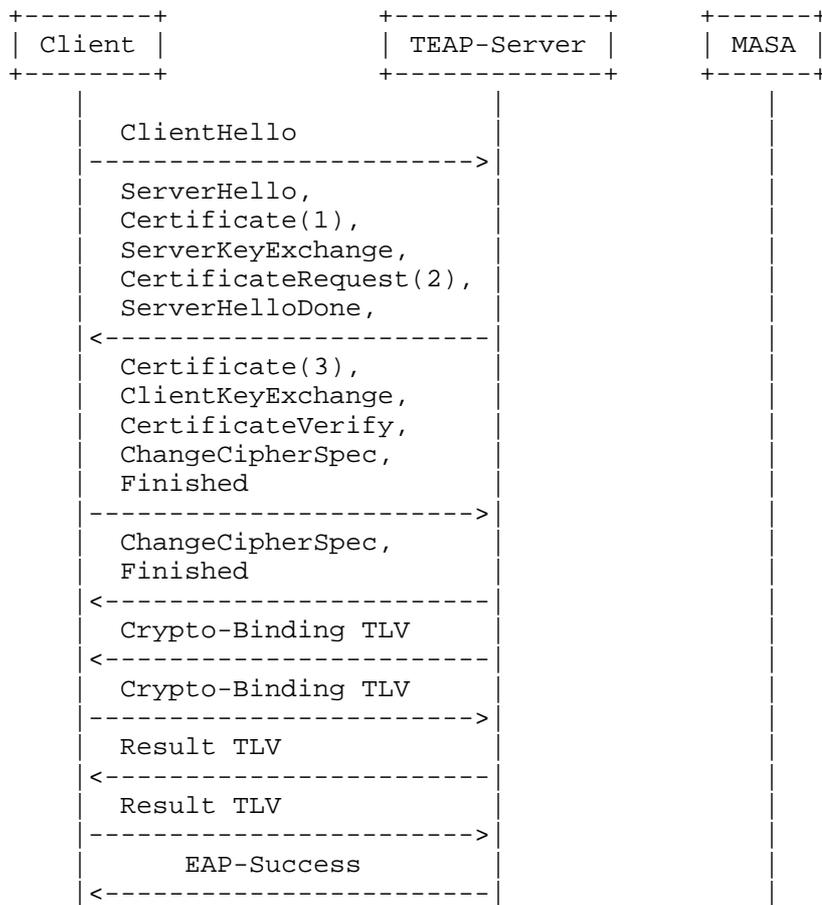


Figure 1: TEAP Server Grants Access

Notes:

(1) If the client has completed the BRSKI flow yet, 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.

6.2. TEAP Server Instructs Client to Perform BRSKI Flow

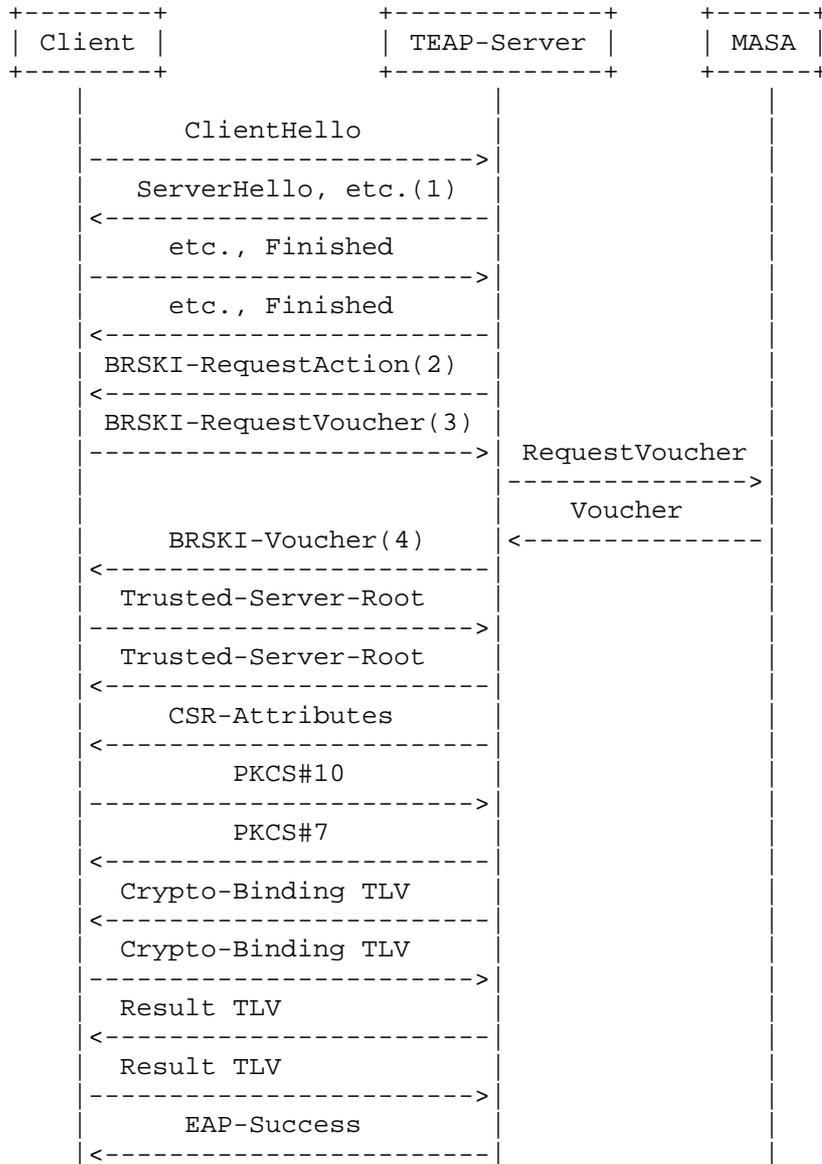


Figure 2: TEAP Server Instructs Client to Perform BRSKI Flow

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.

(2) The server instructs the client to start the BRSKI flow by sending a RequestAction TLV that includes a BRSKI-RequestVoucher TLV.

(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.

6.3. TEAP Server Instructs Client to Reenroll

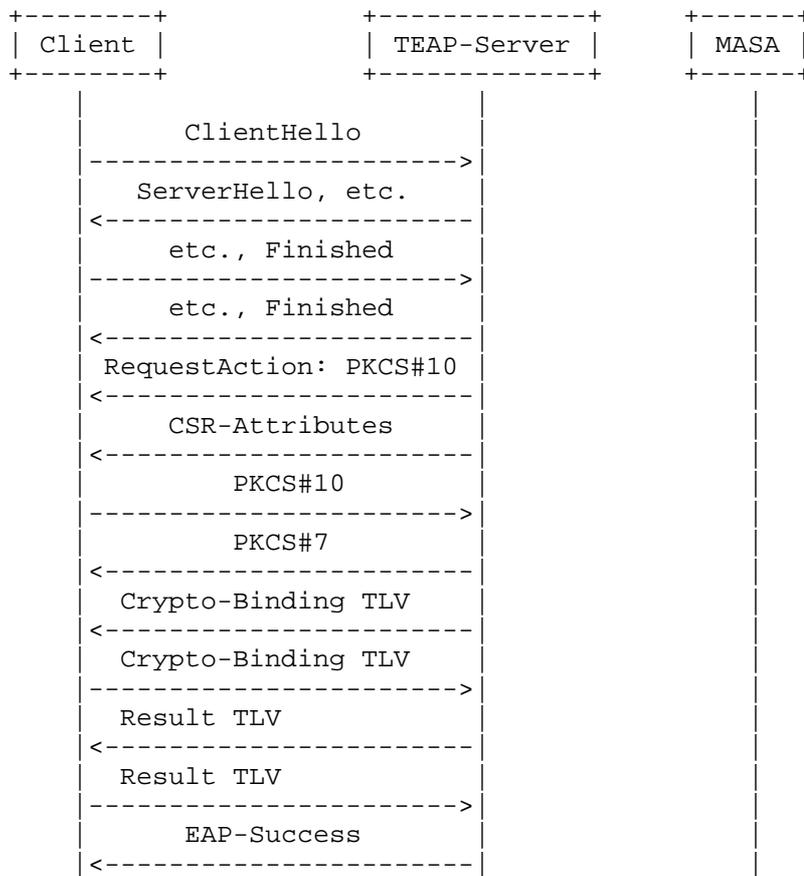


Figure 3: TEAP Server Instructs Client to Reenroll

6.4. 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.

7. TEAP TLV Formats

7.1. BRSKI TLVs

BRSKI defines 3 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.

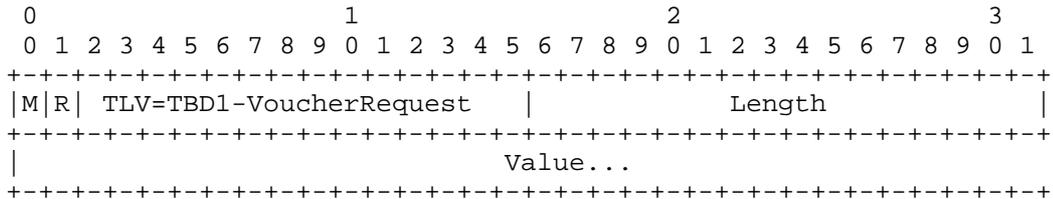
TLV	Message
BRSKI-VoucherRequest	Response
BRSKI-Voucher	Request
CSR-Attributes	Response

These new TLVs are detailed in this section.

7.1.1.1. BRSKI-RequestVoucher TLV

This TLV is used by the server as part of an Action-Request 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.

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].



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

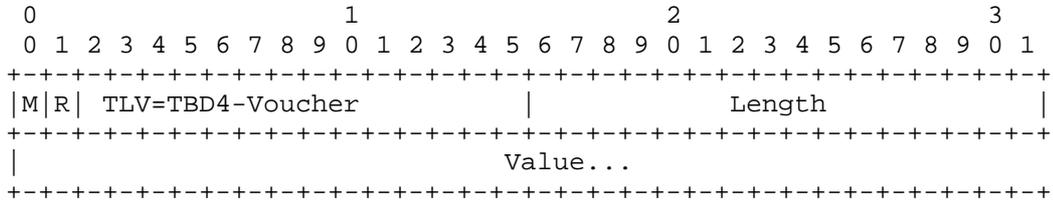
The server is expected to forward the voucher request to the MASA, and then return a voucher in a BRSKI-Voucher TLV as described below. If it is unable to do so, it returns a TEAP Error TLV with one of the defined errors or the following:

- 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 (XXX) minutes.

7.1.2. BRSKI-Voucher TLV

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



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.

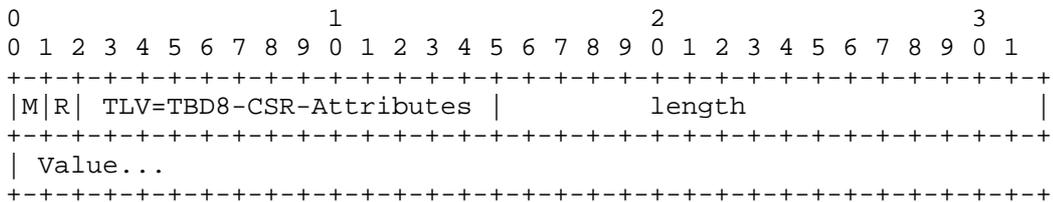
If successful, it installs the new trust anchor contained in the voucher.

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

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

7.1.3. 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].



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:

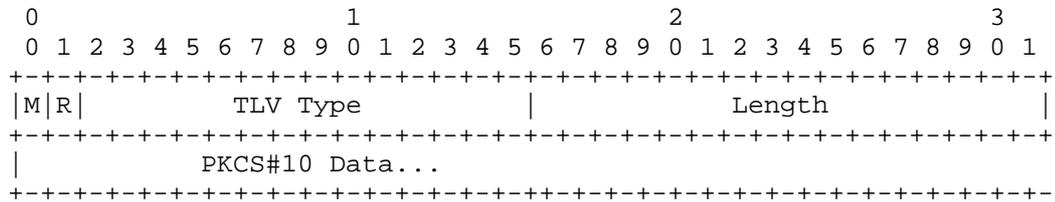
TBD9-CSR-Attribute-Fail Unable to supply the requested attributes.

7.2. 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.

7.2.1. 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. Fragmentation

TLS 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.

9. 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.

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 of the voucher signer is invalid
TBD6-Invalid-Voucher	The form or content of the voucher is not valid
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.

10. Security Considerations

There will be many.

11. Acknowledgments

The authors would like to thank Brian Weis for his assistance, and Alan Dakok for improving language consistency.

12. Informative References

- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-16 (work in progress), June 2018.
- [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>>.

Appendix A. Changes from Earlier Versions

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

Draft -00:

- o Initial revision

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
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

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].

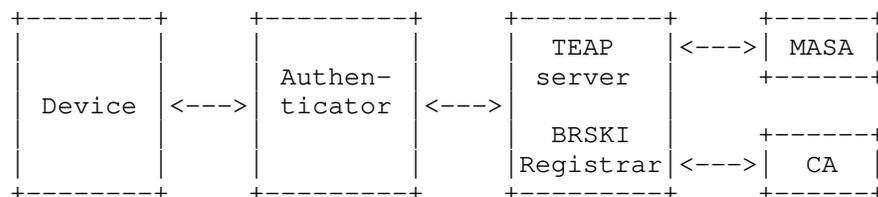
2. TEAP BRSKI Architecture

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 LDevID, 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

[IEEE8021AR] 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.

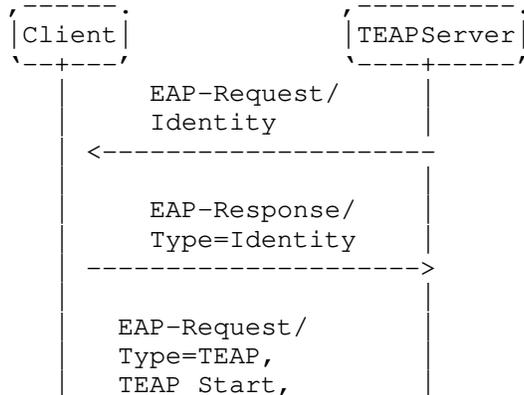
The TEAP BRSKI TLVs are expected to occur at the beginning of the TEAP Phase 2 and MUST occur before the final Crypto-Binding TLV. This draft does not exclude the possibility of having other EAP methods occur following the TEAP BRSKI TLVs and as such, the Crypto-Binding TLV process rules as defined in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



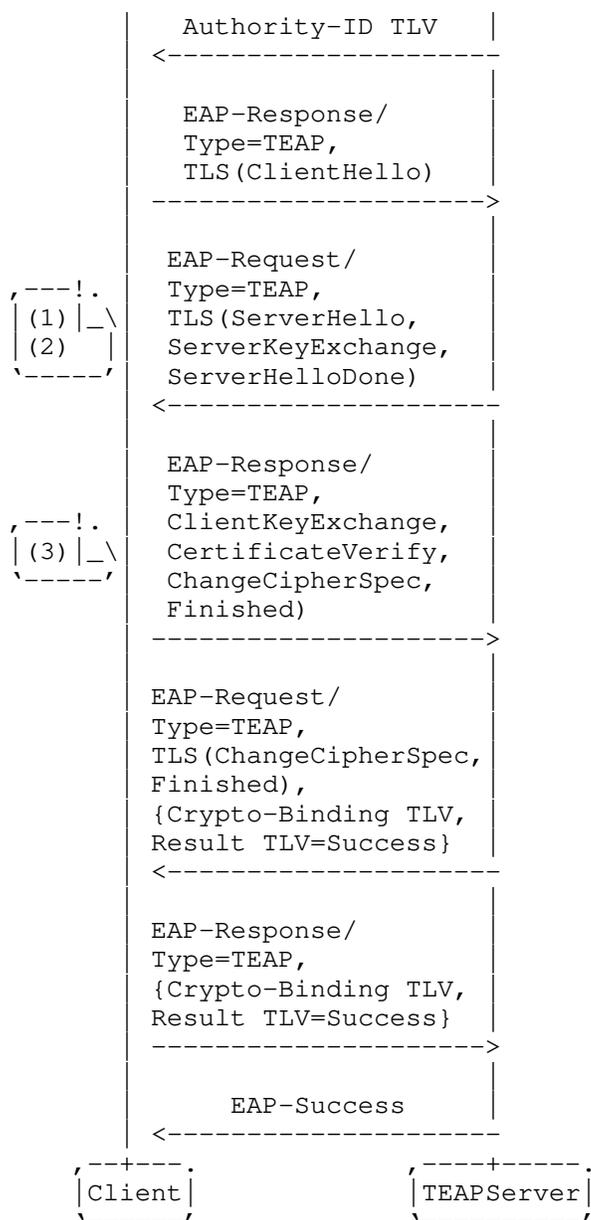


Figure 1: TEAP Server Grants Access

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.

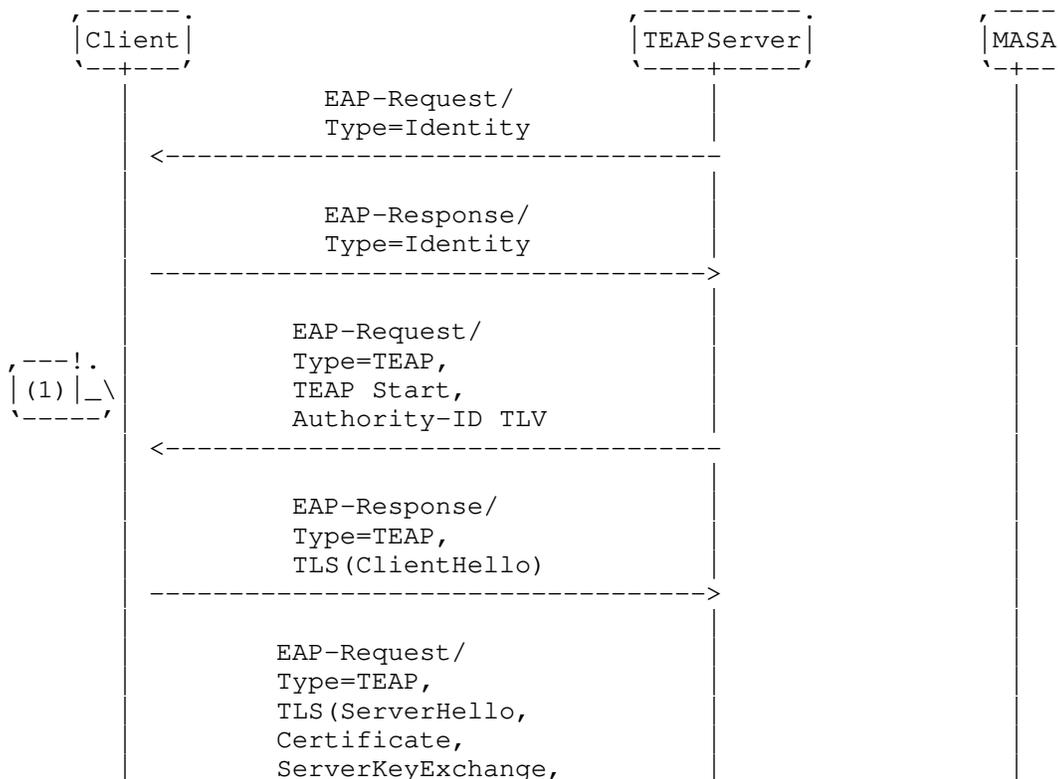
(2) The server may include certificate_authorities field in the CertificateRequest message in order to restrict the identity certificates that the device is allowed present.

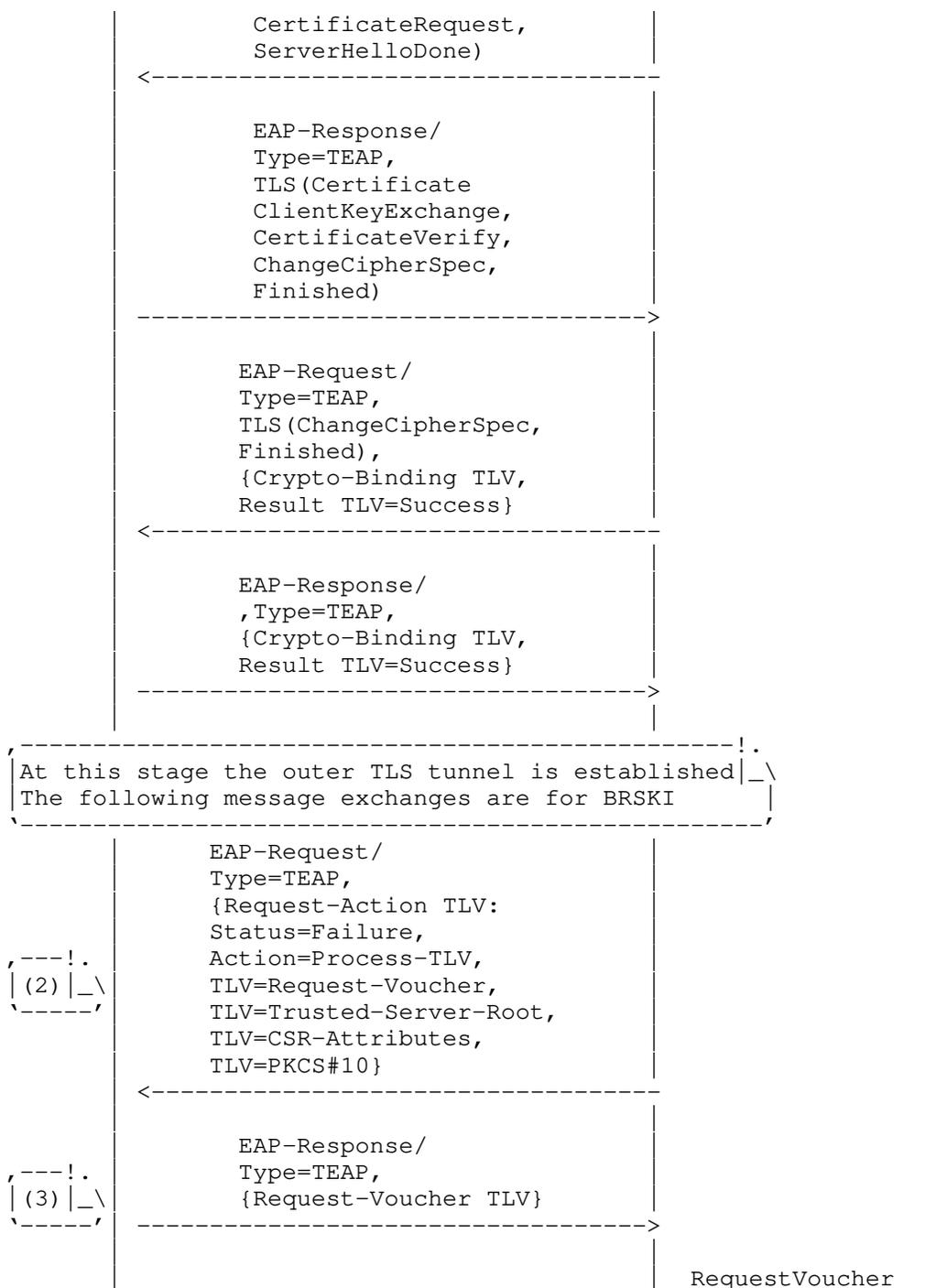
(3) The device will present its LDevID, if it has one, in preference to its IDevID, if allowed by server policy.

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.





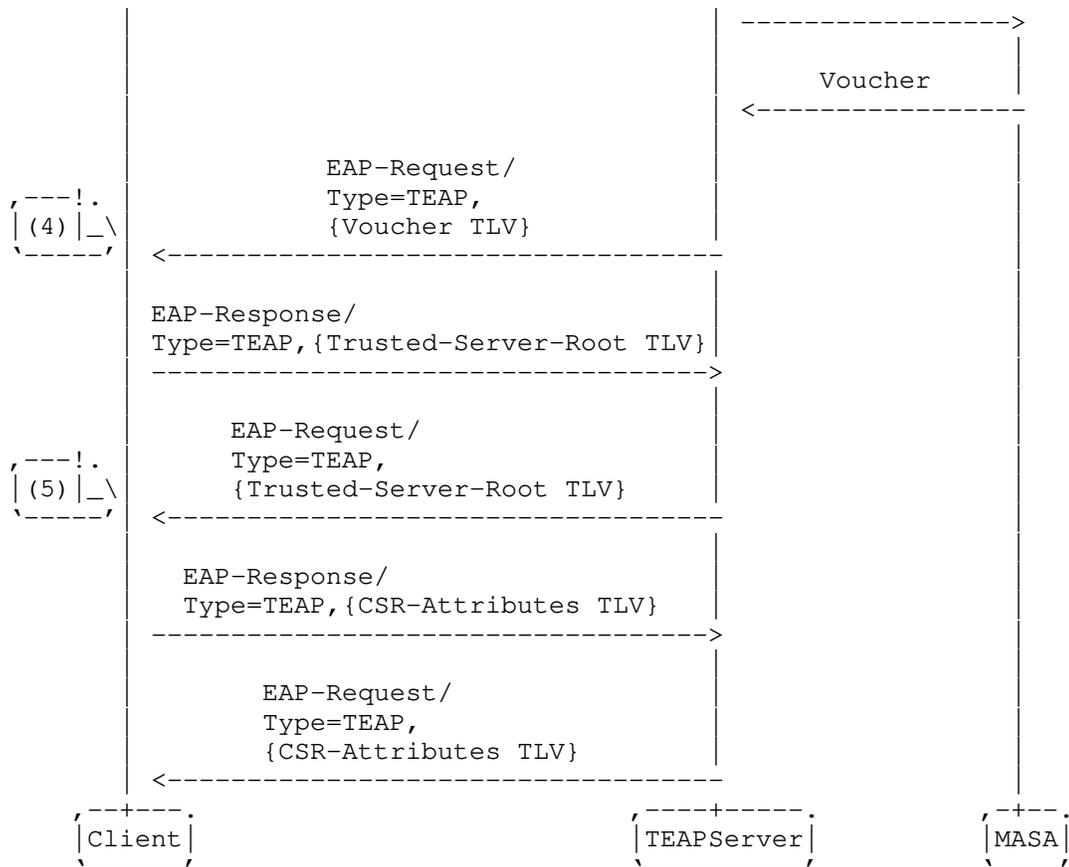


Figure 2: TEAP Server Instructs Client to Perform BRSKI Flow

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

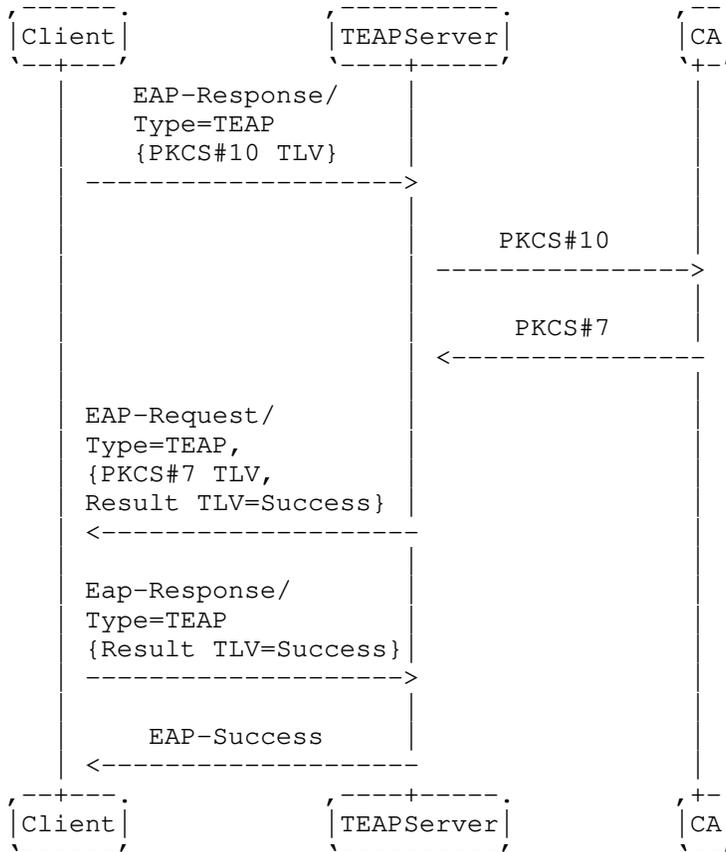


Figure 3: Enrollment after BRSKI Flow

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.

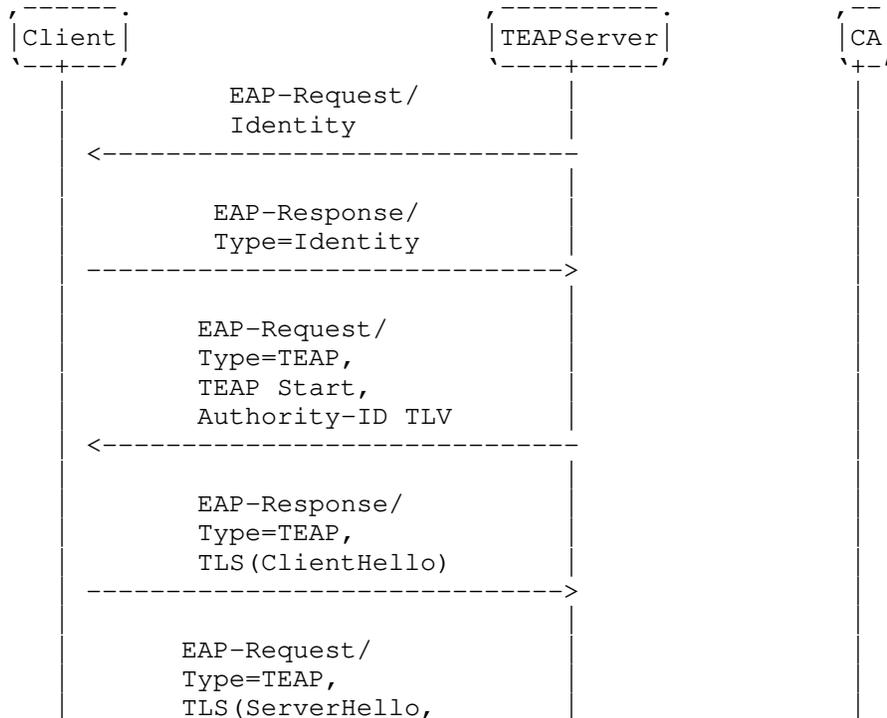
(4) Once the client receives and validates the voucher signed by the MASA, it must verify the certificate it previously received from the server.

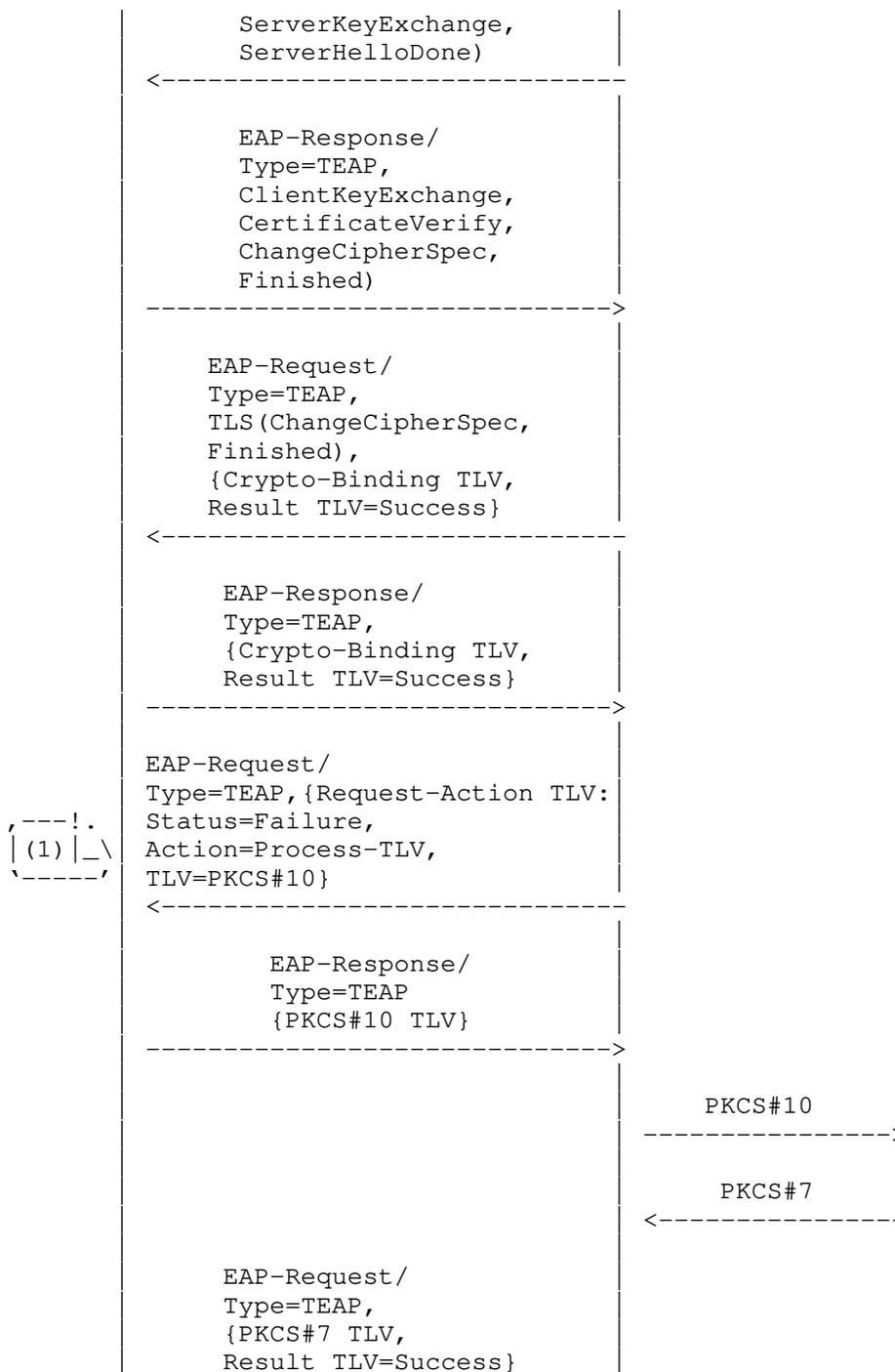
(5) As outlined in [RFC7170], the Trusted-Server-Root TLV is exchanged after the Crypto-Binding TLV exchange, and after the client has used the Voucher to authenticate the TEAP server identity. This is equivalent to section [todo] from [I-D.ietf-anima-bootstrapping-keyinfra].

(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.





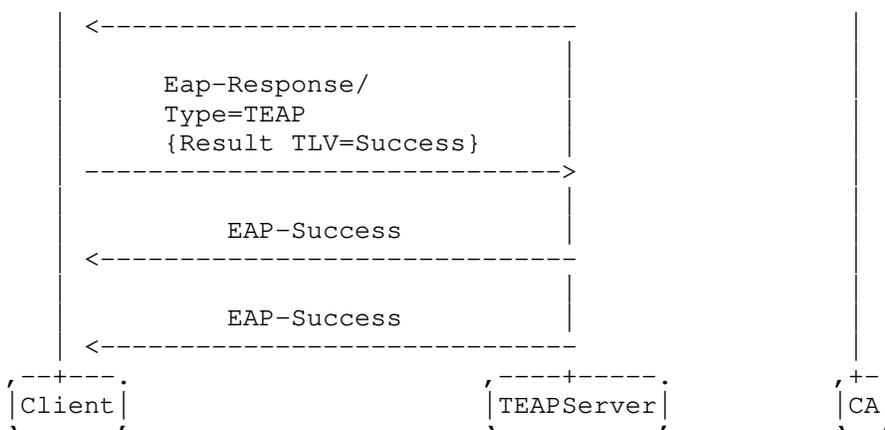


Figure 4: TEAP Server Instructs Client to Reenroll

(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.

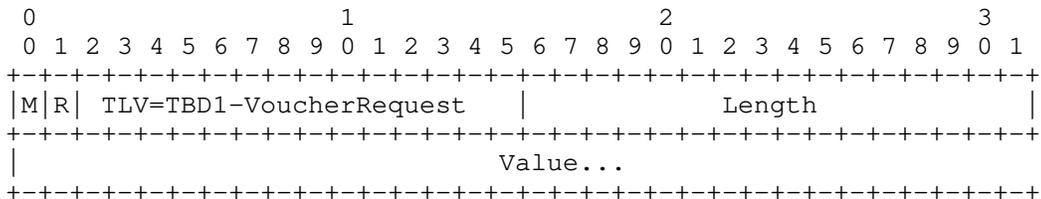
TLV	Message
BRSKI-VoucherRequest	Response
BRSKI-Voucher	Request
CSR-Attributes	Response
Retry-After	Response
NAI-Identity	Request

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].



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

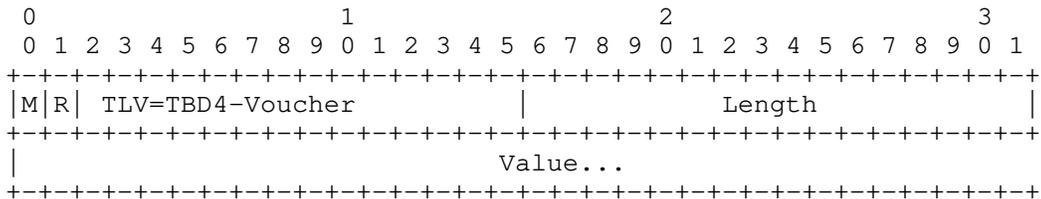
The server is expected to forward the voucher request to the MASA, and then return a voucher in a BRSKI-Voucher TLV as described below. If it is unable to do so, it returns an TEAP Error TLV with one of the defined errors or the following:

- 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].



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.

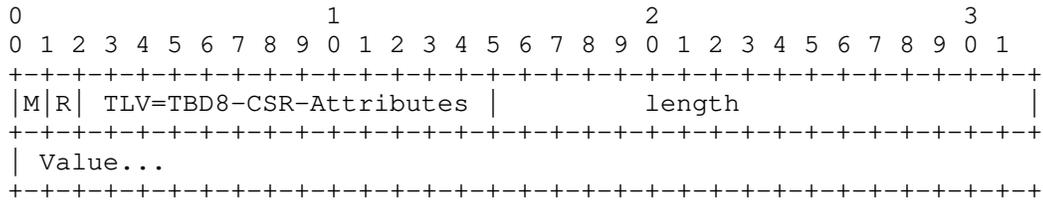
Otherwise, the peer transmits an TEAP error TLV with one of the following error messages:

- 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.

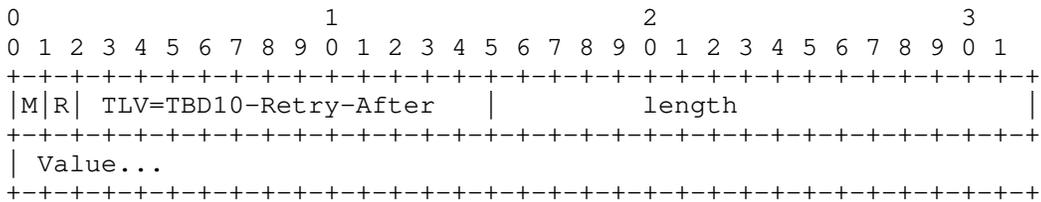


Again, the M and R values are set to 0. In the case where the client is unable to provide the requested attributes, an TEAP-Error is returned as follows:

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

8.1.4. Retry-After TLV

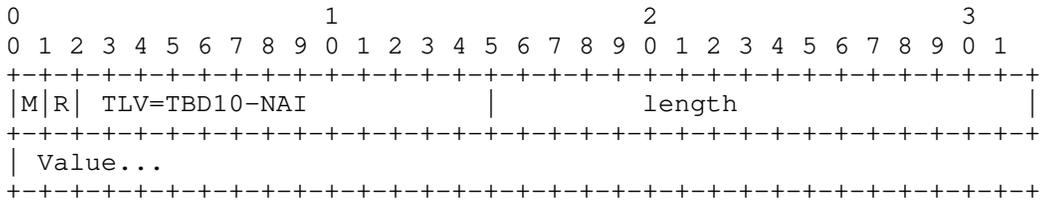
The server MUST transmit this TLV to the peer when replying to a PKCS#10 TLV request from the peer where the server is willing to fulfill the request and issue a certificate via a PKCS#7 response, but is unable to fulfill the request immediately. This TLV is used to tell the peer the minimum length of 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.



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.



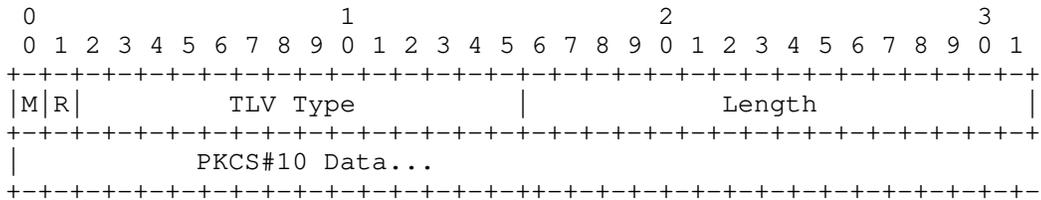
Again, the M and R values are set to 0.

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: April 25, 2019

M. Sethi
J. Mattsson
Ericsson
October 22, 2018

Handling Large Certificates and Long Certificate Chains in EAP-TLS
draft-ms-emu-eaptlscert-01

Abstract

Extensible Authentication Protocol (EAP) provides support for multiple authentication methods. EAP-Transport Layer Security (EAP-TLS) provides means for key derivation and strong mutual authentication with certificates. However, certificates can often be relatively large in size. The certificate chain to the root-of-trust can also be long when multiple intermediate Certification Authorities (CAs) are involved. This implies that EAP-TLS authentication needs to be fragmented into many smaller packets for transportation over the lower-layer. Such fragmentation can not only negatively affect the latency, but also results in implementation challenges. For example, many authenticator (access point) implementations will drop an EAP session if it hasn't finished after 40 - 50 packets. This can result in failed authentication even when the two communicating parties have the correct credentials for mutual authentication. Moreover, there are no mechanisms available to easily recover from such situations. This memo looks at the problem in detail and discusses the solutions available to overcome these deployment challenges.

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 April 25, 2019.

Copyright Notice

Copyright (c) 2018 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	3
4. Handling of Large Certificates and Long Certificate Chains	4
4.1. Updating Certificates	4
4.2. Updating Code	5
4.3. Guidelines for certificates	6
5. IANA Considerations	6
6. Security Considerations	6
7. References	6
7.1. Normative References	6
7.2. Informative References	6
Acknowledgements	7
Authors' Addresses	7

1. Introduction

EAP-TLS is widely deployed and often used for network access authentication of requesting peers. EAP-TLS provides strong mutual authentication with certificates. However, certificates can be large and certificate chains can often be long. This implies that EAP-TLS authentication needs to be fragmented into many smaller packets for transportation over the lower-layer. Such fragmentation can not only negatively affect the latency, but also results in implementation challenges. For example, many authenticator (access point) implementations will drop an EAP session if it hasn't finished after 40-50 packets. This has led to a situation where a client and server cannot authenticate each other even though both the sides have valid credentials for successful authentication and key derivation.

Unlike TLS authentication on the web, where typically only the server is authenticated with certificates; in EAP-TLS both the client and server are authenticated with certificates. Therefore, EAP-TLS authentication involves exchange of larger number of messages than regular TLS authentication on the web. Also, from deployment experience, the end-entity certificate for clients typically has a longer certificate chain to the root-of-trust than the end-entity certificate for the server.

This memo looks at related work and potential tools available for overcoming the implementation challenges induced by large certificates and long certificate chains. It then discusses the solutions available to overcome these deployment challenges. The draft is an early version and aims to foster discussion in the working group.

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 RFC 8174 [RFC8174].

In addition, this document frequently uses the following terms as they have been defined in [RFC5216]:

authenticator The entity initiating EAP authentication.

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

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.

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 that are used for EAP-TLS authentication without requiring changes to the existing code base. In Section 4.2 we look at recommendations that rely on updates to the EAP-TLS implementations which can be deployed with existing certificates. Finally, in Section 4.3, we provide some guidelines when issuing certificates for use with EAP-TLS.

4.1. Updating Certificates

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, the size of public keys with traditional RSA is about 384 bytes, while the size of public keys with ECC is only 32 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 packets. 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.

4.2. Updating Code

TLS allows endpoints to reduce the sizes of 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. When using TLS 1.2 or earlier, only the self-signed certificate that specifies the root certificate authority may be omitted. Therefore, updating TLS implementations to version 1.3 can help to significantly reduce the number of messages exchanged for EAP-TLS authentication.

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].

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.

4.3. Guidelines for certificates

This section provides some recommendations for certificates used for EAP-TLS authentication. Unlike TLS authentication on the web, EAP-TLS messages are often carried over the link-layer

- o Reasonable number of OIDs
- o ...

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

TBD

7. References

7.1. Normative References

- [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>>.
- [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-04 (work in progress), October 2018.
- [IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004. , December 2004.

- [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

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. Levkowitz, 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