Network Working Group
Request for Comments: 5281
Category: Informational

P. Funk
Unaffiliated
S. Blake-Wilson
SafeNet

August 2008

Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

EAP-TTLS is an EAP (Extensible Authentication Protocol) method that encapsulates a TLS (Transport Layer Security) session, consisting of a handshake phase and a data phase. During the handshake phase, the server is authenticated to the client (or client and server are mutually authenticated) using standard TLS procedures, and keying material is generated in order to create a cryptographically secure tunnel for information exchange in the subsequent data phase. During the data phase, the client is authenticated to the server (or client and server are mutually authenticated) using an arbitrary authentication mechanism encapsulated within the secure tunnel. The encapsulated authentication mechanism 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 attacks. The data phase may also be used for additional, arbitrary data exchange.

Table of Contents

1. Introduction 4
2. Motivation 5
3. Requirements Language 7
4. Terminology 7
5. Architectural Model 9
5.1. Carrier Protocols 10
5.2. Security Relationships 10
5.3. Messaging 11
5.4. Resulting Security 12
6. Protocol Layering Model 12
7. EAP-TTLS Overview 13
7.1. Phase 1: Handshake 14
7.2. Phase 2: Tunnel 14
7.3. EAP Identity Information 15
7.4. Piggybacking 15
7.5. Session Resumption 16
7.6. Determining Whether to Enter Phase 2 17
7.7. TLS Version 18
7.8. Use of TLS PRF 18
8. Generating Keying Material 19
9. EAP-TTLS Protocol 20
9.1. Packet Format 20
9.2. EAP-TTLS Start Packet 21
9.2.1. Version Negotiation 21
9.2.2. Fragmentation 22
9.2.3. Acknowledgement Packets 22
10. Encapsulation of AVPs within the TLS Record Layer 23
10.1. AVP Format 23
10.2. AVP Sequences 25
10.3. Guidelines for Maximum Compatibility with AAA Servers 25
11. Tunneled Authentication 26
11.1. Implicit Challenge 26
11.2. Tunneled Authentication Protocols 27
11.2.1. EAP 27
11.2.2. CHAP 29
11.2.3. MS-CHAP 30
11.2.4. MS-CHAP-V2 30
11.2.5. PAP 32
11.3. Performing Multiple Authentications 33
11.4. Mandatory Tunneled Authentication Support 34
11.5. Additional Suggested Tunneled Authentication Support 34
12. Keying Framework 35
12.1. Session-Id 35
12.2. Peer-Id 35
12.3. Server-Id 35
13. AVP Summary 35

14. Security Considerations 36
14.1. Security Claims 36
14.1.1. Authentication Mechanism 36
14.1.2. Ciphersuite Negotiation 37
14.1.3. Mutual Authentication 37
14.1.4. Integrity Protection 37
14.1.5. Replay Protection 37
14.1.6. Confidentiality 37
14.1.7. Key Derivation 37
14.1.8. Key Strength 37
14.1.9. Dictionary Attack Protection 38
14.1.10. Fast Reconnect 38
14.1.11. Cryptographic Binding 38
14.1.12. Session Independence 38
14.1.13. Fragmentation 38
14.1.14. Channel Binding 38
14.2. Client Anonymity 38
14.3. Server Trust 39
14.4. Certificate Validation 39
14.5. Certificate Compromise 40
14.6. Forward Secrecy 40
14.7. Negotiating-Down Attacks 40
15. Message Sequences 41
15.1. Successful Authentication via Tunneled CHAP 41
15.2. Successful Authentication via Tunneled EAP/MD5-Challenge 43
15.3. Successful Session Resumption 46
16. IANA Considerations 47
17. Acknowledgements 48
18. References 48
18.1. Normative References 48
18.2. Informative References 49

1. Introduction

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

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) [RFC5216].

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 [RFC2865], Diameter [RFC3588], or other AAA servers. These include non-EAP protocols such as PAP [RFC1661], CHAP [RFC1661], MS-CHAP [RFC2433], or MS-CHAP-V2 [RFC2759], as well as EAP protocols such as MD5-Challenge [RFC3748].

EAP-TTLS is an EAP method that provides functionality beyond what is available in EAP-TLS. EAP-TTLS has been widely deployed and this specification documents what existing implementations do. It has some limitations and vulnerabilities, however. These are addressed in EAP-TTLS extensions and ongoing work in the creation of standardized tunneled EAP methods at the IETF. Users of EAP-TTLS are strongly encouraged to consider these in their deployments.

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 authentication 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. 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 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.

The main limitation of EAP-TTLS is that its base version lacks support for cryptographic binding between the outer and inner authentication. Please refer to Section 14.1.11 for details and the conditions where this vulnerability exists. It should be noted that an extension for EAP-TTLS [TTLS-EXT] fixed this vulnerability. Users of EAP-TTLS are strongly encouraged to adopt this extension.

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 [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 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 access domains with which the user has no relationship and which will have limited capabilities for routing authentication requests.

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4. 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. Since the access point terminates the server side of the EAP conversation, for the

purposes of this document it is therefore equivalent to the "authenticator", as used in the EAP specification [RFC3748]. Since the access point acts as a client to a AAA server, for the purposes of this document it is therefore also equivalent to the "Network Access Server (NAS)", as used in AAA specifications such as [RFC2865].

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. Since it terminates the client side of the EAP conversation, for the purposes of this document, it is therefore equivalent to the "peer", as used in the EAP specification [RFC3748].

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

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.

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

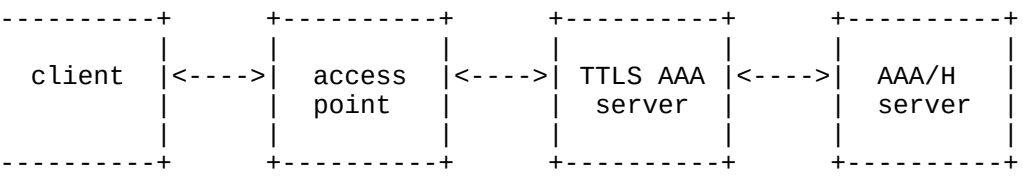
A AAA server which implements EAP-TTLS. This server may also be capable of performing user authentication, or it may proxy the user authentication to a AAA/H.

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.

5. Architectural Model

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



<---- secure password authentication tunnel --->

<---- secure data tunnel ---->

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.

5.1. Carrier Protocols

The entities shown above communicate with each other using carrier protocols capable of encapsulating EAP. The client and access point communicate typically using a link layer carrier protocol such as PPP or EAPOL (EAP over LAN). 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 carrier protocol between client and access point. 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 the EAP-TTLS negotiation. This keying material must be communicated to the access point by the TTLS server using the AAA carrier protocol.

5.2. 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.

5.3. 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 need not include the user's actual identity in this EAP-Response/Identity packet other than for routing purposes (e.g., realm information; see Section 7.3 and [RFC3748], Section 5.1); the user's actual identity need 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 issues 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 (Master Session Key / Extended Master Session Key (MSK/EMSK); see Section 8) is 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).

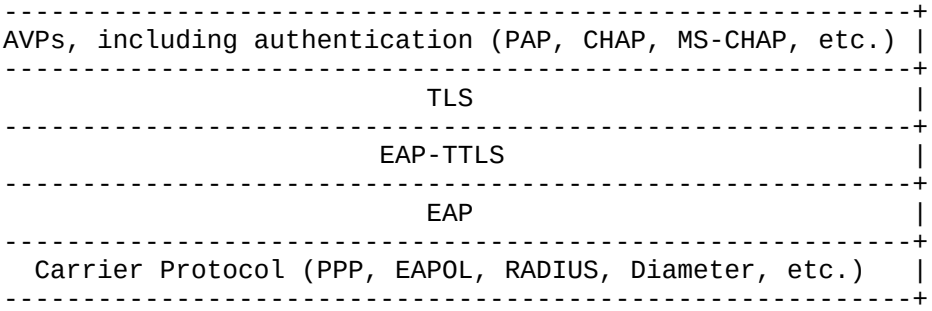
The client and access point now share keying material that they can use to encrypt data traffic between them.

5.4. Resulting Security

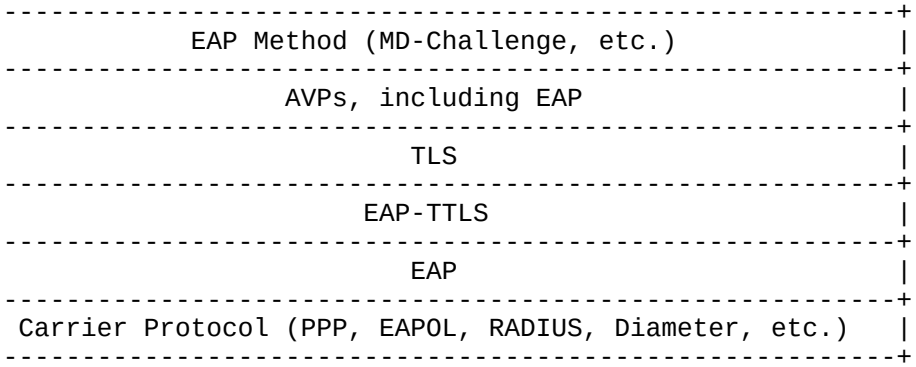
As the diagram above indicates, EAP-TTLS allows user identity and password information to be securely transmitted between client and TTLS server, and generates keying material to allow network data subsequent to authentication to be securely transmitted between client and access point.

6. 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 attribute-value pairs (AVPs) which may carry user authentication or other 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.

7. 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.

7.1. 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.

The TTLS server initiates the EAP-TTLS method with an EAP-TTLS/Start packet, which is an EAP-Request with Type = EAP-TTLS and the S (Start) bit set. This indicates to the client that it should begin the 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 [RFC5216]. 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.

7.2. 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 AAA/H either accepts or rejects the client, resulting in the TTLS server completing the EAP-TTLS negotiation and indicating success or failure to the encapsulating EAP protocol (which normally results in a final EAP-Success or EAP-Failure being sent to the client).

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 final EAP-Success or other success indication.

7.3. EAP Identity Information

The identity of the user is provided during phase 2, where it is protected by the TLS tunnel. However, prior to beginning the EAP-TTLS authentication, the client will typically issue an EAP-Response/Identity packet as part of the EAP protocol, containing a username in clear text. To preserve user anonymity against eavesdropping, this packet specifically SHOULD NOT include the actual name of the user; instead, it SHOULD use a blank or placeholder such as "anonymous". However, this privacy constraint is not intended to apply to any information within the EAP-Response/Identity that is required for routing; thus, the EAP-Response/Identity packet MAY include the name of the realm of a trusted provider to which EAP-TTLS packets should be forwarded; for example, "anonymous@myisp.com".

Note that at the time the initial EAP-Response/Identity packet is sent the EAP method is yet to be negotiated. If, in addition to EAP-TTLS, the client is willing to negotiate use of EAP methods that do not support user anonymity, then the client MAY include the name of the user in the EAP-Response/Identity to meet the requirements of the other candidate EAP methods.

7.4. 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 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.

7.5. Session Resumption

When a client and TTLS server that have previously negotiated an 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 of a session resumption, 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 and indicate success or failure to the encapsulating EAP protocol.

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.]

7.6. Determining Whether to Enter Phase 2

Entering phase 2 is optional, and may be initiated by either client or TTLS server. If no further authentication or other information exchange is required upon completion of phase 1, it is possible to successfully complete the EAP-TTLS negotiation without ever entering phase 2 or tunneling any AVPs.

Scenarios in which phase 2 is never entered include:

- Successful session resumption, with no additional information exchange required,

- Authentication of the client via client certificate during phase 1, with no additional authentication or information exchange required.

The client always has the first opportunity to initiate phase 2 upon completion of phase 1. If the client has no AVPs to send, it either sends an Acknowledgement (see Section 9.2.3) if the TTLS server sends the final phase 1 message, or simply does not piggyback a phase 2 message when it issues the final phase 1 message (as will occur during session resumption).

If the client does not initiate phase 2, the TTLS server, at its option, may either complete the EAP-TTLS negotiation without entering phase 2 or initiate phase 2 by tunneling AVPs to the client.

For example, suppose a successful session resumption occurs in phase 1. The following sequences are possible:

- Neither the client nor TTLS server has additional information to exchange. The client completes phase 1 without piggybacking phase 2 AVPs, and the TTLS server indicates success to the encapsulating EAP protocol without entering phase 2.

- The client has no additional information to exchange, but the TTLS server does. The client completes phase 1 without piggybacking phase 2 AVPs, but the TTLS server extends the EAP-TTLS negotiation into phase 2 by tunneling AVPs in its next EAP-TTLS message.

- The client has additional information to exchange, and piggybacks phase 2 AVPs with its final phase 1 message, thus extending the negotiation into phase 2.

7.7. TLS Version

TLS version 1.0 [RFC2246], 1.1 [RFC4346], or any subsequent version MAY be used within EAP-TTLS. TLS provides for its own version negotiation mechanism.

For maximum interoperability, EAP-TTLS implementations SHOULD support TLS version 1.0.

7.8. Use of TLS PRF

EAP-TTLSv0 utilizes a pseudo-random function (PRF) to generate keying material (Section 8) and to generate implicit challenge material for certain authentication methods (Section 11.1). The PRF used in these computations is the TLS PRF used in the TLS handshake negotiation that initiates the EAP-TTLS exchange.

TLS versions 1.0 [RFC2246] and 1.1 [RFC4346] define the same PRF function, and any EAP-TTLSv0 implementation based on these versions of TLS must use the PRF defined therein. 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 is specified for the underlying TLS version or has been negotiated during the TLS handshake negotiation, then that alternative PRF function must be used in EAP-TTLSv0 computations instead of the TLS 1.0/1.1 PRF.

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

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 TLS 1.0/1.1 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.

8. Generating Keying Material

Upon successful conclusion of an EAP-TTLS negotiation, 128 octets of keying material are generated and exported for use in securing the data connection between client and access point. The first 64 octets of the keying material constitute the MSK, the second 64 octets constitute 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-128(SecurityParameters.master_secret, "ttls
          keying material", SecurityParameters.client_random +
          SecurityParameters.server_random)

MSK = Keying Material [0..63]

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

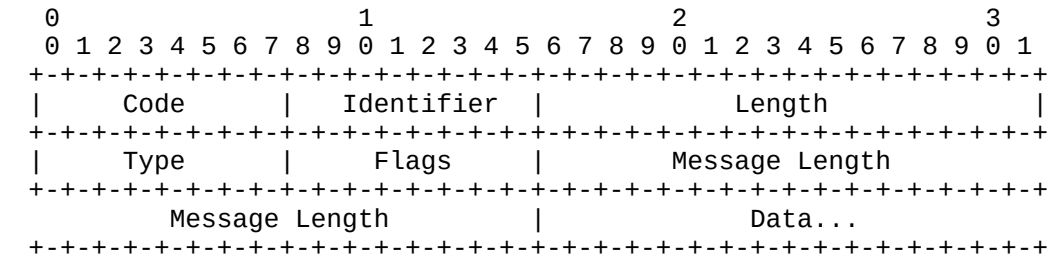
Note that the order of client_random and server_random for EAP-TTLS is reversed from that of the TLS protocol [RFC4346]. This ordering follows the key derivation method of EAP-TLS [RFC5216]. Altering the order of randoms avoids namespace collisions between constant strings defined for EAP-TTLS and those 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 [RFC2548] may be used to distribute the first 32 octets and second 32 octets of the MSK, respectively.

9. EAP-TTLS Protocol

9.1. Packet Format

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



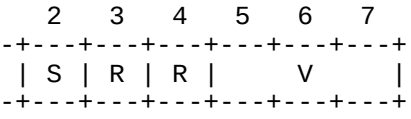
Code
: 1 for request, 2 for response.

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 field 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.

9.2. 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 four-octet boundary relative to the first AVP in the sequence. If an AVP is not a multiple of four octets, it must be padded with zeros to the next four-octet boundary.

9.2.1. 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 MUST set the V field to that version number in all subsequent EAP messages.

9.2.2. Fragmentation

Each EAP-TTLS message contains a single leg of a half-duplex conversation. The EAP carrier protocol (e.g., PPP, EAPOL, RADIUS) may impose constraints on the length 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 the M bit set, the receiver MUST transmit an Acknowledgement packet. The receiver is responsible for reassembly of fragmented packets.

9.2.3. 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:

- A Fragment Acknowledgement is sent in response to an EAP packet with the 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 proceed with the EAP protocol upon completion of EAP-TTLS (typically by sending or causing to be sent a final EAP-Success or EAP-Failure to the client).

10. Encapsulation of AVPs within the TLS Record Layer

Subsequent to the TLS handshake, information may be 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 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.

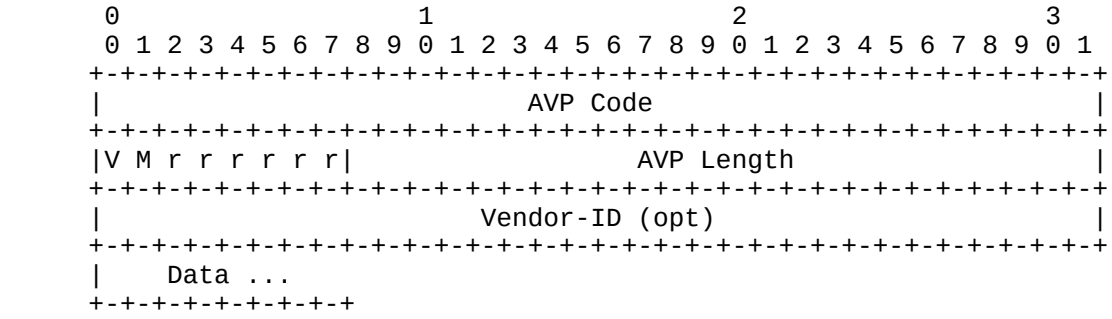
Thus, Diameter is not considered normative except as specified in this document. Specifically, the representation of the Data field of an AVP in EAP-TTLS is identical to that of Diameter.

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.

It is expected that the AVP Codes used in EAP-TTLS will carry roughly the same meaning in EAP-TTLS as they do in Diameter and, by extension, RADIUS. However, although EAP-TTLS uses the same AVP Codes and syntax as Diameter, the semantics may differ, and most Diameter AVPs do not have any well-defined semantics in EAP-TTLS. A separate "EAP-TTLS AVP Usage" registry lists the AVPs that can be used within EAP-TTLS and their semantics in this context (see Section 16 for details). A TTLS server copying AVPs between an EAP-TTLS exchange and a Diameter or RADIUS exchange with a backend MUST NOT make assumptions about AVPs whose usage in either EAP-TTLS or the backend protocol it does not understand. Therefore, a TTLS server MUST NOT copy an AVP between an EAP-TTLS exchange and a Diameter or RADIUS exchange unless the semantics of the AVP are understood and defined in both contexts.

10.1. AVP Format

The format of an AVP is shown below. All items are in network, or big-endian, order; that is, they have the 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 [RFC2865]. AVP numbers 256 and above are defined in Diameter [RFC3588].

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 by the sender and MUST be ignored by the receiver.

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" [RFC3232] 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.

Note that the M bit provides a means for extending the functionality of EAP-TTLS while preserving backward compatibility when desired. By setting the M bit of the appropriate AVP(s) to 0 or 1, the party initiating the function indicates that support of the function by the other party is either optional or required.

10.2. 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 four-octet boundary relative to the first AVP in the sequence. If an AVP is not a multiple of four octets, it must be padded with zeros to the next four-octet boundary.

Note that the AVP Length does not include the padding.

10.3. Guidelines for Maximum Compatibility with AAA Servers

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

- Non-vendor-specific AVPs intended for use with AAA servers 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 intended for use with AAA servers 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.

11. Tunneled Authentication

EAP-TTLS permits user authentication information to be tunneled within the TLS record layer between client and TTLS server, ensuring 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.

11.1. 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 the material requires special handling.

In CHAP, MS-CHAP, and MS-CHAP-V2, for example, the access point issues a challenge to the client, the client then hashes the challenge with the password and forwards the response to the access point. The access point 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-nn(SecurityParameters.master_secret,
                        "ttls challenge",
                        SecurityParameters.client_random +
                        SecurityParameters.server_random);
```

The number of octets to be generated (nn) depends on the authentication method, and is indicated below for each authentication method requiring implicit challenge generation.

11.2. 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.

The client determines which authentication protocol will be used via the initial AVPs it sends to the server, as described in the following sections.

Note that certain of the authentication protocols described below utilize vendor-specific AVPs originally defined for RADIUS. RADIUS and Diameter differ in the encoding of vendor-specific AVPs: RADIUS uses the vendor-specific attribute (code 26), while Diameter uses setting of the V bit to indicate the presence of Vendor-ID. The RADIUS form of the vendor-specific attribute is always convertible to a Diameter AVP with V bit set. All vendor-specific AVPs described below MUST be encoded using the preferred Diameter V bit mechanism; that is, the AVP Code of 26 MUST NOT be used to encode vendor-specific AVPs within EAP-TTLS.

11.2.1. 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.

Note that because Diameter AVPs are not limited to 253 octets of data, as are RADIUS attributes, the RADIUS mechanism of concatenating multiple EAP-Message attributes to represent a longer-than-253-octet EAP packet is not appropriate in EAP-TTLS. Thus, a tunneled EAP packet within a single EAP-TTLS message MUST be contained in a single EAP-Message AVP.

The client initiates EAP by tunneling EAP-Response/Identity to the TTLS server. Depending on the requirements specified for the inner method, the client MAY now place the actual username in this packet; the privacy of the user's identity is now guaranteed by the TLS encryption. This username is typically a Network Access Identifier (NAI) [RFC4282]; that is, it is typically 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, untunneled EAP-Response/Identity packet prior to starting EAP-TTLS, indicates the realm of the TTLS server. The second, occurring as part of the EAP exchange within the EAP-TTLS tunnel, 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. (Refer to Section 7.3 for additional information distinguishing the untunneled and tunneled versions of the EAP-Response/Identity packets.)

Note that TTLS processing of the initial identity exchange is different from plain EAP. The state machine of TTLS is different. However, it is expected that the server side is capable of dealing with client initiation, because even normal EAP protocol runs are client-initiated over AAA. On the client side, there are various implementation techniques to deal with the differences. Even a TTLS-unaware EAP protocol run could be used, if TTLS makes it appear as if an EAP-Request/Identity message was actually received. This is similar to what authenticators do when operating between a client and a AAA server.

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, One-Time Password (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.

Note that EAP-TTLS does not impose special rules on EAP Notification packets; such packets MAY be used within a tunneled EAP exchange according to the rules specified in [RFC3748].

EAP-TTLS provides a reliable transport for the tunneled EAP exchange. However, [RFC3748] assumes an unreliable transport for EAP messages (see Section 3.1), and provides for silent discard of any EAP packet that violates the protocol or fails a method-specific integrity check, on the assumption that such a packet is likely a counterfeit sent by an attacker. But since the tunnel provides a reliable transport for the inner EAP authentication, errors that would result in silent discard according to [RFC3748] presumably represent implementation errors when they occur within the tunnel, and SHOULD be treated as such in preference to being silently discarded. Indeed, silently discarding an EAP message within the tunnel effectively puts a halt to the progress of the exchange, and will result in long timeouts in cases that ought to result in immediate failures.

11.2.2. 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 initiates CHAP by tunneling 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.

11.2.3. 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 initiates MS-CHAP by tunneling 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.

11.2.4. 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 initiates MS-CHAP-V2 by tunneling 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 (that is, with a zero-length Data field). Upon receipt of the empty EAP-TTLS packet from the client, the TTLS server considers the MS-CHAP-V2 authentication to have succeeded.

If the authentication fails, the AAA/H will respond with an Access-Challenge containing the MS-CHAP-Error attribute. This attribute contains a new Ident and a string with additional information such as the error reason and whether a retry is allowed. The TTLS server tunnels this AVP to the client. 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-CHAP-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-CHAP-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, and the client authenticates the AAA/H based on this AVP. Then, the client either abandons the negotiation on failure or sends an EAP-TTLS packet to the TTLS server containing no data (that is, with a zero-length Data field), causing the TTLS server to consider the MS-CHAP-V2 authentication to have succeeded.

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.

11.2.5. PAP

The client initiates PAP by tunneling 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.

11.3. 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 EAP type once the previous authentication completes. Note that each new EAP method is subject to negotiation; that is, the client may respond to the EAP request for a new EAP type with an EAP-Nak, as described in [RFC3748].

For example, a AAA/H wishing to perform an 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 an EAP-Request/Generic Token Card and receive a response. If that response were also satisfactory, it would accept the user.

The entire inner EAP exchange comprising multiple authentications is considered a single EAP sequence, in that each subsequent request MUST contain distinct a EAP Identifier from the previous request, even as one authentication completes and another begins.

The peer identity indicated in the original EAP-Response/Identity that initiated the EAP sequence is intended to apply to each of the sequential authentications. In the absence of an application profile standard specifying otherwise, additional EAP-Identity exchanges SHOULD NOT occur.

The conditions for overall success or failure when multiple authentications are used are a matter of policy on client and server; thus, either party may require that all inner authentications succeed, or that at least one inner authentication succeed, as a condition for success of the overall authentication.

Each EAP method is intended to run to completion. Should the TTLS server abandon a method and start a new one, client behavior is not defined in this document and is a matter of client policy.

Note that it is not always feasible to use the same EAP method twice in a row, since it may not be possible to determine when the first authentication completes and the new authentication begins if the EAP type does not change. Certain EAP methods, such as EAP-TLS, use a Start bit to distinguish the first request, thus allowing each new authentication using that type to be distinguished from the previous. Other methods, such as EAP-MS-CHAP-V2, terminate in a well-defined manner, allowing a second authentication of the same type to commence unambiguously. While use of the same EAP method for multiple authentications is relatively unlikely, implementers should be aware of the issues and avoid cases that would result in ambiguity.

Multiple authentications using non-EAP methods or a mixture of EAP and non-EAP methods is not defined in this document, nor is it known whether such an approach has been implemented.

11.4. Mandatory Tunneled Authentication Support

To ensure interoperability, in the absence of an application profile standard specifying otherwise, an implementation compliant with this specification MUST implement EAP as a tunneled authentication method and MUST implement MD5-Challenge as an EAP type. However, such an implementation MAY allow the use of EAP, any EAP type, or any other tunneled authentication method to be enabled or disabled by administrative action on either client or TTLS server.

In addition, in the absence of an application profile standard specifying otherwise, an implementation compliant with this specification MUST allow an administrator to configure the use of tunneled authentication without the M (Mandatory) bit set on any AVP.

11.5. Additional Suggested Tunneled Authentication Support

The following information is provided as non-normative guidance based on the experience of the authors and reviewers of this specification with existing implementations of EAP-TTLSv0.

The following authentication methods are commonly used, and servers wishing for broad interoperability across multiple media should consider implementing them:

- PAP (both for password and token authentication)
- MS-CHAP-V2
- EAP-MS-CHAP-V2
- EAP-GTC

12. Keying Framework

In compliance with [RFC5247], Session-Id, Peer-Id, and Server-Id are here defined.

12.1. Session-Id

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

Session-Id = 0x15 || client.random || server.random

12.2. Peer-Id

The Peer-Id represents the identity to be used for access control and accounting purposes. When the client presents a certificate as part of the TLS handshake, the Peer-Id is determined based on information in the certificate, as specified in Section 5.2 of [RFC5216]. Otherwise, the Peer-Id is null.

12.3. 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 Section 5.2 of [RFC5216]. Otherwise, the Server-Id is null.

13. AVP Summary

The following table lists each AVP defined in this document, whether the AVP may appear in a packet from server to client ("Request") and/or in a packet from client to server ("Response"), and whether the AVP MUST be implemented ("MI").

Name	Request	Response	MI
User-Name		X	
User-Password		X	
CHAP-Password		X	
Reply-Message	X		
CHAP-Challenge		X	
EAP-Message	X	X	X
MS-CHAP-Response		X	
MS-CHAP-Error	X		
MS-CHAP-NT-Enc-PW		X	
MS-CHAP-Domain	X		
MS-CHAP-Challenge		X	
MS-CHAP2-Response		X	
MS-CHAP2-Success	X		
MS-CHAP2-CPW		X	

14. Security Considerations

14.1. Security Claims

Pursuant to RFC 3748, 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:	Up to 384 bits
Dictionary attack prot.:	Yes
Fast reconnect:	Yes
Cryptographic binding:	No
Session independence:	Yes
Fragmentation:	Yes
Channel binding:	No

14.1.1. Authentication Mechanism

EAP-TTLSv0 utilizes negotiated underlying authentication protocols, both in the phase 1 TLS handshake and the phase 2 tunneled authentication. In a typical deployment, at a minimum the TTLS server authenticates to the client in phase 1, and the client authenticates to the AAA/H server in phase 2. Phase 1 authentication of the TTLS server to the client is typically by certificate; the client may optionally authenticate to the TTLS server by certificate

as well. Phase 2 authentication of the client to the AAA/H server is typically by password or security token via an EAP or supported non-EAP authentication mechanism; this authentication mechanism may provide authentication of the AAA/H server to the client as well (mutual authentication).

14.1.2. Ciphersuite Negotiation

Ciphersuite negotiation is inherited from TLS.

14.1.3. Mutual Authentication

In the recommended minimum configuration, the TTLS server is authenticated to the client in phase 1, and the client and AAA/H server mutually authenticate in phase 2.

14.1.4. Integrity Protection

Integrity protection is inherited from TLS.

14.1.5. Replay Protection

Replay protection is inherited from TLS.

14.1.6. Confidentiality

Confidentiality is inherited from TLS. Note, however, that EAP-TTLSv0 contains no provision for encryption of success or failure EAP packets.

14.1.7. Key Derivation

Both MSK and EMSK are derived. The key derivation PRF is inherited from TLS, and cryptographic agility of this mechanism depends on the cryptographic agility of the TLS PRF.

14.1.8. Key Strength

Key strength is limited by the size of the TLS master secret, which for versions 1.0 and 1.1 is 48 octets (384 bits). Effective key strength may be less, depending on the attack resistance of the negotiated Diffie-Helman (DH) group, certificate RSA/DSA group, etc. BCP 86 [RFC3766], Section 5, offers advice on the required RSA or DH module and DSA subgroup size in bits, for a given level of attack resistance in bits. For example, a 2048-bit RSA key is recommended to provide 128-bit equivalent key strength. The National Institute for Standards and Technology (NIST) also offers advice on appropriate key sizes in [SP800-57].

14.1.9. Dictionary Attack Protection

Phase 2 password authentication is protected against eavesdropping and therefore against offline dictionary attack by TLS encryption.

14.1.10. Fast Reconnect

Fast reconnect is provided by TLS session resumption.

14.1.11. Cryptographic Binding

[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.

Extensions to EAP-TTLSv0 or 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. Such an extension mechanism has been proposed [TTLS-EXT].

14.1.12. Session Independence

TLS guarantees the session independence of its master secret, from which the EAP-TTLSv0 MSK/EMSK is derived.

14.1.13. Fragmentation

Provision is made for fragmentation of lengthy EAP packets.

14.1.14. Channel Binding

Support for channel binding may be added as a future extension, using appropriate AVPs.

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.

Note that if a client authenticating with a certificate wishes to shield its certificate, and hence its identity, from eavesdroppers, it may use the technique described in Section 2.1.4 ("Privacy") of [RFC5216], in which the client sends an empty certificate list, the TTLS server issues a ServerHello upon completion of the TLS handshake to begin a second, encrypted handshake, during which the client will send its certificate list. Note that for this feature to work the client must know in advance that the TTLS server supports it.

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 are authorized to act as a TTLS server and 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 Validation

When either client or server presents a certificate as part of the TLS handshake, it should include the entire certificate chain minus the root to facilitate certificate validation by the other party.

When either client or server receives a certificate as part of the TLS handshake, it should validate the certification path to a trusted root. If intermediate certificates are not provided by the sender, the receiver may use cached or pre-configured copies if available, or may retrieve them from the Internet if feasible.

Clients and servers should implement policies related to the Extended Key Usage (EKU) extension [RFC5280] of certificates it receives, to ensure that the other party's certificate usage conforms to the certificate's purpose. Typically, a client EKU, when present, would

be expected to include id-kp-clientAuth; a server EKU, when present, would be expected to include id-kp-serverAuth. Note that absence of the EKU extension or a value of anyExtendedKeyUsage implies absence of constraint on the certificate's purpose.

14.5. 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 the Online Certificate Status Protocol (OCSP) [RFC2560] during the TLS handshake, as described in [RFC4366].

14.6. 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.

14.7. Negotiating-Down Attacks

EAP-TTLS negotiates its own protocol version prior to, and therefore outside the security established by the TLS tunnel. In principle, therefore, it is subject to a negotiating-down attack, in which an intermediary modifies messages in transit to cause a lower version of the protocol to be agreed upon, each party assuming that the other does not support as high a version as it actually does.

The version of the EAP-TTLS protocol described in this document is 0, and is therefore not subject to such an attack. However, any new version of the protocol using a higher number than 0 should define a mechanism to ensure against such an attack. One such mechanism might be the TTLS server's reiteration of the protocol version that it proposed in an AVP within the tunnel, such AVP to be inserted with M bit clear even when version 0 is agreed upon.

15. Message Sequences

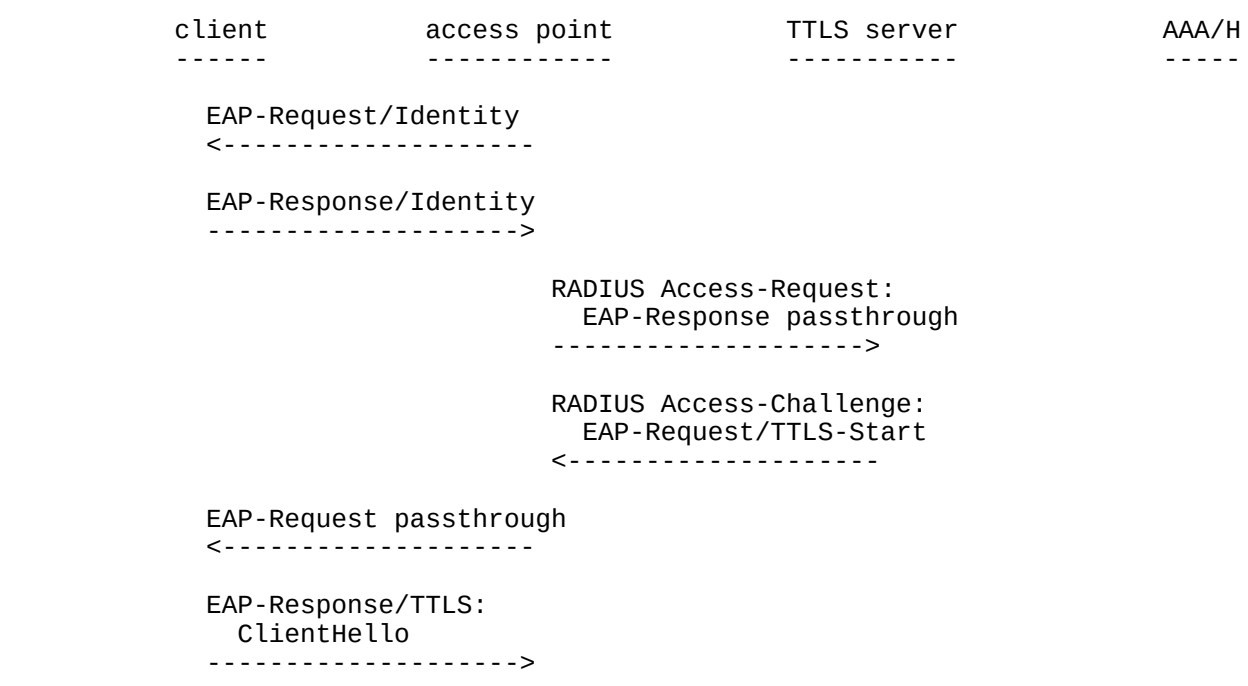
This section presents EAP-TTLS message sequences for various negotiation scenarios. These examples do not attempt to exhaustively depict all possible scenarios.

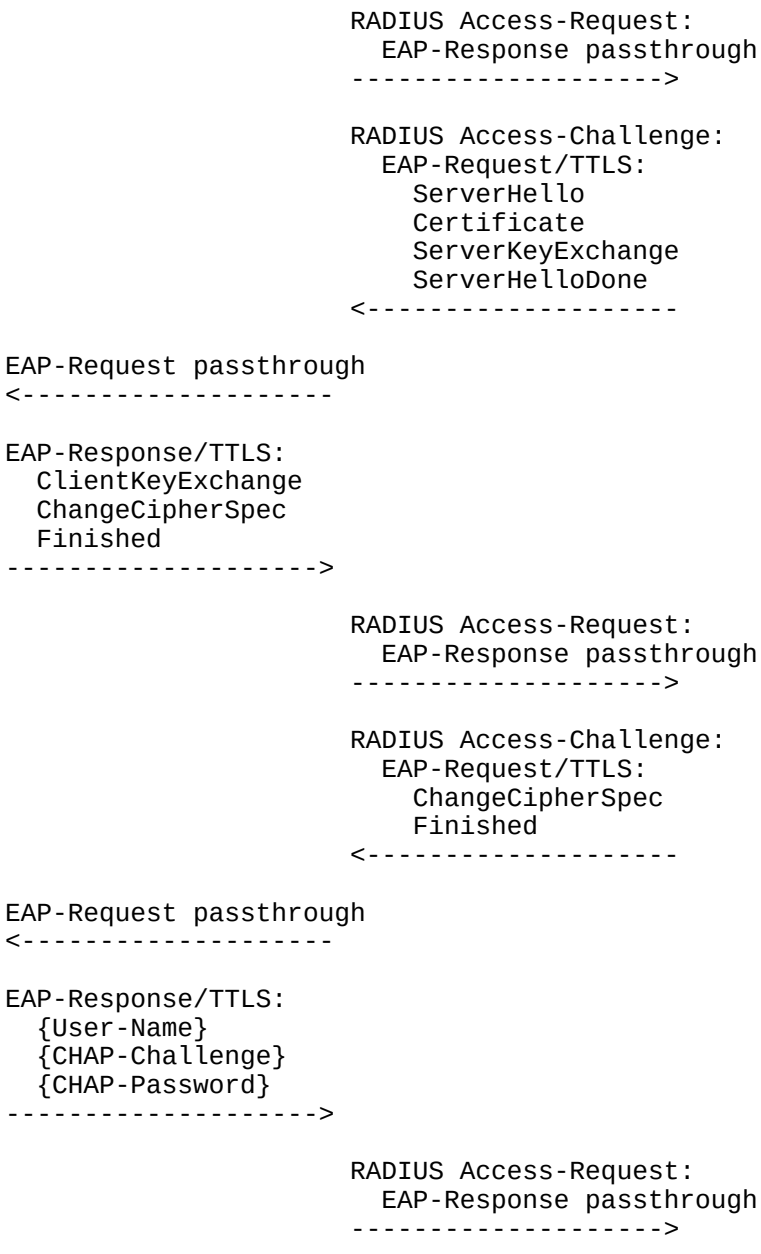
It is assumed that RADIUS is the AAA carrier protocol both between access point and TTLS server, and between TTLS server and AAA/H.

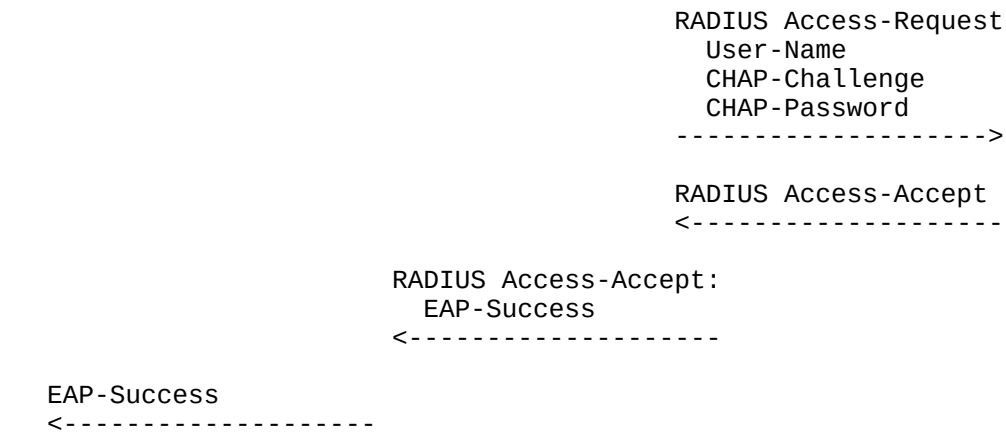
EAP packets that are passed unmodified between client and TTLS server by the access point are indicated as "passthrough". AVPs that are securely tunneled within the TLS record layer are enclosed in curly braces ({}). Items that are optional are suffixed with question mark (?). Items that may appear multiple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

15.1. 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.

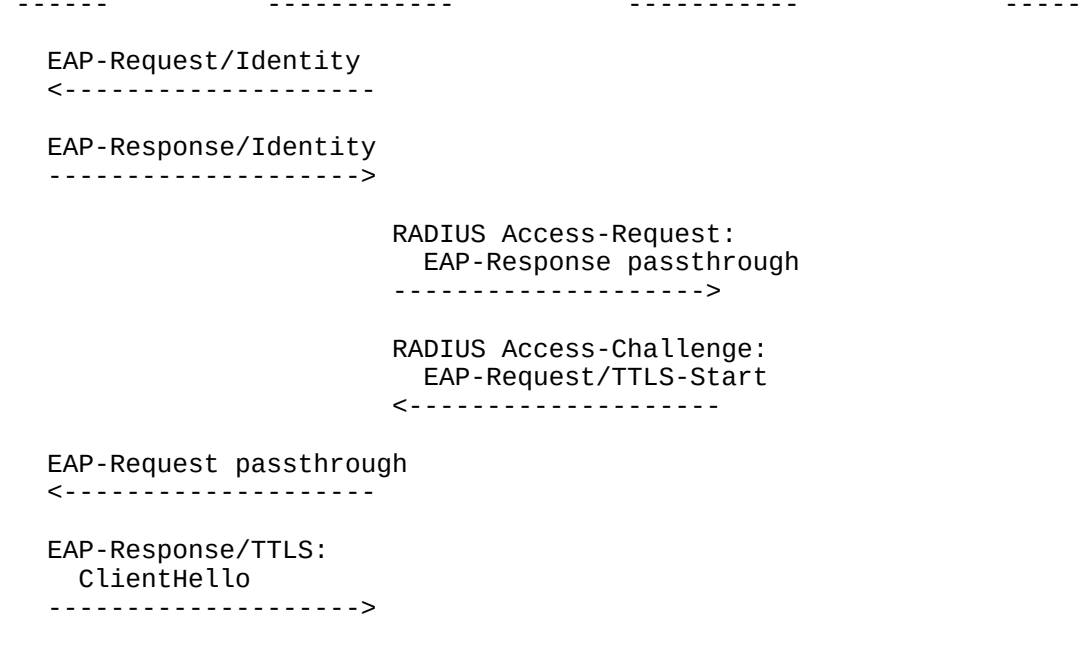


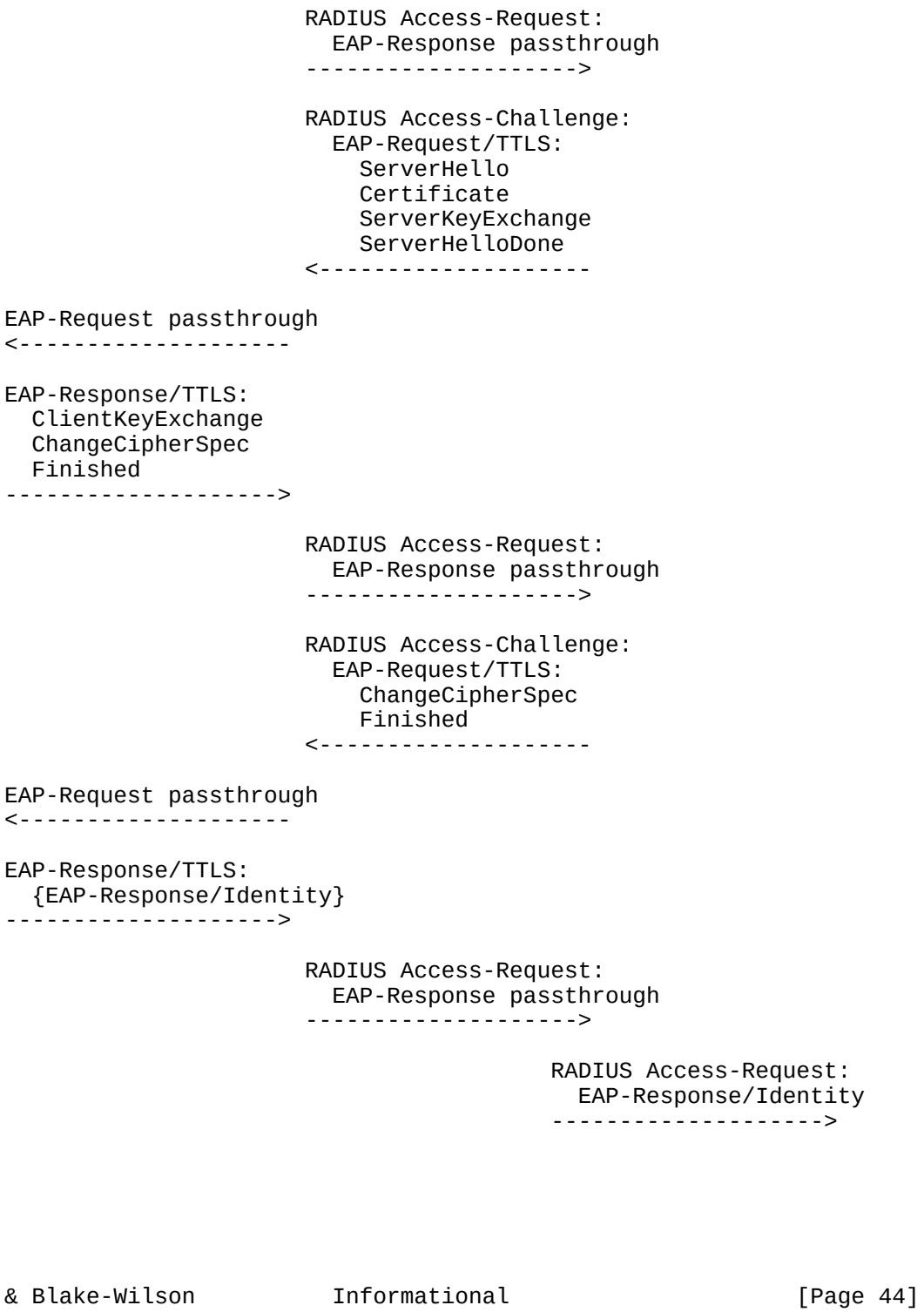

RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Challenge:
EAP-Request/TTLS:
ServerHello
Certificate
ServerKeyExchange
ServerHelloDone
EAP-Request passthrough
EAP-Response/TTLS:
ClientKeyExchange
ChangeCipherSpec
Finished
RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Challenge:
EAP-Request/TTLS:
ChangeCipherSpec
Finished
EAP-Request passthrough
EAP-Response/TTLS:
{User-Name}
{CHAP-Challenge}
{CHAP-Password}
RADIUS Access-Request:
EAP-Response passthrough

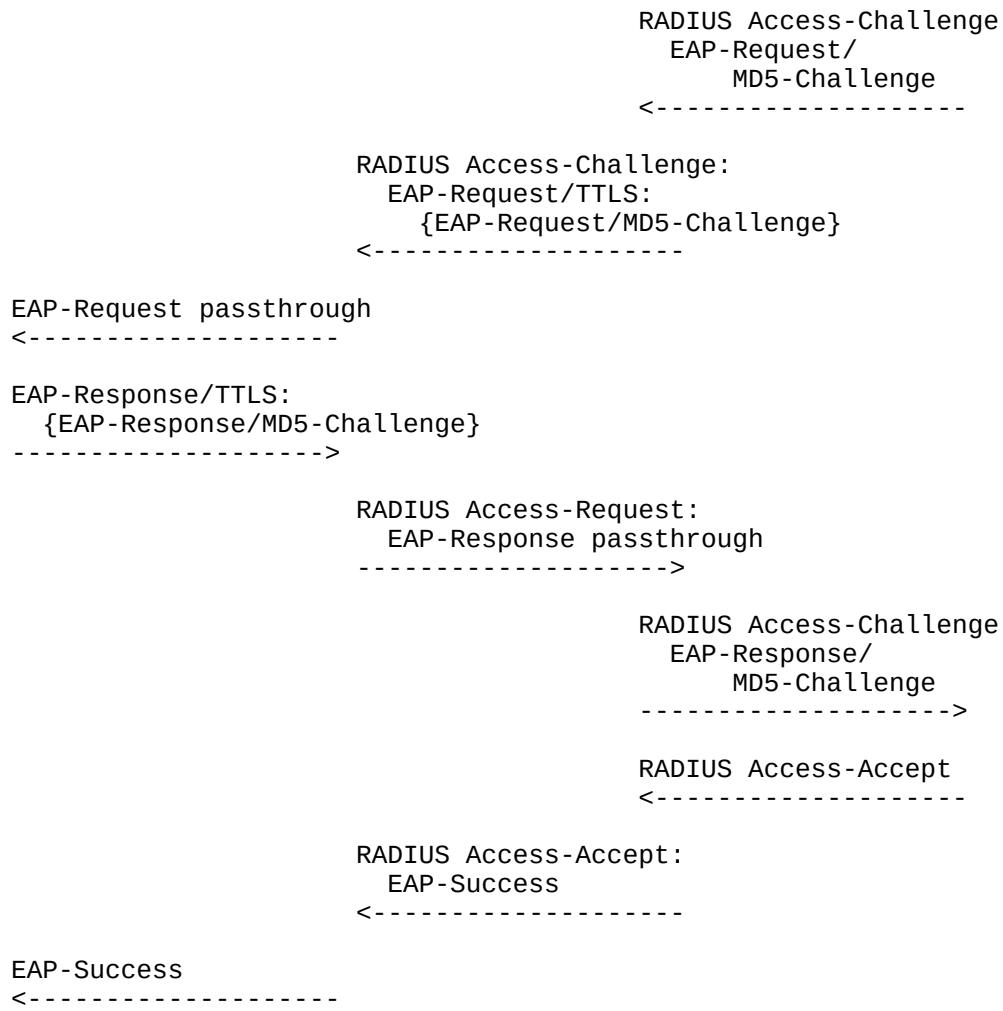


15.2. Successful Authentication via Tunneled EAP/MD5-Challenge

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.

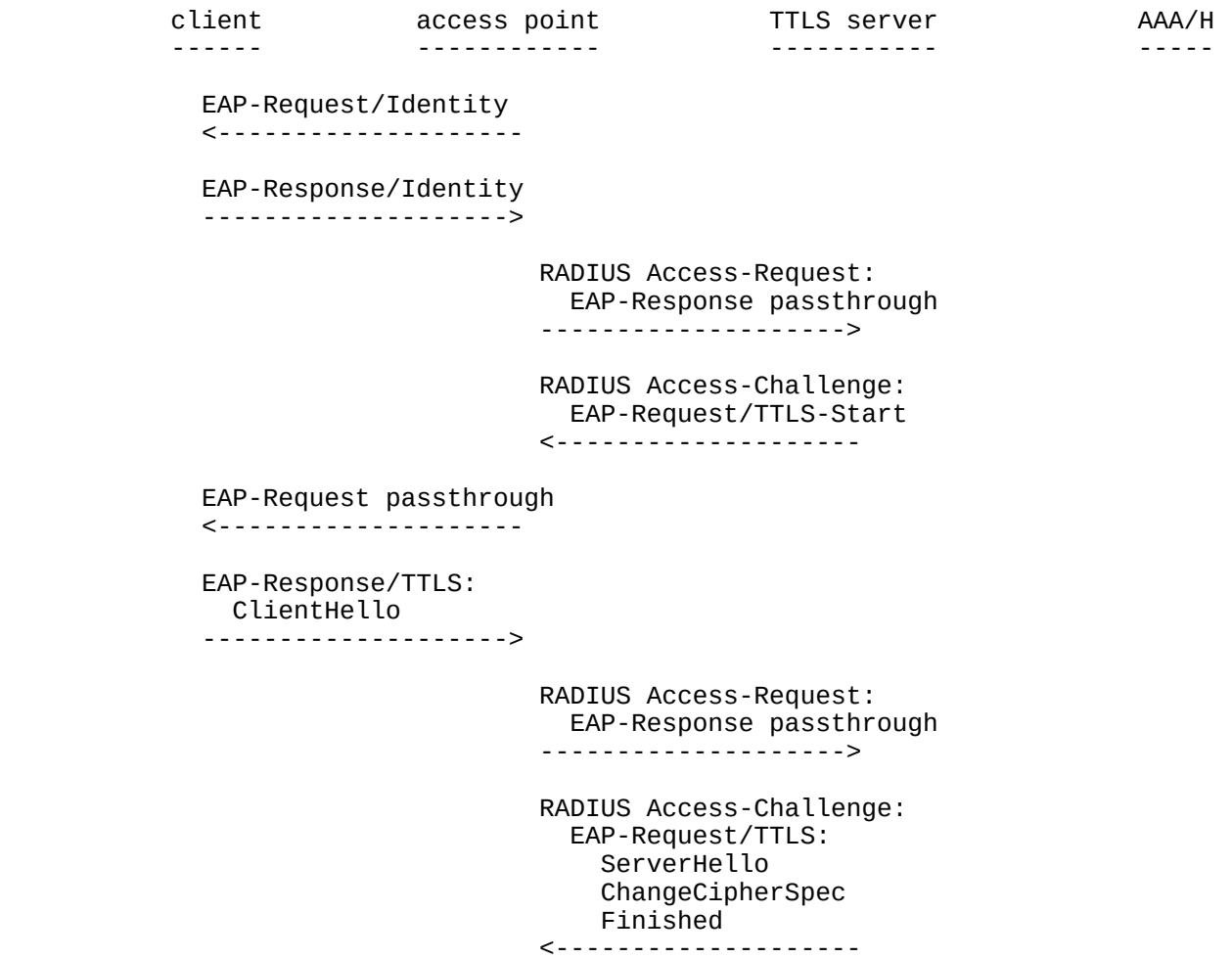



RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Challenge:
EAP-Request/TTLS:
ServerHello
Certificate
ServerKeyExchange
ServerHelloDone
EAP-Request passthrough
EAP-Response/TTLS:
ClientKeyExchange
ChangeCipherSpec
Finished
RADIUS Access-Request:
EAP-Response passthrough
RADIUS Access-Challenge: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

15.3. 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.



```
     EAP-Response/TTLS:
       ChangeCipherSpec
       Finished
     ------------------->

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

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

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

16. IANA Considerations

IANA has assigned the number 21 (decimal) as the method type of the EAP-TTLS protocol. Mechanisms for defining new RADIUS and Diameter AVPs and AVP values are outlined in [RFC2865] and [RFC3588], respectively. No additional IANA registrations are specifically contemplated in this document.

Section 11 of this document specifies how certain authentication mechanisms may be performed within the secure tunnel established by EAP-TTLS. New mechanisms and other functions MAY also be performed within this tunnel. Where such extensions use AVPs that are not vendor-specific, their semantics must be specified in new RFCs; that is, there are TTLS-specific processing rules related to the use of each individual AVP, even though such AVPs have already been defined for RADIUS or DIAMETER.

This specification requires the creation of a new registry -- EAP-TTLS AVP Usage -- to be managed by IANA, listing each non-vendor-specific RADIUS/Diameter AVP that has been defined for use within EAP-TTLS, along with a reference to the RFC or other document that specifies its semantics. The initial list of AVPs shall be those listed in Section 13 of this document. The purpose of this registry is to avoid potential ambiguity resulting from the same AVP being utilized in different functional contexts. This registry does not assign numbers to AVPs, as the AVP numbers are assigned out of the RADIUS and Diameter namespaces as outlined in [RFC2865] and [RFC3588]. Only top-level AVPs -- that is, AVPs not encapsulated within Grouped AVPs -- will be registered. AVPs should be added to this registry based on IETF Review as defined in [RFC5226].

17. Acknowledgements

Thanks to Bernard Aboba, Jari Arkko, Lakshminath Dondeti, Stephen Hanna, Ryan Hurst, Avi Lior, and Gabriel Montenegro for careful reviews and useful comments.

18. References

18.1. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

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

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

[RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

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

[RFC4282] Aboba, B., Beadles, M., Arkko, J. and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

18.2. Informative References

[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.

[802.11] Institute of Electrical and Electronics Engineers, "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, 2007.

[TTLS-EXT] Hanna, S. and P. Funk, "Key Agility Extensions for EAP-TTLSv0", Work in Progress, September 2007.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.

[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.

[SP800-57] National Institute of Standards and Technology, "Recommendation for Key Management", Special Publication 800-57, May 2006.

Authors' Addresses

Paul Funk
43 Linnaean St.
Cambridge, MA 02138
EMail: PaulFunk@alum.mit.edu

Simon Blake-Wilson
SafeNet
Amstelveenseweg 88-90
1054XV, Amsterdam
The Netherlands
EMail: sblakewilson@nl.safenet-inc.com

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

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.