EMU working group
Internet-Draft
Obsoletes: 7170 (if approved)
Updates: 9427 (if approved)
Intended status: Standards Track
Expires: 25 December 2025

A. DeKok
InkBridge Networks

23 June 2025

Tunnel Extensible Authentication Protocol (TEAP) Version 2
draft-dekok-emu-teapv2-00

Abstract

This document defines the Tunnel Extensible Authentication Protocol (TEAP) version 2. It addresses a number of security and interoperability issues in TEAPv1 which was defined in [I-D.ietf-emu-rfc7170bis].

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-dekok-emu-teapv2/.

Discussion of this document takes place on the EMU Working Group mailing list (mailto:emu@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/emu/. Subscribe at https://www.ietf.org/mailman/listinfo/emu/.

Source for this draft and an issue tracker can be found at https://github.com/inkbridgenetworks/teapv2.git.

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 25 December 2025.

Table of Contents

1. Introduction

Tunnel Extensible Authentication Protocol (TEAP) version 1 was first defined in [RFC7170]. However, implementations of that specification were found to have limited interoperability, due to the complexity and under-specification of the cryptographic key deriviations that it defined.

TEAPv1 was updated and clarified in [I-D.ietf-emu-rfc7170bis]. That document described a large amount of potential functionality in the protocol, but also noted in [I-D.ietf-emu-rfc7170bis], Section 5.1 that only a small subset of that functionality was interoperable. In addition, the interoperable parts of the protocol security issues which could allow on-path attackers essentially unlimited control over the data being transported inside of the TLS tunnel.

We do not review the full security issues with TEAPv1 here. Instead, we define new and simpler cryptographic key deriviations. These derivations address all of the known issues with TEAPv1.

1.1. Changes from TEAPv1

Most aspects of TEAPv1 are unchanged. The message and TLV formats are the same, as are the derivations for the session_key_seed ([I-D.ietf-emu-rfc7170bis], Section 6.1), along with the Master Session Key (MSK) and Extended Master Session Key (EMSK) ([I-D.ietf-emu-rfc7170bis], Section 6.4).

The Crypto-Binding TLV [I-D.ietf-emu-rfc7170bis], Section 4.2.13 format is also the same as for TEAPv1, even though the cryptographic derivation has changed.

The main difference between TEAPv1 and TEAPv2 is in the cryptographic calculations. The changes for TEAPv2 address a number of issues in TEAPv1:

* each inner exchange is bound to the TLS tunnel.

* Where an inner exchange in TEAPv2 usd an MSK of all zeros, TEAPv2 defines a pseudo MSK which is derived from the data being exchanged.

* each inner exchange includes a Crypto-Binding TLV.

* inner exchanges which use a challenge (e.g. EAP-MSCHAPv2 [KAMATH]) derive that challenge from the TLS parameters.

* The cryptographic key derivations have been substantially simplified.

The result is a protocol which is simpler and is more extensible.

2. Negotiation

TEAPv2 uses the same version negotiation method as is defined in [I-D.ietf-emu-rfc7170bis], Section 3.1, with the Version field set to two (2) for TEAPv2.

TEAPv2 MUST use TLS 1.3 or later.

3. Cryptographic Calculations

The crytographic calculations for TEAPv2 have been substantially simplified from those defined in [I-D.ietf-emu-rfc7170bis], Section 6.

3.1. TEAP Authentication Phase 1: Key Derivations

The session key seed is the same as defined in [I-D.ietf-emu-rfc7170bis], Section 6.2.1 for TEAPv2. That definition is reproduced here verbatim:

```
session_key_seed = TLS-Exporter(
                     "EXPORTER: teap session key seed",, 40)
```

3.2. Intermediate Compound Key Derivations

Instead of using a complex key deriviation method as was done with TEAPv1, TEAPv2 uses a much simpler method to derive the keys. This method is split into a few steps:

* define a seed which combines data from the current inner message along with data from the previous round.

* Call the TLS-Exporter() function ([RFC8446], Section 7.5) with the above seed as the "context_value", in order to derive keying data.

* Split the resulting keying data into subkeys, which are each used for different purposes.

TEAPv1 mixed data from each inner authentication message into the key derivation, TEAPv2 mixes data from each inner message. Some inner authentication methods do not derive keys (e.g. Basic-Password-Req TLV and Basic-Password-Resp TLV). Other inner message exchanges such as the CSR-Attributes TLV, PKCS#7 TLV, or PKCS#10 TLV also do not derive keys.

Where TEAPv1 uses a Master Session Key (MSK) of all zeros for those inner messages which did not define an MSK, TEAPv2 defines a pseudo MSK which is tied to the TLS tunnel. The pseduo MSK is derived from the data being exchanged, which should better protect the data from on-path attackers. This pseudo MSK is then used in the cryptographic calculations.

The MSK and/or EMSK for each inner message are mixed with a seed from previous rounds beginning with the TEAP Phase 2 session_key_seed, to yield a new set of keys for this round. The seed from the final round is then used to derive the MSK and EMSK for TEAP.

3.2.1. Key Seeding

All intermediate compound key deriviations for TEAPv2 depend on the same structure as input to the key deriviations. For simplicity, we define the structure using the same syntax as is used for TLS [RFC8446]

```
struct {
    opaque PrevRoundKey[40]
    opaque MSK[32];
    opaque EMSK[32]
} RoundSeed
```

The above fields have the following definitions:

PrevRoundKey

A key which ties the current exchange to the previous exchange.

For the first round, this field is taken from the session_key_seed.

For subsequent rounds, this field is the previous rounds RoundKey field, which is taken from the DerivedKey structure defined in the next section.

MSK

The Master Session Key (MSK) from the inner message

If the MSK is longer than 32 octets, the extra octets are not used in this structure.

If the inner method derives an EMSK but not an MSK, then this field MUST be initialized to all zeros.

If the inner message does not perform authentication, or the inner authentication method does not derive an MSK or an EMSK, then a pseudo MSK is derived instead. See (TBD) below for dicussion and definition of the pseudo MSK.

EMSK

The Extended Master Session Key (EMSK) from the inner method.

If the EMSK is longer than 32 octets, the extra octets are not used in this structure.

If the inner method does not derive an EMSK, then this field is initialized to all zeros.

An inner message MUST derive either an MSK (including a pseudo MSK), or an EMSK, or both. The RoundSeed structure MUST NOT have both of the MSK and EMSK fields be all zeros.

3.2.2. Key Derivation

Each round produces a DerivedKey, which is depends on the RoundSeed for this round via the following calculation.

```
DerivedKey = TLS-Exporter(
             "EXPORTER: TEAPv2 Inner Methods Compound Keys",
             RoundSeed, 104)
```

The DerivedKey is 104 octets in length, and assigned to the following structure:

```
struct {
    opaque RoundKey[40];
    opaque CMK[32]
    opaque Challenge[32]
} RoundSeed
```

The above fields have the following definitions:

RoundKey

The key for this round, which is copied to the PrevRoundKey field in the RoundSeed structure, in order to seed the next round.

CMK

The Compound MAC Key (CMK)

The CMK is mixed with with various data from the TEAP negotiation to create the Compound-MAC field of the Crypto-Binding attribute.

Challenge

The implicit challenge used for inner authentication methods such as EAP-MSCHAPv2.

Unlike the implicit challenge in [RFC9427], Section 2.4, this challenge is fixed size in length. The inner method uses only as much of the Challenge as it needs, and the remainder of the Challenge is ignored.

If the inner method does not use a challenge, then the Challenge field is ignored.

TBD: this derivation should be from each message, NOT from each side. That is, instead of each party keeping track of "ours" and "theirs" key data, there should only be one set of data "session". The party which sends the first inner message is the one which is bootstraps this process. Each inner exchange then updates RoundKey.

3.3. Methods which do not generate MSK or EMSK

Where an inner message does not generate MSK (Basic-Password-Resp TLV or PKCS#7 TLV), then a pseudo MSK is calculated which is derived from the inner data. This pseudo MSK ensures that the data from each message is mixed in with the data from previous exchanges. This mixing cryptographically binds every inner message to the protected tunnel (not just inner authentication messages), and binds each message to the previous one. This cryptographic binding prevents on-path attacks.

In contrast, TEAPv1 sets the MSK to zero for these TLVs, which does not tie the data to the TLS session, or prevent on-path attackers.

```
MSK = TLS-Exporter("EXPORTER: TEAPv2 Pseudo MSK",
                   data, 32)
```

Where "data" is the contents of the TLV. That is, everything encoded in the TLV after the four octet TLV header. The following list shows the TLVs which are used in inner message exchanges, and the contents which are used as the "data" fefined above.

* Basic-Password-Req TLV: Value

* Basic-Password-Resp TLV: Userlen, Username, Passlen, Password

* CSR-Attributes TLV: DER Encoded CSR Attributes

* PKCS#7 TLV: PKCS#7 Data

* PKCS#10 TLV: PKCS#10 Data

The sending party MUST inform the receiving party which TLVs are used to calculate this pseudo MSK via the Pseudo-MSK-Contents TLV, which is defined in (TBD), below.

This explicit signal makes implementations easier, because otherwise each non-authentication exchange would require special-purpose code to process the TLVs being sent. It also makes future extensions easier, as any future non-authentication exchanges can simply be listed in the Pseudo-MSK-Contents TLV, and do not need special-purpose code to process.

3.4. Computing the Compound-MAC

The Compound-MAC used in the Crypto-Binding TLV is calculated exactly the same as with TEAPv1:

```
Compound-MAC = the first 20 octets of MAC( CMK, BUFFER )
```

Where CMK is the Compound MAC key derived above for this round, and the definition of BUFFER is the same as in [I-D.ietf-emu-rfc7170bis], Section 6.3 for TEAPv1.

3.5. EAP Master Session Key Generation

TEAP authentication assures that the MSK and EMSK output from running TEAP are combined result of all inner methods. The resulting MSK and ESMK are generated from the final inner method, via the following derivation:

```
MSK  = the first 64 octets of TLS-PRF(RoundSeed,
       "Session Key Generating Function")
EMSK = the first 64 octets of TLS-PRF(RoundSeed,
       "Extended Session Key Generating Function")
```

The value for RoundSeed MUST use the PrevRoundSeed from the previous round, along with the MSK and the EMSK from the final inner message.

3.6. Operation across Multiple Rounds

Unlike TEAPv1, every message for every round in TEAPv2 MUST contain a Crypto-Binding TLV. This cryptographic binding helps protect from on-path attackers.

Any party which sends a message in TEAPv2 MUST include a Crypto-Binding TLV. Any party which receives a message in TEAPv2 MUST verify that it contains a Crypto-Binding TLV

TBD: discuss why use of MSK only in TEAPv1 is likely to avoid cryptographic binding? The session_key_seed is taken from the TLS-Exporter(), which binds it to the tunnel. But subsequent exchanges of MSK-only methods do not bind the results to the tunnel. This may or may not be a problem?

3.7. TEAPv2 Message Format

The TEAPv2 message format is identical to that of TEAPv1 ([I-D.ietf-emu-rfc7170bis], Section 4.1) with only one change: the Ver field is set to "2", to indicate that this is TEAPv2.

3.8. TEAPv2 TLVs

The TEAPv2 TLV format and TLV definitions are identical to that for TEAPv1 ([I-D.ietf-emu-rfc7170bis], Section 4.2), with only the changes and additions noted below.

3.8.1. Crypto-Binding TLV

The format of the TEAPv2 Crypto-Binding TLV is the same as for TEAPv1 ([I-D.ietf-emu-rfc7170bis], Section 4.2.13), with the following changes:

* The Version field MUST set to two (2).

* The Received-Ver field MUST be set to two (2), to indicate TEAPv2.

* The Flags field MUST have value 2, to indicate that only the MSK Compound-MAC is present.

* the Nonce field is not used. It SHOULD be set to zeros by the sender. The receiver MUST ignore it.

* The ESMK Compound-MAC field is not used. It SHOULD be set to zeros by the sender. The receiver MUST ignore it.

* The MSK Compound-MAC field is calculated as described above in Section 3.4.

Note that unlike TEAPv1, only one CMK is derived for each inner message, which also means that only one Compound-MAC is derived. This Compound-MAC is placed into the MSK Compound-MAC field, and the EMSK Compound-MAC field is not used.

It would be possible to redefine the entire contents of the Crypto-Binding TLV, in the interest of minor optimization. However, re-using the existing Crypto-Binding TLV format means that there are minimal changes required to implementations, which is a more useful property than saving a few octets of data being exchanged.

3.8.2. Pseudo-MSK-Contents TLV

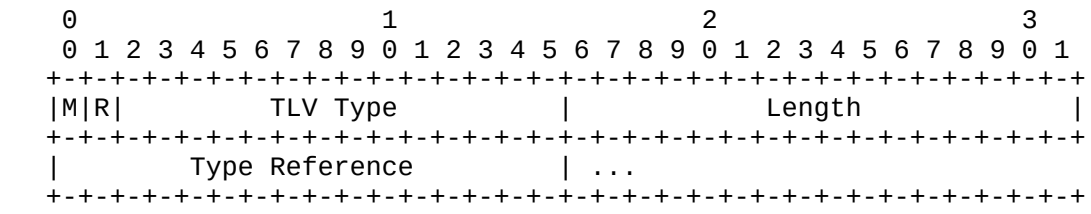
The Pseudo-MSK-Contents TLV provides a mechanism for the sender to inform the receiver as to which attribute or attributes were used to calculate the pseudo MAC.

The Pseudo-MSK-Contents TLV MUST be included in any message where no authentication is taking place. The Pseudo-MSK-Contents TLV MUST be included in any message where the inner authentication method does not derive either an MSK or an EMSK. Where a party expects a Pseudo-MSK-Contents TLV but does not receive it, then the peer or EAP server MUST treat this as a fatal error of 2010 (Missing Pseudo-MSK-Contents TLV).

The Pseudo-MSK-Contents TLV MUST NOT be included in any message where the inner authentication method derives either an MSK or an EMSK. Where a party receives an unexpected Pseudo-MSK-Contents TLV, then the peer or EAP server MUST treat this as a fatal error of 2002 (Unexpected TLVs Exchanged).

Where there is a fatal error due to a problem with the Pseudo-MSK-Contents TLV, the party MUST send a NAK TLV which includes a reference to the Pseudo-MSK-Contents TLV.

TBD: Do we want to add a Vendor-Id as is done with the NAK TLV?



M

1 - Required TLV

R

Reserved, set to zero (0)

TLV Type

20 - Pseudo-MSK-Contents

Length

The Length field is two octets and contains the length of the TLV field in octets. The Length MUST be at least two (2), and MUST be a multiple of two (2).

If the Length field is invalid, implementations MUST behave as if Pseudo-MSK-Contents TLV was missing.

Type Reference

An array of one or more 2-octet values. The values MUST be the TLV Type numbers of the TLVs which were used as input to the pseudo MAC calculation.

3.9. Implicit Challenges

TBD EAP-MSCHAPv2

4. Security Considerations

5. IANA Considerations

TBD - assign new Pseduo-MSK-Contents TLV

TBD - assign Error 2010 (Missing Pseudo-MSK-Contents TLV)

6. References

6.1. Normative References

[BCP14] 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/rfc/rfc8174>.

[I-D.ietf-emu-rfc7170bis] DeKok, A., "Tunnel Extensible Authentication Protocol (TEAP) Version 1", Work in Progress, Internet-Draft, draft-ietf-emu-rfc7170bis-22, 28 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-emu-rfc7170bis-22>.

[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/rfc/rfc8446>.

[RFC9427] DeKok, A., "TLS-Based Extensible Authentication Protocol (EAP) Types for Use with TLS 1.3", RFC 9427, DOI 10.17487/RFC9427, June 2023, <https://www.rfc-editor.org/rfc/rfc9427>.

6.2. Informative References

[KAMATH] Palekar, R. H. and A., "Microsoft EAP CHAP Extensions", June 2007.

[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/rfc/rfc7170>.

Author’s Address

Alan DeKok
InkBridge Networks
Email: alan.dekok@inkbridge.io