EAP
Internet-Draft
Category: Standards Track
<draft-funk-eap-ttls-v0-01.txt>

Paul Funk
Juniper Networks
Simon Blake-Wilson
Basic Commerce &
Industries, Inc.
April 2007

EAP Tunneled TLS Authentication Protocol Version 0 (EAP-TTLSv0)

Status of this Memo

Copyright Notice

Abstract

EAP-TTLS is an EAP protocol that extends EAP-TLS. In EAP-TLS, a TLS handshake is used to mutually authenticate a client and server. EAP-TTLS extends this authentication negotiation by using the secure connection established by the TLS handshake to exchange additional information between client and server. In EAP-TTLS, the TLS handshake may be mutual; or it may be one-way, in which only the server is authenticated to the client. The secure connection established by the handshake may then be used to allow the server to authenticate the client using existing, widely-deployed authentication infrastructures such as RADIUS. The authentication of the client may itself be EAP, or it may be another authentication protocol such as PAP, CHAP, MS-CHAP or MS-CHAP-V2.

Thus, EAP-TTLS allows legacy password-based authentication protocols to be used against existing authentication databases, while protecting the security of these legacy protocols against eavesdropping, man-in-the-middle and other cryptographic attacks.

EAP-TTLS also allows client and server to establish keying material for use in the data connection between the client and access point. The keying material is established implicitly between client and server based on the TLS handshake.

This document describes EAP-TTLSv0; that is, the original version 0 of the EAP-TTLS protocol.

Table of Contents

1. Introduction

Extensible Authentication Protocol (EAP) [RFC3748] defines a standard message exchange that allows a server to authenticate a client based on an authentication protocol agreed upon by both parties. EAP may be extended with additional authentication protocols by registering such protocols with IANA or by defining vendor specific protocols.

Transport Layer Security (TLS) [RFC4346] is an authentication protocol that provides for client authentication of a server or mutual authentication of client and server, as well as secure ciphersuite negotiation and key exchange between the parties. TLS has been defined as an authentication protocol for use within EAP (EAP-TLS) [RFC2716].

Other authentication protocols are also widely deployed. These are typically password-based protocols, and there is a large installed base of support for these protocols in the form of credential databases that may be accessed by RADIUS, Diameter or other AAA servers. These include non-EAP protocols such as PAP, CHAP, MS-CHAP and MS-CHAP-V2, as well as EAP protocols such as MD5-Challenge.

EAP-TTLS is an EAP protocol that extends EAP-TLS. In EAP-TLS, a TLS handshake is used to mutually authenticate a client and server. EAP-TTLS extends this authentication negotiation by using the secure connection established by the TLS handshake to exchange additional information between client and server. In EAP-TTLS, the TLS handshake may be mutual; or it may be one-way, in which only the server is authenticated to the client. The secure connection established by the handshake may then be used to allow the server to authenticate the client using existing, widely-deployed authentication infrastructures such as RADIUS. The authentication of the client may itself be EAP, or it may be another authentication protocol such as PAP, CHAP, MS-CHAP or MS-CHAP-V2.

Thus, EAP-TTLS allows legacy password-based authentication protocols to be used against existing authentication databases, while protecting the security of these legacy protocols against eavesdropping, man-in-the-middle and other cryptographic attacks.

EAP-TTLS also allows client and server to establish keying material for use in the data connection between the client and access point.

The keying material is established implicitly between client and server based on the TLS handshake.

In EAP-TTLS, client and server communicate using attribute-value pairs encrypted within TLS. This generality allows arbitrary functions beyond authentication and key exchange to be added to the EAP negotiation, in a manner compatible with the AAA infrastructure.

2. Motivation

Most password-based protocols in use today rely on a hash of the password with a random challenge. Thus, the server issues a challenge, the client hashes that challenge with the password and forwards a response to the server, and the server validates that response against the user's password retrieved from its database. This general approach describes CHAP, MS-CHAP, MS-CHAP-V2, EAP/MD5-Challenge and EAP/One-Time Password.

An issue with such an approach is that an eavesdropper that observes both challenge and response may be able to mount a dictionary attack, in which random passwords are tested against the known challenge to attempt to find one which results in the known response. Because passwords typically have low entropy, such attacks can in practice easily discover many passwords.

While this vulnerability has long been understood, it has not been of great concern in environments where eavesdropping attacks are unlikely in practice. For example, users with wired or dial-up connections to their service providers have not been concerned that such connections may be monitored. Users have also been willing to entrust their passwords to their service providers, or at least to allow their service providers to view challenges and hashed responses which are then forwarded to their home authentication servers using, for example, proxy RADIUS, without fear that the service provider will mount dictionary attacks on the observed credentials. Because a user typically has a relationship with a single service provider, such trust is entirely manageable.

With the advent of wireless connectivity, however, the situation changes dramatically:

- Wireless connections are considerably more susceptible to

eavesdropping and man-in-the-middle attacks. These attacks may enable dictionary attacks against low-entropy passwords. In addition, they may enable channel hijacking, in which an attacker gains fraudulent access by seizing control of the communications channel after authentication is complete.

- Existing authentication protocols often begin by exchanging the client's username in the clear. In the context of eavesdropping on the wireless channel, this can compromise the client's anonymity and locational privacy.

- Often in wireless networks, the access point does not reside in the administrative domain of the service provider with which the user has a relationship. For example, the access point may reside in an airport, coffee shop, or hotel in order to provide public access via 802.11. Even if password authentications are protected in the wireless leg, they may still be susceptible to eavesdropping within the untrusted wired network of the access point.

- In the traditional wired world, the user typically intentionally connects with a particular service provider by dialing an associated phone number; that service provider may be required to route an authentication to the user's home domain. In a wireless network, however, the user does not get to choose an access domain, and must connect with whichever access point is nearby; providing for the routing of the authentication from an arbitrary access point to the user's home domain may pose a challenge.

Thus, the authentication requirements for a wireless environment that EAP-TTLS attempts to address can be summarized as follows:

- Legacy password protocols must be supported, to allow easy deployment against existing authentication databases.

- Password-based information must not be observable in the communications channel between the client node and a trusted service provider, to protect the user against dictionary attacks.

- The user's identity must not be observable in the communications channel between the client node and a trusted service provider, to protect the user's locational privacy against surveillance,

undesired acquisition of marketing information, and the like.

- The authentication process must result in the distribution of shared keying information to the client and access point to permit encryption and validation of the wireless data connection subsequent to authentication, to secure it against eavesdroppers and prevent channel hijacking.

- The authentication mechanism must support roaming among small access domains with which the user has no relationship and which will have limited capabilities for routing authentication requests.

3. Terminology

AAA

Authentication, Authorization and Accounting - functions that are generally required to control access to a network and support billing and auditing.

AAA protocol

A network protocol used to communicate with AAA servers; examples include RADIUS and Diameter.

AAA server

A server which performs one or more AAA functions: authenticating a user prior to granting network service, providing authorization (policy) information governing the type of network service the user is to be granted, and accumulating accounting information about actual usage.

AAA/H

A AAA server in the user's home domain, where authentication and authorization for that user are administered.

access point

A network device providing users with a point of entry into the network, and which may enforce access control and policy based on information returned by a AAA server. For the purposes of this

document, "access point" and "NAS" are architecturally equivalent. "Access point" is used throughout because it is suggestive of devices used for wireless access; "NAS" is used when more traditional forms of access, such as dial-up, are discussed.

access domain

The domain, including access points and other devices, that provides users with an initial point of entry into the network; for example, a wireless hot spot.

client

A host or device that connects to a network through an access point.

domain

A network and associated devices that are under the administrative control of an entity such as a service provider or the user's home organization.

link layer protocol

A protocol used to carry data between hosts that are connected within a single network segment; examples include PPP and Ethernet.

NAI

A Network Access Identifier [RFC4282], normally consisting of the name of the user and, optionally, the user's home realm.

NAS

A network device providing users with a point of entry into the network, and which may enforce access control and policy based on information returned by a AAA server. For the purposes of this document, "access point" and "NAS" are architecturally equivalent. "Access point" is used throughout because it is suggestive of devices used for wireless access; "NAS" is used when more traditional forms of access, such as dial-up, are discussed.

proxy

A server that is able to route AAA transactions to the appropriate AAA server, possibly in another domain, typically based on the realm portion of an NAI.

realm

The optional part of an NAI indicating the domain to which a AAA transaction is to be routed, normally the user's home domain.

service provider

An organization with which a user has a business relationship, that provides network or other services. The service provider may provide the access equipment with which the user connects, may perform authentication or other AAA functions, may proxy AAA transactions to the user's home domain, etc.

TTLS server

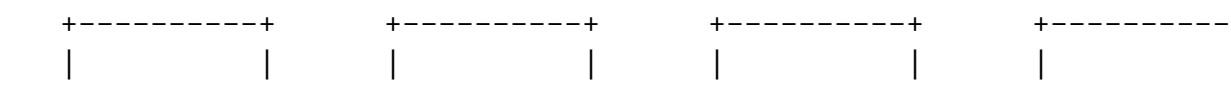
A AAA server which implements EAP-TTLS. This server may also be capable of performing user authentication, or it may proxy the user authentication to a AAA/H.

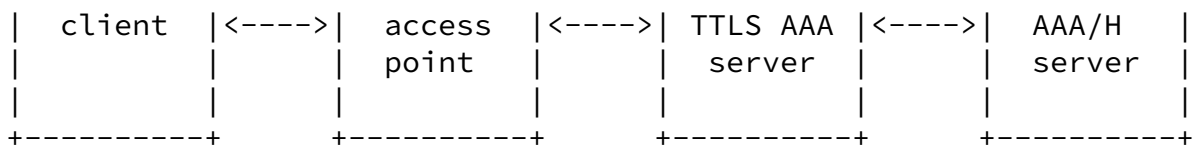
user

The person operating the client device. Though the line is often blurred, "user" is intended to refer to the human being who is possessed of an identity (username), password or other authenticating information, and "client" is intended to refer to the device which makes use of this information to negotiate network access. There may also be clients with no human operators; in this case the term "user" is a convenient abstraction.

4. Architectural Model

The network architectural model for EAP-TTLS usage and the type of security it provides is shown below.





The entities depicted above are logical entities and may or may not correspond to separate network components. For example, the TTLS server and AAA/H server might be a single entity; the access point and TTLS server might be a single entity; or, indeed, the functions of the access point, TTLS server and AAA/H server might be combined

into a single physical device. The above diagram illustrates the division of labor among entities in a general manner and shows how a distributed system might be constructed; however, actual systems might be realized more simply.

Note also that one or more AAA proxy servers might be deployed between access point and TTLS server, or between TTLS server and AAA/H server. Such proxies typically perform aggregation or are required for realm-based message routing. However, such servers play no direct role in EAP-TTLS and are therefore not shown.

4.1 Carrier Protocols

The entities shown above communicate with each other using carrier protocols capable of encapsulating EAP. The client and access point communicate using a link layer carrier protocol such as PPP or EAPOL. The access point, TTLS server and AAA/H server communicate using a AAA carrier protocol such as RADIUS or Diameter.

EAP, and therefore EAP-TTLS, must be initiated via the link layer protocol. In PPP or EAPOL, for example, EAP is initiated when the access point sends an EAP-Request/Identity packet to the client.

The keying material used to encrypt and authenticate the data connection between the client and access point is developed implicitly between the client and TTLS server as a result of EAP-TTLS negotiation. This keying material must be communicated to the

access point by the TTLS server using the AAA carrier protocol.

The client and access point must also agree on an encryption/validation algorithm to be used based on the keying material. In some systems, both these devices may be preconfigured with this information, and distribution of the keying material alone is sufficient. Or, the link layer protocol may provide a mechanism for client and access point to negotiate an algorithm.

In the most general case, however, it may be necessary for both client and access point to communicate their algorithm preferences to the TTLS server, and for the TTLS server to select one and communicate its choice to both parties. This information would be transported between access point and TTLS server via the AAA protocol, and between client and TTLS server via EAP-TTLS in encrypted form.

4.2 Security Relationships

The client and access point have no pre-existing security relationship.

The access point, TTLS server and AAA/H server are each assumed to have a pre-existing security association with the adjacent entity

with which it communicates. With RADIUS, for example, this is achieved using shared secrets. It is essential for such security relationships to permit secure key distribution.

The client and AAA/H server have a security relationship based on the user's credentials such as a password.

The client and TTLS server may have a one-way security relationship based on the TTLS server's possession of a private key guaranteed by a CA certificate which the user trusts, or may have a mutual security relationship based on certificates for both parties.

4.3 Messaging

The client and access point initiate an EAP conversation to negotiate the client's access to the network. Typically, the access point issues an EAP-Request/Identity to the client, which responds

with an EAP-Response/Identity. Note that the client does not include the user's actual identity in this EAP-Response/Identity packet; the user's identity will not be transmitted until an encrypted channel has been established.

The access point now acts as a passthrough device, allowing the TTLS server to negotiate EAP-TTLS with the client directly.

During the first phase of the negotiation, the TLS handshake protocol is used to authenticate the TTLS server to the client and, optionally, to authenticate the client to the TTLS server, based on public/private key certificates. As a result of the handshake, client and TTLS server now have shared keying material and an agreed upon TLS record layer cipher suite with which to secure subsequent EAP-TTLS communication.

During the second phase of negotiation, client and TTLS server use the secure TLS record layer channel established by the TLS handshake as a tunnel to exchange information encapsulated in attribute-value pairs, to perform additional functions such as authentication (one-way or mutual), validation of client integrity and configuration, provisioning of information required for data connectivity, etc.

If a tunneled client authentication is performed, the TTLS server de-tunnels and forwards the authentication information to the AAA/H. If the AAA/H performs a challenge, the TTLS server tunnels the challenge information to the client. The AAA/H server may be a legacy device and needs to know nothing about EAP-TTLS; it only needs to be able to authenticate the client based on commonly used authentication protocols.

Keying material for the subsequent data connection between client and access point may be generated based on secret information developed during the TLS handshake between client and TTLS server.

At the conclusion of a successful authentication, the TTLS server may transmit this keying material to the access point, encrypted based on the existing security associations between those devices (e.g., RADIUS).

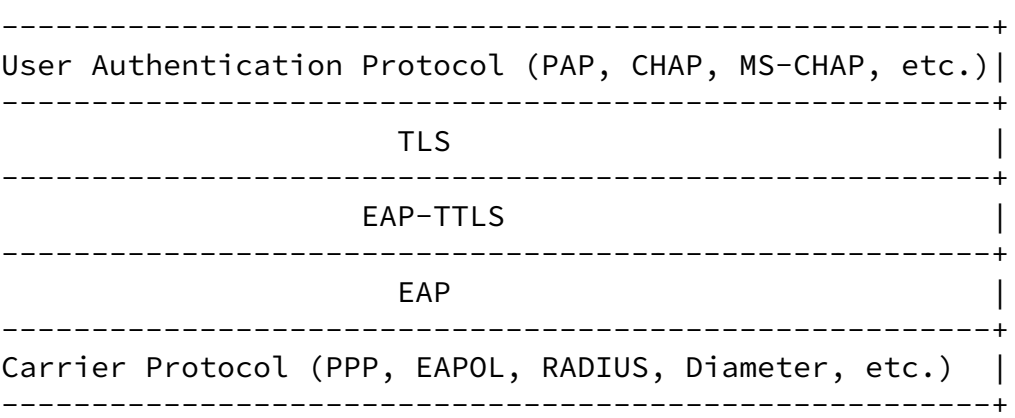
The client and access point now share keying material which they can use to encrypt data traffic between them.

4.4 Resulting Security

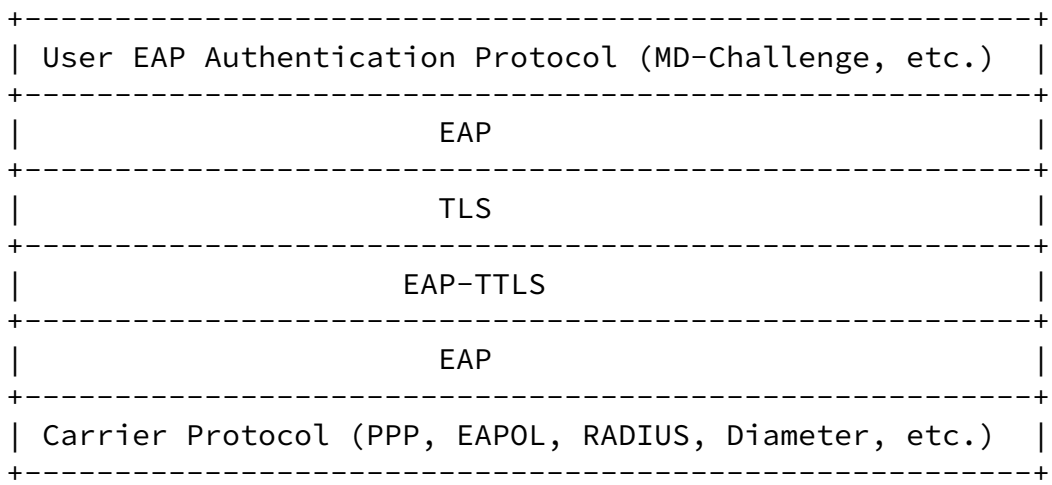
As the diagram above indicates, EAP-TTLS allows user identity and password information to be securely transmitted between client and TTLS server, and performs key distribution to allow network data subsequent to authentication to be securely transmitted between client and access point.

5. Protocol Layering Model

EAP-TTLS packets are encapsulated within EAP, and EAP in turn requires a carrier protocol to transport it. EAP-TTLS packets themselves encapsulate TLS, which is then used to encapsulate user authentication information. Thus, EAP-TTLS messaging can be described using a layered model, where each layer is encapsulated by the layer beneath it. The following diagram clarifies the relationship between protocols:



When the user authentication protocol is itself EAP, the layering is as follows:



Methods for encapsulating EAP within carrier protocols are already defined. For example, PPP [RFC1661] or EAPOL [802.1X] may be used to transport EAP between client and access point; RADIUS [RFC2865] or Diameter [RFC3588] are used to transport EAP between access point and TTLS server.

6. EAP-TTLS Overview

A EAP-TTLS negotiation comprises two phases: the TLS handshake phase and the TLS tunnel phase.

During phase 1, TLS is used to authenticate the TTLS server to the client and, optionally, the client to the TTLS server. Phase 1 results in the activation of a cipher suite, allowing phase 2 to proceed securely using the TLS record layer. (Note that the type and degree of security in phase 2 depends on the cipher suite negotiated during phase 1; if the null cipher suite is negotiated, there will be no security!)

During phase 2, the TLS record layer is used to tunnel information between client and TTLS server to perform any of a number of functions. These might include user authentication, client integrity validation, negotiation of data communication security capabilities, key distribution, communication of accounting information, etc.. Information between client and TTLS server is exchanged via attribute-value pairs (AVPs) compatible with RADIUS and Diameter; thus, any type of function that can be implemented via such AVPs may easily be performed.

EAP-TTLS specifies how user authentication may be performed during phase 2. The user authentication may itself be EAP, or it may be a legacy protocol such as PAP, CHAP, MS-CHAP or MS-CHAP-V2. Phase 2 user authentication may not always be necessary, since the user may

already have been authenticated via the mutual authentication option of the TLS handshake protocol.

Functions other than authentication may also be performed during phase 2. This document does not define any such functions; however, any organization or standards body is free to specify how additional functions may be performed through the use of appropriate AVPs.

EAP-TTLS specifies how keying material for the data connection between client and access point is generated. The keying material is developed implicitly between client and TTLS server based on the results of the TLS handshake; the TTLS server will communicate the keying material to the access point over the carrier protocol.

6.1 Phase 1: Handshake

In phase 1, the TLS handshake protocol is used to authenticate the TTLS server to the client and, optionally, to authenticate the client to the TTLS server.

Phase 1 is initiated when the client sends an EAP-Response/Identity packet to the TTLS server. This packet specifically should not include the name of the user; however, it may include the name of the realm of a trusted provider to which EAP-TTLS packets should be forwarded; for example, "@myisp.com".

The TTLS server responds to the EAP-Response/Identity packet with an EAP-TTLS/Start packet, which is an EAP-Request with Type = EAP-TTLS, the S (Start) bit set, and no data. This indicates to the client that it should begin TLS handshake by sending a ClientHello message.

EAP packets continue to be exchanged between client and TTLS server to complete the TLS handshake, as described in [RFC2716]. Phase 1 is completed when the client and TTLS server exchange ChangeCipherSpec and Finished messages. At this point, additional information may be securely tunneled.

As part of the TLS handshake protocol, the TTLS server will send its certificate along with a chain of certificates leading to the certificate of a trusted CA. The client will need to be configured

with the certificate of the trusted CA in order to perform the authentication.

If certificate-based authentication of the client is desired, the client must have been issued a certificate and must have the private key associated with that certificate

6.2 Phase 2: Tunnel

In phase 2, the TLS Record Layer is used to securely tunnel information between client and TTLS server. This information is encapsulated in sequences of attribute-value pairs (AVPS), whose use and format are described in later sections.

Any type of information may be exchanged during phase 2, according to the requirements of the system. (It is expected that applications utilizing EAP-TTLS will specify what information must be exchanged and therefore which AVPs must be supported.)

The client begins the phase 2 exchange by encoding information in a sequence of AVPs, passing this sequence to the TLS record layer for encryption, and sending the resulting data to the TTLS server.

The TTLS server recovers the AVPs in clear text from the TLS record layer. If the AVP sequence includes authentication information, it forwards this information to the AAA/H server using the AAA carrier protocol. Note that the EAP-TTLS and AAA/H servers may be one and the same, in which case it simply processes the information locally.

The TTLS server may respond with its own sequence of AVPs. The TTLS server passes the AVP sequence to the TLS record layer for encryption and sends the resulting data to the client. For example, the TTLS server may forward an authentication challenge received from the AAA/H.

This process continues until the TTLS server has enough information to issue either an EAP-Success or EAP-Failure. Thus, if the AAA/H rejects the client based on forwarded authentication information, the TTLS server would issue an EAP-Failure. If the AAA/H accepts the client, the TTLS server would issue an EAP-Success.

The TTLS server distributes data connection keying information and other authorization information to the access point in the same AAA carrier protocol message that carries the EAP-Success.

6.3 Piggybacking

While it is convenient to describe EAP-TTLS messaging in terms of two phases, it is sometimes required that a single EAP-TTLS packet to contain both phase 1 and phase 2 TLS messages.

Such "piggybacking" occurs when the party that completes the handshake also has AVPs to send. For example, when negotiating a resumed TLS session, the TTLS server sends its ChangeCipherSpec and Finished messages first, then the client sends its own ChangeCipherSpec and Finished messages to conclude the handshake. If the client has authentication or other AVPs to send to the TTLS server, it MUST tunnel those AVPs within the same EAP-TTLS packet immediately following its Finished message. If the client fails to do this, the TTLS server will incorrectly assume that the client has no AVPs to send, and the outcome of the negotiation could be affected.

6.4 Session Resumption

When a client and TTLS server that have previously negotiated a EAP-TTLS session begin a new EAP-TTLS negotiation, the client and TTLS server MAY agree to resume the previous session. This significantly reduces the time required to establish the new session. This could occur when the client connects to a new access point, or when an access point requires reauthentication of a connected client.

Session resumption is accomplished using the standard TLS mechanism. The client signals its desire to resume a session by including the session ID of the session it wishes to resume in the ClientHello message; the TTLS server signals its willingness to resume that session by echoing that session ID in its ServerHello message.

If the TTLS server elects not to resume the session, it simply does not echo the session ID, causing a new session to be negotiated.

This could occur if the TTLS server is configured not to resume sessions, if it has not retained the requested session's state, or if the session is considered stale. A TTLS server may consider the session stale based on its own configuration, or based on session-limiting information received from the AAA/H (e.g., the RADIUS Session-Timeout attribute).

Tunneled authentication is specifically not performed for resumed sessions; the presumption is that the knowledge of the master secret as evidenced by the ability to resume the session is authentication enough. This allows session resumption to occur without any messaging between the TTLS server and the AAA/H. If periodic reauthentication to the AAA/H is desired, the AAA/H must indicate this to the TTLS server when the original session is established, for example, using the RADIUS Session-Timeout attribute.

The client MAY send other AVPs in its first phase 2 message, to initiate non-authentication functions. If it does not, the TTLS server, at its option, MAY send AVPs to the client to initiate non-authentication functions, or MAY simply complete the EAP-TTLS negotiation by sending EAP-Success or EAP-Failure.

The TTLS server MUST retain authorization information returned by the AAA/H for use in resumed sessions. A resumed session MUST operate under the same authorizations as the original session, and the TTLS server must be prepared to send the appropriate information back to the access point. Authorization information might include the maximum time for the session, the maximum allowed bandwidth, packet filter information and the like. The TTLS server is responsible for modifying time values, such as Session-Timeout, appropriately for each resumed session.

A TTLS server must not permit a session to be resumed if that session did not result in a successful authentication of the user

during phase 2. The consequence of incorrectly implementing this aspect of session resumption would be catastrophic; any attacker could easily gain network access by first initiating a session that succeeds in the TLS handshake but fails during phase 2 authentication, and then resuming that session.

[Implementation note: Toolkits that implement TLS often cache

resumable TLS sessions automatically. Implementers must take care to override such automatic behavior, and prevent sessions from being cached for possible resumption until the user has been positively authenticated during phase 2.]

6.4.1 TTLS Server Guidelines for Session Resumption

When a domain comprises multiple TTLS servers, a client's attempt to resume a session may fail because each EAP-TTLS negotiation may be routed to a different TTLS server.

One strategy to ensure that subsequent EAP-TTLS negotiations are routed to the original TTLS server is for each TTLS server to encode its own identifying information, for example, IP address, in the session IDs that it generates. This would allow any TTLS server receiving a session resumption request to forward the request to the TTLS server that established the original session.

7. Generating Keying Material

Upon successful conclusion of an EAP-TTLS negotiation, 128 octets of keying material is generated and exported for use in securing the data connection between client and access point. The first 64 octets of the keying material constitutes the MSK, the second 64 octets constitutes the EMSK.

The keying material is generated using the TLS PRF function [RFC4346], with inputs consisting of the TLS master secret, the ASCII-encoded constant string "ttls keying material", the TLS client random, and the TLS server random. The constant string is not null-terminated.

```
Keying Material = PRF(SecurityParameters.master_secret,
          "ttls keying material",
          SecurityParameters.client_random +
          SecurityParameters.server_random) [0..127]

MSK = Keying Material [0..63]

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

Note that the order of client_random and server_random for EAP-TTLS is reversed from that of the TLS protocol [RFC4346]. This ordering follows the key derivation method of EAP-TLS [RFC2716]. Altering the

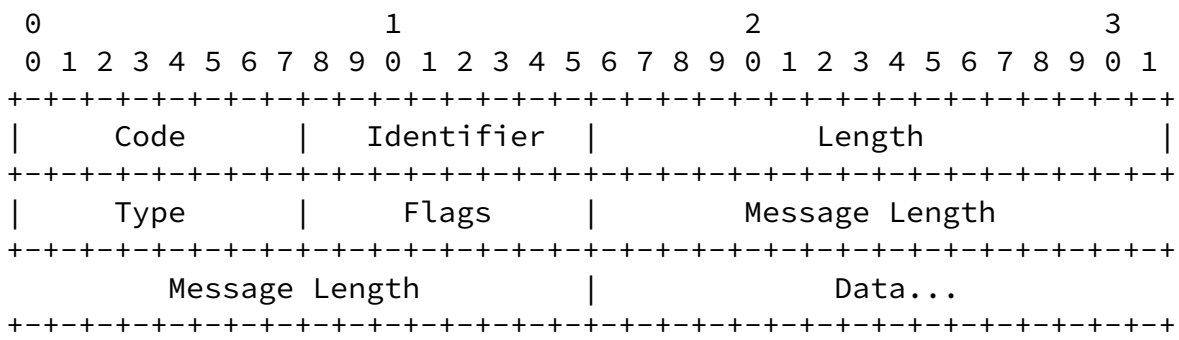
order of randoms avoids namespace collisions between constant strings defined for EAP-TTLS and those defined for the TLS protocol.

The TTLS server distributes this keying material to the access point via the AAA carrier protocol. When RADIUS is the AAA carrier protocol, the MPPE-Recv-Key and MPPE-Send-Key attributes may be used to distribute the first 32 octets and second 32 octets of the MSK, respectively.

8. EAP-TTLS Protocol

8.1 Packet Format

The EAP-TTLS packet format is shown below. The fields are transmitted left to right.



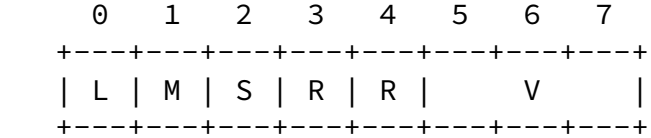
Code
1 for request, 2 for response.

Identifier
The Identifier field is one octet and aids in matching responses with requests. The Identifier field MUST be changed for each request packet and MUST be echoed in each response packet.

Length
The Length field is two octets and indicates the number of octets in the entire EAP packet, from the Code field through the Data field.

Type
21 (EAP-TTLS)

Flags



L = Length included
M = More fragments
S = Start

R = Reserved
V = Version (000 for EAP-TTLSv0)

The L bit is set to indicate the presence of the four octet TLS Message Length field. The M bit indicates that more fragments are to come. The S bit indicates a Start message. The V bit is set to the version of EAP-TTLS, and is set to 000 for EAP-TTLSv0.

Message Length
The Message Length field is four octets, and is present only if the L bit is set. This field provides the total length of the raw data message sequence prior to fragmentation.

Data
For all packets other than a Start packet, the Data field consists of the raw TLS message sequence or fragment thereof. For a Start packet, the Data field may optionally contain an AVP sequence.

8.2 EAP-TTLS Start Packet

The S bit MUST be set on the first packet sent by the server to initiate the EAP-TTLS protocol. It MUST NOT be set on any other packet.

This packet MAY contain additional information in the form of AVPs, which may provide useful hints to the client; for example, the server identity may be useful to the client to allow it to pick the correct TLS session ID for session resumption. Each AVP must begin on a 4-octet boundary relative to the first AVP in the sequence. If an AVP is not a multiple of 4 octets, it must be padded with 0s to the next 4-octet boundary.

8.2.1 Version Negotiation

The version of EAP-TTLS is negotiated in the first exchange between server and client. The server sets the highest version number of

EAP-TTLS that it supports in the V field of its Start message (in the case of EAP-TTLSv0, this is 0). In its first EAP message in response, the client sets the V field to the highest version number that it supports that is no higher than the version number offered by the server. If the client version is not acceptable to the server, it sends an EAP-Failure to terminate the EAP session. Otherwise, the version sent by the client is the version of EAP-TTLS that MUST be used, and both server and client set the V field to that version number in all subsequent EAP messages.

8.2.2 Fragmentation

Each EAP-TTLS message contains a sequence of TLS messages that represent a single leg of a half-duplex conversation. The EAP

carrier protocol (e.g., PPP, EAPOL, RADIUS) may impose constraints on the length of of an EAP message. Therefore it may be necessary to fragment an EAP-TTLS message across multiple EAP messages.

Each fragment except for the last MUST have the M bit set, to indicate that more data is to follow; the final fragment MUST NOT have the M bit set.

If there are multiple fragments, the first fragment MUST have the L bit set and include the length of the entire raw message prior to fragmentation. Fragments other than the first MUST NOT have the L bit set. Unfragmented messages MAY have the L bit set and include the length of the message (though this information is redundant).

Upon receipt of a packet with M bit set, the receiver MUST transmit an Acknowledgement packet. The receiver is responsible for reassembly of fragmented packets.

8.2.3 Acknowledgement Packets

An Acknowledgement packet is an EAP-TTLS packet with no additional data beyond the Flags octet, and with the L, M and S bits of the Flags octet set to 0. (Note, however, that the V field MUST still be set to the appropriate version number.)

An Acknowledgement packet is sent for the following purposes:

- Fragment Acknowledgement

  A Fragment Acknowledgement is sent in response to an EAP packet with M bit set.

- When the final EAP packet of the EAP-TTLS negotiation is sent by the TTLS server, the client must respond with an Acknowledgement packet, to allow the TTLS server to issue its final EAP-Success or EAP-Failure packet.

9. Encapsulation of AVPs within the TLS Record Layer

Subsequent to the TLS handshake, information is tunneled between client and TTLS server through the use of attribute-value pairs (AVPs) encrypted within the TLS record layer.

The AVP format chosen for EAP-TTLS is compatible with the Diameter AVP format. This does not at all represent a requirement that Diameter be supported by any of the devices or servers participating in an EAP-TTLS negotiation. Use of this format is merely a convenience. Diameter is a superset of RADIUS and includes the RADIUS attribute namespace by definition, though it does not limit the size of an AVP as does RADIUS; RADIUS, in turn, is a widely

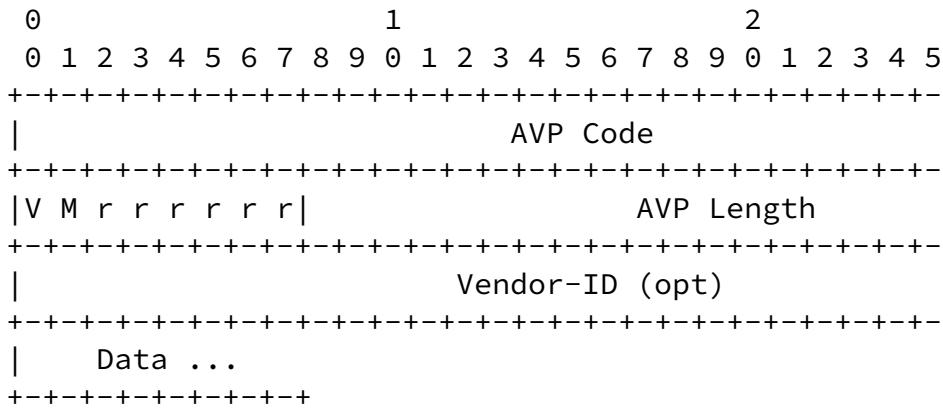
deployed AAA protocol and attribute definitions exist for all commonly used password authentication protocols, including EAP.

Thus, Diameter is not considered normative except as specified in this document. Specifically, the AVP Codes used in EAP-TTLS are semantically equivalent to those defined for Diameter, and, by extension, RADIUS. Also, the representation of the Data field of an AVP in EAP-TTLS is identical to that of Diameter.

Use of the RADIUS/Diameter namespace allows a TTLS server to easily translate between AVPs it uses to communicate to clients and the protocol requirements of AAA servers that are widely deployed. Plus, it provides a well-understood mechanism to allow vendors to extend that namespace for their particular requirements.

9.1 AVP Format

The format of an AVP is shown below. All items are in network, or big-endian, order; that is, they have most significant octet first.



AVP Code

The AVP Code is four octets and, combined with the Vendor-ID field if present, identifies the attribute uniquely. The first 256 AVP numbers represent attributes defined in RADIUS. AVP numbers 256 and above are defined in Diameter.

AVP Flags

The AVP Flags field is one octet, and provides the receiver with information necessary to interpret the AVP.

The 'V' (Vendor-Specific) bit indicates whether the optional Vendor-ID field is present. When set to 1, the Vendor-ID field is present and the AVP Code is interpreted according to the namespace defined by the vendor indicated in the Vendor-ID field.

The 'M' (Mandatory) bit indicates whether support of the AVP is required. If this bit is set to 0, this indicates that the AVP

may be safely ignored if the receiving party does not understand or support it. If set to 1, this indicates that the receiving party must fail the negotiation if it does not understand the AVP; for a TTLS server, this would imply returning EAP-Failure, for a client, this would imply abandoning the negotiation.

The 'r' (reserved) bits are unused and must be set to 0.

AVP Length

The AVP Length field is three octets, and indicates the length of this AVP including the AVP Code, AVP Length, AVP Flags, Vendor-ID (if present) and Data.

Vendor-ID

The Vendor-ID field is present if the 'V' bit is set in the AVP Flags field. It is four octets, and contains the vendor's IANA-assigned "SMI Network Management Private Enterprise Codes" [RFC1700] value. Vendors defining their own AVPs must maintain a consistent namespace for use of those AVPs within RADIUS, Diameter and EAP-TTLS.

A Vendor-ID value of zero is equivalent to absence of the Vendor-ID field altogether.

9.2 AVP Sequences

Data encapsulated within the TLS Record Layer must consist entirely of a sequence of zero or more AVPs. Each AVP must begin on a 4-octet boundary relative to the first AVP in the sequence. If an AVP is not a multiple of 4 octets, it must be padded with 0s to the next 4-octet boundary.

Note that the AVP Length does not include the padding.

9.3 Guidelines for Maximum Compatibility with AAA Servers

For maximum compatibility, the following guidelines for AVP usage are suggested:

- Non-vendor-specific AVPs should be selected from the set of attributes defined for RADIUS; that is, attributes with codes less than 256. This provides compatibility with both RADIUS and Diameter.

- Vendor-specific AVPs should be defined in terms of RADIUS. Vendor-specific RADIUS attributes translate to Diameter (and, hence, to EAP-TTLS) automatically; the reverse is not true. RADIUS vendor-specific attributes use RADIUS attribute 26 and include vendor ID, vendor-specific attribute code and length; see

[RFC2865] for details.

10. Tunneled Authentication

EAP-TTLS permits user authentication information to be tunneled within the TLS record layer between client and TTLS server, guaranteeing the security of the authentication information against active and passive attack between the client and TTLS server. The TTLS server decrypts and forwards this information to the AAA/H over the AAA carrier protocol.

Any type of password or other authentication may be tunneled. Also, multiple tunneled authentications may be performed. Normally, tunneled authentication is used when the client has not been issued a certificate and the TLS handshake provides only one-way authentication of the TTLS server to the client; however, in certain cases it may be desired to perform certificate authentication of the client during the TLS handshake as well as tunneled user authentication afterwards.

10.1 Implicit challenge

Certain authentication protocols that use a challenge/response mechanism rely on challenge material that is not generated by the authentication server, and therefore require special handling.

In CHAP, MS-CHAP and MS-CHAP-V2, for example, the NAS issues a challenge to the client, the client then hashes the challenge with the password and forwards the response to the NAS. The NAS then forwards both challenge and response to a AAA server. But because the AAA server did not itself generate the challenge, such protocols are susceptible to replay attack.

If the client were able to create both challenge and response, anyone able to observe a CHAP or MS-CHAP exchange could pose as that user, even using EAP-TTLS.

To make these protocols secure under EAP-TTLS, it is necessary to provide a mechanism to produce a challenge that the client cannot control or predict. This is accomplished using the same technique described above for generating data connection keying material.

When a challenge-based authentication mechanism is used, both client and TTLS server use the pseudo-random function to generate as many octets as are required for the challenge, using the constant string "ttls challenge", based on the master secret and random values established during the handshake:

```
EAP-TTLS_challenge = PRF(SecurityParameters.master_secret,
                         "ttls challenge",
                         SecurityParameters.client_random +
                         SecurityParameters.server_random);
```

10.2 Tunneled Authentication Protocols

This section describes the methods for tunneling specific authentication protocols within EAP-TTLS.

For the purpose of explication, it is assumed that the TTLS server and AAA/H use RADIUS as a AAA carrier protocol between them. However, this is not a requirement, and any AAA protocol capable of carrying the required information may be used.

10.2.1 EAP

When EAP is the tunneled authentication protocol, each tunneled EAP packet between the client and TTLS server is encapsulated in an EAP-Message AVP, prior to tunneling via the TLS record layer.

The client's first tunneled EAP packet within phase 2 will contain the EAP-Response/Identity. The client places the actual username in this packet; the privacy of the user's identity is now guaranteed by the TLS encryption. This username must be a Network Access Identifier (NAI) [RFC4282]; that is, it must be in the following format:

username@realm

The @realm portion is optional, and is used to allow the TTLS server to forward the EAP packet to the appropriate AAA/H.

Note that the client has two opportunities to specify realms. The first, in the initial EAP-Response/Identity packet, indicates the realm of the TTLS server. The second, in the tunneled authentication, indicates the realm of the client's home network.

Thus, the access point need only know how to route to the realm of the TTLS server; the TTLS server is assumed to know how to route to the client's home realm. This serial routing architecture is anticipated to be useful in roaming environments, allowing access points or AAA proxies behind access points to be configured only with a small number of realms.

Upon receipt of the tunneled EAP-Response/Identity, the TTLS server forwards it to the AAA/H in a RADIUS Access-Request.

The AAA/H may immediately respond with an Access-Reject, in which case the TTLS server completes the negotiation by sending an EAP-Failure to the access point. This could occur if the AAA/H does not recognize the user's identity, or if it does not support EAP.

If the AAA/H does recognize the user's identity and does support EAP, it responds with an Access-Challenge containing an EAP-Request, with the Type and Type-Data fields set according to the EAP protocol with which the AAA/H wishes to authenticate the client; for example MD5-Challenge, OTP or Generic Token Card.

The EAP authentication between client and AAA/H proceeds normally, as described in [RFC3748], with the TTLS server acting as a passthrough device. Each EAP-Request sent by the AAA/H in an Access-Challenge is tunneled by the TTLS server to the client, and each EAP-Response tunneled by the client is decrypted and forwarded by the TTLS server to the AAA/H in an Access-Request.

This process continues until the AAA/H issues an Access-Accept or Access-Reject, at which point the TTLS server completes the negotiation by sending an EAP-Success or EAP-Failure to the access point using the AAA carrier protocol.

10.2.2 CHAP

The CHAP algorithm is described in [RFC1661]; RADIUS attribute formats are described in [RFC2865].

Both client and TTLS server generate 17 octets of challenge material, using the constant string "ttls challenge" as described above. These octets are used as follows:

```
CHAP-Challenge    [16 octets]
CHAP Identifier   [1 octet]
```

The client tunnels User-Name, CHAP-Challenge and CHAP-Password AVPs to the TTLS server. The CHAP-Challenge value is taken from the challenge material. The CHAP-Password consists of CHAP Identifier, taken from the challenge material; and CHAP response, computed according to the CHAP algorithm.

Upon receipt of these AVPs from the client, the TTLS server must verify that the value of the CHAP-Challenge AVP and the value of the CHAP Identifier in the CHAP-Password AVP are equal to the values generated as challenge material. If either item does not match exactly, the TTLS server must reject the client. Otherwise, it forwards the AVPs to the AAA/H in an Access-Request.

The AAA/H will respond with an Access-Accept or Access-Reject. The TTLS server will then issue an EAP-Success or EAP-Failure to the access point.

10.2.3 MS-CHAP

The MS-CHAP algorithm is described in [RFC2433]; RADIUS attribute formats are described in [RFC2548].

Both client and TTLS server generate 9 octets of challenge material, using the constant string "ttls challenge" as described above. These octets are used as follows:

```
MS-CHAP-Challenge [8 octets]
Ident             [1 octet]
```

The client tunnels User-Name, MS-CHAP-Challenge and MS-CHAP-Response AVPs to the TTLS server. The MS-CHAP-Challenge value is taken from the challenge material. The MS-CHAP-Response consists of Ident, taken from the challenge material; Flags, set according the client preferences; and LM-Response and NT-Response, computed according to the MS-CHAP algorithm.

Upon receipt of these AVPs from the client, the TTLS server MUST verify that the value of the MS-CHAP-Challenge AVP and the value of the Ident in the client's MS-CHAP-Response AVP are equal to the

values generated as challenge material. If either item does not match exactly, the TTLS server MUST reject the client. Otherwise, it forwards the AVPs to the AAA/H in an Access-Request.

The AAA/H will respond with an Access-Accept or Access-Reject. The TTLS server will then issue an EAP-Success or EAP-Failure to the access point.

10.2.4 MS-CHAP-V2

The MS-CHAP-V2 algorithm is described in [RFC2759]; RADIUS attribute formats are described in [RFC2548].

Both client and TTLS server generate 17 octets of challenge material, using the constant string "ttls challenge" as described above. These octets are used as follows:

```
MS-CHAP-Challenge [16 octets]
Ident              [1 octet]
```

The client tunnels User-Name, MS-CHAP-Challenge and MS-CHAP2-Response AVPs to the TTLS server. The MS-CHAP-Challenge value is taken from the challenge material. The MS-CHAP2-Response consists of Ident, taken from the challenge material; Flags, set to 0; Peer-Challenge, set to a random value; and Response, computed according to the MS-CHAP-V2 algorithm.

Upon receipt of these AVPs from the client, the TTLS server MUST verify that the value of the MS-CHAP-Challenge AVP and the value of the Ident in the client's MS-CHAP2-Response AVP are equal to the values generated as challenge material. If either item does not match exactly, the TTLS server MUST reject the client. Otherwise, it forwards the AVPs to the AAA/H in an Access-Request.

If the authentication is successful, the AAA/H will respond with an Access-Accept containing the MS-CHAP2-Success attribute. This attribute contains a 42-octet string that authenticates the AAA/H to the client based on the Peer-Challenge. The TTLS server tunnels this AVP to the client. Note that the authentication is not yet complete; the client must still accept the authentication response of the AAA/H.

Upon receipt of the MS-CHAP2-Success AVP, the client is able to authenticate the AAA/H. If the authentication succeeds, the client sends an EAP-TTLS packet to the TTLS server containing no data. Upon receipt of the empty EAP-TTLS packet from the client, the TTLS server now issues an EAP-Success.

If the authentication fails, the AAA/H will respond with an Access-Challenge containing the MS-CHAP2-Error attribute. This attribute contains a new Ident and a string with addition information such as error reason and whether a retry is allowed. If the error reason is an expired password and a retry is allowed, the client may proceed to change the user's password. If the error reason is not an expired password or if the client does not wish to change the user's password, it simply abandons the EAP-TTLS negotiation.

If the client does wish to change the password, it tunnels MS-CHAP-NT-Enc-PW, MS-CHAP2-CPW, and MS-CHAP-Challenge AVPs to the TTLS server. The MS-CHAP2-CPW AVP is derived from the new Ident and Challenge received in the MS-CHAP2-Error AVP. The MS-CHAP-Challenge AVP simply echoes the new Challenge.

Upon receipt of these AVPs from the client, the TTLS server MUST verify that the value of the MS-CHAP-Challenge AVP and the value of the Ident in the client's MS-CHAP2-CPW AVP match the values it sent in the MS-CHAP2-Error AVP. If either item does not match exactly, the TTLS server MUST reject the client. Otherwise, it forwards the AVPs to the AAA/H in an Access-Request.

If the authentication is successful, the AAA/H will respond with an Access-Accept containing the MS-CHAP2-Success attribute. At this point, the negotiation proceeds as described above; the TTLS server tunnels the MS-CHAP2-Success to the client, the client authenticates the AAA/H based on this AVP, it either abandons the negotation on failure or sends an EAP-TTLS packet to the TTLS server containing no data, the TTLS server issues an EAP-Success.

Note that additional AVPs associated with MS-CHAP-V2 may be sent by the AAA/H; for example, MS-CHAP-Domain. The TTLS server MUST tunnel such authentication-related attributes along with the MS-CHAP2-Success.

10.2.5 PAP

The client tunnels User-Name and User-Password AVPs to the TTLS server.

Normally, in RADIUS, User-Password is padded with nulls to a multiple of 16 octets, then encrypted using a shared secret and other packet information.

An EAP-TTLS client, however, does not RADIUS-encrypt the password since no such RADIUS variables are available; this is not a security weakness since the password will be encrypted via TLS anyway. The client SHOULD, however, null-pad the password to a multiple of 16 octets, to obfuscate its length.

Upon receipt of these AVPs from the client, the TTLS server forwards them to the AAA/H in a RADIUS Access-Request. (Note that in the Access-Request, the TTLS server must encrypt the User-Password attribute using the shared secret between the TTLS server and AAA/H.)

The AAA/H may immediately respond with an Access-Accept or Access-Reject. The TTLS server then completes the negotiation by sending an EAP-Success or EAP-Failure to the access point using the AAA carrier protocol.

The AAA/H may also respond with an Access-Challenge. The TTLS server then tunnels the AVPs from the AAA/H's challenge to the client. Upon receipt of these AVPs, the client tunnels User-Name and User-Password again, with User-Password containing new information in response to the challenge. This process continues until the AAA/H issues an Access-Accept or Access-Reject.

At least one of the AVPs tunneled to the client upon challenge MUST be Reply-Message. Normally this is sent by the AAA/H as part of the challenge. However, if the AAA/H has not sent a Reply-Message, the TTLS server MUST issue one, with null value. This allows the client to determine that a challenge response is required.

Note that if the AAA/H includes a Reply-Message as part of an Access-Accept or Access-Reject, the TTLS server does not tunnel this AVP to the client. Rather, this AVP and all other AVPs sent by the AAA/H as part of Access-Accept or Access-Reject are sent to the access point via the AAA carrier protocol.

10.3 Performing Multiple Authentications

In some cases, it is desirable to perform multiple user authentications. For example, a AAA/H may want first to authenticate the user by password, then by token card.

The AAA/H may perform any number of additional user authentications using EAP, simply by issuing a EAP-Request with a new protocol type once the previous authentication succeeded but prior to issuing an EAP-Success or accepting the user via the AAA carrier protocol.

For example, an AAA/H wishing to perform MD5-Challenge followed by Generic Token Card would first issue an EAP-Request/MD5-Challenge and receive a response. If the response is satisfactory, it would then issue EAP-Request/Generic Token Card and receive a response. If that response were also satisfactory, it would issue EAP-Success.

11. Keying Framework

In compliance with [KEYFRAME], Session-Id, Peer-Id and Server-Id are here defined.

11.1 Session-Id

The Session-Id uniquely identifies an authentication exchange between the client and TTLS server. It is defined as follows:

```
Session-Id = 0x015 || client.random || server.random
```

11.2 Peer-Id

For EAP-TTLSv0, the Peer-Id is null.

11.3 Server-Id

The Server-Id identifies the TTLS server. When the TTLS server presents a certificate as part of the TLS handshake, the Server-Id is determined based on information in the certificate, as specified in [RFC2716bis]. Otherwise, the Server-Id is null.

12. Security Claims

Pursuant to RFC3748, security claims for EAP-TTLSv0 are as follows:

Authentication mechanism:	TLS plus arbitrary additional protected authentication(s)
Ciphersuite negotiation:	Yes
Mutual authentication:	Yes, in recommended implementation
Integrity protection:	Yes
Replay protection:	Yes
Confidentiality:	Yes
Key derivation:	Yes
Key strength:	384 bits or higher
Dictionary attack prot.:	Yes
Fast reconnect:	Yes
Crypt. binding:	No
Session independence:	Yes

Fragmentation:	Yes
Channel binding:	Supported via AVPs, though optional

13. Message Sequences

This section presents EAP-TTLS message sequences for various negotiation scenarios. These examples do not attempt to exhaustively depict all possible scenarios.

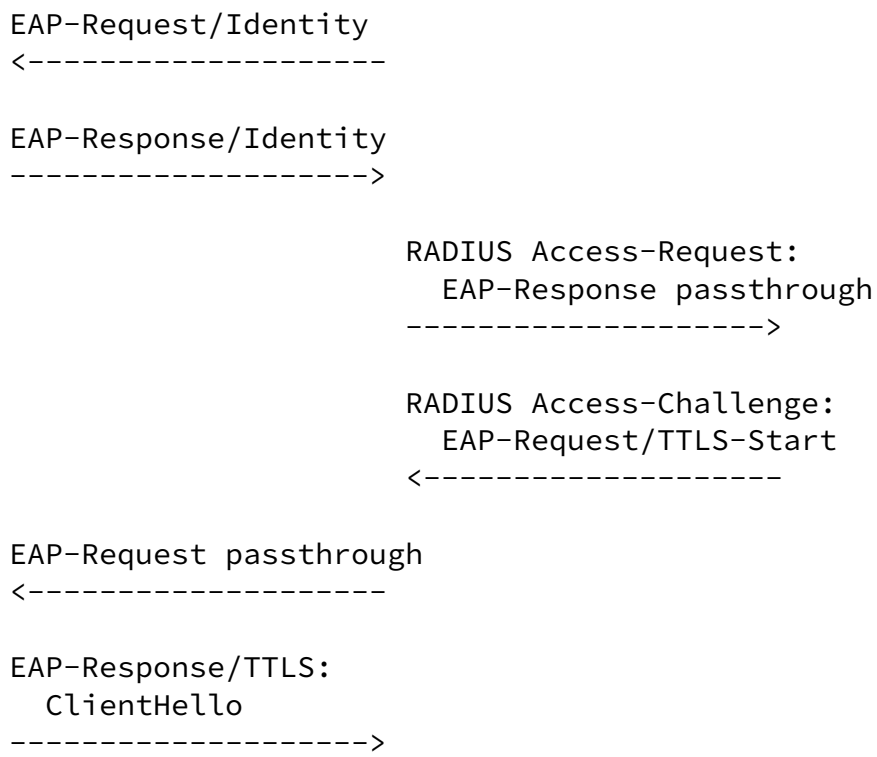
It is assumed that RADIUS is the AAA carrier protocol both between access point and TTLS server, and between TTLS server and AAA/H.

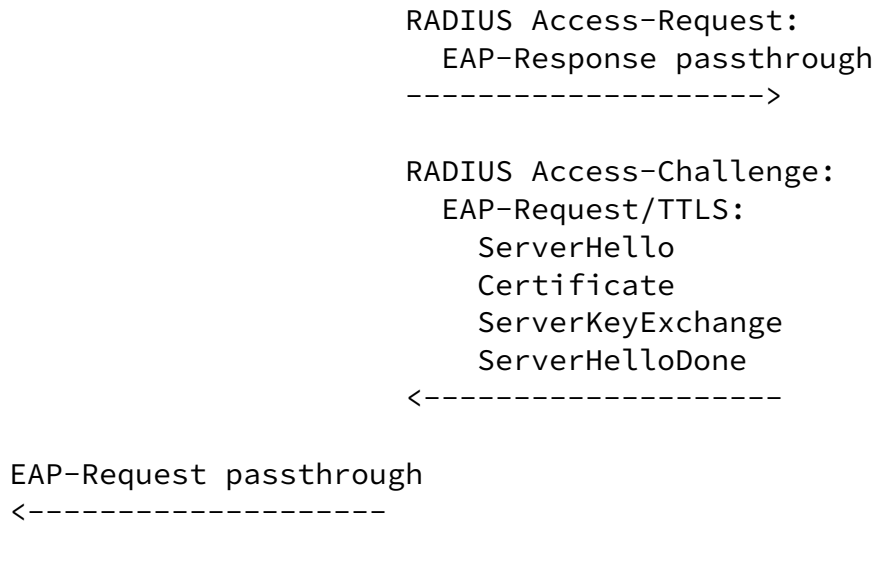
EAP packets that are passed unmodified between client and TTLS server by the access point are indicated as "passthrough". AVPs that are securely tunneled within the TLS record layer are enclosed in curly braces ({}). Items that are optional are suffixed with question mark (?). Items that may appear multiple times are suffixed with plus sign (+).

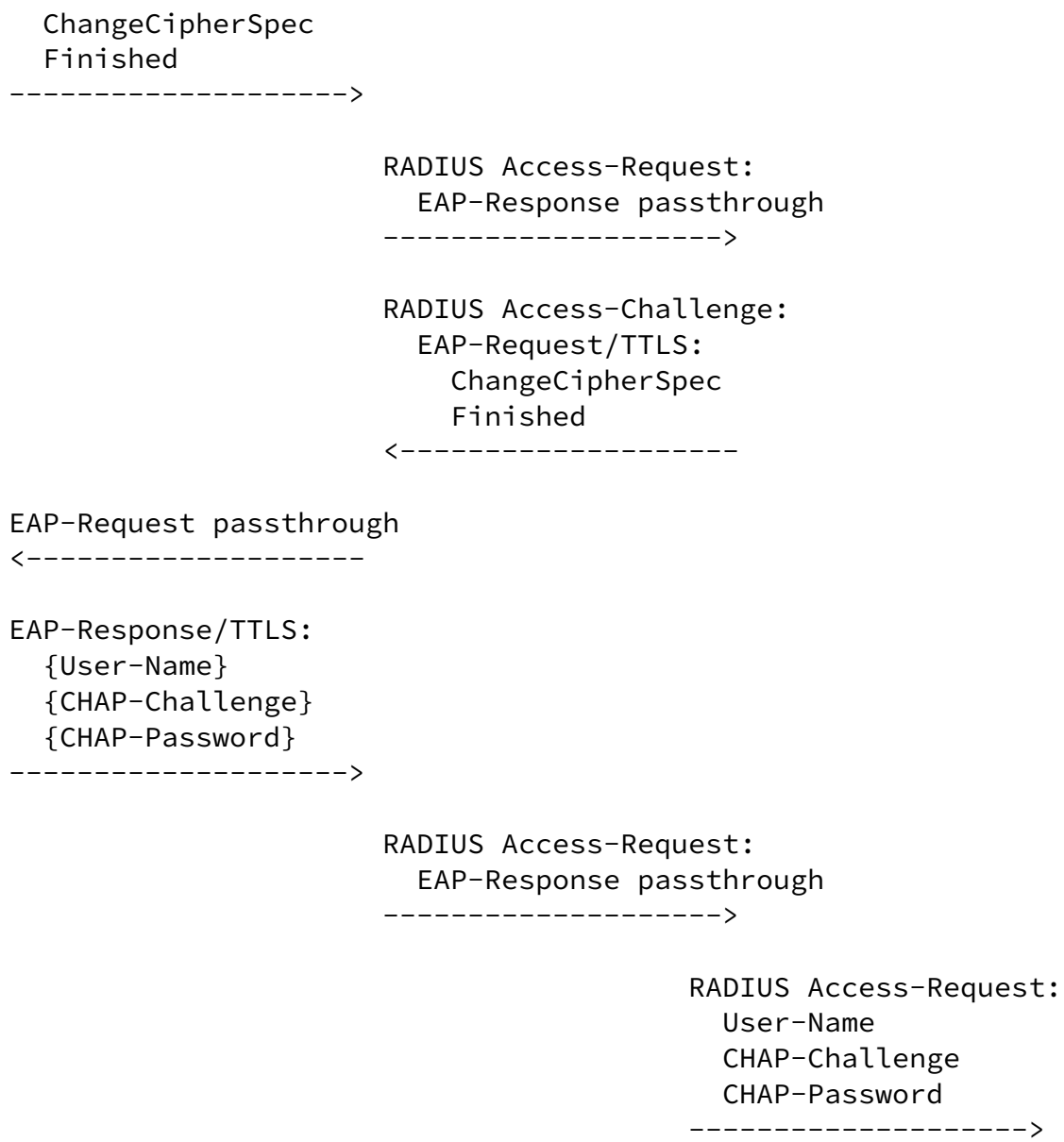
13.1 Successful authentication via tunneled CHAP

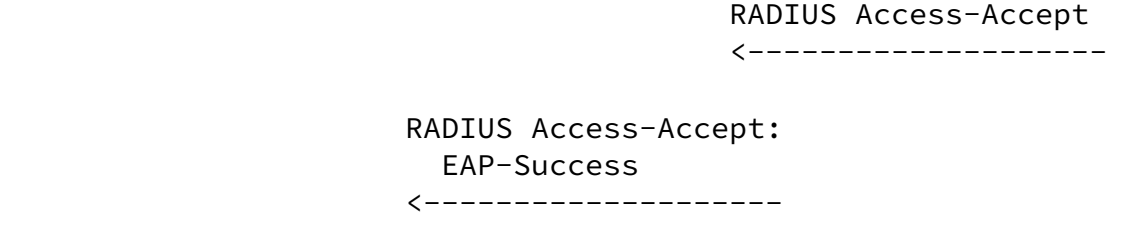
In this example, the client performs one-way TLS authentication of the TTLS server. CHAP is used as a tunneled user authentication mechanism.

client	access point	TTLS server	AAA/H


EAP-Request/Identity
<----------
EAP-Response/Identity
---------->
RADIUS Access-Request:
EAP-Response passthrough
---------->
RADIUS Access-Challenge:
EAP-Request/TTLS-Start
<----------
EAP-Request passthrough
<----------
EAP-Response/TTLS:
ClientHello
---------->


RADIUS Access-Request:
EAP-Response passthrough
---------->
RADIUS Access-Challenge:
EAP-Request/TTLS:
ServerHello
Certificate
ServerKeyExchange
ServerHelloDone
<----------
EAP-Request passthrough
<----------
EAP-Response/TTLS:
ClientKeyExchange


ChangeCipherSpec
Finished
RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Challenge:
EAP-Request/TTLS:
ChangeCipherSpec
Finished
EAP-Request passthrough
EAP-Response/TTLS:
{User-Name}
{CHAP-Challenge}
{CHAP-Password}
RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Request:
User-Name
CHAP-Challenge
CHAP-Password

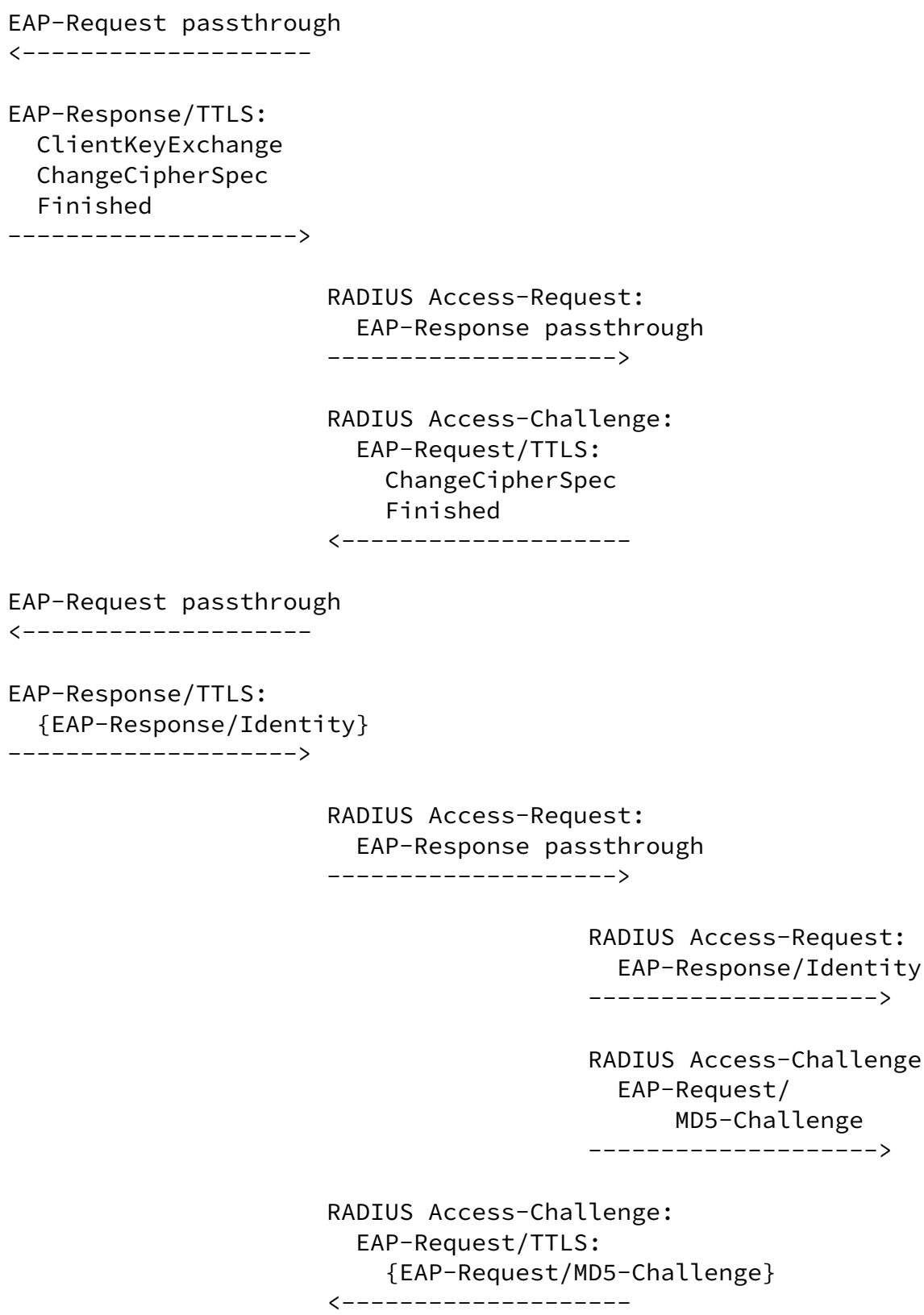

RADIUS Access-Accept
RADIUS Access-Accept:
EAP-Success

```
  EAP-Success passthrough
  <--------------------
```

13.2 Successful authentication via tunneled EAP/MD5-Challenge

In this example, the client performs one-way TLS authentication of the TTLS server and EAP/MD5-Challenge is used as a tunneled user authentication mechanism.

```
client          access point          TTLS server             AAA/H
------          ------------          -----------             -----

  EAP-Request/Identity
  <--------------------

  EAP-Response/Identity
  -------------------->

                        RADIUS Access-Request:
                          EAP-Response passthrough
                        -------------------->

                        RADIUS Access-Challenge:
                          EAP-Request/TTLS-Start
                        <--------------------

  EAP-Request passthrough
  <--------------------

  EAP-Response/TTLS:
    ClientHello
  -------------------->

                        RADIUS Access-Request:
                          EAP-Response passthrough
                        -------------------->

                        RADIUS Access-Challenge:
                          EAP-Request/TTLS:
                            ServerHello
                            Certificate
                            ServerKeyExchange
                            ServerHelloDone
                        <--------------------
```


EAP-Request passthrough
<--------------------
EAP-Response/TTLS:
ClientKeyExchange
ChangeCipherSpec
Finished
-------------------->
RADIUS Access-Request:
EAP-Response passthrough
-------------------->
RADIUS Access-Challenge:
EAP-Request/TTLS:
ChangeCipherSpec
Finished
<--------------------
EAP-Request passthrough
<--------------------
EAP-Response/TTLS:
{EAP-Response/Identity}
-------------------->
RADIUS Access-Request:
EAP-Response passthrough
-------------------->
RADIUS Access-Request:
EAP-Response/Identity
-------------------->
RADIUS Access-Challenge
EAP-Request/
MD5-Challenge
-------------------->
RADIUS Access-Challenge:
EAP-Request/TTLS:
{EAP-Request/MD5-Challenge}
<--------------------
EAP-Request passthrough
<--------------------

```
      EAP-Response/TTLS:
        {EAP-Response/MD5-Challenge}
      ---------------------->
```

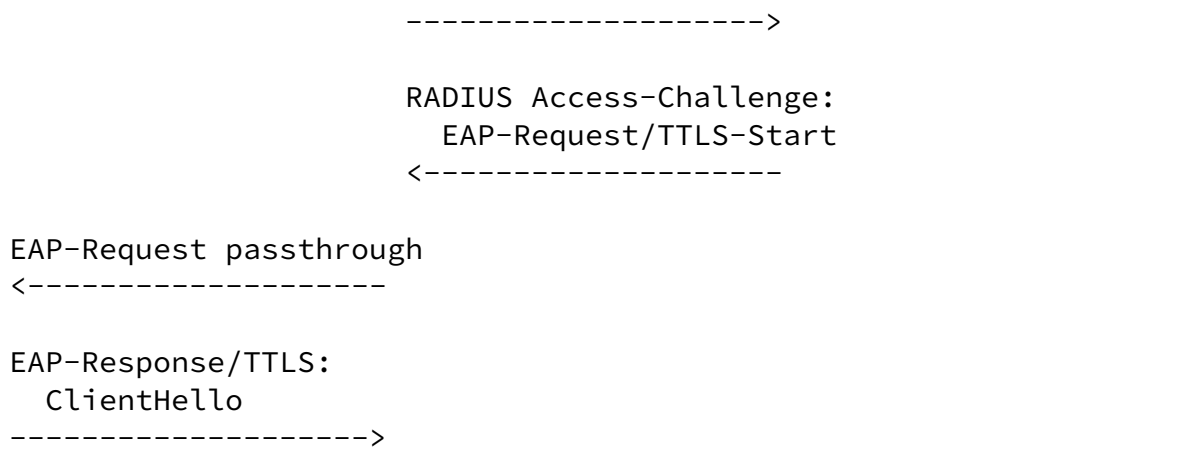


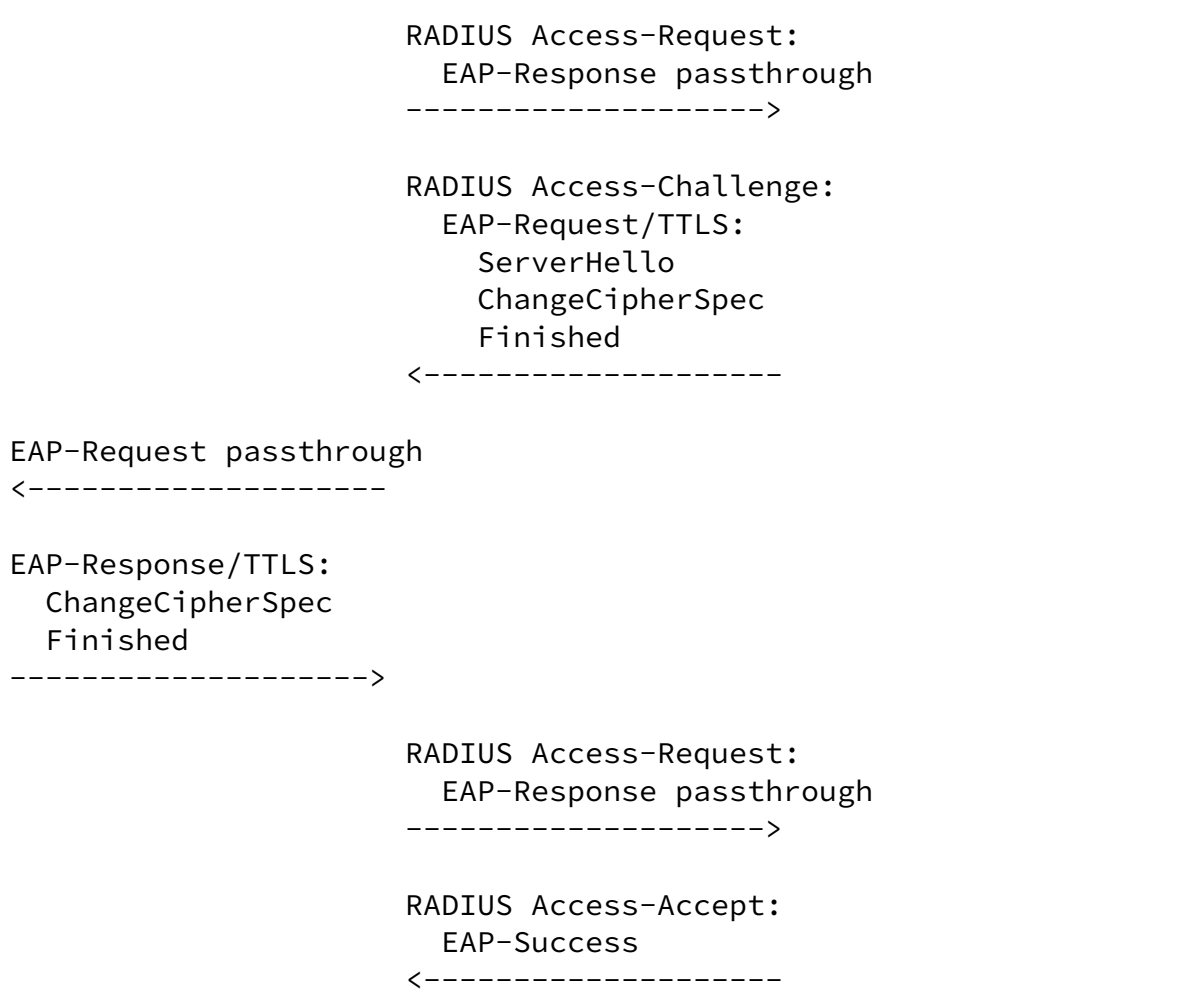
```
                              RADIUS Access-Request:
                                EAP-Response passthrough
                              ---------------------->

                                                  RADIUS Access-Challenge
                                                    EAP-Response/
                                                        MD5-Challenge
                                                  ---------------------->

                                                  RADIUS Access-Accept
                                                  <----------------------

                              RADIUS Access-Accept:
                                EAP-Success
                              <----------------------

      EAP-Success passthrough
      <--------------------
```

13.3 Successful session resumption

In this example, the client and server resume a previous TLS session. The ID of the session to be resumed is sent as part of the ClientHello, and the server agrees to resume this session by sending the same session ID as part of ServerHello.

```
client          access point          TTLS server             AAA/H
------          ------------          -----------             -----

  EAP-Request/Identity
  <--------------------

  EAP-Response/Identity
  ---------------------->

                              RADIUS Access-Request:
                                EAP-Response passthrough
```


RADIUS Access-Challenge:
EAP-Request/TTLS-Start
EAP-Request passthrough
EAP-Response/TTLS:
ClientHello


RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Challenge:
EAP-Request/TTLS:
ServerHello
ChangeCipherSpec
Finished
EAP-Request passthrough
EAP-Response/TTLS:
ChangeCipherSpec
Finished
RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Accept:
EAP-Success

```
   EAP-Success passthrough
   <--------------------
```

14. Security Considerations

14.1 Man-in-the-Middle Attack

[MITM] describes a vulnerability that is characteristic of tunneled authentication protocols, in which an attacker authenticates as a client via a tunneled protocol by posing as an authenticator to a legitimate client using a non-tunneled protocol. When the same proof of credentials can be used in both authentications, the attacker merely shuttles the credential proof between them. EAP-TTLSv0 is vulnerable to such an attack. Care should be taken to avoid using authentication protocols and associated credentials both as inner TTLSv0 methods and as untunneled methods.

A future version of EAP-TTLS should be defined to perform a cryptographic binding of keying material generated by inner authentication methods and the keying material generated by the TLS handshake. This avoids the Man-in-the-Middle problem when used with key-generating inner methods.

14.2 Client Anonymity

Unlike other EAP methods, EAP-TTLS does not communicate a username in the clear in the initial EAP-Response/Identity. This feature is designed to support anonymity and location privacy from attackers eavesdropping the network path between the client and the TTLS server. However implementers should be aware that other factors - both within EAP-TTLS and elsewhere - may compromise a user's identity. For example, if a user authenticates with a certificate during phase 1 of EAP-TTLS, the subject name in the certificate may reveal the user's identity. Outside of EAP-TTLS, the client's fixed MAC address, or in the case of wireless connections, the client's radio signature, may also reveal information. Additionally, implementers should be aware that a user's identity is not hidden from the EAP-TTLS server and may be included in the clear in AAA messages between the access point, the EAP-TTLS server, and the AAA/H server.

14.3 Server Trust

Trust of the server by the client is established via a server certificate conveyed during the TLS handshake. The client should have a means of determining which server identities may be trusted, and should refuse to authenticate with servers it does not trust. The consequence of pursuing authentication with a hostile server is exposure of the inner authentication to attack; e.g. offline dictionary attack against the client password.

14.4 Certificate compromise

Certificates should be checked for revocation to reduce exposure to imposture using compromised certificates.

Checking a server certificate against the most recent revocation list during authentication is not always possible for a client, as it may not have network access until completion of the authentication. This problem can be alleviated through the use of OCSP [RFC2560] during the TLS handshake, as described in [RFC3546].

14.5 Forward secrecy.

With forward secrecy, revelation of a secret does not compromise session keys previously negotiated based on that secret. Thus, when the TLS key exchange algorithm provides forward secrecy, if a TTLS server certificate's private key is eventually stolen or cracked, tunneled user password information will remain secure as long as that certificate is no longer in use. Diffie-Hellman key exchange is an example of an algorithm that provides forward secrecy. A forward secrecy algorithm should be considered if attacks against recorded authentication or data sessions are considered to pose a significant threat.

15. References

[RFC2716] Aboba, B., and D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[RFC2716bis] Simon, D., and B. Aboba, "The EAP TLS Authentication Protocol", Internet Draft (work in progress), draft-

simon-emu-rfc2716bis-08.txt, February 2007.

[KEYFRAME] Aboba, B., Simon, D. and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", Internet Draft (work in progress), draft-ietf-eap-keying-18.txt, February 2007.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "PPP Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4346] Dierks, T., and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, November 1998.

[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.

[RFC1661] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC2865] Rigney, C., Rubens, A., Simpson, W., and S. Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC4282] Aboba, B., Beadles, M., Arkko, J. and P. Eronen, "The Network Access Identifier", RFC 4282, January 1999.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, July 2001.

[RFC1700] Reynolds, J., and J. Postel, "Assigned Numbers", RFC 1700, October 1994.

[RFC2433] Zorn, G., and S. Cobb, "Microsoft PPP CHAP Extensions", RFC 2433, October 1998.

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000.

[RFC2548] Zorn, G., "Microsoft Vendor-specific RADIUS Attributes", RFC 2548, March 1999.

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "Internet X.509 Public Key Infrastructure: Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[MITM] Asokan, N., Niemi, V., and Nyberg, K., "Man-in-the-Middle in Tunneled Authentication", http://www.saunalahti.fi/~asokan/research/mitm.html, Nokia Research Center, Finland, October 24 2002.

16. Authors' Addresses

Questions about this memo can be directed to:

Paul Funk
Juniper Networks
222 Third Street
Cambridge, MA 02142
USA

Phone: +1 617 497-6339
E-mail: pfunk@juniper.net

Simon Blake-Wilson
Basic Commerce & Industries, Inc.
304 Harper Drive, Suite 203
Moorestown, NJ 08057

Phone: +1 856 778-1660
E-mail: sblakewilson@bcisse.com

17. Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license

under such rights might or might not be available; nor does it

represent that it has 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.

18. Disclaimer of Validity

19. Copyright Statement

20. Acknowledgment

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