Network Working Group
Internet Draft
Intended status: Standards Track
Expires: March 2008

Steve Hanna
Juniper Networks
Paul Funk
Juniper Networks
September 24, 2007

**Key Agility Extensions for EAP-TTLSv0**
**draft-hanna-eap-ttls-agility-00.txt**

Status of this Memo



This Internet-Draft will expire on March 24, 2008.

Copyright Notice



Abstract

This document defines new Attribute Value Pairs (AVPs) that add cryptographic algorithm agility and other security features (protected results and cryptographic binding of inner authentications to the outer tunnel) to EAP-TTLSv0.

Table of Contents



## 1. Introduction

EAP-TTLSv0 [TTLSv0] is a "tunneled EAP method". This means that it defines an authentication protocol for use with the Extensible Authentication Protocol [EAP] and that this authentication protocol creates a secure tunnel that can carry other authentication protocols and other data exchanges. EAP-TTLSv0 has many advantages such as extensibility, the ability to more securely carry legacy authentication protocols, and widespread usage. But it also has several issues: lack of cryptographic algorithm agility, vulnerability to possible Man-in-the-Middle attacks when used with

certain legacy protocols as described in [MITM], and vulnerability to truncation attacks.

This specification defines three extensions to EAP-TTLSv0 that address these issues by adding algorithm agility, protected results, and cryptographic binding of inner authentications to the outer tunnel. The extensions are defined using AVPs with semantics that allow for a gradual transition as clients and servers are upgraded to support the extensions. At first, clients can request the extensions in a backward-compatible manner but proceed with the authentication if the server does not support them. Servers can allow clients to use the extensions or not. Over time, clients and servers can change their policy to require the extensions, thus protecting against downgrade attacks. For a description of how the extensions defined in this document provide algorithm agility, protected results, and cryptobinding, see the Security Considerations section.

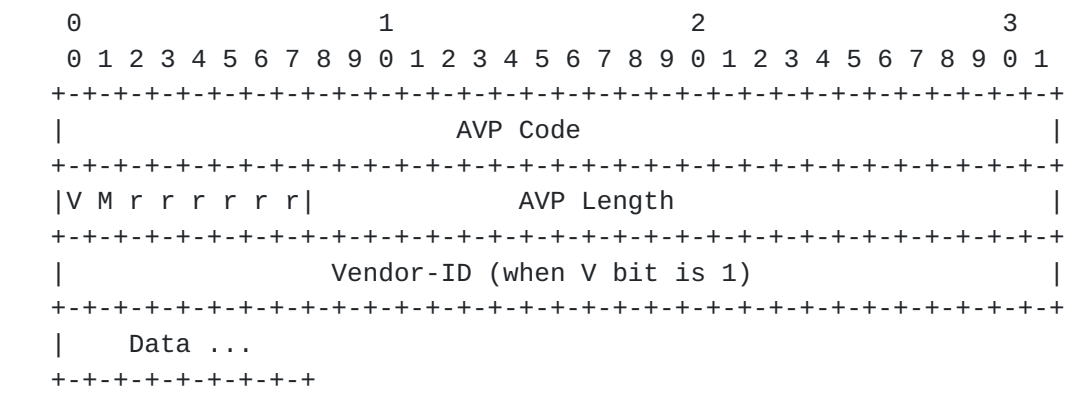
## 2. Terminology

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 [KEYWORDS].

## 3. AVP Format

This document defines several AVPs to be used with EAP-TTLSv0. All of these AVPs begin with the AVP header fields defined in [TTLSv0] and use those fields in the same way. This section describes how those fields are used and interpreted so that each AVP definition below does not need to duplicate this text. Any normative text in this section is normative with respect to all AVPs defined in this document.

The format of an EAP-TTLSv0 AVP is shown below. All items are in network (i.e. big-endian) order.




The IANA will assign AVP Codes for the AVPs defined in this document when this document moves to RFC status (as defined in [TTLSv0] and [RFC3588]). In the mean time, this document defines vendor-specific AVP Codes that may be used by setting the 'V' bit. Once this document moves to RFC status, implementations MUST accept the new standard AVP Codes, MAY accept the vendor-specific AVP Codes, and SHOULD only send the standard AVP Codes.

The 'V' (Vendor-Specific) bit MUST be set to 1 when a vendor-specific AVP Code is used. Otherwise, it MUST be set to 0. When this bit is set to 1, the Vendor-ID field MUST be present. Otherwise, it MUST be absent.

The 'M' (Mandatory) bit may be set to 0 or 1, depending on the preferences and policies of the party sending this AVP. If the M bit is set to 1, the party receiving this AVP MUST either support this AVP Code or fail the authentication. If the M bit is set to 0, the receiving party may safely ignore this AVP.

The 'r' (reserved) bits MUST be set to 0 by the sending party and MUST be ignored by the receiving party, at least for this version of this specification. Future versions of this specification may use these bits for extensibility.

The AVP Length field MUST be set to the length in octets of this AVP including the AVP Code, AVP Length, AVP Flags (V, M, and r), Vendor-ID (if present) and Data.

The Vendor-ID field is omitted when the V flag is 0. The vendor-specific AVP Codes defined in this specification all have a Vendor-ID of 2636.

The contents and meaning of the Data field are different for each AVP defined below.

## 4. Negotiating MSK and EMSK Computation

Secure computation of a Master Session Key (MSK) and Extended Master Session Key (EMSK) is a critical part of any strong EAP method. The mechanism defined in [TTLSv0] for computing the MSK and EMSK always uses the TLS 1.1 PRF based on MD5 and SHA1. This is not algorithm agile. Also, the MSK and EMSK computation method defined in [TTLSv0] does not mix in keying material from inner authentications so cryptographic binding of the inner and outer authentications is not achieved.

The new MSK-Computation AVP allows the TTLS client and TTLS server to negotiate the MSK and EMSK computation method to be used. A new MSK and EMSK computation method defined in section 7.3 of this specification achieves algorithm agility and adds cryptographic binding. To ensure maximum flexibility for the future, other MSK and EMSK computation methods can be defined and negotiated using the same MSK-Computation AVP.

Options for MSK and EMSK computation defined in this specification include:

- Default computation, as defined in [TTLSv0]

- Mixed computation method as defined in section 7.3

In order to negotiate a computation method other than the TTLSv0 default, the client sends an MSK-Computation AVP in the first phase 2 TTLS message sent to the TTLS server. This AVP indicates which MSK computation methods the client is willing to use. If the client does not include an MSK-Computation AVP in the first phase 2 TTLS message sent to the TTLS server, the default computation methods (as specified in [TTLSv0]) are used.

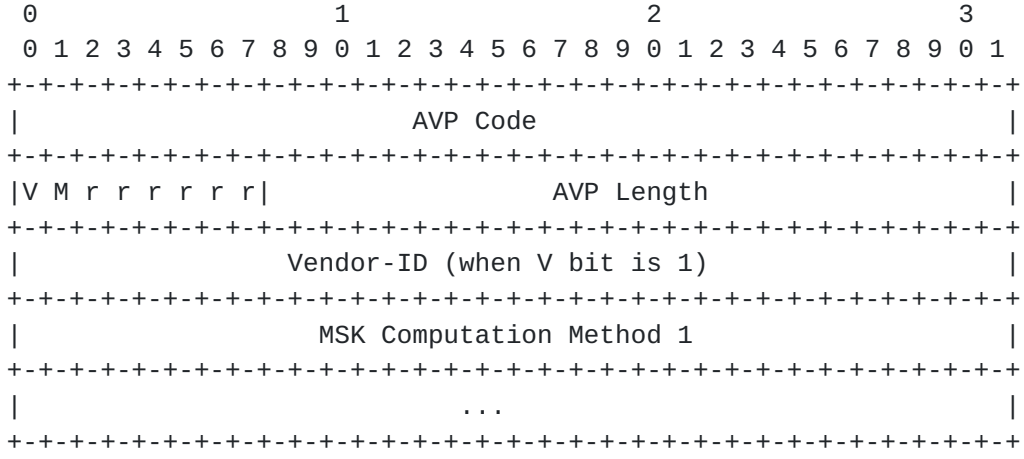
If the default MSK computation method (as specified in [TTLSv0]) is not acceptable to the client, the client SHOULD set the M (Mandatory) bit in the MSK-Computation AVP to indicate that the server must support this AVP or terminate the authentication.

A TTLS server that supports the MSK-Computation AVP MUST respond to an MSK-Computation AVP received from the client by selecting one of those computations or by failing the authentication if it will not accept any of the computations listed by the client. The server MAY make its selection by sending an MSK-Computation AVP in any TTLS message prior to the completion of the authentication; that is, the server is allowed to defer its selection if necessary.

Note that when the TTLS server relays an inner authentication to another server, its ability to mix inner method MSKs with the TLS master secret depends on the MSK of the relayed authentication being conveyed by the other server to the TTLS server. For example, a RADIUS TTLS server may proxy an inner authentication to a home RADIUS server, which would normally return the MSK to the TTLS server via MPPE-Recv-Key and MPPE-Send-Key attributes. Some RADIUS servers may not return the MSK in this manner. In that case, the RADIUS TTLS server will not be able to use an MSK computation method that requires mixing in the inner MSK. That's why the TTLS server is allowed to delay committing to a particular MSK computation method until the end of the handshake. It may not know whether the home RADIUS server will return the MSK (or even whether a home RADIUS server will be used). The TTLS server SHOULD commit to an MSK computation method as soon as possible. The TTLS client MAY refuse to continue with an authentication if the TTLS server has not committed to an MSK computation method by a particular point (e.g. when particularly sensitive credentials are requested).

## 4.1. MSK-Computation AVP

The basic format of the MSK-Computation AVP is defined in section 3 above. This section only describes aspects that are specific to the MSK-Computation AVP.




The AVP Code for the MSK-Computation AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a vendor-specific AVP Code of 256 with a Vendor-ID field of 2636 should be used for experimental purposes.

The body of the MSK-Computation AVP contains one or more 32-bit MSK computation method values (defined in section 4.3 below), each of which specifies a mechanism for computing the MSK and EMSK.

The client MAY list multiple values, in order from most to least preferred. The TTLS server MUST list only one value.

### 4.2. MSK Computation Method Values

A standard set of MSK computation method values for use in the MSK-Computation AVP is defined in this section. To enable extensibility, each value contains a vendor-id in the first (high order) 24 bits and a selector in the last (low order) 8 bits. Values defined in this specification use a vendor-id of 0. Vendors MUST use the vendor's IANA-assigned Private Enterprise Number [PEN].

The following set of standard MSK computation method values are defined at this time:

- 0 Default computation as defined in [TTLSv0]

- 1 Mixed computation mechanism as defined in section 7.3

Additional standard MSK computation method values may be defined at a later time by publishing an RFC that defines these values.

## 5. Key Confirmation

Key Confirmation permits the client and TTLS server each to prove to the other that it is in possession of all keys resulting from inner authentications, in addition to the TLS master secret. This is done to preclude a type of man-in-the-middle attack in which the attacker, acting as a server, induces a legitimate client to perform an authentication which the attacker then acts as a client to feed into the EAP-TTLS negotiation as an inner method (as described in [MITM]).

Note that use of the Mixed mechanism for MSK computation may provide a comparable safeguard, by ensuring that the MSK resulting from the EAP-TTLS negotiation cannot be known by such a man-in-the-middle attacker, and therefore cannot be used in a subsequent data connection, thus allowing authentication to succeed but denying the attacker the fruits of that successful authentication.

Key Confirmation is negotiated using the Key-Confirmation-Option AVP and implemented using the Key-Confirmation AVP. The client MAY issue a Key-Confirmation-Option AVP in its first phase 2 TTLS message to the TTLS server, indicating one or both of the following Key Confirmation Option values:

- Disabled

- Enabled

By listing both Disabled and Enabled options, the client can indicate that it is willing to proceed with or without Key Confirmation. If this is the case, the client should set the M bit for the Key-Confirmation-Option AVP to 0 so that servers that do not support this AVP can be supported. If, on the other hand, the client requires Key Confirmation to be used, it may set the M bit for the Key-Confirmation-Option AVP to 1 so that servers that do not support this AVP will fail the authentication.

The Key-Confirmation-Option AVP MUST appear in the client's first phase 2 TTLS message to the TTLS server if it appears at all. Absence of the Key-Confirmation-Option AVP is equivalent to selecting the "Disabled" option.

If the client transmits a Key-Confirmation-Option AVP that includes an option acceptable to the TTLS server, the TTLS server MUST respond with a Key-Confirmation-Option AVP in its first phase 2 TTLS message to the client indicating its selection (Enabled or Disabled). If the TTLS server is unwilling to accommodate the client's Key Confirmation preference, it MUST fail the authentication.

Once negotiated, Key Confirmation MAY be initiated by the TTLS server in any TTLS message, by issuing a Key-Confirmation AVP to the client. A Key Confirmation that occurs after all MSK-generating inner authentication methods have completed is called the "final Key Confirmation", and any Key Confirmation that occurs early (with fewer than all inner MSKs) is called an "intermediate Key Confirmation".

If the TTLS server indicated "Enabled" in its Key-Confirmation-Option AVP, the TTLS server MAY initiate one or more intermediate Key Confirmations and MUST initiate a final Key Confirmation. If the TTLS server indicated "Disabled" in its Key-Confirmation-Option AP, the TTLS server MUST NOT initiate any intermediate Key Confirmations or final Key Confirmations.

When the client receives a valid Key-Confirmation AVP from the TTLS server, it MUST include its own Key-Confirmation AVP in its next TTLS message to the TTLS server. If the client receives an invalid Key-Confirmation AVP from the TTLS server, it MUST abandon the authentication or otherwise cause it to fail.

If the client wants to require the server to send a final Key Confirmation, it should set the M (Mandatory) bit on the Key-Confirmation-Option AVP to ensure that servers that do not support this AVP will terminate the authentication. If the client sets the M

bit and then the server responds with a first EAP-TTLS message that does not include a Key-Confirmation-Option AVP or includes an invalid or unacceptable Key-Confirmation-Option AVP , or the TTLS server sends an invalid Key-Confirmation AVP at any point, the TTLS client should consider the authentication failed. If the TTLS server sends a Key-Confirmation-Option AVP indicating that key confirmation is Enabled and then the TTLS client receives an EAP-Success message without receiving a final Key Confirmation, the client SHOULD NOT consider the session properly established.

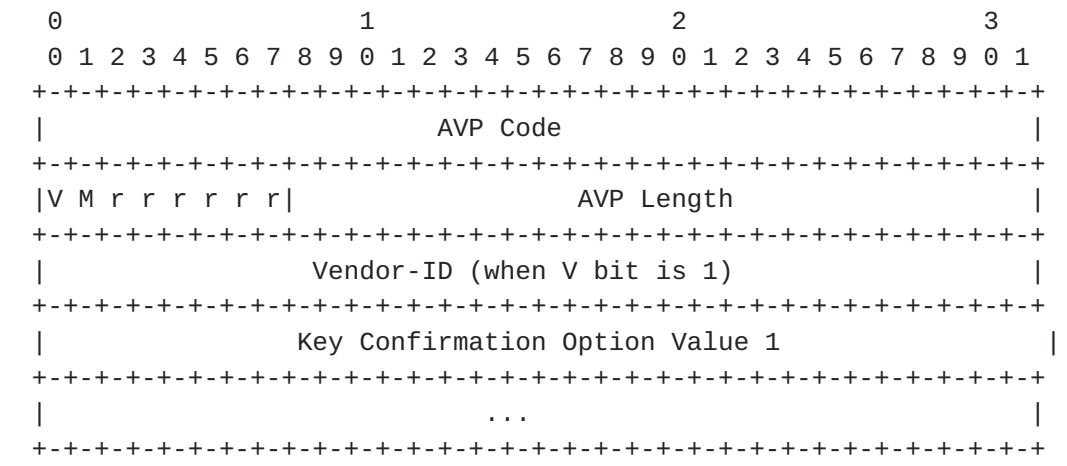
If the TTLS server receives an invalid Key-Confirmation AVP from the TTLS client or fails to receive any Key-Confirmation AVP in response to one sent, the TTLS server SHOULD fail the authentication and send an EAP-Failure message. If secure completion has been negotiated and no TTLS-Success or TTLS-Failure message has been sent, a TTLS-Failure message should be sent before the EAP-Failure message. If secure completion has been negotiated and a TTLS-Success or TTLS-Failure message has already been sent, only an EAP-Failure message should be sent.

Intermediate Key Confirmation might be used when multiple inner authentications are performed and it is desirable to validate the results of a first authentication prior to performing a subsequent authentication, possibly for security or performance reasons. Use of intermediate Key Confirmation is anticipated to be atypical.

Because intermediate Key Confirmations expose information about the results of previous EAP methods, clients SHOULD NOT engage in an intermediate Key Confirmation unless they have already authenticated the server and determined that it is sufficiently trusted to expose this information.

### 5.1. Key-Confirmation-Option AVP

The basic format of the Key-Confirmation-Option AVP is defined in section 3 above. This section only describes aspects that are specific to the Key-Confirmation-Option AVP.




The AVP Code for the Key-Confirmation-Option AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a vendor-specific AVP Code of 257 with a Vendor-ID field of 2636 should be used for experimental purposes.

The body of the Key-Confirmation-Option AVP contains one or more 32-bit Key Confirmation Option Values. To enable extensibility, each value contains a vendor-id in the first (high order) 24 bits and a selector in the last (low order) 8 bits. Standard values (defined in this or future IETF specifications) use a vendor-id of 0. Vendors MUST use the vendor's IANA-assigned Private Enterprise Number [PEN].

The following standard Key Confirmation Option values are defined at this time:

0 Disabled

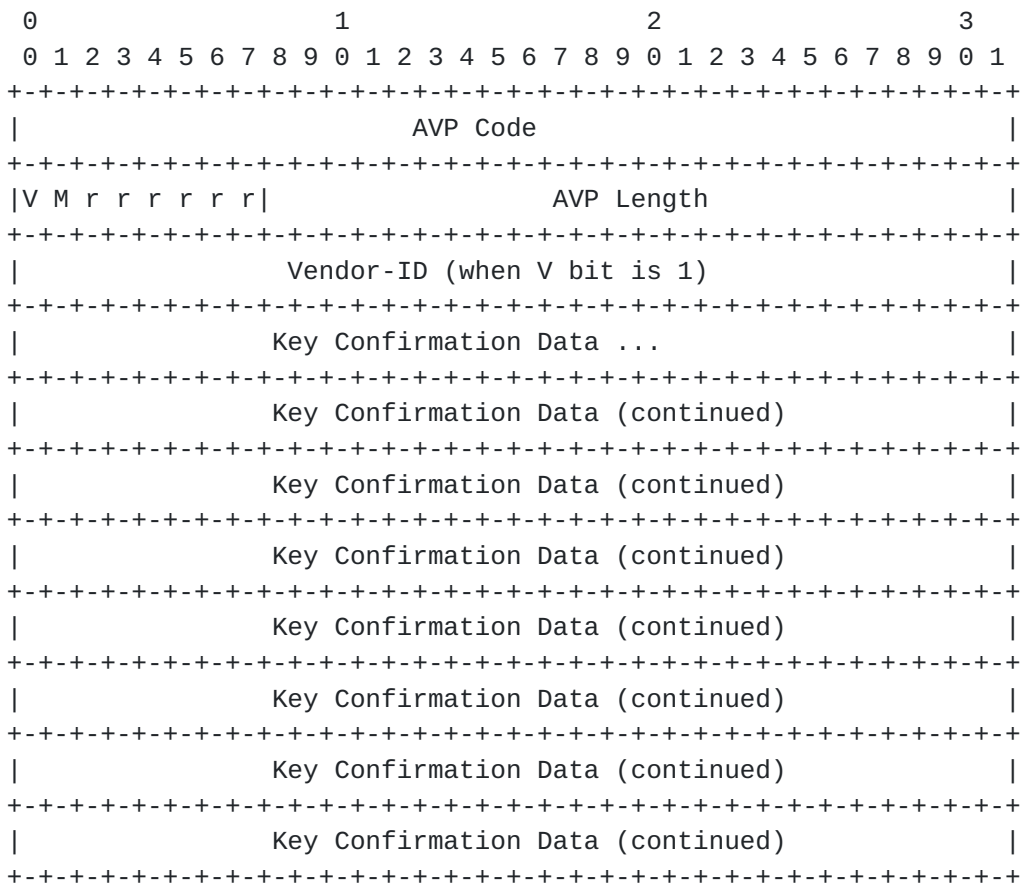
1 Enabled

Additional standard Key Confirmation Option values may be defined at a later time by publishing an RFC that defines these values.

The client MAY list multiple Key Confirmation Option values, in order from most to least preferred. The server MUST list only one value.

## 5.2. Key-Confirmation AVP

The basic format of the Key-Confirmation AVP is defined in section 3 above. This section only describes aspects that are specific to the Key-Confirmation AVP.




The AVP Code for the Key-Confirmation AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a vendor-specific AVP Code of 258 with a Vendor-ID field of 2636 should be used for experimental purposes.

The body of the Key-Confirmation AVP contains 32 octets of binary Key Confirmation Data, computed as described in section 7.4.

## 6. Secure Completion

As defined in [EAP], the EAP-Success and EAP-Failure messages, which indicate the result of an EAP authentication, are unprotected messages sent by the EAP authenticator to the client and not acknowledged by the client. Thus these messages can be changed in transit.

In addition, while the TLS protocol provides a means to securely terminate a conversation, [TTLSv0] does not utilize that mechanism, leaving it vulnerable to possible truncation attack. For example, an attacker might insert a counterfeit EAP-Success to the client prior to completion of the TTLS exchange, or the attacker might contrive to

substitute empty TTLS payloads for real ones to either client or TTLS server in the final TTLS payloads. This substitution would go undetected.

Whether such vulnerabilities could lead to real damage beyond denial of service depends on the underlying inner protocols used within TTLS. However, for maximum security it is advisable to use the Secure Completion mechanism to thwart possible attacks.

In Secure Completion, the TTLS server issues a TTLS-Success or TTLS-Failure AVP as the final AVP of its final TTLS message to the client. The client responds with either a TTLS-Success or TTLS-Failure AVP as the final AVP of its final TTLS message to the TTLS server. Once the TTLS server receives the client's final TTLS message, the unprotected EAP-Success or EAP-Failure (as appropriate) is sent to the client to complete the EAP exchange.

Note that other AVPs may appear in the same TTLS message as TTLS-Success or TTLS-Failure, provided they appear earlier in the message. The TTLS server, for example, could piggyback a TTLS-Success at the end of a final inner EAP-Request message.

The client MUST respond to a TTLS-Success from the TTLS server with either a TTLS-Success or TTLS-Failure. The client MUST respond to a TTLS-Failure from the TTLS server with a TTLS-Failure. If the client fails to include such a value as the final AVP in its final EAP-Response or if the client sends a TTLS-Failure AVP in its final EAP-Response, the server SHOULD send an EAP-Failure. The server SHOULD NOT send an EAP-Failure if both the server and client have just sent TTLS-Success messages since an untrusted intermediary could easily change this EAP-Failure to an EAP-Success without the client detecting that change. In any case, the server SHOULD send only one TTLS-Success or TTLS-Failure message during a particular EAP-TTLSv0 handshake.

When session resumption is employed with EAP-TTLS and secure completion had been negotiated in the prior session, the TTLS server and client MUST still send TTLS-Success or TTLS-Failure AVPs to each other as their last AVPs. This will often result in an additional round trip but the client and server have previously negotiated secure completion and this is the price to be paid for that extra measure of security.

The client MAY issue a Secure-Completion-Option AVP in its first phase 2 TTLS message to the TTLS server, listing one or both of the following Secure Completion Option values (defined in section 6.1 below):

- Disabled

- Enabled

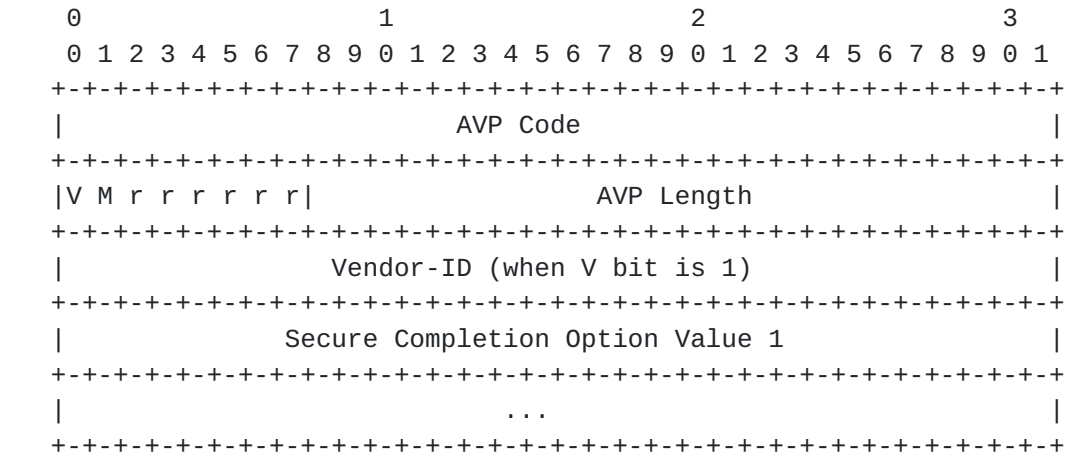
By listing both Disabled and Enabled options, the client can indicate that it is willing to proceed with or without Secure Completion. If this is the case, the client should set the M bit for the Secure-Completion-Option AVP to 0 so that servers that do not support this AVP can be supported. If, on the other hand, the client requires Secure Completion to be used, it may set the M bit for the Secure-Completion-Option AVP to 1 so that servers that do not support this AVP will fail the authentication.

The Secure-Completion-Option AVP MUST appear in the client's first phase 2 TTLS message to the TTLS server if it appears at all. Absence of the Secure-Completion-Option AVP is equivalent to selecting the "Disabled" option.

If the client transmits a Secure-Completion-Option AVP that includes an option acceptable to the TTLS server, the TTLS server MUST respond with a Secure-Completion-Option AVP in its first phase 2 TTLS message to the client indicating its selection (Enabled or Disabled). If the TTLS server is unwilling to accommodate the client's Secure Completion preferences, it MUST fail the authentication.

### 6.1. Secure-Completion-Option AVP

The basic format of the Secure-Completion-Option AVP is defined in section 3 above. This section only describes aspects that are specific to the Secure-Completion-Option AVP.




The AVP Code for the Secure-Completion-Option AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a

vendor-specific AVP Code of 259 with a Vendor-ID field of 2636 should be used for experimental purposes.

The body of the Secure-Completion-Option AVP contains one or more 32-bit Secure Completion Option Values. To enable extensibility, each value contains a vendor-id in the first (high order) 24 bits and a selector in the last (low order) 8 bits. Standard values (defined in this or future IETF specifications) use a vendor-id of 0. Vendors MUST use the vendor's IANA-assigned Private Enterprise Number [PEN].

The following standard Secure Completion Option values are defined:

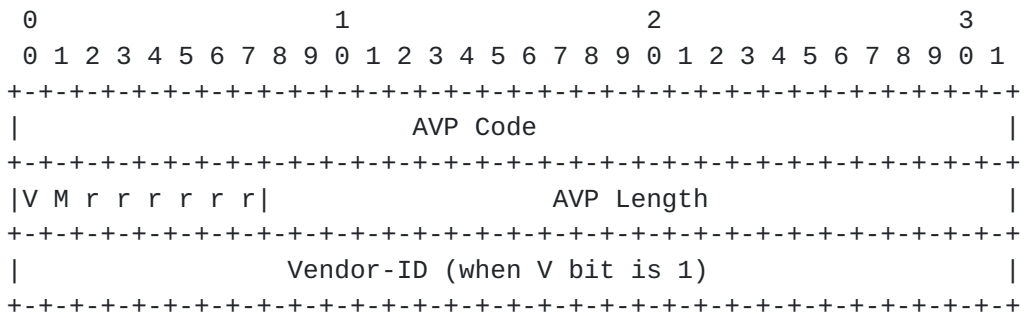
- 0 Disabled (default behavior of [TTLSv0])

- 1 Enabled

The client MAY list multiple Secure Completion Option values, in order from most to least preferred. The server MUST list only one value.

### 6.2. TTLS-Success AVP

The basic format of the TTLS-Success AVP is defined in section 3 above. This section only describes aspects that are specific to the TTLS-Success AVP.

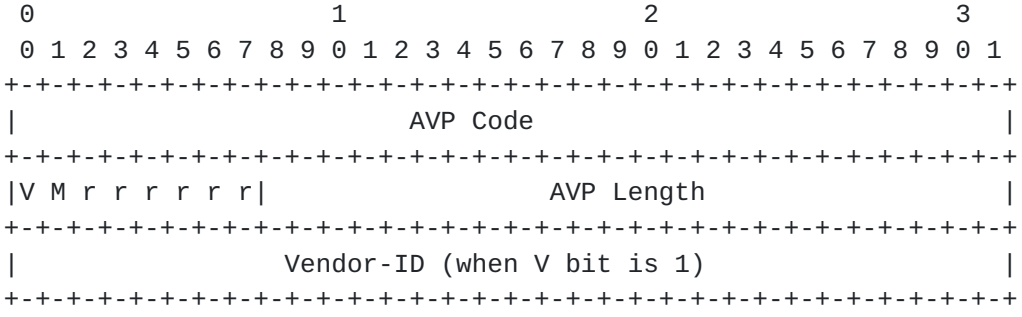



The AVP Code for the TTLS-Success AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a vendor-specific AVP Code of 260 with a Vendor-ID field of 2636 should be used for experimental purposes.

The body of the TTLS-Success AVP is empty (zero length).

### 6.3. TTLS-Failure AVP

The basic format of the TTLS-Failure AVP is defined in section 3 above. This section only describes aspects that are specific to the TTLS-Failure AVP.




The AVP Code for the TTLS-Failure AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a vendor-specific AVP Code of 261 with a Vendor-ID field of 2636 should be used for experimental purposes.

The body of the TTLS-Failure AVP is empty (zero length).

## 7. Cryptographic Computations

### 7.1. Use of TLS PRF

All of the cryptographic mechanisms described in this section use the PRF function used in the TLS handshake negotiation that initiates the TTLS exchange. In many cases, this will be the standard PRF function defined in TLS 1.1 [TLS]; however, it is expected that future versions of or extensions to the TLS protocol will permit alternative PRF functions to be negotiated. If an alternative PRF function has been negotiated during the TLS handshake negotiation, then these mechanisms use that alternative PRF function instead of the TLS 1.1 PRF. This provides algorithm agility since these mechanisms are not tied to the MD5 and SHA-1 hashes used in the TLS 1.1 PRF.

The TLS PRF function used for cryptography described in this specification is denoted as follows:

```
TLS-PRF-nn(secret, label, seed)
```

where:

nn is the number of generated octets

secret is a secret key

label is a string (without null-terminator)

seed is a binary sequence.

The same PRF function that is used in the TLS handshake MUST be used for all cryptographic operations described in this manner.

The TLS 1.1 PRF has invariant output regardless of how many octets are generated. However, it is possible that alternative PRF functions will include the size of the output sequence as input to the PRF function; this means generating 32 octets and generating 64 octets from the same input parameters will no longer result in the first 32 octets being identical. For this reason, the PRF is always specified with an "nn", indicating the number of generated octets.

### 7.2. Composite Key Computation

The Composite Key is a 40-octet secret derived by cryptographically mixing the TLS master secret and all inner MSKs. It is used as a basis for computing the MSK and/or EMSK when the "Mixed" computation mechanism has been negotiated, as well as for computing Key Confirmation values.

Note that when an intermediate Key Confirmation is performed, an intermediate Composite Key must be computed using only those inner MSKs whose values are available at the time that the TTLS server issues the intermediate Key Confirmation. A subsequent Key Confirmation exchange or MSK/EMSK computation using the "Mixed" mechanism will require a new Composite Key computation, as one or more additional inner MSKs will have become available.

```
Composite Key = TLS-PRF-40(master_secret,
             "ttls composite key",
             client_random +
             server_random +
             inner_session_keys)
```

master_secret, client_random and server_random are security parameters from the TLS handshake.

inner_session_keys is the concatenation of all session keys produced by inner authentication methods. This value is calculated as follows. Treating each session key as an unsigned big-endian binary numeric value (perhaps a rather long number), sort this set of session keys from lowest numeric value to highest. Prefix each session key with a

2-octet big-endian value indicating the length of that session key in octets. Then concatenate the length-prefixed session keys in the determined (sorted) order. Finally, append two octets of zero to terminate the sequence.

If there are no inner authentication methods that produce a session key, inner_session_keys will consist only of two octets of zero.

The session keys are ordered by value rather than by the sequence in which they are produced in order to avoid imposing any restrictions against multiple concurrent inner authentications, which might result in ambiguous orderings.

### 7.3. MSK/EMSK computation using the "Mixed" mechanism

The "Mixed" mechanism for computing the MSK and EMSK uses the following formula:

```
Keying Material = TLS-PRF-128(composite_key,
          "ttls mixed keying material",
          <null>)

MSK = Keying Material [0..63]

EMSK = Keying Material [64..127]
```

where <null> indicates that the seed value is of zero-length.

### 7.4. Key Confirmation computation

The Key Confirmation value transmitted by the client is computed according to the following formula:

```
client_key_confirmation = TLS-PRF-32(composite_key,
          "ttls client key confirmation",
          <null>)
```

The Key Confirmation value transmitted by the TTLS server is computed according to the following formula:

```
server_key_confirmation = TLS-PRF-32(composite_key,
          "ttls server key confirmation",
          <null>)
```

## 8. Security Considerations

The EAP-TTLSv0 extensions defined in this document supplement the security protection provided by EAP-TTLSv0 by adding algorithm agility, cryptographic binding of inner authentications to the outer tunnel, and protected results. These additional security features provide protection against failures in cryptographic algorithms, protection against particular kinds of MITM attacks, protection against truncation attacks, and protection against forged results. The following sections show how this protection is provided.

### 8.1. Cryptographic Algorithm Agility

EAP-TTLS uses cryptographic algorithms in several places. First, it is based on EAP-TLS [EAPTLS] which is based on TLS which uses asymmetric and symmetric encryption algorithms and hash algorithms. Some algorithm agility is provided by the ciphersuite negotiation included in TLS 1.1 but the TLS PRF always uses MD5 and SHA1. Fortunately, TLS 1.2 [TLS12] can be used with EAP-TTLS to provide algorithm agility for the TLS PRF. This can be negotiated with the standard TLS version negotiation mechanism.

However, the original design of EAP-TTLSv0 always uses the TLS 1.1 PRF to calculate the Master Session Key (MSK) and Extended Master Session Key (EMSK). This leaves the system open to attacks based on weaknesses in MD5 and SHA1. The MSK-Computation AVP allows the PRF negotiated during the TLS handshake to be used for calculating MSK and EMSK values, thus allowing a smooth transition to other cryptographic algorithms if MD5 and SHA1 are judged to be inadequate.

### 8.2. Protection Against MITM Attacks

As described in [MITM], if a malicious EAP server can convince a client to authenticate using a particular EAP authentication method and then pass this method through a standard tunneled EAP method, the malicious party can typically gain access as if it had performed the authentication itself. Several countermeasures are possible, such as ensuring that clients only authenticate with trusted servers. But one good way to solve the problem is to modify tunneled EAP methods to verify that the same client is terminating both the inner and the outer authentication method.

The MSK-Computation AVP allows the client and/or server to require that the MSK and EMSK is computed using a mixture of the TLS master secret and the MSKs exported from inner methods. If the MSK and EMSK are used to derive keys necessary for later access (as with 802.1X [8021X]), this will ensure that the attack described above will fail

since the attacker will not know the MSKs exported from the inner methods. The KeyConfirmation AVPs provide even stronger protection against this attack, allowing the TTLS server and TTLS client each to verify that the other party knows the MSKs from all inner methods and the TLS master secret before the authentication terminates. When intermediate key confirmation is used, the TTLS server can verify this at any point in the authentication process. If a MITM attack is detected, the TTLS server can terminate authentication promptly to prevent the later authentication steps from taking place. This can avoid revealing sensitive information to the attacker, such as one-time passwords or biometric data.

### 8.3. Protection Against Truncation Attacks

By truncating an EAP-TTLSv0 session, an attacker could cause the client to believe that the session completed successfully when in fact it was unsuccessful or incomplete. The implications of such an attack depend on the particular inner methods in use and the circumstances of the deployment but they could perhaps result in the client using the wrong keying material for later communications, for example. The Secure-Completion-Option, TTLS-Success, and TTLS-Failure AVPs allow the client to require or request that the server and client securely confirm the results of the authentication to each other before the authentication is considered complete. Thus, any truncation can easily be detected.

### 8.4. Protection Against Forged Results

EAP results (EAP-Success and EAP-Failure) are not generally protected in transit from the server to the client. This means that the client can be led to believe that an authentication failed when it actually succeeded or vice versa. The Secure-Completion, TTLS-Success, and TTLS-Failure AVPs allow the client to securely determine the results of the authentication, thus foiling any attempt to forge or modify the results of the authentication.

## 9. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to this specification, in accordance with [IANA]. The term "IETF Consensus" is used in this section with the meaning defined in [IANA].

### 9.1. Allocation of AVP Codes

This specification requires the allocation and assignment of six AVP Codes to identify new AVPs identified in this specification. As

described in RFC 3588, release of blocks of AVPs requires IETF Consensus.

Once this document is approved as an RFC, the IANA is requested to allocate and assign unique AVP Codes for the following six AVPs defined in this specification: MSK-Computation, Key-Confirmation-Option, Key-Confirmation, Secure-Completion-Option, TTLS-Success, and TTLS-Failure.

Once these allocations and assignments have been made, they should be added to this specification. For each AVP Code, there is a paragraph like this one:

The AVP Code for the MSK-Computation AVP will be assigned by IANA when this document moves to RFC status. In the mean time, a vendor-specific AVP Code of 256 with a Vendor-ID field of 2636 should be used for experimental purposes.

Once the AVP Code for the MSK-Computation AVP has been assigned, this paragraph should be changed to read:

The AVP Code for the MSK-Computation AVP is [Assigned-AVP-Code]. In order to allow implementation of this specification before assignment of that code, a vendor-specific AVP Code of 256 with a Vendor-ID field of 2636 has been used for experimental purposes. Implementations of this RFC MUST accept the new standard AVP Code for the MSK-Computation AVP, MAY accept the vendor-specific AVP Code, and SHOULD only send the standard AVP Code.

This subsection (subsection 9.1) of the IANA Considerations section may be removed once the AVP Codes have been assigned and the document is ready to move to RFC status.

### 9.2. Creation of New Registries

This specification requires the creation of three new registries to be managed by IANA: MSK Computation Methods, Key Confirmation Options, and Secure Completion Options.

Each of these registries should be composed of numeric values in the range 0 to 255. Each value should also have a name. Because of the limited number of values available in each of these registries, values should only be added to these registries if they achieve IETF Consensus as defined in [IANA].

A vendor extension mechanism has been defined to allow vendors to define their own values similar in function to the IANA-reserved

values in these registries. This vendor extension mechanism is described earlier in this specification. Such vendor-specific values are not to be tracked or managed by IANA.

Certain values defined in this specification should be prepopulated into these registries. The next three paragraphs enumerate those values.

For the MSK Computation Methods registry, the values to be prepopulated are 0 (with a name of "Default computation as defined in RFC XXXX") and 1 (with a name of "Mixed computation mechanism as defined in RFC YYYY"), where XXXX is replaced by the RFC number assigned to [TTLSv0] and YYYY is replaced by the RFC number assigned to this specification.

For the Key Confirmation Options registry, the values to be prepopulated are 0 (with a name of "Disabled") and 1 (with a name of "Enabled").

For the Secure Completion Options registry, the values to be prepopulated are 0 (with a name of "Disabled") and 1 (with a name of "Enabled").

When this document is ready to become an RFC, this paragraph and the preceding four paragraphs can be deleted.

## 10. References

### 10.1. Normative References

[KEYWORDS] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[TTLSv0] Funk, P. and S. Blake-Wilson, "EAP Tunneled TLS Authentication Protocol Version 0", Internet Draft (work in progress), draft-funk-eap-ttls-v0-01.txt, April 2007.

### 10.2. Informative References

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

[EAPTLS] Simon, D. and B. Aboba, "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[IANA] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[MITM] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunnelled Authentication Protocols", Cryptology ePrint Archive, Report 2002/163, http://eprint.iacr.org, 2002.

[PEN] IANA Private Enterprise Numbers, http://www.iana.org/assignments/enterprise-numbers.

[TLS12] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", Internet Draft (work in progress), draft-ietf-tls-rfc4346bis-04.txt, July 2007.

[8021X] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std 802.1X-2001, June 2001.

## Authors' Addresses

Steve Hanna
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA

Email: shanna@juniper.net

Paul Funk
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA

Email: pfunk@juniper.net

## Intellectual Property Statement

made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

## Copyright Statement



## Disclaimer of Validity



## Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.