EAP-Double-TLS Authentication Protocol
<draft-badra-eap-double-tls-03.txt>

Status

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on November 14, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

1 Abstract

EAP-Double-TLS is an EAP protocol that extends EAP-TLS. In EAP-TLS, a full TLS handshake is used to mutually authenticate a peer and server and to share a secret key. EAP-Double-TLS extends this authentication negotiation by using a secure connection established by the TLS Pre-Shared Key (PSK) handshake to exchange additional information between the peer and the server. The secure connection established by the PSK handshake may then be used to allow the server and the peer to securely exchange their identity and to update security attributes for next sessions.
2 Introduction

The Extensible Authentication Protocol (EAP) [5] defines a mechanism that may be extended with additional authentication protocols within PPP [PPP] such as MD5 [MD5], TLS [TLS] and EAP-TTLS [EAPTTLS].
random numbers to calculate new master_secret and cryptographic keys. This will generate fewer cryptographic computations and less processing time than a full TLS handshake. In addition, it will save the bandwidth which is the bottleneck in the wireless networks.

Shared-key TLS runs as resume sessions using pre-installed secret key. A detailed description may be found in [SKTLS]. However, it may be an advantageous to use shared-key authentication handshake instead of PKI based certificates. Further, shared-key TLS does not require any asymmetric cryptographic operation like RSA encrypt/decrypt or certificates verification.

EAP-Double-TLS is an EAP protocol that extends EAP-TLS. In EAP-TLS, a TLS handshake is used to mutually authenticate a peer and a server. EAP-Double-TLS extends this authentication negotiation by using the secure connection established by the TLS PSK handshake to securely exchange additional information between the peer and the server. In EAP-Double-TLS, the authentication is established using pre-installed key on both the peer and the server. It uses symmetric encryption to allay the PKI requirements of EAP-TLS, PEAP or EAP-TTLS. The secure connection established by the resumed handshake may then be used to allow the server to authenticate the peer using certificate authentication infrastructures, PSK or smart cards.

EAP-Double-TLS allows anonymous exchanges and identity protection against eavesdropping, man-in-the-middle and other cryptographic attacks. It allows also the peer and the server to update security attributes for next sessions and then to ensure the PFS (Perfect Forward Secrecy).

Further, EAP-Double-TLS provides a mechanism for session key establishment for encryption protocols within PPP such as PPP-DES [PPPDES] and PPP-3DES [PPP3DES] protocols.

2.1 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 RFC-2119.

3 Protocol overview

3.1 EAP identity protection

At the beginning of an EAP session, EAP identity (EAP-ID) is transmitted in clear text, in the identity response message. This parameter is used by the authenticator to forward EAP packets to the authentication server which in turn uses it as an index for users’ database management.

In EAP-Double-TLS, EAP-ID SHOULD be replaced either by the TLS session_id value (see 3.2) or by the session_id concatenated to the authentication server address (session_id@server.com).

This process will protect the user’s privacy against surveillance
and make the subscriber’s EAP exchanges untraceable to eavesdroppers. In fact, the current session_id will be replaced by a new one computing during phase 2 (see 3.2).

3.2 Overview of the EAP-Double-TLS conversation

In order to apply the use of shared key TLS, we suggest sharing a TLS session between the peer and the server. The session is identified by the session_id (in TLS terminology). It corresponds to, among others, the value of the master_secret and the cipher_suite. The cipher_suite represents the cryptographic option supported by both the server and the client and it is initialized by both the peer and the server to a particular option.

The shared session may be stocked on a client smartcard.

In general, EAP-Double-TLS negotiation comprises two phases (figure 1):

During the first phase, TLS resumed handshake is used for mutual authentication and key generation. This phase uses a cipher suite allowing phase 2 to securely exchange TLS records or other security protocols payloads.

The second phase of EAP-Double-TLS may be a full TLS handshake with mutual authentication, only server-side authentication, or anonymous. Furthermore, this phase may be established using any mechanism that can generate a session key like Kerberos [Kerberos]. On the other hand, if servers or peers don’t care about anonymity or PFS, they can be satisfied by the phase 1 negotiation. For authorization or additional authentication requirements, the peer and the server can use any mechanism they implement (e.g. MD5, MSCHAP, etc.).

During this second phase, TLS records are exchanged in an encrypted manner using the tunnel established during the first phase in order to optionally perform authorization and authentication mechanisms and to refresh the parameters of the shared session (e.g. session_id and master_secret). This phase is detailed in the following section.
3.1.1 Phase 1: TLS PSK Handshake

In the first phase, the EAP-Double-TLS begins with the authenticator sending an EAP-Request/Identity packet to the peer. The peer will respond with an EAP-Response/Identity packet to the authenticator, containing the peer’s UserId (User Identifier). Once this is established, the authenticator MAY act as a pass-through device, with the EAP packets received from the peer being encapsulated for transmission to a RADIUS server or back-end security server.

When the server receives the peer’s Identity, it MUST respond with an EAP-Double-TLS/Start packet. This is an EAP-Request packet with EAP-Type= EAP-Double-TLS, the Start (S) bit set and no data.

When receiving this message, the peer will answer by EAP-Response packet with EAP-Type= EAP-Double-TLS. The data field encapsulates the TLS client_hello resumed handshake message. This message contains, among others parameters, a random number and the session_Id corresponds to the shared session the peer wishes to use.

The server then checks its sessions’ database for a match. If a match is found, the server replays with EAP-Request with an EAP-Type= EAP-Double-TLS. This packet will encapsulate the TLS server_hello handshake message with the same session_id value and another random number.

After the hello messages, the server will send its TLS change cipher spec message and proceed directly to finished message. The finished message will serve to authenticate the server to the peer since it is encrypted and MACed using keys derived from the shared key.

If the EAP server authenticates unsuccessfully, the peer MAY send an EAP-Response packet of EAP-Type= EAP-Double-TLS containing a TLS Alert message identifying the reason for the failed authentication. A fatal error message results in the immediate termination of the connection.

In order to make sure that the server receives the TLS alert message, the peer MUST wait for the server to reply before
terminating the conversation. Like in [EAPTLS], the server MUST reply with an EAP-Failure packet since server authentication failure is a terminal condition.

If the EAP server authenticates successfully (the peer decrypts the finished message and verify the MAC), the peer MUST send an EAP-Response packet of EAP-Type= EAP-Double-TLS, that transports the change cipher spec and the finished messages. Once this establishment is complete, the peer and the server MAY start the second phase. Otherwise, the server will send data connection keying information and other authorization information to the authenticator.

If the peer authenticates unsuccessfully, the server MAY send an EAP-Response packet of EAP-Type= EAP-Double-TLS containing a TLS Alert message identifying the reason for the failed authentication. Alert messages with a level of fatal result in the immediate termination of the connection.

In order to make sure that the peer receives the TLS alert message, the server MUST wait for the peer to reply before terminating the conversation.

3.1.2 Phase 2

EAP-Double-TLS Phase 2 will occur if the establishment of its first phase is successfully terminated. It may be used to ensure additional services such as peer identity protection and PFS and to apply more authentication and authorization policies. As we cited above, this phase may be established using anonymous TLS sessions or any other mechanism that can generate a session key like Kerberos.

3.1.1.1 Case 1: TLS Handshake

During this phase, the TLS Record layer is used to securely tunnel data related to the second phase. When an anonymous TLS session is chosen to be used, the TLS session information will be encapsulated in sequences of TLS attributes, whose use and format are described in [TLS]. In this case, the server must send the EAP-Double-TLS/HelloRequest to the peer. The aim of this message is to inform the peer that the first phase was successfully terminated and to invite him to start the anonymous TLS session. The hello request message is defined in [TLS] and may be sent by the server at any time.

The anonymous TLS session cannot be established unless both the peer and server agree. In this case, the peer and the server continue to exchange EAP packets until either the anonymous TLS session is successfully established or an error occurs. If the session is successfully established, the peer must send an EAP-Response packet of EAP-Type= EAP-Double-TLS, and no data. The EAP-Server must then respond with an EAP-Success message.

At this point, the server distributes data connection keying information and other authorization data to the authenticator. Note
that EAP-Double-TLS products, as part of the new TLS handshake protocol, new session key and new master_secret key. In this case, the server will distribute to the authenticator, data connection keying and authorization information derived from the shared key that was used during the first phase. Next, the server and the peer replace the security parameters (e.g. the shared key and its identity) with the security parameters that are generated during the anonymous TLS session. These new TLS security parameters will be then used during the next EAP-Double-TLS session.

3.1.1.2 Case 2: Other authentication and authorization mechanisms

In order to extend the EAP-Double-TLS with other authentication and authorization mechanisms, the second phase MAY be established using any mechanism that can generate a session key like Kerberos. In this case and if the first phase has successfully terminated, the server MAY request for additional authentication and authorization information. Thus, the server sends an EAP request indicating the EAP method type (method has the ability of generating session keys).

3.2. Retry behavior

See section 3.2 of RFC 2716.

3.3. Fragmentation

See section 3.3 of RFC 2716.

3.4. Key derivation

EAP-Double-TLS derives keying material after each successful negotiation in each phase. The first phase allows the peer and the server to generate a 48-byte master_secret (MS1) by applying the TLS-PRF (Pseudo Random Function) [TLS] on the shared key (SK), ClientHello.random and ServerHello.random:

\[
MS1 = PRF(SK, "master_secret", 
\text{ClientHello.random + ServerHello.random})[0..48]
\]

This key is used to derive keying material used to encrypt and to calculate the MAC of each message. Keys derived from MS1 are then delivered to the authenticator for additional keys computation.

During the EAP-Double-TLS second phase, the peer and the server will exchange new random values. The peer is also able to randomly generate a secret key (the pre_master_secret in TLS terminology). This key is sent securely to the server using the server public key and it is used to generate a new and fresh master_secret (FMS) key by applying the PRF on, among others, the pre_master_secret (PMS).

The generated key will be then used during the future EAP-Double-TLS session.

\[
FMS = PRF(PMS, "master_secret", 
\text{ClientHello.random + ServerHello.random})[0..48]
\]
3.5. CCP and CCP negotiation

See section 3.6 and 3.7 of RFC 2716.

3.6. Inner method encapsulation

As stated before, EAP-Double-TLS uses the TLS record layer to tunnel information between the peer and the server to, among others operations, perform additional authentication and authorization mechanisms. In this optic, EAP-Double-TLS reuses the attribute-value pairs (AVPs) defined in [EAPTLS].

3.7. Examples

The following exchanges show where TLS Handshake with certificate or anonymous key exchange change within Phase 2, the conversation will be as follows:

```
Authenticating Peer           Authenticator
-------------------------------           -------------
<- EAP-Request/Identity
EAP-Response/Identity ->
<- EAP-Request/
   EAP-Type= EAP-Double-TLS
   (EAP-Double-TLS Start)
EAP-Response/
   EAP-Type= EAP-Double-TLS
   (TLS client_hello) ->
      <- EAP-Request/
         EAP-Type= EAP-Double-TLS
         (TLS server_hello,
          TLS change_cipher_spec,
          TLS finished)
EAP-Response/
   EAP-Type= EAP-Double-TLS
   (TLS change_cipher_spec,
    TLS finished) ->
      <- EAP-Request/
         EAP-Type= EAP-Double-TLS
         (TLS Hello Request)
EAP-Response/
   EAP-Type= EAP-Double-TLS
   (TLS client_hello) ->
      <- EAP-Request/
         EAP-Type= EAP-Double-TLS
         (TLS server_hello,
          [TLS certificate],
          [TLS server_key_exchange],
          [TLS certificate_request],
          TLS server_hello_done)
EAP-Response/
   EAP-Type= EAP-Double-TLS
   ([TLS certificate],
    TLS client_key_exchange,
    TLS server_hello_done) ->
```
The following exchanges show where EAP/Type=X change within Phase 2, the conversation will be as follows:

<table>
<thead>
<tr>
<th>Authenticating Peer</th>
<th>Authenticator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- EAP-Request/Identity</td>
<td></td>
</tr>
<tr>
<td>EAP-Response/Identity -&gt;</td>
<td>&lt;- EAP-Request/</td>
</tr>
<tr>
<td>EAP-Type= EAP-Double-TLS (EAP-Double-TLS Start)</td>
<td>EAP-Type= EAP-Double-TLS</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>EAP-Response/</td>
<td>EAP-Response/</td>
</tr>
<tr>
<td>EAP-Type= EAP-Double-TLS</td>
<td>EAP-Type= EAP-Double-TLS</td>
</tr>
<tr>
<td>(TLS client_hello) -&gt;</td>
<td>(TLS change_cipher_spec,</td>
</tr>
<tr>
<td></td>
<td>TLS finished) -&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- EAP-Request/</td>
<td>EAP-Response/</td>
</tr>
<tr>
<td>EAP-Type= EAP-Double-TLS</td>
<td>EAP-Type= EAP-Double-TLS</td>
</tr>
<tr>
<td>(AVP [EAP-Request/EAP-type=X])</td>
<td>(AVP [EAP-Response/EAP-type=X])</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- EAP-Request/</td>
<td>EAP-Response/</td>
</tr>
<tr>
<td>EAP-Type= EAP-Double-TLS</td>
<td>EAP-Type= EAP-Double-TLS</td>
</tr>
<tr>
<td>(AVP [EAP-Request/EAP-type=X])</td>
<td>(AVP [EAP-Response/EAP-type=X])</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- EAP-Request/</td>
<td>EAP-Response/</td>
</tr>
<tr>
<td>EAP-Type= EAP-Double-TLS</td>
<td>EAP-Type= EAP-Double-TLS</td>
</tr>
<tr>
<td>(AVP [EAP-Request/EAP-type=X, Success])</td>
<td>(AVP [EAP-Request/EAP-type=X])</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- EAP-Request/</td>
<td>EAP-Response/</td>
</tr>
<tr>
<td>EAP-Type= EAP-Double-TLS</td>
<td>EAP-Type= EAP-Double-TLS</td>
</tr>
<tr>
<td>(AVP [EAP-Request/EAP-type=X])</td>
<td>(AVP [EAP-Response/EAP-type=X]) -&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- EAP-Success</td>
<td>EAP-Success</td>
</tr>
</tbody>
</table>

The following exchanges show where TLS Handshake with certificate or
anonymous key exchange change within Phase 2, and fragmentation is
required (during the phase 1, no fragmentation is required), the
conversation (during the phase 2) will be as follows:

<table>
<thead>
<tr>
<th>Authenticating Peer</th>
<th>Authenticator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Badra &amp; Urien</td>
<td>Informational - Expires November 2005</td>
</tr>
</tbody>
</table>

INTERNET DRAFT EAP-Double-TLS May 2005

```plaintext
<- EAP-Request/
   EAP-Type= EAP-Double-TLS
   (TLS Hello Request, S bit set)

EAP-Response/
EAP-Type= EAP-Double-TLS
(TLS client_hello) ->
   <- EAP-Request/
   EAP-Type= EAP-Double-TLS
   (TLS server_hello, [TLS certificate],
   [TLS server_key_exchange],
   [TLS certificate_request],
   TLS server_hello_done)
   (Fragment 1: L, M bits set)

EAP-Response/
EAP-Type= EAP-Double-TLS ->
   <- EAP-Request/
   EAP-Type= EAP-Double-TLS
   (Fragment 2: M bits set)

EAP-Response/
EAP-Type= EAP-Double-TLS ->
   <- EAP-Request/
   EAP-Type= EAP-Double-TLS
   (Fragment 3)
   ...

EAP-Response/
EAP-Type= EAP-Double-TLS
([TLS certificate],
 TLS client_key_exchange,
 [TLS certificate_verify],
 TLS change_cipher_spec,
 TLS finished) (Fragment 1: L, M bits set) ->
   ...
   <- EAP-Success
```

During the phase 1 and in the case where the server authenticates to
the peer successfully, but the peer fails to authenticate to the
server, the conversation will be as follows:

<table>
<thead>
<tr>
<th>Authenticating Peer</th>
<th>Authenticator</th>
</tr>
</thead>
</table>
| EAP-Response/Identity ->
   <- EAP-Request/Identity |
| EAP-Response/Identity ->
   <- EAP-Request/
   EAP-Type= EAP-Double-TLS
   (TLS Start) |

EAP-Response/
EAP-Type= EAP-Double-TLS
(TLS client_hello) ->
   <- EAP-Request/
```
During the phase 1 and in the case where server authentication is unsuccessful, the conversation will be as follows:

**Authenticating Peer**           **Authenticator**  
----------------------------------           -------------  
<- EAP-Request/                   <- EAP-Request/  
  EAP-Type= EAP-Double-TLS        EAP-Type= EAP-Double-TLS  
  (TLS Start)                   (TLS Start)  

EAP-Response/  
EAP-Type= EAP-Double-TLS  
(TLS client_hello)  
<- EAP-Failure  
(User Disconnected)  

5 Detailed description of the EAP-Double-TLS protocol

This section shows the conversation between the peer and the authenticator using EAP-Double-TLS protocol. It takes the same notifications introduced in the section 4 of RFC2716 [EAPTLS].

5.1 EAP-Double-TLS Packet Format

A summary of the EAP-Double-TLS Request/Response packet format is shown below. The fields are transmitted from left to right.
The description of the EAP/Response/identity is detailed according to the IETF RFC 2284.

5.2 EAP-Double-TLS Request Packet

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

<table>
<thead>
<tr>
<th>Code=01</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Flag</td>
<td>EAP-Double-TLS Message Length</td>
</tr>
<tr>
<td>EAP-Double-TLS Message Length</td>
<td>EAP-Double-TLS Data...</td>
<td></td>
</tr>
</tbody>
</table>

Code

1

Identifier

The Identifier field is one octet and aids in matching responses with requests. The Identifier field MUST be changed on each Request packet.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and Double-TLS Response fields.

Type

TBD - EAP Double TLS

Flags

0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
| L M S R R R R |
+-+-+-+-+-+-+-+-+

L = Length included
M = More fragments
S = EAP-Double-TLS start

R = Reserved
The L bit (length included) is set to indicate the presence of the four octet Double-TLS Message Length field, and MUST be set for the first fragment of a fragmented EAP-Double-TLS message or set of messages. The M bit (more fragments) is set on all but the last fragment. The S Bit (EAP-Double-TLS start) is set in an EAP-Double-TLS Start message. This differentiates the EAP-Double-TLS Start message from a fragment acknowledgement.

Double-TLS Message Length

The Double-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 Double-TLS message or set of messages that is being fragmented.

Double-TLS data

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

5.3 EAP-Double-TLS Response Packet

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

```
 0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code=01   |  Identifier   |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Flag      | EAP-Double-TLS Message Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| EAP-Double-TLS Message Length | EAP-Double-TLS Data... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Code

2

Identifier

The Identifier field is one octet and 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 Double-TLS Response fields.
The L bit (length included) is set to indicate the presence of the four octet Double-TLS Message Length field, and MUST be set for the first fragment of a fragmented EAP-Double-TLS message or set of messages. The M bit (more fragments) is set on all but the last fragment. The S bit (EAP-Double-TLS start) is set in an EAP-Double-TLS Start message. This differentiates the EAP-Double-TLS Start message from a fragment acknowledgement.

**Double-TLS Message Length**

The Double-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 Double-TLS message or set of messages that is being fragmented.

**Double-TLS data**

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

### 6 Security Considerations

The EAP-Double-TLS server MUST stock the TLS session in a secured and protected manner in order to prevent attackers from retrieving the master_secret values and session’ parameters.

#### 6.1 Security claims

This section describes EAP-Double-TLS in terms of specific security terminology as required by [EAP].

**6.1.1 Authentication, confidentiality, and Integrity. Replay, man-in-the-middle and dictionary attack protection**

EAP-Double-TLS provides mutual authentication using the shared key during the first phase. It mitigates man-in-the-middle vulnerabilities because of the mutual authentication established during the first phase. The confidentiality and integrity are provided using the negotiated cryptographic algorithms as well as encryption and authentication keys derived from the shared key. Furthermore and during the second phase, messages notification (failure or success) are also protected against man-in-the-middle and eavesdropping attacks. This is because they are encrypted with the tunnel established during the first phase. On the other hand and like TLS, EAP-Double-TLS natively protects against replay protection attacks using sequence numbers. The use of sequence numbers and of strong cryptographic algorithms (offered by TLS and therefore by,
EAP-Double-TLS) defends the protocol against dictionary attacks.

6.1.2 Session independence or perfect forward secrecy

EAP-Double-TLS protocol independently generates keys per session and it uses ephemeral public/private keys during its second phase. As a result, it provides Perfect Forward Secrecy (i.e. ephemeral Diffie-Hellman and RSA public keys are both supported by EAP-Double-TLS phase 2 as key exchange methods). Added to that, passive attacks (such as capture of the EAP conversation) or active attacks (including the recovery of the shared key) do not entail the compromising of prior shared keys and are thus incapable of decrypting previous sessions.

6.1.3 Protected cipher suite negotiation and user identity protection

EAP-Double-TLS ensures cipher suite negotiation in a protected manner. In fact, it uses the same TLS principle that offers an integrated mechanism to protect cipher suite negotiation. This is because at the end of the first phase, the peer and server exchange the finished messages. These messages are always sent immediately after a change cipher spec message to verify that the key exchange and authentication processes were successful. They are the first messages protected with the just-negotiated algorithms and the secret key, and it is computed in function of, among others, all handshake messages data, including the negotiated cipher suite.

Concerning the identity protection and as we cited above, the shared key and its identity are replaced with new values if the second phase of EAP-Double-TLS is successfully terminated. This process will protect the user’s privacy and identity against surveillance and make the subscriber’s EAP exchanges untraceable to eavesdroppers. In fact, the EAP-ID value used at the beginning and the session_id used during the first phase will be replaced by a new session_id securely computing during the EAP-Double-TLS second phase.

6.1.4 Key strength

EAP-Double-TLS reuses the TLS-PRF for keys generation (See [TLS]).

6.1.5 Channel binding

EAP-Double-TLS does not explicitly include any channel binding.

6.1.6 Fast reconnect

Due to the nature of wireless connections, the peer may be disconnected at any time. Fortunately, the EAP-Double-TLS peer and the server don't have to go through the entire process every time they want to communicate. While EAP-Double-TLS is based on TLS, fast reconnection option is implicitly included; executing TLS resumed Handshake (as described in phase 1).
This document requires IANA to allocate a new EAP Type for EAP-Double-TLS.

Acknowledgements

This EAP method has been inspired by [EAPTLS] and [TLS]. Thus, it reused extracts of these documents.

References


Appendix A. EAP-Double-TLS protocol within EAP Smartcards

EAP-support in smartcards is described and detailed by an Internet draft [EAPSC]. It is an opened, ISO 7816 microcontroller supporting most authentication protocols. An EAP smartcard implements an EAP method (EAP-TLS, etc) and works in cooperation with a smartcard interface entity, which transparently sends and receives EAP messages to and from this component.

Smartcard is one of the news technologies added to the world of information technology. In fact, they can make significant impact on current computer systems and network environments because of their inherent security and mobility. Further, they are an effective means of adding enhanced protection to wireless networks; namely 802.11 wireless LAN. Added to that, they are widely used in the Global System for Mobile Communication (GSM) [GSM] in the form of a SIM (Subscriber Identity Module) card for secure access to the mobile network, for storing basic network information and for accounting/billing procedures.

Smartcards have a bear particular attraction, as they generally considered as the most secure computing platform. In fact, they offer good tamper resistance. This means that certain physical hardware and software protections are used, which makes it difficult to extract or modify private and secret information in the module. So it seems a good idea to store the (strong) master_secret keys on a smartcard. Further, smartcard deployment in a typical network such as WLAN 802.11 [802.11] offers the enhanced functionality of tighter authentication.

A.1 Fragmentation issues

Data is exchanged between the terminal and the smartcard through a card acceptance device (CAD) in the form of messages exchanged from the terminal to the card and vice versa. Data transport is established by using Data Pocket called Application Protocol Data Unit (APDU). Each APDU consists of two fields: 5 bytes header and 0-255 bytes of data. The ISO [ISOAPDU] standard defines these command/response packets that are used for reading, writing and exchanging data between the host and the smartcard. These packets transferred from the CAD to the module (command APDU) are followed
by a response APDU from the module back to the CAD.

The TLS Record Layer fragments information blocks into TLS records carrying data in chunks of 16384 bytes or less [TLS]. Furthermore, TLS message may carry multiple TLS records. Since the IEEE 802.3 MAC may not send frames greater than 1518 bytes in length and because fragmentation support is not provided by EAP, it is the responsibility of EAP methods to provide the fragmentation required. For that, EAP-Double-TLS extends the EAP-TLS segmentation method, which defines a segmentation process that splits TLS messages in smaller blocks, acknowledged by the recipient. In our implementation, the RADIUS server generates acknowledged requests and the supplicant answers by acknowledged responses.

EAP-TLS defines the fragmentation mechanism for data exchanged between the server and the terminal. It will not define the data segmentation between the terminal and the smartcard because the latter is not readable to the EAP-TLS server. For that and in order to allow smartcards use, a double segmentation mechanism was introduces by our EAP-Double-TLS to forward TLS packets to the smartcard. We defined this mechanism as following. First, TLS server

Badra & Urien Informational - Expires November 2005 19
INTERNET DRAFT EAP-Double-TLS May 2005

messages are divided in smaller segments (E1, E2), whose size is typically 1400 bytes or less (figure 2). Next, the segments are encapsulated in EAP-Double-TLS packets that are split in a collection of APDUs (A1 . . . A1p . . . An1 . . . Anq) in the form of ISO7816 commands. Afterwards, the APDUs (each APDUs size is around 240 bytes) are forwarded to the EAP-Double-TLS smartcard. Note that for each APDU received by the smartcard, an APDU response, with 2 bytes of data, is generated to inform the supplicant of the APDUs status (if the APDU was arrived and correctly processed or no).

<table>
<thead>
<tr>
<th>EAP-Double-TLS</th>
<th>Supplicant Smartcard interface</th>
<th>Authentication server</th>
</tr>
</thead>
<tbody>
<tr>
<td>+---------------+-------------------------------+----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TLS</td>
<td>EAP-Double-TLS</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------</td>
</tr>
</tbody>
</table>

Send: TLS message M1 = E1 .. En
EAP-Double-TLS:
E1 <= 1400 octets
<-- Frag E1 = A11 .. A1p

<-- APDU : Frag A11
(<= 240 octets)

APDU -->
Ack A11
.
.
.

<-- APDU : Frag A1p
(<= 240 octets)

APDU -->
Ack A1p

Ack E1 -->
However, for the smartcard part and in order to prevent multiple segmentation and re-assembly operations, the maximum EAP message length of a no fragmented packet SHALL be set to 240 bytes. For a fragmented EAP message, the maximum length value shall be 240 bytes.

As defined in EAP-TLS, when the EAP-Double-TLS smartcard receives an
EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-TLS and no data. This serves as a fragment ACK.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

Information on the IETF’s procedures with respect to rights in IETF 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.

Disclaimer of Validity

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 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.

Copyright Statement

Copyright (C) The Internet Society (2004). 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.

Acknowledgment

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