EAP Working Group
Internet-Draft
Obsoletes: 2284 (if approved)
Expires: July 2, 2003

L. Blunk
Merit Network, Inc
J. Vollbrecht
Vollbrecht Consulting LLC
B. Aboba
Microsoft
J. Carlson
Sun
H. Levkowetz, Ed.
ipUnplugged
January 2003

Extensible Authentication Protocol (EAP)
<draft-ietf-eap-rfc2284bis-02.txt>

Status of this Memo

Copyright Notice

Abstract

This document defines the Extensible Authentication Protocol (EAP), an authentication framework which supports multiple authentication mechanisms. EAP typically runs directly over the link layer without requiring IP, but is reliant on lower layer ordering guarantees as in PPP and IEEE 802. EAP does provide its own support for duplicate

elimination and retransmission. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this. While EAP was originally developed for use with PPP, it is also now in use with IEEE 802.

This document obsoletes RFC 2284. A summary of the changes between this document and RFC 2284 is available in Appendix B.

Table of Contents

1. Introduction

This document defines the Extensible Authentication Protocol (EAP), an authentication framework which supports multiple authentication mechanisms. EAP typically runs directly over the link layer without requiring IP, but is reliant on lower layer ordering guarantees as in PPP and IEEE 802. EAP does provide its own support for duplicate elimination and retransmission. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this.

EAP may be used on dedicated links as well as switched circuits, and wired as well as wireless links. To date, EAP has been implemented with hosts and routers that connect via switched circuits or dial-up lines using PPP [RFC1661]. It has also been implemented with switches and access points using IEEE 802 [IEEE.802.1990]. EAP encapsulation on IEEE 802 wired media is described in [IEEE.802-1X.2001].

One of the advantages of the EAP architecture is its flexibility. EAP is used to select a specific authentication mechanism, typically after the authenticator requests more information in order to determine the specific authentication mechanism(s) to be used. Rather than requiring the authenticator to be updated to support each new authentication method, EAP permits the use of a backend authentication server which may implement some or all authentication methods, with the authenticator acting as a pass-through for some or all methods and users.

Within this document, authenticator requirements apply regardless of whether the authenticator is operating as a pass-through or not. Where the requirement is meant to apply to either the authenticator or backend authentication server, depending on where the EAP authentication is terminated, the term "EAP server" will be used.

1.1 Specification of Requirements

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

1.2 Terminology

This document frequently uses the following terms:

authenticator
: The end of the EAP link initiating the EAP authentication methods. [Note: This terminology is also used in

[IEEE.802-1X.2001], and has the same meaning in this document].

peer
: The end of the EAP Link that responds to the authenticator. [Note:In [IEEE.802-1X.2001], 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.2001].]

Displayable Message
: This is interpreted to be a human readable string of characters, and MUST NOT affect operation of the protocol. The message encoding MUST follow the UTF-8 transformation format [RFC2279].

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.

Silently Discard
: This means the implementation discards the packet without further processing. The implementation SHOULD provide the capability of logging the event, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.

Security claims terminology for EAP Methods (see Section 7.2):

Mutual authentication
: This refers to an EAP method in which, within an interlocked exchange, the authenticator authenticates the peer and the peer authenticates the authenticator. Two independent one-way methods, running in opposite directions do not provide mutual authentication as defined here.

Integrity protection
: This refers to providing data origin authentication and protection against unauthorized modification of information

for EAP packets (including EAP Requests and Responses). When making this claim, a method specification MUST describe the EAP packets and fields within the EAP packet that are protected.

Replay protection
: This refers to protection against replay of EAP messages, including EAP Requests and Responses, and method-specific success and failure indications.

Confidentiality
: This refers to encryption of EAP messages, including EAP Requests and Responses, and method-specific success and failure indications. A method making this claim MUST support identity protection.

Key derivation
: This refers to the ability of the EAP method to derive a Master Key which is not exported, as well as a ciphersuite-independent Master Session Keys. Both the Master Key and Master Session Keys are used only for further key derivation, not directly for protection of the EAP conversation or subsequent data.

Key strength
: If the effective key strength is N bits, the best currently known methods to recover the key (with non-negligible probability) require an effort comparable to 2^N operations of a typical block cipher.

Dictionary attack resistance
: Where password authentication is used, users are notoriously prone to select poor passwords. A method may be said to be dictionary attack resistant if, when there is a weak password in the secret, the method does not allow an attack more efficient than brute force.

Fast reconnect
: The ability, in the case where a security association has been previously established, to create a new or refreshed security association in a smaller number of round-trips.

Man-in-the-Middle resistance
: The ability for the peer to demonstrate to the authenticator that it has acted as the peer for each method within the conversation. Similarly, the authenticator demonstrates to the peer that it has acted as the authenticator for each method within the conversation. If

this is not possible, then the authentication sequence or tunnel may be vulnerable to a man-in-the-middle attack.

Acknowledged result indications
: The ability of the authenticator to provide the peer with an indication of whether the peer has successfully authenticated to it, and for the peer to acknowledge receipt, as well as providing an indication of whether the authenticator has successfully authenticated to the peer. Since EAP Success and Failure packets are neither acknowledged nor integrity protected, this claim requires implementation of a method- specific result exchange that is integrity protected.

2. Extensible Authentication Protocol (EAP)

The EAP authentication exchange proceeds as follows:

[1] The authenticator sends a Request to authenticate the peer. The Request has a type field to indicate what is being requested. Examples of Request types include Identity, MD5-challenge, etc. The MD5-challenge type corresponds closely to the CHAP authentication protocol [RFC1994]. Typically, the authenticator will send an initial Identity Request; however, an initial Identity Request is not required, and MAY be bypassed. For example, the identity may not be required where it is determined by the port to which the peer has connected (leased lines, dedicated switch or dial-up ports); or where the identity is obtained in another fashion (via calling station identity or MAC address, in the Name field of the MD5-Challenge Response, etc.).

[2] The peer sends a Response packet in reply to a valid Request. As with the Request packet the Response packet contains a Type field, which corresponds to the Type field of the Request.

[3] The authenticator sends an additional Request packet, and the peer replies with a Response. The sequence of Requests and Responses continues as long as needed. EAP is a 'lock step' protocol, so that other than the initial Request, a new Request cannot be sent prior to receiving a valid Response. The Authenticator MUST NOT send a Success or Failure packet as a result of a timeout. After a suitable number of timeouts have elapsed, the Authenticator SHOULD end the EAP conversation.

[4] The conversation continues until the authenticator cannot authenticate the peer (unacceptable Responses to one or more Requests), in which case the authenticator implementation MUST

transmit an EAP Failure (Code 4). Alternatively, the authentication conversation can continue until the authenticator determines that successful authentication has occurred, in which case the authenticator MUST transmit an EAP Success (Code 3).

Since EAP is a peer-to-peer protocol, an independent and simultaneous authentication may take place in the reverse direction. Both peers may act as authenticators and authenticatees at the same time.

Advantages

The EAP protocol can support multiple authentication mechanisms without having to pre-negotiate a particular one.

Devices (e.g. a NAS, switch or access point) do not have to understand each authentication method and MAY act as a pass-through agent for a backend authentication server. Support for pass-through is optional. An authenticator MAY authenticate local users while at the same time acting as a pass-through for non-local users and authentication methods it does not implement locally.

For sessions in which the authenticator acts as a pass-through, it MUST determine the outcome of the authentication solely based on the Accept/Reject indication sent by the backend authentication server; the outcome MUST NOT be determined by the contents of an EAP packet sent along with the Accept/Reject indication, or the absence of such an encapsulated EAP packet.

Separation of the authenticator from the backend authentication server simplifies credentials management and policy decision making.

Disadvantages

For use in PPP, EAP does require the addition of a new authentication type to PPP LCP and thus PPP implementations will need to be modified to use it. It also strays from the previous PPP authentication model of negotiating a specific authentication mechanism during LCP. Similarly, switch or access point implementations need to support [IEEE.802-1X.2001] in order to use EAP.

Where the authenticator is separate from the backend authentication server, this complicates the security analysis and, if needed, key distribution.

2.1 Support for sequences

An EAP conversation MAY utilize a sequence of methods. A common example of this is an Identity request followed by a single EAP authentication method such as an MD5-Challenge. However a peer MUST utilize only one authentication method (Type 4 or greater) within an EAP conversation, after which the authenticator MUST send a Success or Failure packet. As a result, Identity Requery is not supported.

Once a peer has sent a Response of the same Type as the initial Request, an authenticator MUST NOT send a Request of a different Type prior to completion of the final round of a given method (with the exception of a Notification-Request) and MUST NOT send a Request for an additional method of any Type after completion of the initial authentication method.

Supporting multiple authentication methods within an EAP conversation would add complexity to the EAP protocol, would enable man-in-the-middle attacks (see Section 7.4), and would result in interoperability problems, since existing EAP implementations typically do not support multiple authentication methods.

If an additional authentication method is requested by the authenticator, or if the authenticator sends a Request of a different Type prior to completion of the final round of a given method, the peer SHOULD silently discard the Request. A peer MUST NOT send a Nak (legacy or expanded) in reply to a Request, after an initial non-Nak Response has been sent. Since spoofed EAP Request packets may be sent by an attacker, an an authenticator receiving an unexpected Nak SHOULD silently discard it and log the event.

Where a single EAP authentication method is utilized, but other methods are run within it (e.g. "tunneled" methods) the prohibition against multiple authentication methods does not apply. Such "tunneled" methods appear as a single authentication method to EAP. Backward compatibility can be provided, since a peer not supporting a "tunneled" method can reply to the initial EAP-Request with a Nak. To address security vulnerabilities, "tunneled" methods MUST support protection against man-in-the-middle attacks.

Within or associated with each authenticator, it is not anticipated that a particular named peer will support a choice of methods. This would make the peer vulnerable to attacks that negotiate the least secure method from among a set (negotiation attacks, described in Section 7.8). Instead, for each named peer there SHOULD be an indication of exactly one method used to authenticate that peer name. If a peer needs to make use of different authentication methods under different circumstances, then distinct identities SHOULD be employed,

each of which identifies exactly one authentication method.

2.2 EAP multiplexing model

Conceptually, EAP implementations consist of the following components:

[a] Lower layer. The lower layer is responsible for transmitting and receiving EAP frames between the peer and authenticator. EAP has been run over a variety of lower layers including PPP; wired IEEE 802 LANs [IEEE.802-1X.2001]; IEEE 802.11 wireless LANs [IEEE.802-11.1999]; UDP (L2TP [RFC2661] and ISAKMP [PIC]); and TCP [PIC]. Lower layer behavior is discussed in Section 3.

[b] EAP layer. The EAP layer receives and transmits EAP packets via the lower layer, implements the EAP state machine, and delivers and receives EAP messages to and from EAP methods.

[c] EAP method. EAP methods implement the authentication algorithms and receive and transmit EAP messages via the EAP layer. Since fragmentation support is not provided by EAP itself, this is the responsibility of EAP methods, which are discussed in Section 5.

The EAP multiplexing model is illustrated in figure 1 below. Note that there is no requirement that an implementation conform to this model, as long as the on-the-wire behavior is consistent with it.

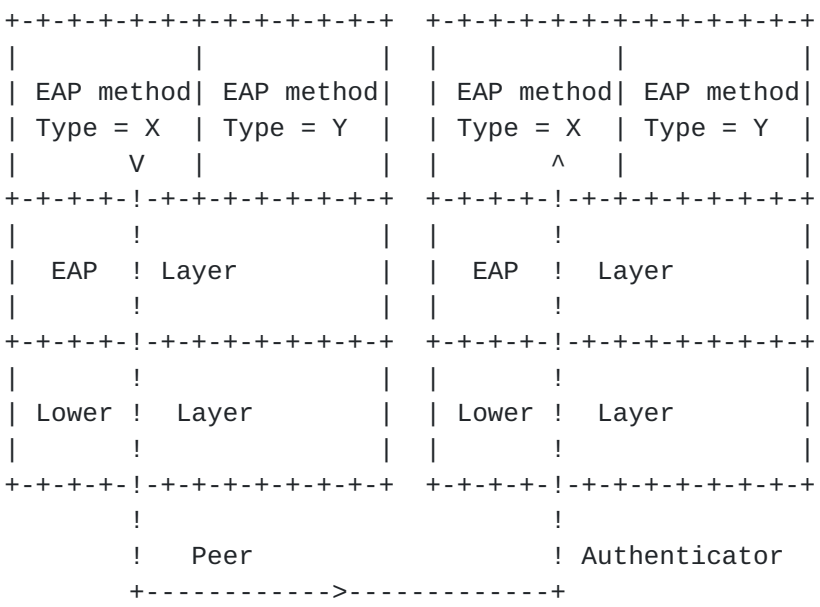


Figure 1: EAP Multiplexing Model

Within EAP, the Type field functions much like a port number in UDP or TCP. With the exception of Types handled by the EAP layer, it is assumed that the EAP layer multiplexes incoming EAP packets according to their Type, and delivers them only to the EAP method corresponding to that Type code, with one exception.

Since EAP methods may wish to access the Identity, the Identity Response can be assumed to be stored within the EAP layer so as to be available to methods of Types other than 1 (Identity). The Identity Type is discussed in Section 5.1.

A Notification Response is only used as confirmation that the peer received the Notification Request, not that it has processed it, or displayed the message to the user. It cannot be assumed that the contents of the Notification Request or Response is available to another method. The Notification Type is discussed in Section 5.2.

The Nak method is utilized for the purposes of method negotiation. Peers MUST respond to an EAP Request for an unacceptable Type with a Nak Response (legacy or expanded). It cannot be assumed that the contents of the Nak Response is available to another method. The Nak Type is discussed in Section 5.3.

EAP packets with codes of Success or Failure do not include a Type, and therefore are not delivered to an EAP method. Success and Failure are discussed in Section 4.2.

Given these considerations, the Success, Failure, Nak Response and

Notification Request/Response messages MUST NOT be used to carry data destined for delivery to other EAP methods.

Where a pass-through authenticator is present, it forwards packets back and forth between the peer and a backend authentication server, based on the EAP layer header fields (Code, Identifier, Length). Since pass-through authenticators rely on a backend authenticator server to implement methods, the EAP method layer header fields (Type, Type-Data) are not examined as part of the forwarding decision. The forwarding model is illustrated in Figure 2. Compliant pass-through authenticator implementations MUST by default be capable of forwarding packets from any EAP method.

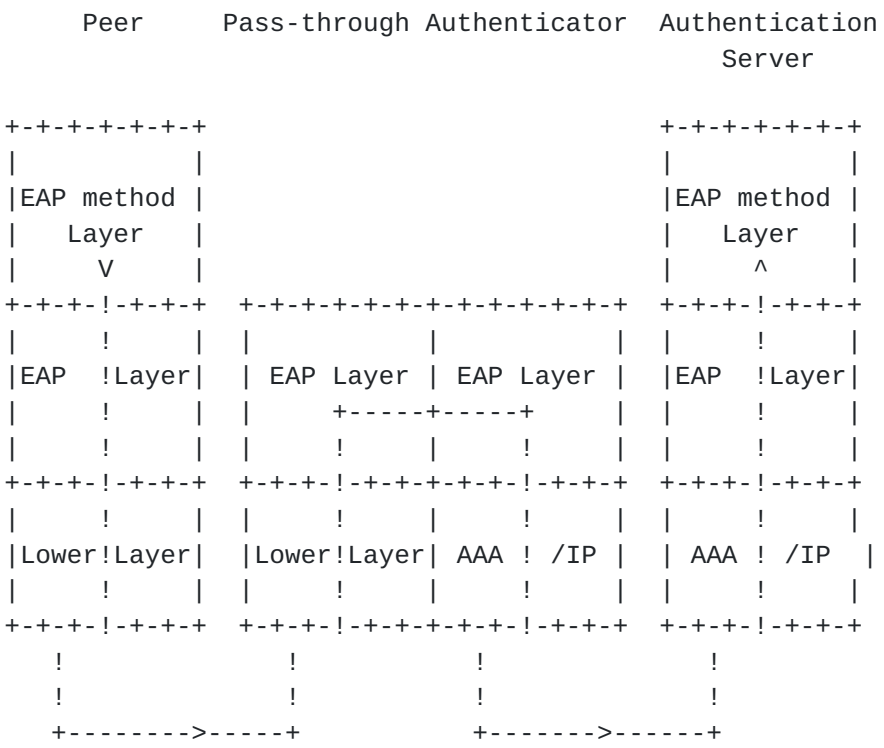


Figure 2: Pass-through Authenticator

3. Lower layer behavior

3.1 Lower layer requirements

EAP makes the following assumptions about lower layers:

[1] Lower layer CRC or checksum is not necessary. In EAP, the authenticator retransmits Requests that have not yet received Responses, so that EAP does not assume that lower layers are

reliable. Since EAP defines its own retransmission behavior, when run over a reliable lower layer, it is possible (though undesirable) for retransmission to occur both in the lower layer and the EAP layer.

If lower layers exhibit a high loss rate, then retransmissions are likely, and since EAP Success and Failure are not retransmitted, timeouts are also likely to result. EAP methods such as EAP TLS [RFC2716] include a message integrity check (MIC) and regard MIC errors as fatal. Therefore if a checksum or CRC is not provided by the lower layer, then some methods may not behave well.

[2] Lower layer data security. After EAP authentication is complete, the peer will typically transmit data to the network, through the authenticator. In order to provide assurance that the peer transmitting data is the one that successfully completed EAP authentication, it is necessary for the lower layer to provide per- packet integrity, authentication and replay protection that is bound to the original EAP authentication, or for the lower layer to be physically secure. Otherwise it is possible for subsequent data traffic to be hijacked, or replayed.

As a result of these considerations, EAP SHOULD be used only when lower layers provide physical security for data (e.g. wired PPP or IEEE 802 links), or for insecure links, where per-packet authentication, integrity and replay protection is provided. Where keying material for the lower layer ciphersuite is itself provided by EAP, typically the lower layer ciphersuite cannot be enabled until late in the EAP conversation, after key derivation has completed. Thus it may only be possible to use the lower layer ciphersuite to protect a portion of the EAP conversation, such as the EAP Success or Failure packet.

[3] Known MTU. The EAP layer does not support fragmentation and reassembly. However, EAP methods SHOULD be capable of handling fragmentation and reassembly. As a result, EAP is capable of functioning across a range of MTU sizes, as long as the MTU is known.

[4] Possible duplication. Where the lower layer is reliable, it will provide the EAP layer with a non-duplicated stream of packets. However, while it is desirable that lower layers provide for non-duplication, this is not a requirement. The Identifier field provides both the peer and authenticator with the ability to detect duplicates.

[5] Ordering guarantees. EAP does not require the Identifier to be monotonically increasing, and so is reliant on lower layer ordering guarantees for correct operation. Also, EAP was originally defined to run on PPP, and [RFC1661] Section 1 has an ordering requirement:

"The Point-to-Point Protocol is designed for simple links which transport packets between two peers. These links provide full-duplex simultaneous bi-directional operation, and are assumed to deliver packets in order."

Lower lower transports for EAP MUST preserve ordering between a source and destination, at a given priority level (the level of ordering guarantee provided by [IEEE.802.1990]).

3.2 EAP usage within PPP

In order to establish communications over a point-to-point link, each end of the PPP link must first send LCP packets to configure the data link during Link Establishment phase. After the link has been established, PPP provides for an optional Authentication phase before proceeding to the Network-Layer Protocol phase.

By default, authentication is not mandatory. If authentication of the link is desired, an implementation MUST specify the Authentication-Protocol Configuration Option during Link Establishment phase.

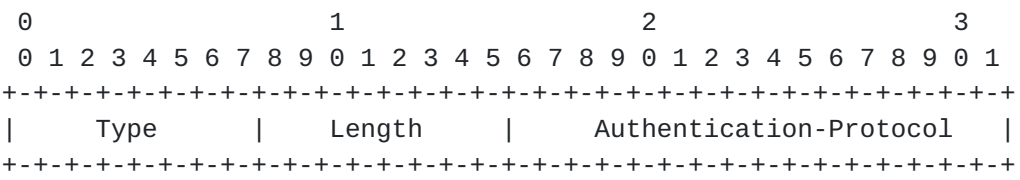
If the identity of the peer has been established in the Authentication phase, the server can use that identity in the selection of options for the following network layer negotiations.

When implemented within PPP, EAP does not select a specific authentication mechanism at PPP Link Control Phase, but rather postpones this until the Authentication Phase. This allows the authenticator to request more information before determining the specific authentication mechanism. This also permits the use of a "back-end" server which actually implements the various mechanisms while the PPP authenticator merely passes through the authentication exchange. The PPP Link Establishment and Authentication phases, and the Authentication-Protocol Configuration Option, are defined in The Point-to-Point Protocol (PPP) [RFC1661].

3.2.1 PPP Configuration Option Format

A summary of the PPP Authentication-Protocol Configuration Option format to negotiate the EAP Authentication Protocol is shown below. The fields are transmitted from left to right.

Exactly one EAP packet is encapsulated in the Information field of a PPP Data Link Layer frame where the protocol field indicates type hex C227 (PPP EAP).



Type

3

Length

4

Authentication-Protocol

C227 (Hex) for PPP Extensible Authentication Protocol (EAP)

3.3 EAP usage within IEEE 802

The encapsulation of EAP over IEEE 802 is defined in [IEEE.802-1X.2001]. The IEEE 802 encapsulation of EAP does not involve PPP, and IEEE 802.1X does not include support for link or network layer negotiations. As a result, within IEEE 802.1X it is not possible to negotiate non-EAP authentication mechanisms, such as PAP or CHAP [RFC1994].

3.4 Link layer indications

The reliability and security of link layer indications is dependent on the medium. Since EAP is media independent, the presence or absence of link layer security is not taken into account in the processing of EAP messages.

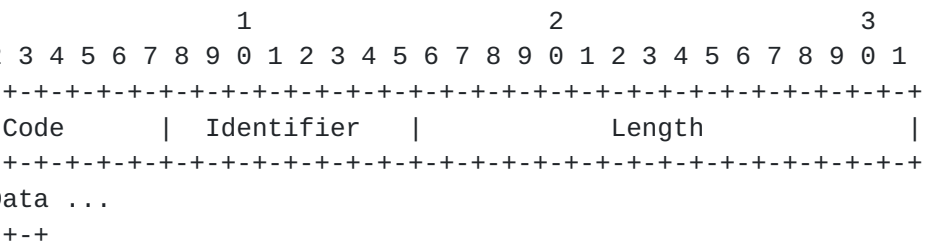
Link layer failure indications provided to EAP by the link layer MUST be processed and will cause an EAP exchange in progress to be aborted. However, link layer success indications MUST NOT affect EAP message processing so that an EAP implementation MUST NOT conclude that authentication has succeeded based on those indications. This ensures that an attacker spoofing link layer indications can at best succeed in a denial of service attack.

A discussion of some reliability and security issues with link layer indications in PPP, IEEE 802 wired networks and IEEE 802.11 wireless LANs can be found in the Security Considerations, Section 7.12.

In IEEE 802.11 a "link down" indication is an unreliable indication of link failure, since wireless signal strength can come and go and may be influenced by radio frequency interference generated by an attacker. To avoid unnecessary resets, it is advisable to damp these indications, rather than passing them directly to the EAP. Since EAP supports retransmission, it is robust against transient connectivity losses.

4. EAP Packet format

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



Code

The Code field is one octet and identifies the type of EAP packet. EAP Codes are assigned as follows:

1	Request
2	Response
3	Success
4	Failure

Since EAP only defines Codes 1-4, EAP packets with other codes MUST be silently discarded by both authenticators and peers.

Identifier

The Identifier field is one octet and aids in matching Responses with Requests.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length 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.

Data

The Data field is zero or more octets. The format of the Data field is determined by the Code field.

4.1 Request and Response

Description

The Request packet (Code field set to 1) is sent by the authenticator to the peer. Each Request has a Type field which serves to indicate what is being requested. Additional Request packets MUST be sent until a valid Response packet is received, or an optional retry counter expires.

Retransmitted Requests MUST be sent with the same Identifier value in order to distinguish them from new Requests. The contents of the data field is dependent on the Request type. The peer MUST send a Response packet in reply to a valid Request packet. Responses MUST only be sent in reply to a valid Request and never retransmitted on a timer.

The Identifier field of the Response MUST match that of the currently outstanding Request. An authenticator receiving a Response whose Identifier value does not match that of the currently outstanding Request MUST silently discard the Response. The Type field of a Response MUST either match that of the Request, or correspond to a legacy or expanded Nak (see Section 5.3).

Implementation Note: The authenticator is responsible for retransmitting Request messages. If the Request message is obtained from elsewhere (such as from a backend authentication server), then the authenticator will need to save a copy of the Request in order to accomplish this. The peer is responsible for detecting and handling duplicate Request messages before processing them in any way, including passing them on to an outside party. The authenticator is also responsible for discarding Response messages with a non-matching Identifier value before acting on them in any way, including passing them on to the backend authentication server for verification. Since the authenticator can retransmit before receiving a Response from the peer, the authenticator can receive multiple Responses, each with a matching Identifier. Until a new Request

is received by the authenticator, the Identifier value is not updated, so that the authenticator forwards Responses to the backend authentication server, one at a time.

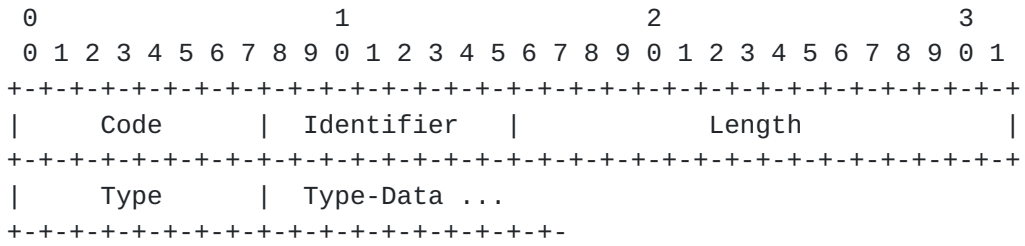
Because the authentication process will often involve user input, some care must be taken when deciding upon retransmission strategies and authentication timeouts. By default, where EAP is run over an unreliable lower layer, the EAP retransmission timer SHOULD be computed as described in [RFC2988]. This includes use of Karn's algorithm to filter RTT estimates resulting from retransmissions. A maximum of 3-5 retransmissions is suggested.

When run over a reliable lower layer (e.g. EAP over ISAKMP/TCP, as within [PIC]), the authenticator retransmission timer SHOULD be set to an infinite value, so that retransmissions do not occur at the EAP layer. Note that in this case the peer may still maintain a timeout value so as to avoid waiting indefinitely for a Request.

Where the authentication process requires user input, the measured round trip times are largely determined by user responsiveness rather than network characteristics, so that RTO estimation is not helpful. Instead, the retransmission timer SHOULD be set so as to provide sufficient time for the user to respond, with longer timeouts required in certain cases, such as where Token Cards (see Section 5.6) are involved.

In order to provide the EAP authenticator with guidance as to the appropriate timeout value, a hint can be communicated to the authenticator by the backend authentication server (such as via the RADIUS Session-Timeout attribute).

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



Code

1 for Request
2 for Response

Identifier

The Identifier field is one octet. The Identifier field MUST be the same if a Request packet is retransmitted due to a timeout while waiting for a Response. Any new (non-retransmission) Requests MUST modify the Identifier field. In order to avoid confusion between new Requests and retransmissions, the Identifier value chosen for each new Request need only be different from the previous Request, but need not be unique within the conversation. One way to achieve this is to start the Identifier at an initial value and increment it for each new Request. Initializing the first Identifier with a random number rather than starting from zero is recommended, since it makes sequence attacks somewhat harder.

Since the Identifier space is unique to each session, authenticators are not restricted to only 256 simultaneous authentication conversations. Similarly, with re-authentication, an EAP conversation might continue over a long period of time, and is not limited to only 256 roundtrips.

If a peer receives a valid duplicate Request for which it has already sent a Response, it MUST resend its original Response. If a peer receives a duplicate Request before it has sent a Response, but after it has determined the initial Request to be valid (i.e. it is waiting for user input), it MUST silently discard the duplicate Request. An EAP message may be found invalid for a variety of reasons: failed lower layer CRC or checksum, malformed EAP packet, EAP method MIC failure, etc.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and Type-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

The Type field is one octet. This field indicates the Type of Request or Response. A single Type MUST be specified for each EAP Request or Response. Normally, the Type field of the Response

will be the same as the Type of the Request. However, there is also a Nak Response Type for indicating that a Request type is unacceptable to the peer. An initial specification of Types follows in a later section of this document.

Type-Data

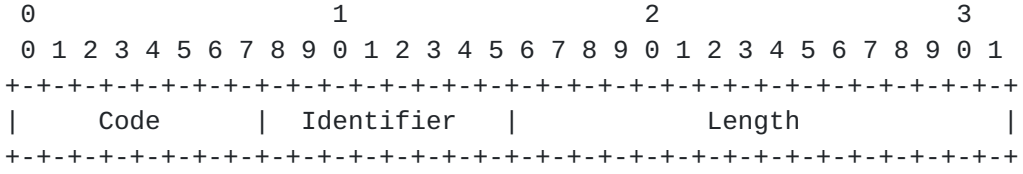
The Type-Data field varies with the Type of Request and the associated Response.

4.2 Success and Failure

The Success packet is sent by the authenticator to the peer to acknowledge successful authentication. The authenticator MUST transmit an EAP packet with the Code field set to 3 (Success). If the authenticator cannot authenticate the peer (unacceptable Responses to one or more Requests) then the implementation MUST transmit an EAP packet with the Code field set to 4 (Failure). An authenticator MAY wish to issue multiple Requests before sending a Failure response in order to allow for human typing mistakes. Success and Failure packets MUST NOT contain additional data.

Implementation Note: Because the Success and Failure packets are not acknowledged, the authenticator cannot know whether they have been received. As a result, these packets are not retransmitted by the authenticator. If acknowledged success and failure indications are desired, these MAY be implemented within individual EAP methods. Since only a single EAP authentication method is supported within an EAP conversation, a peer that successfully authenticates the authenticator MAY, in the event that an EAP Success is not received, conclude that the EAP Success packet was lost and enable the link.

A summary of the Success and Failure packet format is shown below. The fields are transmitted from left to right.



Code

3 for Success
4 for Failure

Identifier

The Identifier field is one octet and aids in matching replies to Responses. The Identifier field MUST match the Identifier field of the Response packet that it is sent in response to.

Length

4

4.2.1 Processing of success and failure

EAP Success or Failure packets MUST NOT be sent by an authenticator prior to completion of the final round of a given method. A peer EAP implementation receiving a Success or Failure packet prior to completion of the method in progress MUST silently discard it. By default, an EAP peer MUST silently discard a "canned" EAP Success message (an EAP Success message sent immediately upon connection). This ensures that a rogue authenticator will not be able to bypass mutual authentication by sending an EAP Success prior to conclusion of the EAP method conversation.

5. Initial EAP Request/Response Types

This section defines the initial set of EAP Types used in Request/Response exchanges. More Types may be defined in follow-on documents. The Type field is one octet and identifies the structure of an EAP Request or Response packet. The first 3 Types are considered special case Types.

The remaining Types define authentication exchanges. The Nak Type is valid only for Response packets, it MUST NOT be sent in a Request. The Nak Type MUST only be sent in response to a Request which uses an authentication Type code (i.e., Type of 4 or greater).

All EAP implementations MUST support Types 1-4, which are defined in this document, and SHOULD support Type 254. Follow-on RFCs MAY define additional EAP Types.

1	Identity
2	Notification
3	Nak (Response only)
4	MD5-Challenge
5	One Time Password (OTP)
6	Generic Token Card (GTC)
254	Expanded types
255	Experimental use

5.1 Identity

Description

The Identity Type is used to query the identity of the peer. Generally, the authenticator will issue this as the initial Request. An optional displayable message MAY be included to prompt the peer in the case where there expectation of interaction with a user. A Response of Type 1 (Identity) SHOULD be sent in Response to a Request with a Type of 1 (Identity).

Since Identity Requests and Responses are not protected, from a security perspective, it may be preferable for protected method-specific Identity exchanges to be used instead.

Implementation Note: The peer MAY obtain the Identity via user input. It is suggested that the authenticator retry the Identity Request in the case of an invalid Identity or authentication failure to allow for potential typos on the part of the user. It is suggested that the Identity Request be retried a minimum of 3 times before terminating the authentication phase with a Failure reply. The Notification Request MAY be used to indicate an invalid authentication attempt prior to transmitting a new Identity Request (optionally, the failure MAY be indicated within the message of the new Identity Request itself).

Type

1

Type-Data

This field MAY contain a displayable message in the Request, containing UTF-8 encoded ISO 10646 characters [RFC2279]. The Response uses this field to return the Identity. If the Identity is unknown, this field should be zero bytes in length. The field MUST NOT be null terminated. The length of this field is derived

from the Length field of the Request/Response packet and hence a null is not required.

5.2 Notification

Description

The Notification Type is optionally used to convey a displayable message from the authenticator to the peer. An authenticator MAY send a Notification Request to the peer at any time when there is no outstanding Request. The peer MUST respond to a Notification Request with a Notification Response; a Nak Response MUST NOT be sent.

The peer SHOULD display this message to the user or log it if it cannot be displayed. The Notification Type is intended to provide an acknowledged notification of some imperative nature, but it is not an error indication, and therefore does not change the state of the peer. Examples include a password with an expiration time that is about to expire, an OTP sequence integer which is nearing 0, an authentication failure warning, etc. In most circumstances, Notification should not be required.

Type

2

Type-Data

The Type-Data field in the Request contains a displayable message greater than zero octets in length, containing UTF-8 encoded ISO 10646 characters [RFC2279]. The length of the message is determined by Length field of the Request packet. The message MUST NOT be null terminated. A Response MUST be sent in reply to the Request with a Type field of 2 (Notification). The Type-Data field of the Response is zero octets in length. The Response should be sent immediately (independent of how the message is displayed or logged).

5.3 Nak

5.3.1 Legacy Nak

Description

The legacy Nak Type is valid only in Response messages. It is sent in reply to a Request where the desired authentication Type is unacceptable. Authentication Types are numbered 4 and above. The Response contains one or more authentication Types desired by the Peer. Type zero (0) is used to indicate that the sender has no viable alternatives.

Since the legacy Nak Type is valid only in Responses and has very limited functionality, it MUST NOT be used as a general purpose error indication, such as for communication of error messages, or negotiation of parameters specific to a particular EAP method.

Code

2 for Response.

Identifier

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field of a legacy Nak Response MUST match the Identifier field of the Request packet that it is sent in response to.

Length

>=6

Type

3

Type-Data

Where any peer receives a Request for an unacceptable Type in the range (1-253,255), or a peer lacking support for Expanded Types receives a Request for Type 254, a legacy Nak Response MUST be sent. The Type-Data field of the legacy Nak Response MUST contain one or more octets indicating the desired authentication Type(s), one octet per Type, or the value zero (0) to indicate no proposed alternative. A peer supporting Expanded Types that receives a Request for an unacceptable Type (1-253, 255) MAY include the value 254 in the legacy Nak Response in order to indicate the desire for an Expanded authentication Type. If the authenticator can accomodate this preference, it will respond with an Expanded Type Request.

5.3.2 Expanded Nak

Description

The Expanded Nak Type is valid only in Response messages. It MUST be sent only in reply to a Request of Type 254 (Expanded Type) where the authentication Type is unacceptable. The Expanded Nak Type uses the Expanded Type format itself, and the Response contains one or more authentication Types desired by the peer, all in Expanded Type format. Type zero (0) is used to indicate that the sender has no viable alternatives. The general format of the Expanded Type is described in Section 5.7.

Since the Expanded Nak Type is valid only in Responses and has very limited functionality, it MUST NOT be used as a general purpose error indication, such as for communication of error messages, or negotiation of parameters specific to a particular EAP method.

Code

2 for Response.

Identifier

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field of a Expanded Nak Response MUST match the Identifier field of the Request packet that it is sent in response to.

Length

>=40

Type

254

Vendor-Id

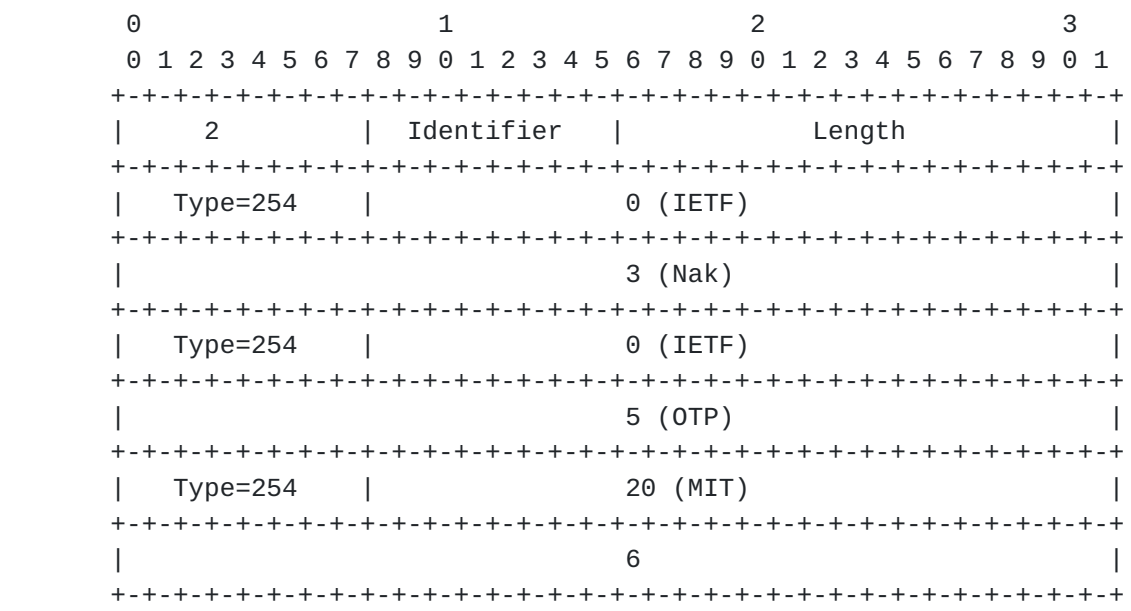
0 (IETF)

Vendor-Type

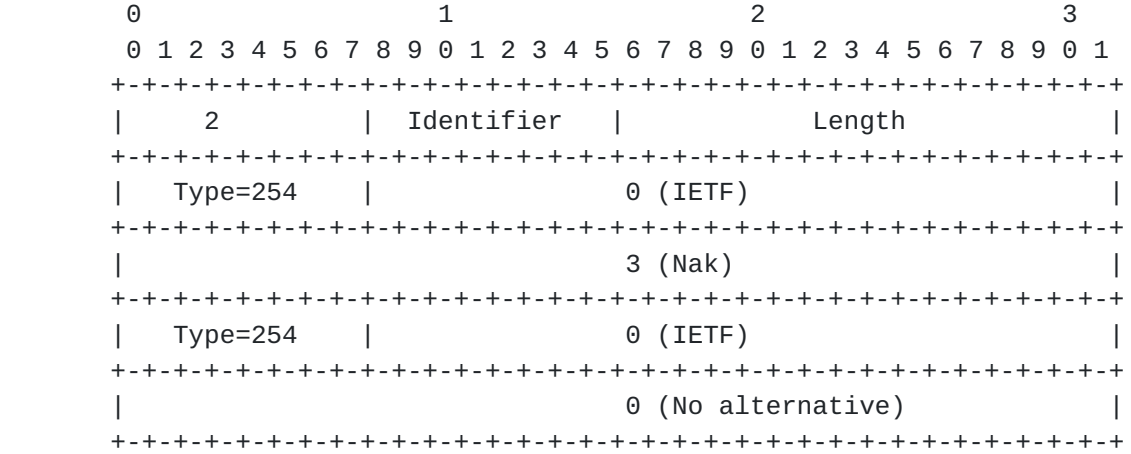
3 (Nak)

Vendor-Data

The Expanded Nak Type is only sent when the Request contains an Expanded Type (254) as defined in Section 5.7. The Vendor-Data field of the Nak Response MUST contain one or more authentication Types (4 or greater), all in expanded format, 8 octets per Type, or the value zero (0), also in Expanded Type format, to indicate no proposed alternative. The desired authentication Types may include a mixture of Vendor-Specific and IETF Types. For example, an Expanded Nak Response indicating a preference for OTP (Type 5), and an MIT (Vendor-Id=20) Expanded Type of 6 would appear as follows:



An Expanded Nak Response indicating a no desired alternative would appear as follows:



5.4 MD5-Challenge

Description

The MD5-Challenge Type is analogous to the PPP CHAP protocol [RFC1994] (with MD5 as the specified algorithm). The Request contains a "challenge" message to the peer. A Response MUST be sent in reply to the Request. The Response MAY be either of Type 4 (MD5-Challenge) or Type 3 (Nak). The Nak reply indicates the peer's desired authentication Type(s). EAP peer and EAP server implementations MUST support the MD5-Challenge mechanism. An authenticator that supports only pass-through MUST allow communication with a backend authentication server that is capable of supporting MD5-Challenge, although the EAP authenticator implementation need not support MD5-Challenge itself. However, if the EAP authenticator can be configured to authenticate peers locally (e.g. not operate in pass-through), then the requirement for support of the MD5-Challenge mechanism applies.

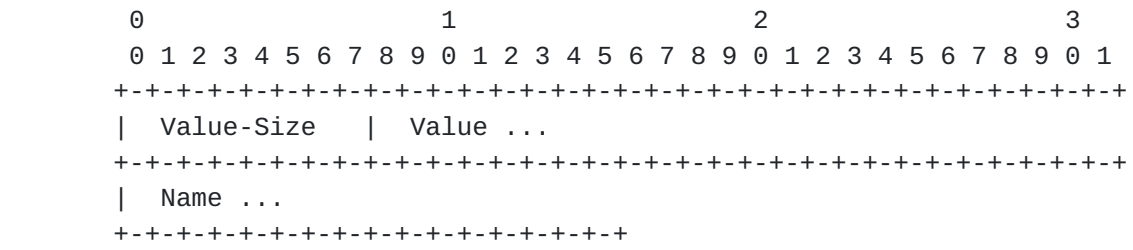
Note that the use of the Identifier field in the MD5-Challenge Type is different from that described in [RFC1994]. EAP allows for retransmission of MD5-Challenge Request packets while [RFC1994] states that both the Identifier and Challenge fields MUST change each time a Challenge (the CHAP equivalent of the MD5-Challenge Request packet) is sent.

Type

4

Type-Data

The contents of the Type-Data field is summarized below. For reference on the use of these fields see the PPP Challenge Handshake Authentication Protocol [RFC1994].



Security Claims (see Section 7.2):

Intended use:	Wired networks, including PPP, PPPOE, and IEEE 802 wired media. Use over the Internet or with wireless media only when protected.
Mechanism:	Password or pre-shared key.
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key Derivation:	No
Key strength:	N/A
Dictionary attack prot:	No
Key hierarchy:	N/A
Fast reconnect:	No
MiTM resistance:	No
Acknowledged S/F:	No

5.5 One-Time Password (OTP)

Description

The One-Time Password system is defined in "A One-Time Password System" [RFC2289] and "OTP Extended Responses" [RFC2243]. The Request contains a displayable message containing an OTP challenge. A Response MUST be sent in reply to the Request. The Response MUST be of Type 5 (OTP) or Type 3 (Nak). The Nak Response indicates the peer's desired authentication Type(s).

Type

5

Type-Data

The Type-Data field contains the OTP "challenge" as a displayable message in the Request. In the Response, this field is used for the 6 words from the OTP dictionary [RFC2289]. The messages MUST NOT be null terminated. The length of the field is derived from the Length field of the Request/Reply packet.

Security Claims (see Section 7.2):

Intended use:	Wired networks, including PPP, PPPOE, and IEEE 802 wired media. Use over the Internet or with wireless media only when protected.
Mechanism:	One-Time Password
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key Derivation:	No
Key strength:	N/A
Dictionary attack prot:	No
Key hierarchy:	N/A
Fast reconnect:	No
MiTM resistance:	No
Acknowledged S/F:	No

5.6 Generic Token Card (GTC)

Description

The Generic Token Card Type is defined for use with various Token Card implementations which require user input. The Request contains a displayable message and the Response contains the Token Card information necessary for authentication. Typically, this would be information read by a user from the Token card device and entered as ASCII text. A Response MUST be sent in reply to the Request. The Response MUST be of Type 6 (GTC) or Type 3 (Nak). The Nak Response indicates the peer's desired authentication Type(s).

Type

6

Type-Data

The Type-Data field in the Request contains a displayable message greater than zero octets in length. The length of the message is determined by the Length field of the Request packet. The message MUST NOT be null terminated. A Response MUST be sent in reply to the Request with a Type field of 6 (Generic Token Card). The Response contains data from the Token Card required for authentication. The length of the data is determined by the Length field of the Response packet.

Security Claims (see Section 7.2):

Intended use:	Wired networks, including PPP, PPPOE, and IEEE 802 wired media. Use over the Internet or with wireless media only when protected.
Mechanism:	Hardware token.
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key Derivation:	No
Key strength:	N/A
Dictionary attack prot:	No
Key hierarchy:	N/A
Fast reconnect:	No
MiTM resistance:	No
Acknowledged S/F:	No

5.7 Expanded types

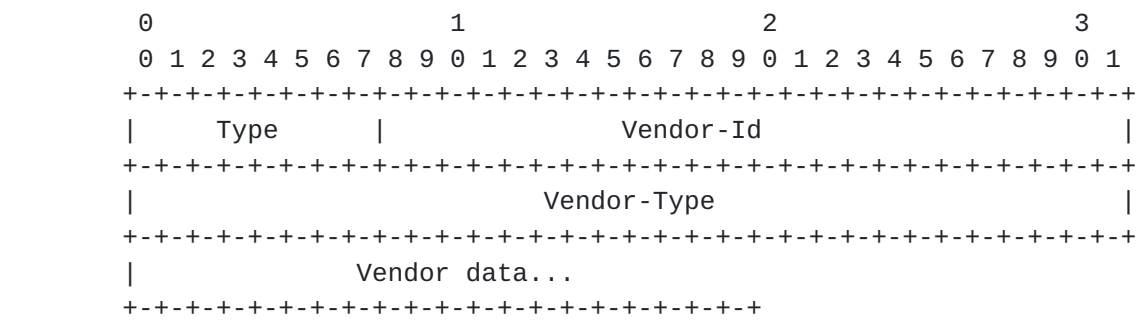
Description

Due to EAP's popularity, the original Method Type space, which only provides for 255 values, is being allocated at a pace which if continued would result in exhaustion within a few years. Since many of the existing uses of EAP are vendor-specific, the Expanded Method Type is available to allow vendors to support their own Expanded Types not suitable for general usage.

The Expanded type is also used to expand the global Method Type space beyond the original 255 values. A Vendor-Id of 0 maps the original 255 possible types onto a namespace of 2^32-1 possible types, allowing for virtually unlimited expansion. (Type 0 is only used in a Nak Response, to indicate no acceptable alternative)

An implementation that supports the Expanded attribute MUST treat EAP types that are less than 256 equivalently whether they appear as a single octet or as the 32-bit Vendor-Type within a Expanded type where Vendor-Id is 0. Peers not equipped to interpret the Expanded Type MUST send a Nak as described in Section 5.3.1, and negotiate a more suitable authentication method.

A summary of the Expanded Type format is shown below. The fields are transmitted from left to right.



Type

254 for Expanded type

Vendor-Id

The Vendor-Id is 3 octets and represents the SMI Network Management Private Enterprise Code of the Vendor in network byte order, as allocated by IANA. A Vendor-Id of zero is reserved for use by the IETF in providing an expanded global EAP Type space.

Vendor-Type

The Vendor-Type field is four octets and represents the vendor-specific Method Type.

If Vendor-Id is zero, the Vendor-Type field is an extension and superset of the existing namespace for EAP types. The first 256 types are reserved for compatibility with single-octet EAP types that have already been assigned or may be assigned in the future. Thus, EAP types from 0 through 255 are semantically identical whether they appear as single octet EAP types or as Vendor-Types when Vendor-Id is zero.

Vendor-Data

The Vendor-Data field is defined by the vendor. Where a Vendor-Id of zero is present, the Vendor-Data field will be used for transporting the contents of EAP methods of Types defined by the IETF.

5.8 Experimental

Description

The experimental type has no fixed format or content. It is intended for use when experimenting with new EAP types. This type

is intended for experimental and testing purposes. No guarantee is made for interoperability between peers using this type, as outlined in [IANA-EXP].

Type

255

Type-Data

Undefined

6. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the EAP protocol, in accordance with BCP 26, [RFC2434].

There are two name spaces in EAP that require registration: Packet Codes and Method Types.

EAP is not intended as a general-purpose protocol, and allocations SHOULD NOT be made for purposes unrelated to authentication.

6.1 Definition of Terms

The following terms are used here with the meanings defined in BCP 26: "name space", "assigned value", "registration".

The following policies are used here with the meanings defined in BCP 26: "Private Use", "First Come First Served", "Expert Review", "Specification Required", "IETF Consensus", "Standards Action".

6.2 Recommended Registration Policies

For registration requests where a Designated Expert should be consulted, the responsible IESG area director should appoint the Designated Expert. For Designated Expert with Specification Required, the request is posted to the EAP WG mailing list (or, if it has been disbanded, a successor designated by the Area Director) for comment and review, and MUST include a pointer to a public specification. Before a period of 30 days has passed, the Designated Expert will either approve or deny the registration request and publish a notice of the decision to the EAP WG mailing list or its successor. A denial notice must be justified by an explanation and, in the cases where it is possible, concrete suggestions on how the request can be modified so as to become acceptable.

For registration requests requiring Expert Review, the EAP mailing list should be consulted. If the EAP mailing list is no longer operational, an alternative mailing list may be designated by the responsible IESG Area Director.

Packet Codes have a range from 1 to 255, of which 1-4 have been allocated. Because a new Packet Code has considerable impact on interoperability, a new Packet Code requires Standards Action, and should be allocated starting at 5.

The original EAP Method Type space has a range from 1 to 255, and is the scarcest resource in EAP, and thus must be allocated with care. Method Types 1-36 have been allocated, with 20 available for re-use. Method Types 37-191 may be allocated on the advice of a Designated Expert, with Specification Required.

Allocation of blocks of Method Types (more than one for a given purpose) should require IETF Consensus. EAP Type Values 192-253 are reserved and allocation requires Standards Action.

Method Type 254 is allocated for the Expanded Type. Where the Vendor-Id field is non-zero, the Expanded Type is used for functions specific only to one vendor's implementation of EAP, where no interoperability is deemed useful. When used with a Vendor-Id of zero, Method Type 254 can also be used to provide for an expanded IETF Method Type space. Method Type values 256-4294967295 may be allocated after Type values 1-191 have been allocated.

Method Type 255 is allocated for Experimental use, such as testing of new EAP methods before a permanent Type code is allocated.

7. Security Considerations

EAP was designed for use with dialup PPP [RFC1661] and was later adapted for use in wired IEEE 802 networks [IEEE.802.1990] in [IEEE.802-1X.2001]. On these networks, an attacker would need to gain physical access to the telephone or switch infrastructure in order to mount an attack. While such attacks have been documented, such as in [DECEPTION], they are assumed to be rare.

However, subsequently EAP has been proposed for use on wireless networks, and over the Internet, where physical security cannot be assumed. On such networks, the security vulnerabilities are greater, as are the requirements for EAP security.

This section defines the threat model and security terms and describes the security claims section required in EAP method specifications. We then discuss threat mitigation.

7.1 Threat model

On physically insecure networks, it is possible for an attacker to gain access to the physical medium. This enables a range of attacks, including the following:

[1] An adversary may try to discover user identities by snooping authentication traffic.

[2] An adversary may try to modify or spoof EAP packets.

[3] An adversary may launch denial of service attacks by spoofing layer 2 indications or EAP layer success/failure indications, replaying EAP packets, or generating packets with overlapping Identifiers.

[4] An adversary might attempt to recover the pass-phrase by mounting an offline dictionary attack.

[5] An adversary may attempt to convince the peer to connect to an untrusted network, by mounting a man-in-the-middle attack.

[6] An adversary may attempt to disrupt the EAP negotiation in order to weaken the authentication.

[7] An attacker may attempt to recover the key by taking advantage of weak key derivation techniques used within EAP methods.

[8] An attacker may attempt to take advantage of weak ciphersuites subsequently used after the EAP conversation is complete.

Where EAP is used over wireless networks, an attacker needs to be within the coverage area of the wireless medium in order to carry out these attacks. However, where EAP is used over the Internet, no such restrictions apply.

7.2 Security claims

In order to clearly articulate the security provided by an EAP method, EAP method specifications MUST include a Security Claims section including the following declarations:

[a] Intended use. This includes a statement of whether the method is intended for use over a physically secure or insecure network, as well as a statement of the applicable media.

[b] Mechanism. This is a statement of the authentication technology: certificates, pre-shared keys, passwords, token cards, etc.

[c] Security claims. This is a statement of the claimed security properties of the method, using terms defined in Section 1.2: mutual authentication, integrity protection, replay protection, confidentiality, key derivation, key strength, dictionary attack resistance, fast reconnect, man-in-the-middle resistance, acknowledged result indications. The Security Claims section of an EAP method specification SHOULD provide justification for the claims that are made. This can be accomplished by including a proof in an Appendix, or including a reference to a proof.

[d] Key strength. If the method derives keys, then the effective key strength MUST be estimated. This estimate is meant for potential users of the method to determine if the keys produced are strong enough for the intended application.

The effective key strength SHOULD be stated as number of bits, defined as follows: If the effective key strength is N bits, the best currently known methods to recover the key (with non-negligible probability) require an effort comparable to 2^N operations of a typical block cipher. The statement SHOULD be accompanied by a short rationale, explaining how this number was arrived at. This explanation SHOULD include the parameters required to achieve N bits of entropy based on current knowledge of the algorithms.

(Note: Although it is difficult to define what "comparable effort" and "typical block cipher" exactly mean, reasonable approximations are sufficient here. Refer to e.g. [SILVERMAN] for more discussion.)

The key strength depends on the methods used to derive the keys. For instance, if keys are derived from a shared secret (such as a password or master key), and possibly some public information such as nonces, the effective key strength is limited by the entropy of the long-term secret (assuming that the derivation procedure is computationally simple). To take another example, when using public key algorithms, the strength of the symmetric key depends on the strength of the public keys used.

[e] Description of key hierarchy. EAP methods deriving keys MUST either provide a reference to a key hierarchy specification, or describe how keys used for authentication/integrity, encryption and IVs are to be derived from the provided keying material, and how cryptographic separation is maintained between keys used for different purposes.

[f] Indication of vulnerabilities. In addition to the security claims that are made, the specification MUST indicate which of the security claims detailed in Section 1.2 are NOT being made.

7.3 Identity protection

An Identity exchange is optional within the EAP conversation. Therefore, it is possible to omit the Identity exchange entirely, or to postpone it until later in the conversation once a protected channel has been established.

However, where roaming is supported as described in [RFC2607], it may be necessary to locate the appropriate backend authentication server before the authentication conversation can proceed. The realm portion of the Network Access Identifier (NAI) [RFC2486] is typically included within the Identity-Response in order to enable the authentication exchange to be routed to the appropriate backend authentication server. Therefore while the peer-name portion of the NAI may be omitted in the Identity- Response, where proxies or relays are present, the realm portion may be required.

7.4 Man-in-the-middle attacks

Where a sequence of methods is utilized for authentication or EAP is tunneled within another protocol that omits peer authentication, there exists a potential vulnerability to man-in-the-middle attack.

Where a sequence of EAP methods is utilized for authentication, the peer might not have proof that a single entity has acted as the authenticator for all EAP methods within the sequence. For example, an authenticator might terminate one EAP method, then forward the next method in the sequence to another party without the peer's knowledge or consent. Similarly, the authenticator might not have proof that a single entity has acted as the peer for all EAP methods within the sequence.

This enables an attack by a rogue EAP authenticator tunneling EAP to a legitimate server. Where the tunneling protocol is used for key establishment but does not require peer authentication, an attacker convincing a legitimate peer to connect to it will be able to tunnel EAP packets to a legitimate server, successfully authenticating and obtaining the key. This allows the attacker to successfully establish itself as a man-in-the-middle, gaining access to the network, as well as the ability to decrypt data traffic between the legitimate peer and server.

This attack may be mitigated by the following measures:

[a] Requiring mutual authentication within EAP tunneling mechanisms.

[b] Requiring cryptographic binding between EAP methods executed within a sequence or between the EAP tunneling protocol and the tunneled EAP methods. Where cryptographic binding is supported, a mechanism is also needed to protect against downgrade attacks that would bypass it.

[c] Limiting the EAP methods authorized for use without protection, based on peer and authenticator policy.

[d] Avoiding the use of sequences or tunnels when a single, strong method is available.

7.5 Packet modification attacks

While individual EAP methods may support per-packet data origin authentication, integrity and replay protection, EAP itself does not provide built-in support for this.

Since the Identifier is only a single octet, it is easy to guess, allowing an attacker to successfully inject or replay EAP packets. An attacker may also modify EAP headers within EAP packets where the header is unprotected. This could cause packets to be inappropriately discarded or misinterpreted.

In the case of PPP and IEEE 802 wired links, it is assumed that such attacks are restricted to attackers who can gain access to the physical link. However, where EAP is run over physically insecure lower layers such as IEEE 802.11 or the Internet (such as within protocols supporting PPP, EAP or Ethernet Tunneling), this assumption is no longer valid and the vulnerability to attack is greater.

To protect EAP messages sent over physically insecure lower layers, methods providing mutual authentication and key derivation, as well as per-packet origin authentication, integrity and replay protection SHOULD be used. Method-specific MICs may be used to provide protection. Since EAP messages of Types Identity, Notification, and Nak do not include their own MIC, it may be desirable for the EAP method MIC to cover information contained within these messages, as well as the header of each EAP message. To provide protection, EAP also may be encapsulated within a protected channel created by protocols such as ISAKMP [RFC2408] as is done in [PIC] or within TLS [RFC2246]. However, as noted in Section 7.4, EAP tunneling may result in a man-in-the-middle vulnerability.

7.6 Dictionary attacks

Password authentication algorithms such as EAP-MD5, MS-CHAPv1 [RFC2433] and Kerberos V [RFC1510] are known to be vulnerable to dictionary attacks. MS-CHAPv1 vulnerabilities are documented in [PPTPv1]; Kerberos vulnerabilities are described in [KRBATTACK], [KRBLIM], and [KERB4WEAK].

In order to protect against dictionary attacks, an authentication algorithm resistant to dictionary attack (as defined in Section 7.2) may be used. This is particularly important when EAP runs over media which are not physically secure.

If an authentication algorithm is used that is known to be vulnerable to dictionary attack, then the conversation may be tunneled within a protected channel, in order to provide additional protection. However, as noted in Section 7.4, EAP tunneling may result in a man-in-the-middle vulnerability, and therefore dictionary attack resistant methods are preferred.

7.7 Connection to an untrusted network

With EAP methods supporting one-way authentication, such as EAP-MD5, the authenticator's identity is not verified. Where the lower layer is not physically secure (such as where EAP runs over wireless media or IP), this enables the peer to connect to a rogue authenticator. As a result, where the lower layer is not physically secure, a method supporting mutual authentication is recommended.

In EAP there is no requirement that authentication be full duplex or that the same protocol be used in both directions. It is perfectly acceptable for different protocols to be used in each direction. This will, of course, depend on the specific protocols negotiated. However, in general, completing a single unitary mutual authentication is preferable to two one-way authentications, one in each direction. This is because separate authentications that are not bound cryptographically so as to demonstrate they are part of the same session are subject to man-in-the-middle attacks, as discussed in Section 7.4.

7.8 Negotiation attacks

In a negotiation attack, the attacker attempts to convince the peer and authenticator to negotiate a less secure EAP method. EAP does not provide protection for the Nak packet, although it is possible for a method to include coverage of Nak Responses within a method-specific MIC.

To avoid negotiation attacks in situations where EAP runs over physically insecure media, for each named peer there SHOULD be an indication of exactly one method used to authenticate that peer name, as described in Section 2.1.

7.9 Implementation idiosyncrasies

The interaction of EAP with lower layer transports such as PPP and IEEE 802 are highly implementation dependent.

For example, upon failure of authentication, some PPP implementations do not terminate the link, instead limiting traffic in Network-Layer Protocols to a filtered subset, which in turn allows the peer the opportunity to update secrets or send mail to the network administrator indicating a problem. Similarly, while in IEEE 802.1X an authentication failure will result in denied access to the controlled port, limited traffic may be permitted on the uncontrolled port.

In EAP there is no provision for retries of failed authentication. However, in PPP the LCP state machine can renegotiate the authentication protocol at any time, thus allowing a new attempt. Similarly, in IEEE 802.1X the Supplicant or Authenticator can re-authenticate at any time. It is recommended that any counters used for authentication failure not be reset until after successful authentication, or subsequent termination of the failed link.

7.10 Key derivation

It is possible for the peer and EAP server to mutually authenticate, and derive a Master Key (MK). The MK is unique to the peer and EAP server and MUST NOT be exported by the EAP method, or used directly to protect the EAP conversation or subsequent data. As a result, possession of the MK represents proof of a successful authentication, and this is potentially useful in enabling features such as fast reconnect, or fast handoff.

In order to provide keying material for use in a subsequently negotiated ciphersuite, the EAP method exports a Master Session Key (MSK). Like the EAP Master Key, EAP Master Session Keys are also not used directly to protect data; however, they are of sufficient size to enable subsequent derivation of Transient Session Keys (TSKs) for use with the selected ciphersuite.

EAP methods provide Master Session Keys and not Transient Session Keys so as to allow EAP methods to be ciphersuite and media independent. Depending on the lower layer, EAP methods may run before or after ciphersuite negotiation, so that the selected ciphersuite

may not be known to the EAP method. By providing keying material usable with any ciphersuite, EAP methods can used with a wide range of ciphersuites and media. Since the peer and EAP client reside on the same machine, TSKs can be provided to the lower layer security module without needing to leave the machine.

In the case where the backend authentication server and authenticator reside on different machines, there are several implications for security:

[a] Mutual authentication may occur between the peer and the backend authentication server, if the negotiated EAP method supports this. However, where the authenticator and backend authentication server are separate, the peer and authenticator do not mutually authenticate within EAP. However, subsequent to completion of the EAP conversation, the lower layer may support mutual authentication between the peer and authenticator. For example, IEEE 802.11i includes a Transient Session Key derivation protocol known as the 4-way handshake, which guarantees liveness of the TSKs, provides for mutual authentication between the peer and authenticator, replay protection, and protected ciphersuite negotiation.

[b] The MSK negotiated between the peer and backend authentication server will need to be transmitted to the authenticator. The specification of this transit mechanism is outside the scope of this document.

This specification does not provide detailed guidance on how EAP methods are to derive the MK and MSK. Key derivation is an art that is best practiced by professionals; rather than inventing new key derivation algorithms, reuse of existing algorithms such as those specified in IKE [RFC2409], or TLS [RFC2246] is recommended.

However, some general guidelines can be provided:

[1] The MK is for use only by the EAP authenticator and peer and MUST NOT be exported by the EAP method or provided to a third party.

[2] Since the MSK is exported by the EAP method, while the MK is not, possession of the MSK MUST NOT provide information useful in determining the MK.

[3] The MSK and TSKs MUST be fresh. Otherwise it is infeasible to detect messages replayed from prior sessions.

[4] TSKs MUST be cryptographically independent from each other so that if an attacker obtains one of them, he will not have gained information useful in determining the other ones.

[5] There MUST be a way to determine whether TSKs belong to this or to some other session.

[6] The MSK derived by EAP methods MUST be bound to the peers as well as to the authentication method, so as to avoid a man-in-the-middle attack (see Section 7.4).

7.11 Weak ciphersuites

If after the initial EAP authentication, data packets are sent without per-packet authentication, integrity and replay protection, an attacker with access to the media can inject packets, "flip bits" within existing packets, replay packets, or even hijack the session completely. Without per-packet confidentiality, it is possible to snoop data packets.

As a result, as noted in Section 3.1, where EAP is used over a physically insecure lower layer, per-packet authentication, integrity and replay protection SHOULD be used, and per-packet confidentiality is also recommended.

7.12 Link layer

There exists a number of reliability and security issues with link layer indications in PPP, IEEE 802 wired networks and IEEE 802.11 wireless LANs:

[a] PPP. In PPP, link layer indications such as LCP-Terminate (a link failure indication) and NCP (a link success indication) are not authenticated or integrity protected. They can therefore be spoofed by an attacker with access to the physical medium.

[b] IEEE 802 wired networks. On wired networks, IEEE 802.1X messages are sent to a non-forwardable multicast MAC address. As a result, while the IEEE 802.1X EAPOL-Start and EAPOL-Logoff frames are not authenticated or integrity protected, only an attacker with access to the physical link can spoof these messages.

[c] IEEE 802.11 wireless LANs. In IEEE 802.11, link layer indications include Disassociate and Deauthenticate frames (link failure indications), and Association and Reassociation Response frames (link success indications). These messages are not authenticated or integrity protected, and although they are not forwardable,

they are spoofable by an attacker within range.

In IEEE 802.11, IEEE 802.1X data frames are sent as Class 3 unicast data frames, are therefore forwardable. This implies that while EAPOL-Start and EAPOL-Logoff messages may be authenticated and integrity protected, they can be spoofed by an authenticated attacker far from the target when "pre-authentication" is enabled.

7.13 Separation of EAP server and authenticator

It is possible for the EAP peer and authenticator to mutually authenticate, and derive a Master Session Key (MSK) for a ciphersuite used to protect subsequent data traffic. This does not present an issue on the peer, since the peer and EAP client reside on the same machine; all that is required is for the EAP client module to derive and pass a Transient Session Key (TSK) to the ciphersuite module.

However, in the case where the EAP server and authenticator reside on different machines, there are several implications for security.

[a] Authentication will occur between the peer and the EAP server, not between the peer and the authenticator. This means that it is not possible for the peer to validate the identity of the NAS or tunnel server that it is speaking to, using EAP alone.

[b] As discussed in [RFC2869bis], the authenticator is dependent on the AAA protocol in order to know the outcome of an authentication conversation, and does not look at the encapsulated EAP packet (if one is present) to determine the outcome. In practice this means that the AAA protocol spoken between the authenticator and authentication server MUST support per-packet authentication, integrity and replay protection.

[c] A EAP Master Session Key (MSK) negotiated between the peer and EAP server will need to be transmitted to the authenticator. Therefore a mechanism needs to be provided to transmit the MSK from the EAP server to the authenticator or tunnel server that needs it. The specification of the key transport and wrapping mechanism is outside the scope of this document.

7.14 Strict Interpretation

An EAP method wishing to enforce tighter security than is provided by the packet processing rules of Section 2.1 and Section 4.2.1 MAY indicate within their specification that they follow a "strict

interpretation" of EAP.

When requested by a method, "strict interpretation" causes the EAP implementation to impose inbound filter rules from the point where an initial Request is answered with a Response of the same Type, until the method completes. "Strict interpretation" also implies that on completion the peer method will indicate whether it succeeded (was able to authenticate the authenticator) or failed (did not succeed in authenticating the authenticator).

An EAP method making use of "strict interpretation" must include a definition of completion for both the peer and authenticator, and also must indicate the conditions under which successful completion will be indicated.

The filter rules are as follows:

[a] On the peer, all EAP packets are silently discarded, except for those with Code=1 (Request) and Type=Method-Type. This implies that methods supporting "strict interpretation" do not utilize Notification, but instead support their own method-specific error messages.

[b] On the peer, once the method completes unsuccessfully, the EAP conversation is terminated, the link is not enabled and Success packets are silently discarded. If the conversation completes successfully, then Failure packets are silently discarded.

[c] On the EAP server, once the initial EAP Request is responded to with an EAP Response of the same Type, all EAP packets are silently discarded, except those with Code=2 (Response) and Type=EAP-Method-Type.

Implementation note: While none of the methods defined in this document support strict interpretation, EAP-TLS [RFC2716] implementations SHOULD support strict interpretation.

8. Acknowledgments

This protocol derives much of its inspiration from Dave Carrel's AHA draft as well as the PPP CHAP protocol [RFC1994]. Valuable feedback was provided by Yoshihiro Ohba of Toshiba America Research, Jari Arkko of Ericsson, Sachin Seth of Microsoft, Glen Zorn of Cisco Systems, Jesse Walker of intel, Nick Petroni, Paul Funk of Funk Software and Pasi Eronen of Nokia.

Normative References

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

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

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

[RFC2243] Metz, C., "OTP Extended Responses", RFC 2243, November 1997.

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[RFC2289] Haller, N., Metz, C., Nesser, P. and M. Straw, "A One-Time Password System", RFC 2289, February 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[IEEE.802.1990] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Overview and Architecture", IEEE Standard 802, 1990.

[IEEE.802-1X.2001] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X, September 2001.

Informative References

[DECEPTION] Slatalla, M. and J. Quittner, "Masters of Deception", HarperCollins , New York, 1995.

[RFC1510] Kohl, J. and B. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[RFC2246] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A.

and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2284] Blunk, L. and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

[RFC2486] Aboba, B. and M. Beadles, "The Network Access Identifier", RFC 2486, January 1999.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2408] Maughan, D., Schneider, M. and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

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

[RFC2607] Aboba, B. and J. Vollbrecht, "Proxy Chaining and Policy Implementation in Roaming", RFC 2607, June 1999.

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

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

[KRBATTACK] Wu, T., "A Real-World Analysis of Kerberos Password Security", Stanford University Computer Science Department, http://theory.stanford.edu/~tjw/krbpass.html.

[KRBLIM] Bellovin, S. and M. Merrit, "Limitations of the Kerberos authentication system", Proceedings of the 1991 Winter USENIX Conference, pp. 253-267, 1991.

[KERB4WEAK] Dole, B., Lodin, S. and E. Spafford, "Misplaced trust: Kerberos 4 session keys", Proceedings of the Internet Society Network and Distributed System Security Symposium, pp. 60-70, March 1997.

[PIC] Aboba, B., Krawczyk, H. and Y. Sheffer, "PIC, A Pre-IKE Credential Provisioning Protocol", draft-ietf-ipsra-pic-06 (work in progress), October 2002.

[PPTPv1] Schneier, B. and Mudge, "Cryptanalysis of Microsoft's Point-to- Point Tunneling Protocol", Proceedings of the 5th ACM Conference on Communications and Computer Security, ACM Press, November 1998.

[IEEE.802-3.1996] Institute of Electrical and Electronics Engineers, "Information technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific requirements - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE Standard 802.3, 1996.

[IEEE.802-11.1999] Institute of Electrical and Electronics Engineers, "Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Network - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, 1999.

[SILVERMAN] Silverman, Robert D., "A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths", RSA Laboratories Bulletin 13, April 2000 (Revised November 2001), http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html.

[RFC2869bis] Aboba, B. and P. Calhoun, "RADIUS Support For Extensible Authentication Protocol (EAP)", draft-aboba-radius-rfc2869bis-09 (work in progress), February 2003.

[IANA-EXP] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", draft-narten-iana-experimental-allocations-03 (work in progress), December 2002.

Authors' Addresses

Larry J. Blunk
Merit Network, Inc
4251 Plymouth Rd., Suite 2000
Ann Arbor, MI 48105-2785
USA

Phone: +1 734-647-9563
Fax: +1 734-647-3185
EMail: ljb@merit.edu

John R. Vollbrecht
Vollbrecht Consulting LLC
9682 Alice Hill Drive
Dexter, MI 48130
USA

Phone:
EMail: jrv@umich.edu

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 706 6605
Fax: +1 425 936 6605
EMail: bernarda@microsoft.com

James Carlson
Sun Microsystems, Inc
1 Network Drive
Burlington, MA 01803-2757
USA

Phone: +1 781 442 2084
Fax: +1 781 442 1677
EMail: james.d.carlson@sun.com

Henrik Levkowetz
ipUnplugged AB
Arenavagen 33
Stockholm S-121 28
SWEDEN

Phone: +46 8 725 9513
EMail: henrik@levkowetz.com

Appendix A. Method Specific Behavior

A.1 Message Integrity Checks

Today, EAP methods commonly define message integrity checks (MICs) that cover more than one EAP packet. For example, EAP-TLS [RFC2716] defines a MIC over a TLS record that could be split into multiple fragments; within the FINISHED message, the MIC is computed over previous messages. Where the MIC covers more than one EAP packet, a MIC validation failure is typically considered a fatal error..

If a per-packet MIC is employed within an EAP method, then peers, authentication servers, and authenticators not operating in pass-through mode MUST validate the MIC. MIC validation failures SHOULD be logged. Whether a MIC validation failure is considered a fatal error or not is determined by the EAP method specification.

Within EAP-TLS [RFC2716] a MIC validation failures is treated as a fatal error, since that is what is specified in TLS [RFC2246]. However, it is also possible to develop EAP methods that support per-packet MICs, and respond to verification failures by silently discarding the offending packet.

In this document, descriptions of EAP message handling assume that per-packet MIC validation, where it occurs, is effectively performed as though it occurs before sending any responses or changing the state of the host which received the packet.

Appendix B. Changes from RFC 2284

This section lists the major changes between [RFC2284] and this document. Minor changes, including style, grammar, spelling and editorial changes are not mentioned here.

- The Terminology section (Section 1.2) has been expanded, defining more concepts and giving more exact definitions.

- In Section 2, it is explicitly specified that more than one exchange of Request and Response packets may occur as part of the

EAP authentication exchange. How this may and may not be used is specified in detail in Section 2.1.

- Also in Section 2, some requirements on the authenticator when acting in pass-through mode has been made explicit.

- An EAP multiplexing model (Section 2.2) has been added, to illustrate a typical implementation of EAP. There is no requirement that an implementation conforms to this model, as long as the on-the-wire behavior is consistent with it.

- As EAP is now in use with a variety of lower layers, not just PPP for which it was first designed, Section 3 on lower layer behavior has been added.

- In the description of the EAP Request and Response interaction (Section 4.1), it has been more exactly specified when packets should be silently discarded, and also the behavior on receiving duplicate requests. The implementation notes in this section has been substantially expanded.

- In Section 4.2, it has been clarified that Success and Failure packets must not contain additional data, and the implementation note has been expanded. A subsection giving requirements on processing of success and failure packets has been added.

- Section 5 on EAP Request/Response Types lists two new type values: the Expanded type (Section 5.7), which is used to expand the type value number space, and the Experimental type. In the Expanded type number space, the new Expanded Nak (Section 5.3.2) type has been added. Clarifications have been made in the description of most of the existing types. Security claims summaries have been added for authentication methods.

- An IANA Considerations section (Section 6) has been added, giving registration policies for the numbering spaces defined for EAP.

- The Security Considerations (Section 7) have been greatly expanded, aiming at giving a much more comprehensive coverage of possible threats and other security considerations.

Appendix C. Open issues

(This section should be removed by the RFC editor before publication)

Open issues relating to this specification are tracked on the following web site:

http://www.drizzle.com/~aboba/EAP/eapissues.html

The current working documents for this draft are available at this web site:

http://www.levkowetz.com/pub/ietf/drafts/eap/

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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