EMU Working Group
Internet-Draft
Expires: October 21, 2006

T. Otto
H. Tschofenig
Siemens AG
April 19, 2006

The EAP-TLS-PSK Authentication Protocol
draft-otto-emu-eap-tls-psk-00

Status of this Memo

Copyright Notice

Abstract

The Extensible Authentication Protocol (EAP), defined in RFC 3748, is a network access authentication framework which provides support for multiple authentication methods. One proposal is EAP-TLS, which relies on the Transport Layer Security (TLS) protocol and allows for certificate-based authentication. This document specifies EAP-TLS-PSK, which also relies on TLS, but allows for shared secret-based authentication. EAP-TLS-PSK supports the pre-shared key ciphersuites specified in RFC 4279.

Table of Contents

1. Introduction 3
1.1. Overview about pre-shared key TLS ciphersuites 3
1.2. Requirements notation 4
1.3. Terminology 5
2. Protocol Overview 6
2.1. Overview of the EAP-TLS-PSK Conversation 6
2.2. Retry Behavior 10
2.3. Fragmentation 10
2.4. Identity Verification 12
2.5. Key Hierarchy 13
2.6. Ciphersuite and Compression Negotiation 13
3. EAP-TLS-PSK Packet Format 15
4. IANA Considerations 17
5. Security Considerations 18
5.1. Mutual Authentication 18
5.2. Protected Result Indications 18
5.3. Integrity Protection 18
5.4. Replay Protection 18
5.5. Dictionary Attacks 18
5.6. Key Derivation 19
5.7. Session Independence 19
5.8. Exposition of the PSK 19
5.9. Fragmentation 20
5.10. Channel Binding 20
5.11. Fast Reconnect 20
5.12. Identity Protection 20
5.13. Protected Ciphersuite Negotiation 20
5.14. Confidentiality 20
5.15. Cryptographic Binding 21
5.16. Security Claims 21
6. Acknowledgments 23
7. References 24
7.1. Normative References 24
7.2. Informative References 24
Authors' Addresses 27
Intellectual Property and Copyright Statements 28

1. Introduction

The Extensible Authentication Protocol (EAP), described in [RFC3748], provides a standard mechanism for support of multiple authentication methods. Through the use of EAP, support for a number of authentication schemes may be added, including smart cards, Kerberos, Public Key, One Time Passwords, and others.

In 1998, EAP-TLS ([RFC2716]) was published. It performs mutual authentication based on the Transport Layer Security (TLS) protocol. EAP-TLS allows an EAP peer to take advantage of the protected ciphersuite negotiation, mutual authentication and key management capabilities of the TLS protocol, described in [RFC2246bis]. Nonetheless, EAP-TLS restricts on certificate-based authentication.

In December 2005, IETF standardized pre-shared key ciphersuites for TLS [RFC4279]. At IETF 65 in March 2006, the EMU Working Group agreed on leaving EAP-TLS in its current form, i.e. not to enhance it by support of the pre-shared key ciphersuites, but rather to specify a new EAP method for this purpose.

This is the rationale for the EAP method specified in this document, called EAP-TLS-PSK.

1.1. Overview about pre-shared key TLS ciphersuites

The goal of this subsection is to survey the pre-shared key TLS ciphersuites specified in [RFC4279]. These ciphersuites are divided into three sets, which distinguish in the underlying key exchange mechanism and in the way the premaster_secret is computed. The three key exchange mechanisms are henceforth referred to as PSK, DHE_PSK, and RSA_PSK, in compliance with [RFC4279].

Basically, the pre-shared key extensions are realized by adding attributes to the TLS client_key_exchange and TLS server_key_exchange message, or also by changing the semantic of existing attributes.

For instance, all three sets extend the the client_key_exchange message by a PSK identity, and the server_key_exchange message by an attribute "PSK identity hint". This attribute is optional, and can be used by the server to send some hint to the client which identity to choose.

The three key exchange mechanisms PSK, DHE_PSK and RSA_PSK shall be contrasted as next.

PSK

These ciphersuites fully relies on symmetric key algorithms for authentication, are thus very efficient and therefore well suited for constrainted environments. where

The premaster_secret results from a concatenation of the pre-shared key and its length.

DHE_PSK

DHE_PSK uses a PSK to authenticate a Diffie-Hellman key exchange. The Diffie-Hellman public keys are exchanged within the server_key_exchange and client_key_exchange messages. The DHE_PSK ciphersuites provide Perfect Forward Secrecy (PFS).

The premaster_secret results from a concatenation of the pre-shared key, its length, and the Diffie-Hellman shared secret and its length.

RSA_PSK

RSA_PSK combines public key authentication of the server (using RSA and certificates) with pre-shared key authentication of the client. The server sends a certificate message to the client which contains his public key. In this sense, RSA_PSK equals TLS ciphersuites with RSA key exchange. However, [RFC4279] does not further specify what the certificates contain. The client_key_exchange message contains next to the PSK identity a

parameter "EncryptedPreMasterSecret" in length of 48 byte. The first two octets are the TLS version number, the other 46 byte are random data. For a RSA-based ciphersuite, this value is exactly the TLS premaster_secret. For the pre-shared key ciphersuites with RSA_PSK key exchange, however, the premaster_secret results from a concatenation of its 48-byte value with the pre-shared key and its length.

For further information on the respective mechanism, however, please refer to the original specification [RFC4279].

1.2. Requirements notation

In this document, several words are used to signify the requirements of the specification. 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].

1.3. Terminology

This document frequently uses the following terms:

authenticator:

The end of the link initiating EAP authentication. The term authenticator is used in [IEEE-802.1X], and has the same meaning in this document.

peer:

The end of the link that responds to the authenticator. In [IEEE-802.1X], this end is known as the Supplicant.

backend authentication server:

A backend authentication server is an entity that provides an authentication service to an authenticator. When used, this server typically executes EAP methods for the authenticator. This terminology is also used in [IEEE-802.1X].

EAP server:

The entity that terminates the EAP authentication method with the peer. In the case where no backend authentication server is used, the EAP server is part of the authenticator. In the case where the authenticator operates in pass-through mode, the EAP server is located on the backend authentication server.

Master Session Key (MSK):

Keying material that is derived between the EAP peer and server and exported by the EAP method. The MSK is at least 64 octets in length.

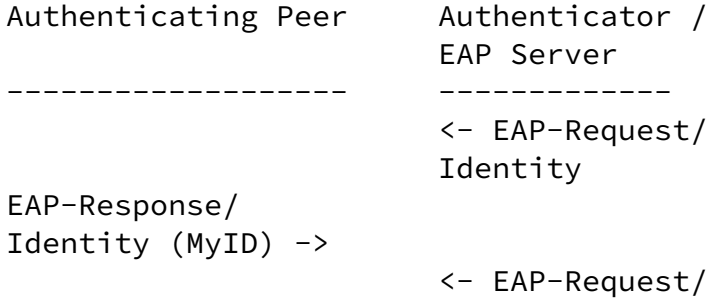
Extended Master Session Key (EMSK):

Additional keying material derived between the EAP client and server that is exported by the EAP method. The EMSK is at least 64 octets in length.

2. Protocol Overview

2.1. Overview of the EAP-TLS-PSK Conversation

The following figure depicts the EAP-TLS-PSK message flow in the successful case.



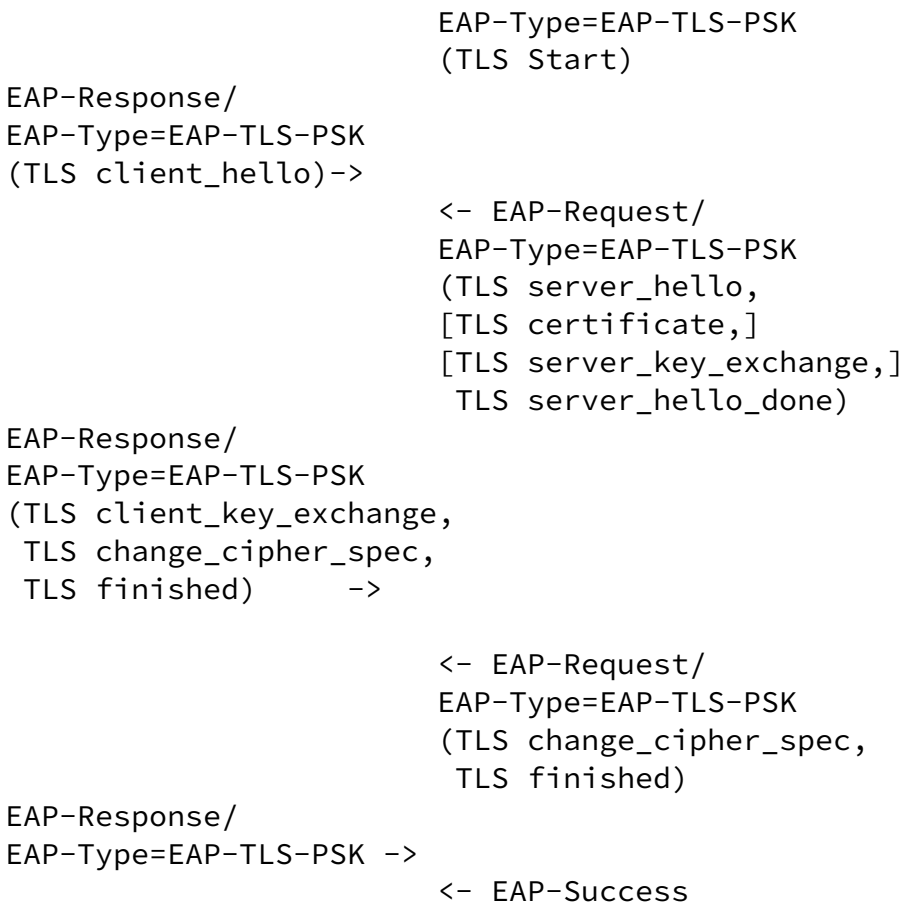


Figure 1: EAP-TLS-PSK message flow

As described in [RFC3748], the EAP-TLS-PSK conversation will typically begin with the authenticator and the peer negotiating EAP. The authenticator will then typically send an EAP-Request/Identity packet to the peer, and the peer will respond with an EAP-Response/ Identity packet to the authenticator, containing the peer's userId.

From this point forward, while nominally the EAP conversation occurs between the EAP authenticator and the peer, the authenticator MAY act as a passthrough device, with the EAP packets received from the peer being encapsulated for transmission to a backend security server. In the discussion that follows, we will use the term "EAP server" to denote the ultimate endpoint conversing with the peer.

Once having received the peer's Identity, the EAP server MUST respond

with an EAP-TLS-PSK/Start packet, which is an EAP-Request packet with EAP-Type=EAP-TLS-PSK, the Start (S) bit set, and no data. The EAP-TLS-PSK conversation will then begin, with the peer sending an EAP-Response packet with EAP-Type=EAP-TLS-PSK. The data field of that packet will encapsulate one or more TLS records in TLS record layer format, containing a TLS client_hello handshake message.

The current cipher spec for the TLS records will be TLS_NULL_WITH_NULL_NULL and null compression. This current cipher spec remains the same until the change_cipher_spec message signals that subsequent records will have the negotiated attributes for the remainder of the handshake.

The client_hello message contains the client's TLS version number, a sessionId, a random number, and a set of ciphersuites supported by the client. The version offered by the client MUST correspond to TLS v1.0 or later.

The EAP server will then respond with an EAP-Request packet with EAP-Type=EAP-TLS-PSK. The data field of this packet will encapsulate one or more TLS records. These will contain a TLS server_hello handshake message, possibly followed by TLS server_key_exchange, server_hello_done and/or finished handshake messages, and/or a TLS change_cipher_spec message. The server_hello handshake message contains a TLS version number, another random number, a sessionId, and a ciphersuite. The version offered by the server MUST correspond to TLS v1.0 or later.

If the client's sessionId is null or unrecognized by the server, the server MUST choose the sessionId to establish a new session; otherwise, the sessionId will match that offered by the client, indicating a resumption of the previously established session with that sessionID. The server will also choose a ciphersuite from those offered by the client; if the session matches the client's, then the ciphersuite MUST match the one negotiated during the handshake protocol execution that established the session.

The purpose of the sessionId within the TLS protocol is to allow for improved efficiency in the case where a client repeatedly attempts to authenticate to an EAP server within a short period of time.

As a result, it is left up to the peer whether to attempt to continue

a previous session, thus shortening the TLS conversation. Typically the peer's decision will be made based on the time elapsed since the previous authentication attempt to that EAP server. Based on the sessionId chosen by the peer, and the time elapsed since the previous authentication, the EAP server will decide whether to allow the continuation, or whether to choose a new session.

In the case where the EAP server and authenticator reside on the same device, then client will only be able to continue sessions when connecting to the same NAS or tunnel server. Should these devices be set up in a rotary or round-robin then it may not be possible for the peer to know in advance the authenticator it will be connecting to, and therefore which sessionId to attempt to reuse. As a result, it is likely that the continuation attempt will fail. In the case where the EAP authentication is remoted then continuation is much more likely to be successful, since multiple NAS devices and tunnel servers will remote their EAP authentications to the same backend authentication server.

If the EAP server is resuming a previously established session, then it MUST include only a TLS change_cipher_spec message and a TLS finished handshake message after the server_hello message. The finished message contains the EAP server's authentication response to the peer. If the EAP server is not resuming a previously established session, then it MUST include a TLS server_certificate handshake message, and a server_hello_done handshake message MUST be the last handshake message encapsulated in this EAP-Request packet.

In case of a RSA_PSK ciphersuite, the server sends a certificate message. This message MUST contain the server's public key. In accordance to EAP-TLS, the certificate message contains a public key certificate chain for either a key exchange public key (such as an RSA or Diffie-Hellman key exchange public key) or a signature public key (such as an RSA or DSS signature public key). In the latter case, a TLS server_key_exchange handshake message MUST also be included to allow the key exchange to take place.

In an EAP-TLS-PSK message exchange, the client will never send a certificate message and certificate_verify message, and the server will never send a certificate_request message.

The peer MUST respond to the EAP-Request with an EAP-Response packet of EAP-Type=EAP-TLS-PSK. The data field of this packet will encapsulate one or more TLS records containing a TLS client_key_exchange, change_cipher_spec and and finished handshake message.

If the preceding server_hello message sent by the EAP server in the preceding EAP-Request packet indicated the resumption of a previous session, then the peer MUST send only the change_cipher_spec and finished handshake messages. The finished message contains the peer's authentication response to the EAP server.

If the preceding server_hello message sent by the EAP server in the preceeding EAP-Request packet did not indicate the resumption of a previous session, then the peer MUST send, in addition to the change_cipher_spec and finished messages, a client_key_exchange message, which completes the exchange of a shared master secret between the peer and the EAP server.

If the peer's authentication is unsuccessful, the EAP server SHOULD send an EAP-Request packet with EAP-Type=EAP-TLS-PSK, encapsulating a TLS record containing the appropriate TLS alert message. In particular, if the server does not recognize the PSK identity, it MUST respond with either an "unknown_psk_identity" TLS alert pessage or continue with the protocol and send a TLS "decrypt_error" alert, which stands for an incorrect key.

The server SHOULD send a TLS alert message rather immediately terminating the conversation so as to allow the peer to inform the user of the cause of the failure and possibly allow for a restart of the conversation.

To ensure that the peer receives the TLS alert message, the EAP server MUST wait for the peer to reply with an EAP-Response packet. The EAP-Response packet sent by the peer MAY encapsulate a TLS client_hello handshake message, in which case the EAP server MAY allow the EAP-TLS-PSK conversation to be restarted, or it MAY contain an EAP-Response packet with EAP-Type=EAP-TLS-PSK and no data, in which case the EAP server MUST send an EAP-Failure packet, and terminate the conversation. It is up to the EAP server whether to allow restarts, and if so, how many times the conversation can be restarted. An EAP server implementing restart capability SHOULD impose a limit on the number of restarts, so as to protect against denial of service attacks.

If the peers authenticates successfully, the EAP server MUST respond with an EAP-Request packet with EAP-Type=EAP-TLS-PSK, which includes, in the case of a new TLS session, one or more TLS records containing TLS change_cipher_spec and finished handshke messages. The latter contains the EAP server's authentication response to the peer. The peer will then verify the hash in order to authenticate the EAP server.

If the EAP server authenticates unsuccessfully, the peer MAY send an

EAP-Response packet of EAP-Type=EAP-TLS-PSK containing a TLS Alert message identifying the reason for the failed authentication. The peer MAY send a TLS alert message rather than immediately terminating the conversation so as to allow the EAP server to log the cause of the error for examination by the system administrator.

To ensure that the EAP server receives the TLS alert message, the peer MUST wait for the EAP server to reply before terminating the conversation. The EAP server MUST reply with an EAP-Failure packet since server authentication failure is a terminal condition.

If the EAP server authenticates successfully, the peer MUST send an EAP-Response packet of EAP-Type=EAP-TLS-PSK, and no data. The EAP server then MUST respond with an EAP-Success message.

2.2. Retry Behavior

As with other EAP protocols, the EAP server is responsible for retry behavior. This means that if the EAP server does not receive a reply from the peer, it MUST resend the EAP-Request for which it has not yet received an EAP-Response. However, the peer MUST NOT resend EAP-Response packets without first being prompted by the EAP server.

For example, if the initial EAP-TLS-PSK start packet sent by the EAP server were to be lost, then the peer would not receive this packet, and would not respond to it. As a result, the EAP-TLS-PSK start packet would be resent by the EAP server. Once the peer received the EAP-TLS-PSK start packet, it would send an EAP-Response encapsulating the client_hello message. If the EAP-Response were to be lost, then the EAP server would resend the initial EAP-TLS-PSK start, and the peer would resend the EAP-Response.

As a result, it is possible that a peer will receive duplicate EAP-Request messages, and may send duplicate EAP-Responses. Both the peer and the EAP server should be engineered to handle this possibility.

2.3. Fragmentation

A single TLS record may be up to 16384 octets in length, but a TLS message may span multiple TLS records, and a TLS certificate message may in principle be as long as 16MB. The group of EAP-TLS-PSK messages sent in a single round may thus be larger than the PPP MTU size, the maximum RADIUS packet size of 4096 octets, or even the Multilink Maximum Received Reconstructed Unit (MRRU). As described in [RFC1990], the multilink MRRU is negotiated via the Multilink MRRU LCP option, which includes an MRRU length field of two octets, and thus can support MRRUs as large as 64 KB.

However, note that in order to protect against reassembly lockup and denial of service attacks, it may be desirable for an implementation to set a maximum size for one such group of TLS messages. Since a typical certificate chain is rarely longer than a few thousand octets, and no other field is likely to be anwhere near as long, a reasonable choice of maximum acceptable message length might be 64 KB.

If this value is chosen, then fragmentation can be handled via the multilink PPP fragmentation mechanisms described in [RFC1990]. While this is desirable, there may be cases in which multilink or the MRRU LCP option cannot be negotiated. As a result, an EAP-TLS-PSK implementation MUST provide its own support for fragmentation and reassembly.

Since EAP is a simple ACK-NAK protocol, fragmentation support can be added in a simple manner. In EAP, fragments that are lost or damaged in transit will be retransmitted, and since sequencing information is provided by the Identifier field in EAP, there is no need for a fragment offset field as is provided in IPv4.

EAP-TLS-PSK fragmentation support is provided through addition of a flags octet within the EAP-Response and EAP-Request packets, as well as a TLS Message Length field of four octets. Flags include the Length included (L), More fragments (M), and EAP-TLS-PSK Start (S) bits. The L flag is set to indicate the presence of the four octet TLS Message Length field, and MUST be set for the first fragment of a fragmented TLS message or set of messages. The M flag is set on all but the last fragment. The S flag is set only within the EAP-TLS-PSK start message sent from the EAP server to the peer. The TLS Message Length field is four octets, and provides the total length of the TLS message or set of messages that is being fragmented; this simplifies

buffer allocation.

When an EAP-TLS-PSK peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-TLS-PSK and no data. This serves as a fragment ACK. The EAP server MUST wait until it receives the EAP-Response before sending another fragment. In order to prevent errors in processing of fragments, the EAP server MUST increment the Identifier field for each fragment contained within an EAP-Request, and the peer MUST include this Identifier value in the fragment ACK contained within the EAP-Response. Retransmitted fragments will contain the same Identifier value.

Similarly, when the EAP server receives an EAP-Response with the M bit set, it MUST respond with an EAP-Request with EAP-Type=EAP-TLS-PSK and no data. This serves as a fragment ACK. The EAP peer MUST

wait until it receives the EAP-Request before sending another fragment. In order to prevent errors in the processing of fragments, the EAP server MUST use increment the Identifier value for each fragment ACK contained within an EAP-Request, and the peer MUST include this Identifier value in the subsequent fragment contained within an EAP- Response.

2.4. Identity Verification

As noted in [RFC3748] Section 5.1: It is RECOMMENDED that the Identity Response be used primarily for routing purposes and selecting which EAP method to use. EAP Methods SHOULD include a method-specific mechanism for obtaining the identity, so that they do not have to rely on the Identity Response.

As part of the EAP-TLS-PSK authentication, the peer uses the client_key_exchange message to transmit his identity.

For RSA_PSK ciphersuites, the server presents a certificate to the peer, which contains his identity.

EAP-TLS-PSK therefore provides a mechanism for determining both the peer and server identities.

o The peer identity (Peer-ID in is exactly the PSK identity of the

client_key_exchange message.

- The server identity (Server-ID in is for ciphersuites of type "RSA_PSK" either directly determined from the altSubjectName in the server certificate or by a mapping of the altSubjectName to the Server-ID using a directory service.
  For ciphersuites of type "PSK" and "DHE_PSK", the server identity is uniquely defined by means of the pre-shared key, which is shared exclusively between an EAP peer and EAP server.

For RSA_PSK, the peer MUST verify the validity of the EAP server certificate, and SHOULD also examine the EAP server name presented in the certificate, in order to determine whether the EAP server can be trusted. Please note that in the case where the EAP authentication is remoted that the EAP server will not reside on the same machine as the authenticator, and therefore the name in the EAP server's certificate cannot be expected to match that of the intended destination. In this case, a more appropriate test might be whether the EAP server's certificate is signed by a CA controlling the intended destination and whether the EAP server exists within a target sub-domain.

2.5. Key Hierarchy

In EAP-TLS-PSK, the MSK, EMSK and IV are derived from the TLS master secret via a one-way function. This ensures that the TLS master secret cannot be derived from the MSK, EMSK or IV unless the one-way function (TLS PRF) is broken. Since the MSK is derived from the the TLS master secret, if the TLS master secret is compromised then the MSK is also compromised.

The notation X[A..B] means byte A to B of X. The notation TLS-PRF-X means that the TLS-PRF is iterated as long as possible to generate X byte output.

Having this, the EAP-TLS-PSK key derivation procedure looks as follows:

```
MSK  = TLS-PRF-128 (master_secret, "client EAP encryption",
     client.random || server.random)[0..63]
```

```
EMSK = TLS-PRF-128 (master_secret, "client EAP encryption",
     client.random || server.random)[64..127]

IV   = TLS-PRF-64 ("", "client EAP encryption",
     client.random || server.random)[0..63]
```

The TLS-negotiated ciphersuite is used to set up a protected channel for use in protecting the EAP conversation, keyed by the derived TEKs. The TEK derivation proceeds as follows:

```
master_secret = TLS-PRF-48 (pre_master_secret, "master secret",
   client.random || server.random)

TEK = TLS-PRF-X (master_secret, "key expansion",
   server.random || client.random)
```

To meet the requirements of [I-D.ietf-eap-keying], EAP-TLS-PSK defines a Method-ID, which is used for computing the Session-ID and key names. In the current version, the Method-ID is set to the concatenation of the server and client Finished messages. The Method-ID uniquely identifies an EAP-TLS-PSK session, because the hashes in the Finished message contain the random values exchanged with the ClientHello- and ServerHello messages as well as the identities of client and EAP server.

2.6. Ciphersuite and Compression Negotiation

Since TLS supports ciphersuite negotiation, peers completing the TLS negotiation will also have selected a ciphersuite, which includes

encryption and hashing methods. Since the ciphersuite negotiated within EAP-TLS-PSK applies only to the EAP conversation, TLS ciphersuite negotiation SHOULD NOT be used to negotiate the ciphersuites used to secure data.

EAP-TLS-PSK is intended to be used only with the ciphersuites defined in [RFC4279]. For convenience, these ciphersuites are summarized below.

```
CipherSuite TLS_PSK_WITH_RC4_128_SHA          = { 0x00, 0x8A };
CipherSuite TLS_PSK_WITH_3DES_EDE_CBC_SHA     = { 0x00, 0x8B };
```

```
CipherSuite TLS_PSK_WITH_AES_128_CBC_SHA      = { 0x00, 0x8C };
CipherSuite TLS_PSK_WITH_AES_256_CBC_SHA      = { 0x00, 0x8D };
CipherSuite TLS_DHE_PSK_WITH_RC4_128_SHA      = { 0x00, 0x8E };
CipherSuite TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA = { 0x00, 0x8F };
CipherSuite TLS_DHE_PSK_WITH_AES_128_CBC_SHA  = { 0x00, 0x90 };
CipherSuite TLS_DHE_PSK_WITH_AES_256_CBC_SHA  = { 0x00, 0x91 };
CipherSuite TLS_RSA_PSK_WITH_RC4_128_SHA      = { 0x00, 0x92 };
CipherSuite TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA = { 0x00, 0x93 };
CipherSuite TLS_RSA_PSK_WITH_AES_128_CBC_SHA  = { 0x00, 0x94 };
CipherSuite TLS_RSA_PSK_WITH_AES_256_CBC_SHA  = { 0x00, 0x95 };
```

TLS also supports compression as well as ciphersuite negotiation. Since compression negotiated within EAP-TLS-PSK applies only to the EAP conversation, TLS compression negotiation MUST NOT be used to negotiate compression mechanisms to be applied to data.

3. EAP-TLS-PSK Packet Format

A summary of the EAP TLS Request/Response packet format is shown below. The fields are transmitted from left to right.

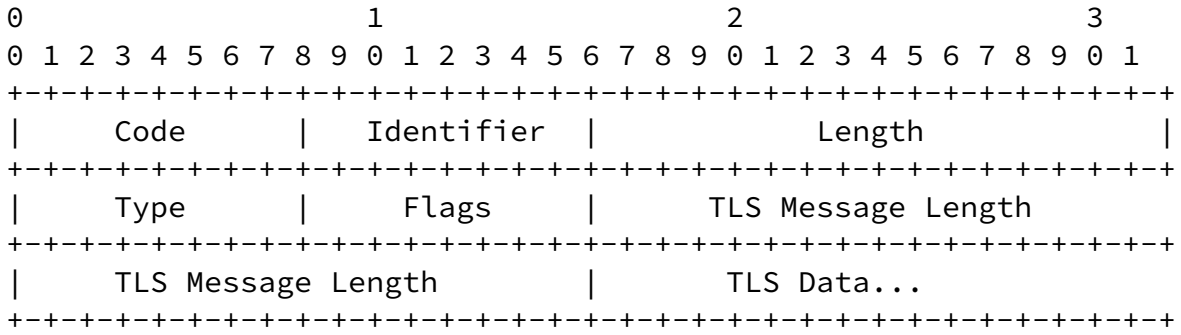


Figure 3: EAP-TLS-PSK packet format

Code

1 - EAP-TLS-PSK Request (short: Request)
2 - EAP-TLS-PSK Response (short: Response)

Identifier

The identifier field is one octet and aids in matching responses with requests. If the message is of type Response, then the identifier MUST match the Identifier field from the corresponding request.

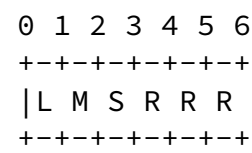
Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception.

Type

TBD - EAP-TLS-PSK

Flags



L = Length included
M = More fragments
S = EAP-TLS-PSK start
R = Reserved

The L bit (length included) is set to indicate the presence of the four octet TLS Message Length field, and MUST be set for the first fragment of a fragmented TLS message or set of messages. The M bit (more fragments) is set on all but the last fragment. The S bit (EAP-TLS-PSK start) is set in an EAP-TLS-PSK Start message. This differentiates the EAP-TLS-PSK Start message from a fragment acknowledgement. Implementations of this specification MUST set the reserved bits to zero, and MUST ignore them on reception.

TLS Message Length

The TLS Message Length field is four octets, and is present only if the L bit is set. This field provides the total length of the TLS message or set of messages that is being fragmented.

TLS Data

The TLS data consists of the encapsulated TLS packet in TLS record format.

4. IANA Considerations

This document requires IANA to allocate a new EAP Type for EAP-TLS-PSK.

5. Security Considerations

[RFC3748] highlights several attacks that are possible against EAP as EAP does not provide any robust security mechanism. This section discusses the claimed security properties of EAP-TLS-PSK as well as vulnerabilities and security recommendations in the threat model of [RFC3748].

5.1. Mutual Authentication

EAP-TLS-PSK provides mutual authentication. This holds for all three modes PSK, DHE_PSK and RSA_PSK.

5.2. Protected Result Indications

EAP-TLS-PSK does not provide protected result indications.

5.3. Integrity Protection

EAP-TLS-PSK provides integrity protection thanks to the TLS Finished message, which contains a Message Authentication Code computed over the whole previous conversation. That is, the verification of the Finished message serves as guarantee of the conversation's integrity.

5.4. Replay Protection

EAP-TLS-PSK provides replay protection of its mutual authentication part thanks to the use of random numbers in the client_hello and server_hello messages. These random numbers are 16 byte long. One expects to have to record 2**64 (i.e. approximately 1.84*10**19) EAP-TLS-PSK successful authentication before an authentication can be replayed. A good source for randomness is cruicial for the security of EAP-TLS-PSK.

5.5. Dictionary Attacks

For PSK and DHE_PSK, mutual authentication is based on a shared secret. While [RFC4279] does not specify the length of this pre-shared key, EAP-TLS-PSK does so. The pre-shared key MUST be at least 16 byte long and have full entropy. For these two modes, it is highly discouraged having derived the pre-shared key from low entropy source, e.g. a password.

For RSA_PSK, the pre-shared key SHOULD also be at least 16 byte long. In contrast to PSK and DHE_PSK, the pre-shared key may also be derived from a low entropy source, e.g. a password. This becomes possible because the EAP server authenticates himself through public key techniques and prior than the client.

In this version of EAP-TLS-PSK, however, the pre-shared key MUST BE at least 16 byte long and MUST HAVE full entropy. Then, EAP-TLS-PSK is not susceptible for dictionary attacks.

5.6. Key Derivation

EAP-TLS-PSK supports key derivation according to [RFC3748] and [I-D.ietf-eap-keying].

EAP-TLS-PSK exports keys, namely a 64-byte MSK, a 64-byte EMSK and a 64-byte IV.

Some remarks regarding the key strength of EAP-TLS-PSK. In case of RSA_PSK, the server's private key size should be chosen accordingly to the length of the pre-shared key. Section 5 in [RFC3766] contrasts symmetric key sizes with their public key counterparts, to obtain roughly the same overall key strength. Based on the table below, a 3072-bit RSA key is required to provide 128-bit equivalent key strength.

Attack Resistance (bits)	RSA or DH Modulus size (bits)	DSA subgroup size (bits)
70	947	128
80	1228	145
90	1553	153

100	1926	184
150	4575	279
200	8719	373
250	14596	475

5.7. Session Independence

Thanks to its key derivation mechanisms, EAP-PSK provides session independence: passive attacks (such as capture of the EAP conversation) or active attacks (including compromise of the MSK or EMSK) does not enable compromise of subsequent or prior MSKs or EMSKs. The assumption that the random numbers of the TLS client_hello and server_hello messages are random is central for the security of EAP-TLS-PSK in general and session independance in particular.

5.8. Exposition of the PSK

EAP-TLS-PSK specifies three sets of ciphersuites, which distinguish in the underlying key derivation mechanism.

- The PSK and RSA_PSK ciphersuites do not provide perfect forward secrecy. Compromise of the pre-shared key or the pre-shared key and the server's private key (in case of RSA_PSK) leads to compromise of recorded past sessions.

- DHE_PSK ciphersuites provide perfect forward secrecy, if a fresh Diffie-Hellman private key is generated for each handshake.

5.9. Fragmentation

EAP-TLS-PSK supports fragmentation and reassembly. The mechanism is inherited from EAP-TLS ([RFC2716]).

5.10. Channel Binding

EAP-TLS-PSK does not provide channel binding.

5.11. Fast Reconnect

EAP-TLS-PSK relies on the Transport Layer Security protocol, which specifies a fast resumption mode. If peer and server agree on continuing a previously established session, the session's master secret can be re-used. That is, related computations can be omitted, which make the fast resumption mode very efficient.

For PSK ciphersuites, the speedup is believed to be minimal, since these ciphersuites rely on symmetric key operations only. It is expected, that DHE_PSK and RSA_PSK may benefit from the fast resumption mode.

5.12. Identity Protection

EAP-TLS-PSK does not provide user identity protection. The client_key_exchange message contains the peer's identity. This message is sent in plaintext.

5.13. Protected Ciphersuite Negotiation

Since EAP-TLS-PSK relies on TLS, it also supports ciphersuite negotiation. This is done in a securely manner, because the TLS Finished messages authenticate the whole handshake. Therefore, EAP-TLS-PSK provides protected ciphersuite negotiation.

5.14. Confidentiality

EAP-TLS-PSK does not support this feature. According to Section 7.2.1 of [RFC3748], this feature would mandate for the feature 'identity protection', which is also not addressed by EAP-TLS-PSK.

5.15. Cryptographic Binding

This feature is not applicable for EAP-TLS-PSK.

5.16. Security Claims

This section provides the security claims required by [RFC3748].

[a] Mechanism.

* For PSK and DHE_PSK: Pre-shared key.

* For RSA_PSK: Server via Public key, Peer via Pre-shared key.

[b] Security claims. EAP-TLS-PSK provides:

* Mutual authentication (see Section 5.1)

* Integrity protection (see Section 5.3)

* Replay protection (see Section 5.4)

* Key derivation (see Section 5.6)

* Dictionary attack resistance (see Section 5.5)

* Session independence (see section Section 5.5)

* Fast reconnect (see Section 5.11)

* Fragmentation (see Section 5.9)

* Protected cipher suite negotiation (see Section 5.13)

* Perfect Forward Secrecy (at least partially; see Section 5.6)

[c] Key strength. EAP-TLS-PSK provides at least a 16-byte effective key strength.

[d] Indication of vulnerabilities. EAP-TLS-PSK does not provide:

* Identity protection (see Section 5.12)

* Confidentiality (see Section 5.14)

* Cryptographic binding (see Section 5.15)

* Key agreement: the session key is chosen by the peer (see Section 5.6)

* Channel binding (see Section 5.10)

6. Acknowledgments

The EAP-TLS-PSK specification lends parts of both the EAP-TLS and EAP-PSK specifications. The authors would like to thank Bernard Aboba and Dan Simon and Florent Bersani for their challenging work.

7. References

7.1. Normative References

[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.

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

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

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

7.2. Informative References

[I-D.ietf-eap-keying]
Aboba, B., "Extensible Authentication Protocol (EAP) Key Management Framework", draft-ietf-eap-keying-12 (work in progress), April 2006.

[IEEE-802.11]
Institute of Electrical and Electronics Engineers, "Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, 1999.

[IEEE-802.11i]
Institute of Electrical and Electronics Engineers, "Approved Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security",

IEEE 802.11i-2004, June 2004.

[IEEE-802.16e]
Institute of Electrical and Electronics Engineers, "Standard for Local and Metropolitan Area Networks: Part 16: Air Interface for Fixed and Mobile Broadband Wireless

Access Systems: Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operations in Licensed Bands" IEEE 802.16e", IEEE 802.16e, August 2005.

[IEEE-802.1X]
Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X, September 2001.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1570] Simpson, W., "PPP LCP Extensions", RFC 1570, January 1994.

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

[RFC1662] Simpson, W., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

[RFC1990] Sklower, K., Lloyd, B., McGregor, G., Carr, D., and T. Coradetti, "The PPP Multilink Protocol (MP)", RFC 1990, August 1996.

[RFC2419] Sklower, K. and G. Meyer, "The PPP DES Encryption Protocol, Version 2 (DESE-bis)", RFC 2419, September 1998.

[RFC2420] Kummert, H., "The PPP Triple-DES Encryption Protocol (3DESE)", RFC 2420, September 1998.

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

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol",

RFC 2637, July 1999.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, August 1999.

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

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC4334] Housley, R. and T. Moore, "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)", RFC 4334, February 2006.

[TLSPSK-Perf] Fang-Chun Kuo, Hannes Tschofenig, Fabian Meyer, and Xiaoming Fu, "Comparison Studies between Pre-Shared Key and Public Key Exchange Mechanisms for Transport Layer Security (TLS)", IFI-TB-2006-01 URL: http://user.informatik.uni-goettingen.de/~fkuo/publications/ptls-ifi-tb-2006-01.pdf, January 2006.

Authors' Addresses

Thomas Otto
Siemens AG
Otto-Hahn-Ring 6
Munich 81739
Germany

Email: thomas.g.otto@googlemail.com

Hannes Tschofenig
Siemens AG
Otto-Hahn-Ring 6
Munich 81739
Germany

Email: hannes.tschofenig@siemens.com

Intellectual Property Statement

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.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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