EMU working group
Internet-Draft
Obsoletes: 7170 (if approved)
Updates: 9427 (if approved)
Intended status: Standards Track
Expires: 16 April 2026

A. DeKok
InkBridge Networks

13 October 2025

Tunnel Extensible Authentication Protocol (TEAP) Version 2
draft-dekok-emu-teapv2-01

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 16 April 2026.

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 between TEAPv1 and TEAPv2 simplify the protocol substantially, and address issues seen with TEAPv1:

* The Crypto-Binding TLV calculation in TEAPv1 had significant differences between theory and practice. TEAPv2 redefines the deriviation of the Crypto-Binding TLV to make it simpler.

* A number of inner message exchanges did not tie the Crypto-Binding TLV to the data being exchanged, as the MSK was set to all zeroes. This setting negated most of the utility of the Crypto-Binding TLVs. TEAPv2 simplifies this deriviation, and avoids the use of the Crypto-Binding TLV where it is not necessary.

* The other 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. Initial-Binding TLV

NOTE: We should use either the Initial-Binding TLV _OR_ the Crypto-Binding TLV. We do not need both.

The Initial-Binding TLV is used to prove that both the peer and server participated in the tunnel establishment and sequence of authentications. It also provides verification of the TEAP type, version negotiated, and Outer TLVs exchanged before the TLS tunnel establishment.

The Initial-Binding TLV MUST be included in the first message sent by each party, and MUST NOT be included in any subseqent messages.

A party receiving an initial message from the other party MUST check that the message contains an Initial-Binding TLV. The Initial-Binding TLV MUST be validated by the receiving part before processing any other field is process.

The Initial-Binding TLV is valid only if the following checks pass on its contents:

* The Version field contain a known value,

* The Received-Ver field matches the TEAP version sent by the receiver during the EAP version negotiation,

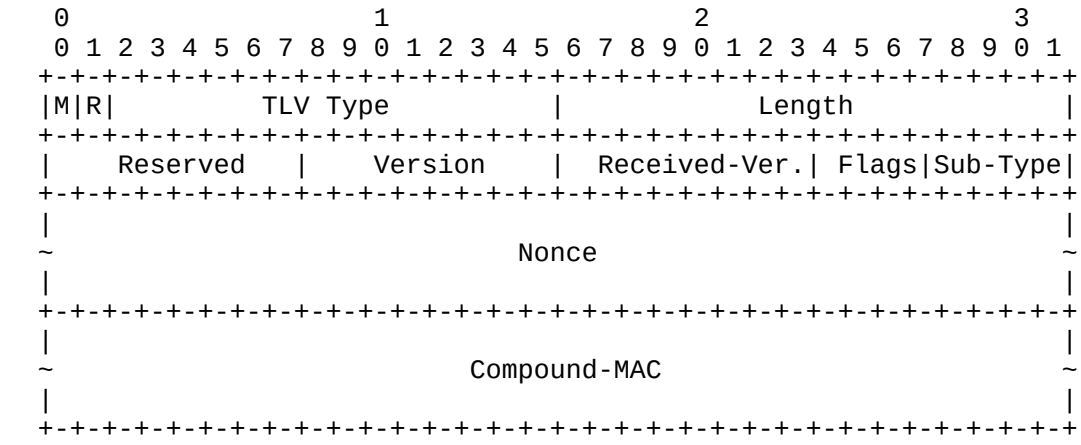
* The Sub-Type field is set to the correct value for this exchange,

* The Compound-MAC verifies correctly.

If any of the above checks fails, then the TLV is invalid. An invalid Initial-Binding TLV is a fatal error and is handled as described in ((RFC7170bis phase 2 errors))

Once the Initial-Binding TLV(s) have been exchanged and verified, both parties know that the TLS tunnel is secure. Any subsequent Crypto-Binding TLV exchange is unnecessary.

The Crypto-Binding TLV is defined as follows:



M

Mandatory, set to one (1)

R

Reserved, set to zero (0)

TLV Type

TBD - Initial-Binding TLV

Length

24

Reserved

Reserved, set to zero (0)

Version

The Version field is a single octet, which is set to the version of Crypto-Binding TLV the TEAP method is using. For an implementation compliant with TEAPv1, the version number MUST be set to one (1).

Received-Ver

The Received-Ver field is a single octet and MUST be set to the TEAP version number received during version negotiation. Note that this field only provides protection against downgrade attacks, where a version of EAP requiring support for this TLV is required on both sides.

For TEAPv1, this version number MUST be set to one (1).

Flags

The Flags field is four bits.

The field MUST be set to zero.

All other values of the Flags field are invalid.

Sub-Type

The Sub-Type field is four bits. Defined values include

0 Binding Request

MUST be used by authentication server.

1 Binding Response

MUST be used by peer.

All other values of the Sub-Type field are invalid.

Nonce

The Nonce field is 32 octets. It contains a 256-bit nonce that is temporally unique, used for Compound-MAC key derivation at each end. The nonce in a request MUST have its least significant bit

set to zero (0), and the nonce in a response MUST have the same value as the request nonce except the least significant bit MUST be set to one (1).

Compound-MAC

The Compound-MAC field is 20 octets. The computation of the MAC is described in Section 3.6.

Note that this field is always 20 octets in length. Any larger MAC is simply truncated. All validations or comparisons MUST be done on the truncated value.

###

The CMK used to calculate the Compound-MAC is defined as

```
CMK = the first 20 octets of TLS-PRF(session-key-seed,
             "Inner Methods Compound Keys",
             Nonce)
```

Note that the Nonce is different for the Binding Request and Binding Response.

3.2.1. MSK and EMSK

3.3. EAP Master Session Key Generation

When the Initial-Binding TLV is used, the final TEAP MSK and EMSK are not bound to the inner methods. The MSK and EMSK are derived as:

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

3.4. Intermediate Compound Key Derivations

NOTE: We should use either the Initial-Binding TLV _OR_ the Crypto-Binding TLV. We do not need both. If we use the Initial-Binding TLV, then the intermediate compound key deriviations are not necessary.

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 exchange into the key derivation, TEAPv2 mixes data only from inner exchanges which derive an MSK and/or an EMSK. For inner exchanges which do not derive an MSK or EMSK, TEAPv2 omits the Crypto-Binding TLV.

That is, some inner authentication methods do not derive MSK or EMSK, such as 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 MSK or EMSK. Inner messages which contain those TLVs MUST NOT contain a Crypto-Binding TLV.

Where an inner authentication methed derives the MSK and/or EMSK, those keys 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.

3.4.1. 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 authentication method.

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

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

EMSK

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

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

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

Where the inner message is not an authentication method, or the inner authentication method does not derive MSK or EMSK (e.g. Basic-Password-Resp TLV), then the RoundSeed structure MUST NOT be modified. The inner message also MUST NOT send a Crypto-Binding TLV.

3.4.2. 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]
} DerivedKey
```

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.

3.5. Methods which do not generate MSK or EMSK

Where an inner round has not generated MSK or EMSK, then the resulting inner message MUST NOT contain a Crypto-Binding TLV.

For these inner messages, the RoundSeed contents MUST remain unchanged. That is, the Crypto-Binding TLV is calculated, and the RoundSeed updated only for inner authentication methods which derive MSK or EMSK. The Crypto-Binding TLV is therefore calculated as if other inner exchanges do not exist.

3.6. 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.

3.7. 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 (if available) from the final inner exchange.

Where no inner exchange derives an MSK or EMSK, then the RoundSeed structure is unchanged from its initial value, which has PrevRoundKey set to Seed, and the MSK and EMSK fields are all zero.

3.8. 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?

3.9. 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.

3.10. 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.

3.10.1. 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 set to random values. There is no need to set the least significant bit to zero or one. If the least significant bit is set to a particular value, it has no impact on the protocol.

* 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 3.6.

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.

3.11. Implicit Challenges

TBD EAP-MSCHAPv2

4. Security Considerations

5. IANA Considerations

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