

Network Working Group
Request for Comments: 3748
Obsoletes: 2284
Category: Standards Track

B. Aboba
Microsoft
L. Blunk
Merit Network, Inc
J. Vollbrecht
Vollbrecht Consulting LLC
J. Carlson
Sun
H. Levkowitz, Ed.
ipUnplugged
June 2004

Extensible Authentication Protocol (EAP)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2004).

Abstract

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

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

Table of Contents

1.	Introduction	3
1.1.	Specification of Requirements	4
1.2.	Terminology	4
1.3.	Applicability	6
2.	Extensible Authentication Protocol (EAP).	7
2.1.	Support for Sequences	9
2.2.	EAP Multiplexing Model.	10
2.3.	Pass-Through Behavior	12
2.4.	Peer-to-Peer Operation.	14
3.	Lower Layer Behavior.	15
3.1.	Lower Layer Requirements.	15
3.2.	EAP Usage Within PPP.	18
3.2.1.	PPP Configuration Option Format.	18
3.3.	EAP Usage Within IEEE 802	19
3.4.	Lower Layer Indications	19
4.	EAP Packet Format	20
4.1.	Request and Response.	21
4.2.	Success and Failure	23
4.3.	Retransmission Behavior	26
5.	Initial EAP Request/Response Types.	27
5.1.	Identity.	28
5.2.	Notification.	29
5.3.	Nak	31
5.3.1.	Legacy Nak	31
5.3.2.	Expanded Nak	32
5.4.	MD5-Challenge	35
5.5.	One-Time Password (OTP)	36
5.6.	Generic Token Card (GTC).	37
5.7.	Expanded Types.	38
5.8.	Experimental.	40
6.	IANA Considerations	40
6.1.	Packet Codes.	41
6.2.	Method Types.	41
7.	Security Considerations	42
7.1.	Threat Model.	42
7.2.	Security Claims	43
7.2.1.	Security Claims Terminology for EAP Methods.	44
7.3.	Identity Protection	46
7.4.	Man-in-the-Middle Attacks	47
7.5.	Packet Modification Attacks	48
7.6.	Dictionary Attacks.	49
7.7.	Connection to an Untrusted Network.	49
7.8.	Negotiation Attacks	50
7.9.	Implementation Idiosyncrasies	50
7.10.	Key Derivation.	51
7.11.	Weak Ciphersuites	53

7.12. Link Layer.	53
7.13. Separation of Authenticator and Backend Authentication Server.	54
7.14. Cleartext Passwords	55
7.15. Channel Binding	55
7.16. Protected Result Indications.	56
8. Acknowledgements.	58
9. References.	59
9.1. Normative References.	59
9.2. Informative References.	60
Appendix A. Changes from RFC 2284.	64
Authors' Addresses	66
Full Copyright Statement	67

1. Introduction

This document defines the Extensible Authentication Protocol (EAP), an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP. EAP provides its own support for duplicate elimination and retransmission, but is reliant on lower layer ordering guarantees. 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]. EAP encapsulation on IEEE 802 wired media is described in [IEEE-802.1X], and encapsulation on IEEE wireless LANs in [IEEE-802.11i].

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 method 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 peers.

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

peer

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

Supplicant

The end of the link that responds to the authenticator in [IEEE-802.1X]. In this document, this end of the link is called the peer.

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].

AAA

Authentication, Authorization, and Accounting. AAA protocols with EAP support include RADIUS [RFC3579] and Diameter [DIAM-EAP]. In this document, the terms "AAA server" and "backend authentication server" are used interchangeably.

Displayable Message

This is interpreted to be a human readable string of characters. 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.

Successful Authentication

In the context of this document, "successful authentication" is an exchange of EAP messages, as a result of which the authenticator decides to allow access by the peer, and the peer decides to use this access. The authenticator's decision typically involves both authentication and authorization aspects; the peer may successfully authenticate to the authenticator, but access may be denied by the authenticator due to policy reasons.

Message Integrity Check (MIC)

A keyed hash function used for authentication and integrity protection of data. This is usually called a Message Authentication Code (MAC), but IEEE 802 specifications (and this document) use the acronym MIC to avoid confusion with Medium Access Control.

Cryptographic Separation

Two keys (x and y) are "cryptographically separate" if an adversary that knows all messages exchanged in the protocol cannot compute x from y or y from x without "breaking" some cryptographic assumption. In particular, this definition allows that the adversary has the knowledge of all nonces sent in cleartext, as well as all predictable counter values used in the protocol. Breaking a cryptographic assumption would typically require inverting a one-way function or predicting the outcome of a cryptographic pseudo-random number generator without knowledge of the secret state. In other words, if the keys are cryptographically separate, there is no shortcut to compute x from y or y from x, but the work an adversary must do to perform this computation is equivalent to performing an exhaustive search for the secret state value.

Master Session Key (MSK)

Keying material that is derived between the EAP peer and server and exported by the EAP method. The MSK is at least 64 octets in length. In existing implementations, a AAA server acting as an EAP server transports the MSK to the authenticator.

Extended Master Session Key (EMSK)

Additional keying material derived between the EAP client and server that is exported by the EAP method. The EMSK is at least 64 octets in length. The EMSK is not shared with the authenticator or any other third party. The EMSK is reserved for future uses that are not defined yet.

Result indications

A method provides result indications if after the method's last message is sent and received:

- 1) The peer is aware of whether it has authenticated the server, as well as whether the server has authenticated it.
- 2) The server is aware of whether it has authenticated the peer, as well as whether the peer has authenticated it.

In the case where successful authentication is sufficient to authorize access, then the peer and authenticator will also know if the other party is willing to provide or accept access. This may not always be the case. An authenticated peer may be denied access due to lack of authorization (e.g., session limit) or other reasons. Since the EAP exchange is run between the peer and the server, other nodes (such as AAA proxies) may also affect the authorization decision. This is discussed in more detail in Section 7.16.

1.3. Applicability

EAP was designed for use in network access authentication, where IP layer connectivity may not be available. Use of EAP for other purposes, such as bulk data transport, is NOT RECOMMENDED.

Since EAP does not require IP connectivity, it provides just enough support for the reliable transport of authentication protocols, and no more.

EAP is a lock-step protocol which only supports a single packet in flight. As a result, EAP cannot efficiently transport bulk data, unlike transport protocols such as TCP [RFC793] or SCTP [RFC2960].

While EAP provides support for retransmission, it assumes ordering guarantees provided by the lower layer, so out of order reception is not supported.

Since EAP does not support fragmentation and reassembly, EAP authentication methods generating payloads larger than the minimum EAP MTU need to provide fragmentation support.

While authentication methods such as EAP-TLS [RFC2716] provide support for fragmentation and reassembly, the EAP methods defined in this document do not. As a result, if the EAP packet size exceeds the EAP MTU of the link, these methods will encounter difficulties.

EAP authentication is initiated by the server (authenticator), whereas many authentication protocols are initiated by the client (peer). As a result, it may be necessary for an authentication algorithm to add one or two additional messages (at most one roundtrip) in order to run over EAP.

Where certificate-based authentication is supported, the number of additional roundtrips may be much larger due to fragmentation of certificate chains. In general, a fragmented EAP packet will require as many round-trips to send as there are fragments. For example, a certificate chain 14960 octets in size would require ten round-trips to send with a 1496 octet EAP MTU.

Where EAP runs over a lower layer in which significant packet loss is experienced, or where the connection between the authenticator and authentication server experiences significant packet loss, EAP methods requiring many round-trips can experience difficulties. In these situations, use of EAP methods with fewer roundtrips is advisable.

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 is responsible for retransmitting requests as described in Section 4.1. After a suitable number of retransmissions, the authenticator **SHOULD** end the EAP conversation. The authenticator **MUST NOT** send a Success or Failure packet when retransmitting or when it fails to get a response from the peer.
- [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).

Advantages:

- o The EAP protocol can support multiple authentication mechanisms without having to pre-negotiate a particular one.
- o Network Access Server (NAS) devices (e.g., a 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 peers, while at the same time acting as a pass-through for non-local peers and authentication methods it does not implement locally.
- o Separation of the authenticator from the backend authentication server simplifies credentials management and policy decision making.

Disadvantages:

- o For use in PPP, EAP requires 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] in order to use EAP.
- o 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, the peer and authenticator 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.

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; a peer receiving such Requests MUST treat them as invalid, and silently discard them. As a result, Identity Requery is not supported.

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 authenticator receiving an unexpected Nak SHOULD discard it and log the event.

Multiple authentication methods within an EAP conversation are not supported due to their vulnerability to man-in-the-middle attacks (see Section 7.4) and incompatibility with existing implementations.

Where a single EAP authentication method is utilized, but other methods are run within it (a "tunneled" method), 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

(legacy or expanded). To address security vulnerabilities, "tunneled" methods MUST support protection against man-in-the-middle attacks.

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], IEEE 802.11 wireless LANs [IEEE-802.11], UDP (L2TP [RFC2661] and IKEv2 [IKEv2]), 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 duplicate detection and retransmission, and delivers and receives EAP messages to and from the EAP peer and authenticator layers.
- [c] EAP peer and authenticator layers. Based on the Code field, the EAP layer demultiplexes incoming EAP packets to the EAP peer and authenticator layers. Typically, an EAP implementation on a given host will support either peer or authenticator functionality, but it is possible for a host to act as both an EAP peer and authenticator. In such an implementation both EAP peer and authenticator layers will be present.
- [d] EAP method layers. EAP methods implement the authentication algorithms and receive and transmit EAP messages via the EAP peer and authenticator layers. 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.

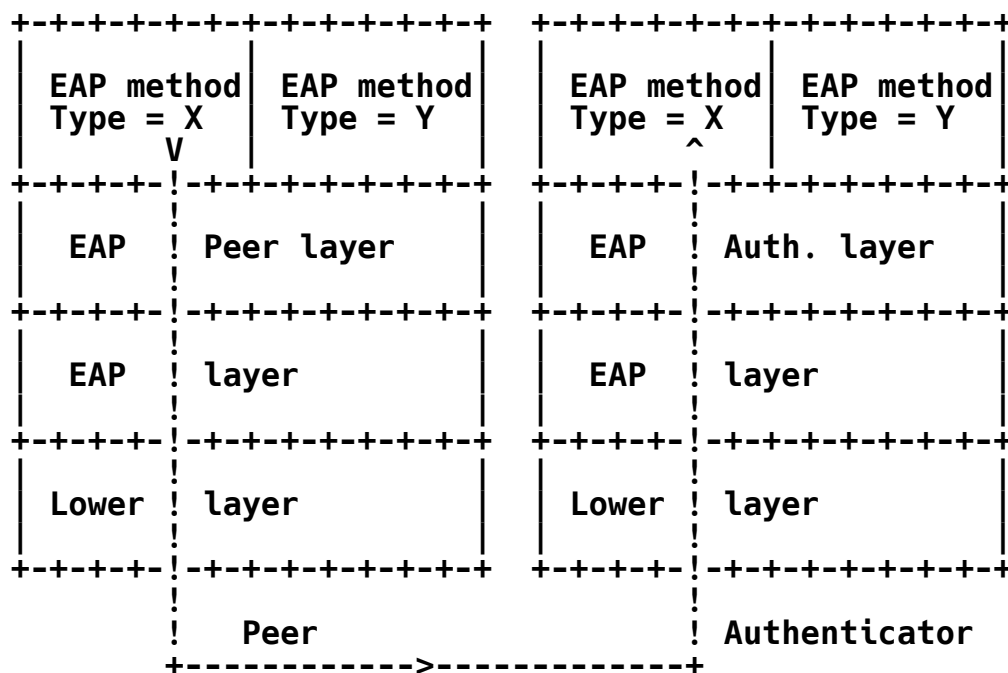


Figure 1: EAP Multiplexing Model

Within EAP, the Code field functions much like a protocol number in IP. It is assumed that the EAP layer demultiplexes incoming EAP packets according to the Code field. Received EAP packets with Code=1 (Request), 3 (Success), and 4 (Failure) are delivered by the EAP layer to the EAP peer layer, if implemented. EAP packets with Code=2 (Response) are delivered to the EAP authenticator layer, if implemented.

Within EAP, the Type field functions much like a port number in UDP or TCP. It is assumed that the EAP peer and authenticator layers demultiplex incoming EAP packets according to their Type, and deliver them only to the EAP method corresponding to that Type. An EAP method implementation on a host may register to receive packets from the peer or authenticator layers, or both, depending on which role(s) it supports.

Since EAP authentication methods may wish to access the Identity, implementations SHOULD make the Identity Request and Response accessible to authentication methods (Types 4 or greater), in addition to the Identity method. 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 are available to another method. The Notification Type is discussed in Section 5.2.

Nak (Type 3) or Expanded Nak (Type 254) are utilized for the purposes of method negotiation. Peers respond to an initial EAP Request for an unacceptable Type with a Nak Response (Type 3) or Expanded Nak Response (Type 254). It cannot be assumed that the contents of the Nak Response(s) are available to another method. The Nak Type(s) are discussed in Section 5.3.

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

Given these considerations, the Success, Failure, Nak Response(s), and Notification Request/Response messages MUST NOT be used to carry data destined for delivery to other EAP methods.

2.3. Pass-Through Behavior

When operating as a "pass-through authenticator", an authenticator performs checks on the Code, Identifier, and Length fields as described in Section 4.1. It forwards EAP packets received from the peer and destined to its authenticator layer to the backend authentication server; packets received from the backend authentication server destined to the peer are forwarded to it.

A host receiving an EAP packet may only do one of three things with it: act on it, drop it, or forward it. The forwarding decision is typically based only on examination of the Code, Identifier, and Length fields. A pass-through authenticator implementation MUST be capable of forwarding EAP packets received from the peer with Code=2 (Response) to the backend authentication server. It also MUST be capable of receiving EAP packets from the backend authentication server and forwarding EAP packets of Code=1 (Request), Code=3 (Success), and Code=4 (Failure) to the peer.

Unless the authenticator implements one or more authentication methods locally which support the authenticator role, the EAP method layer header fields (Type, Type-Data) are not examined as part of the forwarding decision. Where the authenticator supports local authentication methods, it MAY examine the Type field to determine whether to act on the packet itself or forward it. Compliant pass-through authenticator implementations MUST by default forward EAP packets of any Type.

EAP packets received with Code=1 (Request), Code=3 (Success), and Code=4 (Failure) are demultiplexed by the EAP layer and delivered to the peer layer. Therefore, unless a host implements an EAP peer layer, these packets will be silently discarded. Similarly, EAP packets received with Code=2 (Response) are demultiplexed by the EAP layer and delivered to the authenticator layer. Therefore, unless a host implements an EAP authenticator layer, these packets will be silently discarded. The behavior of a "pass-through peer" is undefined within this specification, and is unsupported by AAA protocols such as RADIUS [RFC3579] and Diameter [DIAM-EAP].

The forwarding model is illustrated in Figure 2.

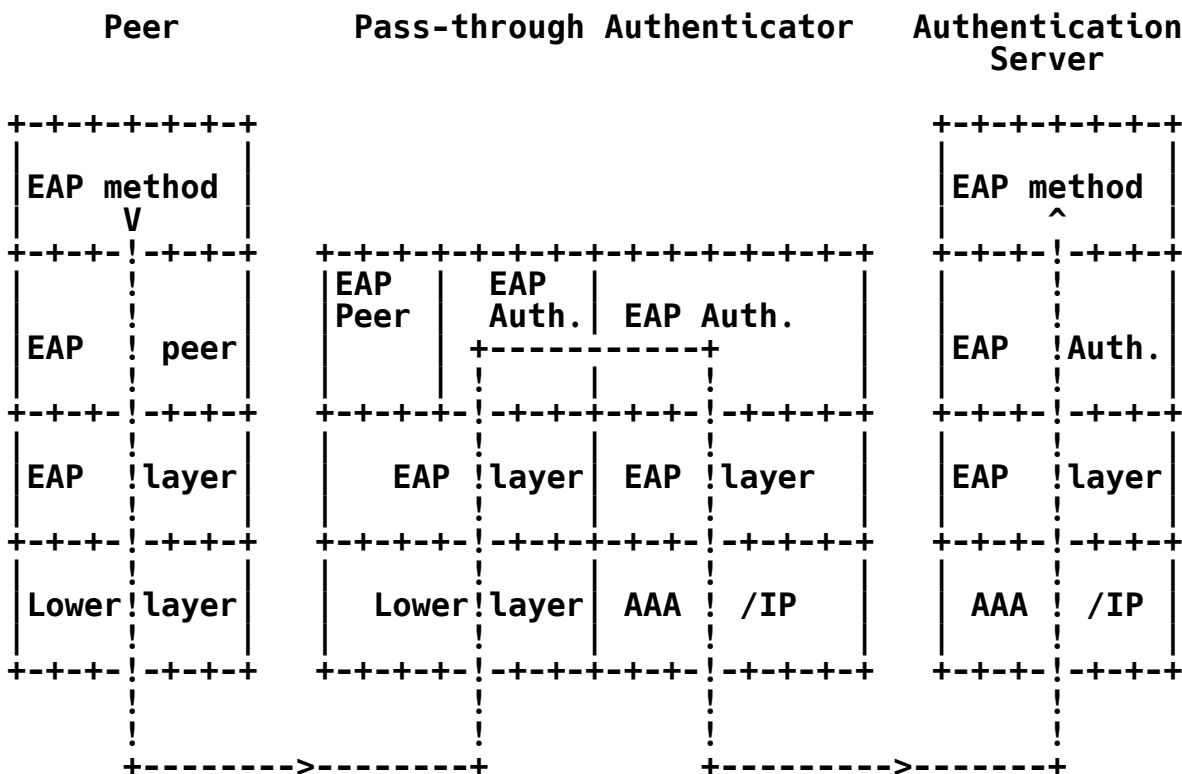


Figure 2: Pass-through Authenticator

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.

2.4. Peer-to-Peer Operation

Since EAP is a peer-to-peer protocol, an independent and simultaneous authentication may take place in the reverse direction (depending on the capabilities of the lower layer). Both ends of the link may act as authenticators and peers at the same time. In this case, it is necessary for both ends to implement EAP authenticator and peer layers. In addition, the EAP method implementations on both peers must support both authenticator and peer functionality.

Although EAP supports peer-to-peer operation, some EAP implementations, methods, AAA protocols, and link layers may not support this. Some EAP methods may support asymmetric authentication, with one type of credential being required for the peer and another type for the authenticator. Hosts supporting peer-to-peer operation with such a method would need to be provisioned with both types of credentials.

For example, EAP-TLS [RFC2716] is a client-server protocol in which distinct certificate profiles are typically utilized for the client and server. This implies that a host supporting peer-to-peer authentication with EAP-TLS would need to implement both the EAP peer and authenticator layers, support both peer and authenticator roles in the EAP-TLS implementation, and provision certificates appropriate for each role.

AAA protocols such as RADIUS/EAP [RFC3579] and Diameter EAP [DIAM-EAP] only support "pass-through authenticator" operation. As noted in [RFC3579] Section 2.6.2, a RADIUS server responds to an Access-Request encapsulating an EAP-Request, Success, or Failure packet with an Access-Reject. There is therefore no support for "pass-through peer" operation.

Even where a method is used which supports mutual authentication and result indications, several considerations may dictate that two EAP authentications (one in each direction) are required. These include:

- [1] Support for bi-directional session key derivation in the lower layer. Lower layers such as IEEE 802.11 may only support uni-directional derivation and transport of transient session keys. For example, the group-key handshake defined in [IEEE-802.11i] is uni-directional, since in IEEE 802.11 infrastructure mode, only the Access Point (AP) sends multicast/broadcast traffic. In IEEE 802.11 ad hoc mode, where either peer may send multicast/broadcast traffic, two uni-directional group-key

exchanges are required. Due to limitations of the design, this also implies the need for unicast key derivations and EAP method exchanges to occur in each direction.

- [2] Support for tie-breaking in the lower layer. Lower layers such as IEEE 802.11 ad hoc do not support "tie breaking" wherein two hosts initiating authentication with each other will only go forward with a single authentication. This implies that even if 802.11 were to support a bi-directional group-key handshake, then two authentications, one in each direction, might still occur.
- [3] Peer policy satisfaction. EAP methods may support result indications, enabling the peer to indicate to the EAP server within the method that it successfully authenticated the EAP server, as well as for the server to indicate that it has authenticated the peer. However, a pass-through authenticator will not be aware that the peer has accepted the credentials offered by the EAP server, unless this information is provided to the authenticator via the AAA protocol. The authenticator **SHOULD** interpret the receipt of a key attribute within an Accept packet as an indication that the peer has successfully authenticated the server.

However, it is possible that the EAP peer's access policy was not satisfied during the initial EAP exchange, even though mutual authentication occurred. For example, the EAP authenticator may not have demonstrated authorization to act in both peer and authenticator roles. As a result, the peer may require an additional authentication in the reverse direction, even if the peer provided an indication that the EAP server had successfully authenticated to it.

3. Lower Layer Behavior

3.1. Lower Layer Requirements

EAP makes the following assumptions about lower layers:

- [1] Unreliable transport. 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, it is possible (though undesirable) for retransmission to occur both in the lower layer and the EAP layer when EAP is run over a reliable lower layer.

Note that EAP Success and Failure packets are not retransmitted. Without a reliable lower layer, and with a non-negligible error rate, these packets can be lost, resulting in timeouts. It is therefore desirable for implementations to improve their resilience to loss of EAP Success or Failure packets, as described in Section 4.2.

- [2] Lower layer error detection. While EAP does not assume that the lower layer is reliable, it does rely on lower layer error detection (e.g., CRC, Checksum, MIC, etc.). EAP methods may not include a MIC, or if they do, it may not be computed over all the fields in the EAP packet, such as the Code, Identifier, Length, or Type fields. As a result, without lower layer error detection, undetected errors could creep into the EAP layer or EAP method layer header fields, resulting in authentication failures.

For example, EAP TLS [RFC2716], which computes its MIC over the Type-Data field only, regards MIC validation failures as a fatal error. Without lower layer error detection, this method, and others like it, will not perform reliably.

- [3] Lower layer security. EAP does not require lower layers to provide security services such as per-packet confidentiality, authentication, integrity, and replay protection. However, where these security services are available, EAP methods supporting Key Derivation (see Section 7.2.1) can be used to provide dynamic keying material. This makes it possible to bind the EAP authentication to subsequent data and protect against data modification, spoofing, or replay. See Section 7.1 for details.
- [4] Minimum MTU. EAP is capable of functioning on lower layers that provide an EAP MTU size of 1020 octets or greater.

EAP does not support path MTU discovery, and fragmentation and reassembly is not supported by EAP, nor by the methods defined in this specification: Identity (1), Notification (2), Nak Response (3), MD5-Challenge (4), One Time Password (5), Generic Token Card (6), and expanded Nak Response (254) Types.

Typically, the EAP peer obtains information on the EAP MTU from the lower layers and sets the EAP frame size to an appropriate value. Where the authenticator operates in pass-through mode, the authentication server does not have a direct way of determining the EAP MTU, and therefore relies on the authenticator to provide it with this information, such as via the Framed-MTU attribute, as described in [RFC3579], Section 2.4.

While methods such as EAP-TLS [RFC2716] support fragmentation and reassembly, EAP methods originally designed for use within PPP where a 1500 octet MTU is guaranteed for control frames (see [RFC1661], Section 6.1) may lack fragmentation and reassembly features.

EAP methods can assume a minimum EAP MTU of 1020 octets in the absence of other information. EAP methods **SHOULD** include support for fragmentation and reassembly if their payloads can be larger than this minimum EAP MTU.

EAP is a lock-step protocol, which implies a certain inefficiency when handling fragmentation and reassembly. Therefore, if the lower layer supports fragmentation and reassembly (such as where EAP is transported over IP), it may be preferable for fragmentation and reassembly to occur in the lower layer rather than in EAP. This can be accomplished by providing an artificially large EAP MTU to EAP, causing fragmentation and reassembly to be handled within the lower layer.

- [5] 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.
- [6] Ordering guarantees. EAP does not require the Identifier to be monotonically increasing, and so is reliant on lower layer ordering guarantees for correct operation. 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 layer transports for EAP **MUST** preserve ordering between a source and destination at a given priority level (the ordering guarantee provided by [IEEE-802]).

Reordering, if it occurs, will typically result in an EAP authentication failure, causing EAP authentication to be re-run. In an environment in which reordering is likely, it is therefore expected that EAP authentication failures will be common. It is **RECOMMENDED** that EAP only be run over lower layers that provide ordering guarantees; running EAP over raw IP or UDP transport is

NOT RECOMMENDED. Encapsulation of EAP within RADIUS [RFC3579] satisfies ordering requirements, since RADIUS is a "lockstep" protocol that delivers packets in order.

3.2. EAP Usage Within PPP

In order to establish communications over a point-to-point link, each end of the PPP link first sends LCP packets to configure the data link during the 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 the 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 the 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 "backend" 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 EAP follows. 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).

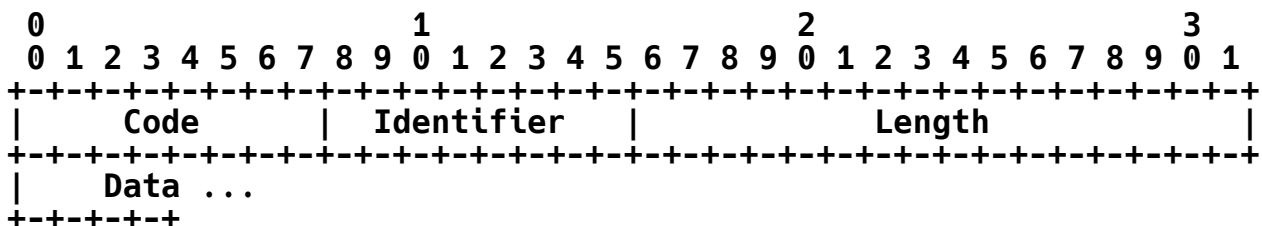
this, it is necessary for the lower layer to provide per-packet integrity, authentication and replay protection, and to bind these per-packet services to the keys derived during EAP authentication. Otherwise, it is possible for subsequent data traffic to be modified, spoofed, or replayed.

Where keying material for the lower layer ciphersuite is itself provided by EAP, ciphersuite negotiation and key activation are controlled by the lower layer. In PPP, ciphersuites are negotiated within ECP so that it is not possible to use keys derived from EAP authentication until the completion of ECP. Therefore, an initial EAP exchange cannot be protected by a PPP ciphersuite, although EAP re-authentication can be protected.

In IEEE 802 media, initial key activation also typically occurs after completion of EAP authentication. Therefore an initial EAP exchange typically cannot be protected by the lower layer ciphersuite, although an EAP re-authentication or pre-authentication exchange can be protected.

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, in octets, 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 MUST be ignored upon reception. A message with the Length field set to a value larger than the number of received octets MUST be silently discarded.

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

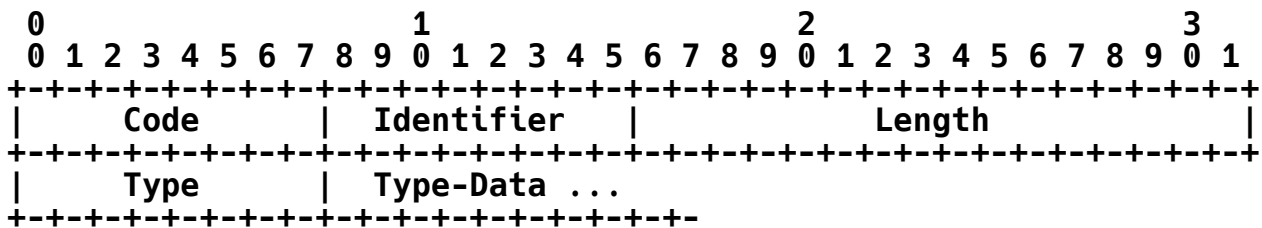
Description

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, an optional retry counter expires, or a lower layer failure indication is received.

Retransmitted Requests MUST be sent with the same Identifier value in order to distinguish them from new Requests. The content 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 be retransmitted on a timer.

If a peer receives a valid duplicate Request for which it has already sent a Response, it MUST resend its original Response without reprocessing the Request. Requests MUST be processed in the order that they are received, and MUST be processed to their completion before inspecting the next Request.

A summary of the Request and Response packet format follows. 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.

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.

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 more difficult.

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.

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.

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 MUST be ignored upon reception. A message with the Length field set to a value larger than the number of received octets MUST be silently discarded.

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. An initial specification of Types follows in Section 5 of this document.

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) indicating that a Request Type is unacceptable to the peer. A peer MUST NOT send a Nak (legacy or expanded) in response to a Request, after an initial non-Nak Response has been sent. An EAP server receiving a Response not meeting these requirements MUST silently discard it.

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 after completion of an EAP authentication method (Type 4 or greater) to indicate that the peer has authenticated successfully to the authenticator. 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 after unsuccessful completion of the EAP method in progress, 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.

Success and Failure packets MUST NOT be sent by an EAP authenticator if the specification of the given method does not explicitly permit the method to finish at that point. A peer EAP implementation receiving a Success or Failure packet where sending one is not explicitly permitted MUST silently discard it. By default, an EAP peer MUST silently discard a "canned" Success packet (a Success packet sent immediately upon connection). This ensures that a rogue authenticator will not be able to bypass mutual authentication by sending a Success packet prior to conclusion of the EAP method conversation.

Implementation Note: Because the Success and Failure packets are not acknowledged, they are not retransmitted by the authenticator, and may be potentially lost. A peer MUST allow for this circumstance as described in this note. See also Section 3.4 for guidance on the processing of lower layer success and failure indications.

As described in Section 2.1, only a single EAP authentication method is allowed within an EAP conversation. EAP methods may implement result indications. After the authenticator sends a failure result indication to the peer, regardless of the response from the peer, it MUST subsequently send a Failure packet. After the authenticator sends a success result indication to the peer and receives a success result indication from the peer, it MUST subsequently send a Success packet.

On the peer, once the method completes unsuccessfully (that is, either the authenticator sends a failure result indication, or the peer decides that it does not want to continue the conversation, possibly after sending a failure result indication), the peer MUST terminate the conversation and indicate failure to the lower layer. The peer MUST silently discard Success packets and MAY silently discard Failure packets. As a result, loss of a Failure packet need not result in a timeout.

On the peer, after success result indications have been exchanged by both sides, a Failure packet MUST be silently discarded. The peer MAY, in the event that an EAP Success is not received, conclude that the EAP Success packet was lost and that authentication concluded successfully.

If the authenticator has not sent a result indication, and the peer is willing to continue the conversation, the peer waits for a Success or Failure packet once the method completes, and **MUST NOT** silently discard either of them. In the event that neither a Success nor Failure packet is received, the peer **SHOULD** terminate the conversation to avoid lengthy timeouts in case the lost packet was an EAP Failure.

If the peer attempts to authenticate to the authenticator and fails to do so, the authenticator **MUST** send a Failure packet and **MUST NOT** grant access by sending a Success packet. However, an authenticator **MAY** omit having the peer authenticate to it in situations where limited access is offered (e.g., guest access). In this case, the authenticator **MUST** send a Success packet.

Where the peer authenticates successfully to the authenticator, but the authenticator does not send a result indication, the authenticator **MAY** deny access by sending a Failure packet where the peer is not currently authorized for network access.

A summary of the Success and Failure 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	2	3	4	5	6	7	8	9	0	1								
Code										Identifier										Length																			

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

4.3. Retransmission Behavior

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 dynamically estimated. 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. 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 may be determined by user responsiveness rather than network characteristics, so that dynamic RT0 estimation may not be 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).

In order to dynamically estimate the EAP retransmission timer, the algorithms for the estimation of SRTT, RTTVAR, and RT0 described in [RFC2988] are **RECOMMENDED**, including use of Karn's algorithm, with the following potential modifications:

- [a] In order to avoid synchronization behaviors that can occur with fixed timers among distributed systems, the retransmission timer is calculated with a jitter by using the RT0 value and randomly adding a value drawn between $-RT0min/2$ and $RT0min/2$. Alternative calculations to create jitter **MAY** be used. These **MUST** be pseudo-random. For a discussion of pseudo-random number generation, see [RFC1750].
- [b] When EAP is transported over a single link (as opposed to over the Internet), smaller values of $RT0initial$, $RT0min$, and $RT0max$ **MAY** be used. Recommended values are $RT0initial=1$ second, $RT0min=200ms$, and $RT0max=20$ seconds.

- [c] When EAP is transported over a single link (as opposed to over the Internet), estimates MAY be done on a per-authenticator basis, rather than a per-session basis. This enables the retransmission estimate to make the most use of information on link-layer behavior.
- [d] An EAP implementation MAY clear SRTT and RTTVAR after backing off the timer multiple times, as it is likely that the current SRTT and RTTVAR are bogus in this situation. Once SRTT and RTTVAR are cleared, they should be initialized with the next RTT sample taken as described in [RFC2988] equation 2.2.

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 future 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. Nak (Type 3) or Expanded Nak (Type 254) are valid only for Response packets, they MUST NOT be sent in a Request.

All EAP implementations MUST support Types 1-4, which are defined in this document, and SHOULD support Type 254. Implementations MAY support other Types defined here or in future RFCs.

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

EAP methods MAY support authentication based on shared secrets. If the shared secret is a passphrase entered by the user, implementations MAY support entering passphrases with non-ASCII characters. In this case, the input should be processed using an appropriate stringprep [RFC3454] profile, and encoded in octets using UTF-8 encoding [RFC2279]. A preliminary version of a possible stringprep profile is described in [SASLPREP].

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 is an 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).

Some EAP implementations piggy-back various options into the Identity Request after a NUL-character. By default, an EAP implementation SHOULD NOT assume that an Identity Request or Response can be larger than 1020 octets.

It is RECOMMENDED that the Identity Response be used primarily for routing purposes and selecting which EAP method to use. EAP Methods SHOULD include a method-specific mechanism for obtaining the identity, so that they do not have to rely on the Identity Response. Identity Requests and Responses are sent in cleartext, so an attacker may snoop on the identity, or even modify or spoof identity exchanges. To address these threats, it is preferable for an EAP method to include an identity exchange that supports per-packet authentication, integrity and replay protection, and confidentiality. The Identity Response may not be the appropriate identity for the method; it may have been truncated or obfuscated so as to provide privacy, or it may have been decorated for routing purposes. Where the peer is configured to only accept authentication methods supporting protected identity exchanges, the peer MAY provide an abbreviated Identity Response (such as omitting the peer-name portion of the NAI [RFC2486]). For further discussion of identity protection, see Section 7.3.

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. 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]. Where the Request contains a null, only the portion of the field prior to the null is displayed. If the Identity is unknown, the Identity Response field should be zero bytes in length. The Identity Response field MUST NOT be null terminated. In all cases, the length of the Type-Data field is derived from the Length field of the Request/Response packet.

Security Claims (see Section 7.2):

Auth. mechanism:	None
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	N/A
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	No
Channel binding:	No

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, prior to completion of an EAP authentication method. The peer MUST respond to a Notification Request with a Notification Response unless the EAP authentication method specification prohibits the use of Notification messages. In any case, a Nak Response MUST NOT be sent in response to a Notification Request. Note that the default maximum length of a Notification Request is 1020 octets. By default, this leaves at most 1015 octets for the human readable message.

An EAP method MAY indicate within its specification that Notification messages must not be sent during that method. In this case, the peer MUST silently discard Notification Requests from the point where an initial Request for that Type is answered with a Response of the same Type.

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 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 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).

Security Claims (see Section 7.2):

Auth. mechanism:	None
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	N/A
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	No
Channel binding:	No

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, and therefore the authenticator SHOULD NOT send another Request after receiving a Nak Response containing a zero value.

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 a peer receives a Request for an unacceptable authentication Type (4-253,255), or a peer lacking support for Expanded Types receives a Request for Type 254, a Nak Response (Type 3) MUST be sent. The Type-Data field of the Nak Response (Type 3) 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 authentication Type (4-253, 255) MAY include the value 254 in the Nak Response (Type 3) to indicate the desire for an Expanded authentication Type. If the authenticator can accommodate this preference, it will respond with an Expanded Type Request (Type 254).

Security Claims (see Section 7.2):

Auth. mechanism:	None
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	N/A
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	No
Channel binding:	No

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 an Expanded Nak Response **MUST** match the Identifier field of the Request packet that it is sent in response to.

Length

>=20

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:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	2									Identifier											Length=28										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	Type=254																				0 (IETF)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
																					3 (Nak)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	Type=254																				0 (IETF)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
																					5 (OTP)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	Type=254																				20 (MIT)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
																					6										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

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

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	2									Identifier											Length=20										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	Type=254																				0 (IETF)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
																					3 (Nak)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	Type=254																				0 (IETF)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
																					0 (No alternative)										
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Security Claims (see Section 7.2):

Auth. mechanism:	None
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	N/A
Fast reconnect:	No
Crypt. binding:	N/A

Session independence:	N/A
Fragmentation:	No
Channel binding:	No

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), Nak (Type 3), or Expanded Nak (Type 254). 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.

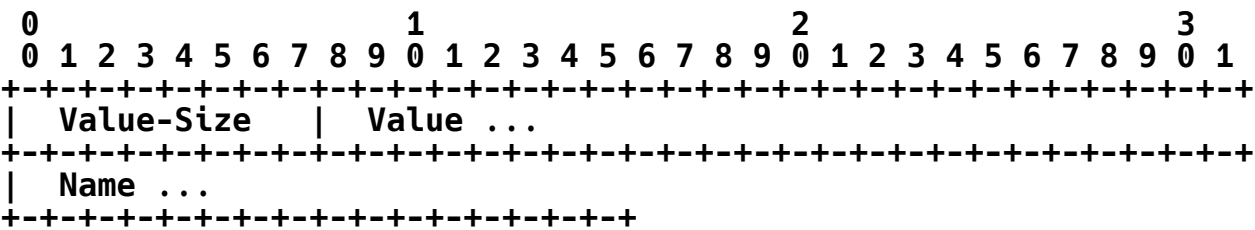
Note: [RFC1994] treats the shared secret as an octet string, and does not specify how it is entered into the system (or if it is handled by the user at all). EAP MD5-Challenge implementations MAY support entering passphrases with non-ASCII characters. See Section 5 for instructions how the input should be processed and encoded into octets.

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):

Auth. mechanism:	Password or pre-shared key.
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	No
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	No
Channel binding:	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 an OTP challenge in the format described in [RFC2289]. A Response MUST be sent in reply to the Request. The Response MUST be of Type 5 (OTP), Nak (Type 3), or Expanded Nak (Type 254). The Nak Response indicates the peer's desired authentication Type(s). The EAP OTP method is intended for use with the One-Time Password system only, and MUST NOT be used to provide support for cleartext passwords.

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.

Note: [RFC2289] does not specify how the secret pass-phrase is entered by the user, or how the pass-phrase is converted into octets. EAP OTP implementations **MAY** support entering passphrases with non-ASCII characters. See Section 5 for instructions on how the input should be processed and encoded into octets.

Security Claims (see Section 7.2):

Auth. mechanism:	One-Time Password
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	Yes
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	No
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	No
Channel binding:	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), Nak (Type 3), or Expanded Nak (Type 254). The Nak Response indicates the peer's desired authentication Type(s). The EAP GTC method is intended for use with the Token Cards supporting challenge/response

authentication and **MUST NOT** be used to provide support for cleartext passwords in the absence of a protected tunnel with server authentication.

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.

EAP GTC implementations **MAY** support entering a response with non-ASCII characters. See Section 5 for instructions how the input should be processed and encoded into octets.

Security Claims (see Section 7.2):

Auth. mechanism:	Hardware token.
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	No
Key strength:	N/A
Dictionary attack prot.:	No
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	No
Channel binding:	No

5.7. Expanded Types

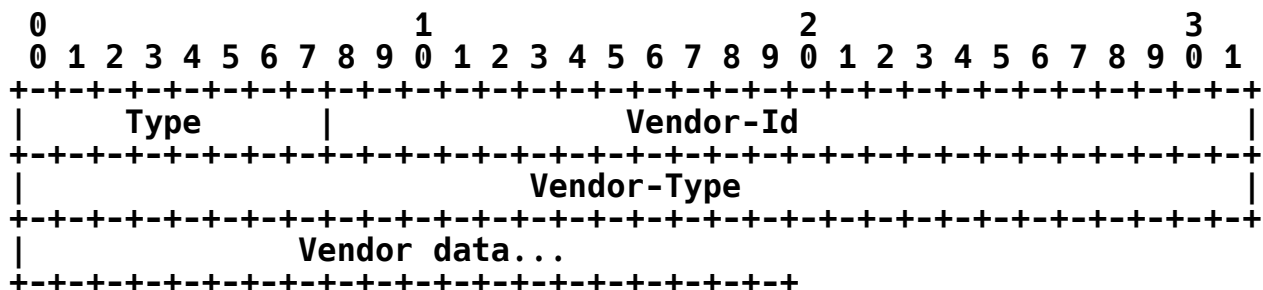
Description

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 space of $2^{32}-1$ possible Types. (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 an 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 the 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. There is one exception to this rule: Expanded Nak and Legacy Nak packets share the same Type, but must be treated differently because they have a different format.

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 [RFC3692].

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.

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".

For registration requests where a Designated Expert should be consulted, the responsible IESG area director should appoint the Designated Expert. The intention is that any allocation will be accompanied by a published RFC. But in order to allow for the allocation of values prior to the RFC being approved for publication, the Designated Expert can approve allocations once it seems clear that an RFC will be published. The Designated expert will post a request to the EAP WG mailing list (or a successor designated by the Area Director) for comment and review, including an Internet-Draft. 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, as well as informing IANA. 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 should be provided.

6.1. Packet Codes

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.

6.2. Method Types

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 1-45 have been allocated, with 20 available for re-use. Method Types 20 and 46-191 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, on the advice of a Designated Expert, with Specification Required.

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

7. Security Considerations

This section defines a generic threat model as well as the EAP method security claims mitigating those threats.

It is expected that the generic threat model and corresponding security claims will be used to define EAP method requirements for use in specific environments. An example of such a requirements analysis is provided in [IEEE-802.11i-req]. A security claims section is required in EAP method specifications, so that EAP methods can be evaluated against the requirements.

7.1. Threat Model

EAP was developed for use with PPP [RFC1661] and was later adapted for use in wired IEEE 802 networks [IEEE-802] in [IEEE-802.1X]. Subsequently, EAP has been proposed for use on wireless LAN networks and over the Internet. In all these situations, it is possible for an attacker to gain access to links over which EAP packets are transmitted. For example, attacks on telephone infrastructure are documented in [DECEPTION].

An attacker with access to the link may carry out a number of attacks, including:

- [1] An attacker may try to discover user identities by snooping authentication traffic.
- [2] An attacker may try to modify or spoof EAP packets.
- [3] An attacker may launch denial of service attacks by spoofing lower layer indications or Success/Failure packets, by replaying EAP packets, or by generating packets with overlapping Identifiers.
- [4] An attacker may attempt to recover the pass-phrase by mounting an offline dictionary attack.
- [5] An attacker may attempt to convince the peer to connect to an untrusted network by mounting a man-in-the-middle attack.
- [6] An attacker may attempt to disrupt the EAP negotiation in order to cause a weak authentication method to be selected.
- [7] An attacker may attempt to recover keys 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.
- [9] An attacker may attempt to perform downgrading attacks on lower layer ciphersuite negotiation in order to ensure that a weaker ciphersuite is used subsequently to EAP authentication.
- [10] An attacker acting as an authenticator may provide incorrect information to the EAP peer and/or server via out-of-band mechanisms (such as via a AAA or lower layer protocol). This includes impersonating another authenticator, or providing inconsistent information to the peer and EAP server.

Depending on the lower layer, these attacks may be carried out without requiring physical proximity. Where EAP is used over wireless networks, EAP packets may be forwarded by authenticators (e.g., pre-authentication) so that the attacker need not be within the coverage area of an authenticator in order to carry out an attack on it or its peers. Where EAP is used over the Internet, attacks may be carried out at an even greater distance.

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] Mechanism. This is a statement of the authentication technology: certificates, pre-shared keys, passwords, token cards, etc.
- [b] Security claims. This is a statement of the claimed security properties of the method, using terms defined in Section 7.2.1: mutual authentication, integrity protection, replay protection, confidentiality, key derivation, dictionary attack resistance, fast reconnect, cryptographic binding. 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.
- [c] 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 a 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, on average, an effort comparable to $2^{(N-1)}$ operations of a typical block cipher. The statement **SHOULD** be accompanied by a short rationale, explaining how this number was derived. This explanation **SHOULD** include the parameters required to achieve the stated key strength 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 a long-term secret), and possibly some public information such as nonces, the effective key strength is limited by the strength 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.

- [d] Description of key hierarchy. EAP methods deriving keys **MUST** either provide a reference to a key hierarchy specification, or describe how Master Session Keys (MSKs) and Extended Master Session Keys (EMSKs) are to be derived.
- [e] Indication of vulnerabilities. In addition to the security claims that are made, the specification **MUST** indicate which of the security claims detailed in Section 7.2.1 are **NOT** being made.

7.2.1. Security Claims Terminology for EAP Methods

These terms are used to describe the security properties of EAP methods:

Protected ciphersuite negotiation

This refers to the ability of an EAP method to negotiate the ciphersuite used to protect the EAP conversation, as well as to integrity protect the negotiation. It does not refer to the ability to negotiate the ciphersuite used to protect data.

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 an EAP method or its messages, including success and failure result indications.

Confidentiality

This refers to encryption of EAP messages, including EAP Requests and Responses, and success and failure result indications. A method making this claim MUST support identity protection (see Section 7.3).

Key derivation

This refers to the ability of the EAP method to derive exportable keying material, such as the Master Session Key (MSK), and Extended Master Session Key (EMSK). The MSK is used only for further key derivation, not directly for protection of the EAP conversation or subsequent data. Use of the EMSK is reserved.

Key strength

If the effective key strength is N bits, the best currently known methods to recover the key (with non-negligible probability) require, on average, an effort comparable to $2^{(N-1)}$ operations of a typical block cipher.

Dictionary attack resistance

Where password authentication is used, passwords are commonly selected from a small set (as compared to a set of N-bit keys), which raises a concern about dictionary attacks. A method may be said to provide protection against dictionary attacks if, when it uses a password as a secret, the method does not allow an offline attack that has a work factor based on the number of passwords in an attacker's dictionary.

Fast reconnect

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

Cryptographic binding

The demonstration of the EAP peer to the EAP server that a single entity has acted as the EAP peer for all methods executed within a tunnel method. Binding MAY also imply that the EAP server demonstrates to the peer that a single entity has acted as the EAP server for all methods executed within a tunnel method. If executed correctly, binding serves to mitigate man-in-the-middle vulnerabilities.

Session independence

The demonstration that passive attacks (such as capture of the EAP conversation) or active attacks (including compromise of the MSK or EMSK) does not enable compromise of subsequent or prior MSKs or EMSKs.

Fragmentation

This refers to whether an EAP method supports fragmentation and reassembly. As noted in Section 3.1, EAP methods should support fragmentation and reassembly if EAP packets can exceed the minimum MTU of 1020 octets.

Channel binding

The communication within an EAP method of integrity-protected channel properties such as endpoint identifiers which can be compared to values communicated via out of band mechanisms (such as via a AAA or lower layer protocol).

Note: This list of security claims is not exhaustive. Additional properties, such as additional denial-of-service protection, may be relevant as well.

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 use a method-specific identity exchange 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 EAP-Response/Identity 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 EAP-Response/Identity where proxies or relays are present, the realm portion may be required.

It is possible for the identity in the identity response to be different from the identity authenticated by the EAP method. This may be intentional in the case of identity privacy. An EAP method **SHOULD** use the authenticated identity when making access control decisions.

7.4. Man-in-the-Middle Attacks

Where EAP is tunneled within another protocol that omits peer authentication, there exists a potential vulnerability to a man-in-the-middle attack. For details, see [BINDING] and [MITM].

As noted in Section 2.1, EAP does not permit untunneled sequences of authentication methods. Were a sequence of EAP authentication methods to be permitted, 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.

Tunneling EAP within another protocol 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 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. For further details on cryptographic binding, see [BINDING].

- [c] Limiting the EAP methods authorized for use without protection, based on peer and authenticator policy.
- [d] Avoiding the use of tunnels when a single, strong method is available.

7.5. Packet Modification Attacks

While EAP methods may support per-packet data origin authentication, integrity, and replay protection, support is not provided within the EAP layer.

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 (Code, Identifier, Length, Type) within EAP packets where the header is unprotected. This could cause packets to be inappropriately discarded or misinterpreted.

To protect EAP packets against modification, spoofing, or replay, methods supporting protected ciphersuite negotiation, mutual authentication, and key derivation, as well as integrity and replay protection, are recommended. See Section 7.2.1 for definitions of these security claims.

Method-specific MICs may be used to provide protection. 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.

It is **RECOMMENDED** that methods providing integrity protection of EAP packets include coverage of all the EAP header fields, including the Code, Identifier, Length, Type, and Type-Data fields.

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 [IKEv2] or within TLS [RFC2246]. However, as noted in Section 7.4, EAP tunneling may result in a man-in-the-middle vulnerability.

Existing EAP methods 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.

Within EAP-TLS [RFC2716], a MIC validation failure 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.

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]; MS-CHAPv2 vulnerabilities are documented in [PPTPv2]; Kerberos vulnerabilities are described in [KRBATTACK], [KRBLIM], and [KERB4WEAK].

In order to protect against dictionary attacks, authentication methods resistant to dictionary attacks (as defined in Section 7.2.1) are recommended.

If an authentication algorithm is used that is known to be vulnerable to dictionary attacks, 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 peer does not authenticate the authenticator, making the peer vulnerable to attack by a rogue authenticator. Methods supporting mutual authentication (as defined in Section 7.2.1) address this vulnerability.

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 Nak Response packets, although it is possible for a method to include coverage of Nak Responses within a method-specific MIC.

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. 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.

7.9. Implementation Idiosyncrasies

The interaction of EAP with lower layers 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 an authentication failure will result in denied access to the controlled port in [IEEE-802.1X], 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 keys. In order to provide keying material for use in a subsequently negotiated ciphersuite, an EAP method supporting key derivation **MUST** export a Master Session Key (MSK) of at least 64 octets, and an Extended Master Session Key (EMSK) of at least 64 octets. EAP Methods deriving keys **MUST** provide for mutual authentication between the EAP peer and the EAP Server.

The MSK and EMSK **MUST NOT** be used directly to protect data; however, they are of sufficient size to enable derivation of a AAA-Key subsequently used to derive Transient Session Keys (TSKs) for use with the selected ciphersuite. Each ciphersuite is responsible for specifying how to derive the TSKs from the AAA-Key.

The AAA-Key is derived from the keying material exported by the EAP method (MSK and EMSK). This derivation occurs on the AAA server. In many existing protocols that use EAP, the AAA-Key and MSK are equivalent, but more complicated mechanisms are possible (see [KEYFRAME] for details).

EAP methods **SHOULD** ensure the freshness of the MSK and EMSK, even in cases where one party may not have a high quality random number generator. A **RECOMMENDED** method is for each party to provide a nonce of at least 128 bits, used in the derivation of the MSK and EMSK.

EAP methods export the MSK and EMSK, but not Transient Session Keys so as to allow EAP methods to be ciphersuite and media independent. Keying material exported by EAP methods **MUST** be independent of the ciphersuite negotiated to protect data.

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 be used with a wide range of ciphersuites and media.

In order to preserve algorithm independence, EAP methods deriving keys **SHOULD** support (and document) the protected negotiation of the ciphersuite used to protect the EAP conversation between the peer and server. This is distinct from the ciphersuite negotiated between the peer and authenticator, used to protect data.

The strength of Transient Session Keys (TSKs) used to protect data is ultimately dependent on the strength of keys generated by the EAP method. If an EAP method cannot produce keying material of sufficient strength, then the TSKs may be subject to a brute force

attack. In order to enable deployments requiring strong keys, EAP methods supporting key derivation **SHOULD** be capable of generating an MSK and EMSK, each with an effective key strength of at least 128 bits.

Methods supporting key derivation **MUST** demonstrate cryptographic separation between the MSK and EMSK branches of the EAP key hierarchy. Without violating a fundamental cryptographic assumption (such as the non-invertibility of a one-way function), an attacker recovering the MSK or EMSK **MUST NOT** be able to recover the other quantity with a level of effort less than brute force.

Non-overlapping substrings of the MSK **MUST** be cryptographically separate from each other, as defined in Section 7.2.1. That is, knowledge of one substring **MUST NOT** help in recovering some other substring without breaking some hard cryptographic assumption. This is required because some existing ciphersuites form TSKs by simply splitting the AAA-Key to pieces of appropriate length. Likewise, non-overlapping substrings of the EMSK **MUST** be cryptographically separate from each other, and from substrings of the MSK.

The EMSK is reserved for future use and **MUST** remain on the EAP peer and EAP server where it is derived; it **MUST NOT** be transported to, or shared with, additional parties, or used to derive any other keys. (This restriction will be relaxed in a future document that specifies how the EMSK can be used.)

Since EAP does not provide for explicit key lifetime negotiation, EAP peers, authenticators, and authentication servers **MUST** be prepared for situations in which one of the parties discards the key state, which remains valid on another party.

This specification does not provide detailed guidance on how EAP methods derive the MSK and EMSK, how the AAA-Key is derived from the MSK and/or EMSK, or how the TSKs are derived from the AAA-Key.

The development and validation of key derivation algorithms is difficult, and as a result, EAP methods **SHOULD** re-use well established and analyzed mechanisms for key derivation (such as those specified in IKE [RFC2409] or TLS [RFC2246]), rather than inventing new ones. EAP methods **SHOULD** also utilize well established and analyzed mechanisms for MSK and EMSK derivation. Further details on EAP Key Derivation are provided within [KEYFRAME].

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.

To protect against data modification, spoofing, or snooping, it is recommended that EAP methods supporting mutual authentication and key derivation (as defined by Section 7.2.1) be used, along with lower layers providing per-packet confidentiality, authentication, integrity, and replay protection.

Additionally, if the lower layer performs ciphersuite negotiation, it should be understood that EAP does not provide by itself integrity protection of that negotiation. Therefore, in order to avoid downgrading attacks which would lead to weaker ciphersuites being used, clients implementing lower layer ciphersuite negotiation **SHOULD** protect against negotiation downgrading.

This can be done by enabling users to configure which ciphersuites are acceptable as a matter of security policy, or the ciphersuite negotiation **MAY** be authenticated using keying material derived from the EAP authentication and a MIC algorithm agreed upon in advance by lower-layer peers.

7.12. Link Layer

There are reliability and security issues with link layer indications in PPP, IEEE 802 LANs, 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 link.
- [b] IEEE 802. IEEE 802.1X EAPOL-Start and EAPOL-Logoff frames are not authenticated or integrity protected. They can therefore be spoofed by an attacker with access to the link.
- [c] IEEE 802.11. In IEEE 802.11, link layer indications include Disassociate and Deauthenticate frames (link failure indications), and the first message of the 4-way handshake (link success indication). 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 may be sent as Class 3 unicast data frames, and 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.

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.

7.13. Separation of Authenticator and Backend Authentication Server

It is possible for the EAP peer and EAP server to mutually authenticate and derive a AAA-Key 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 client to derive the AAA-Key from the MSK and EMSK exported by the EAP method, and to subsequently pass a Transient Session Key (TSK) to the ciphersuite module.

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

- [a] Authentication will occur between the peer and the authentication server, not between the peer and the authenticator. This means that it is not possible for the peer to validate the identity of the authenticator that it is speaking to, using EAP alone.
- [b] As discussed in [RFC3579], 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 implies that the AAA protocol spoken between the authenticator and authentication server MUST support per-packet authentication, integrity, and replay protection.
- [c] After completion of the EAP conversation, where lower layer security services such as per-packet confidentiality, authentication, integrity, and replay protection will be enabled, a secure association protocol SHOULD be run between the peer and authenticator in order to provide mutual authentication between

the peer and authenticator, guarantee liveness of transient session keys, provide protected ciphersuite and capabilities negotiation for subsequent data, and synchronize key usage.

- [d] A AAA-Key derived from the MSK and/or EMSK negotiated between the peer and authentication server MAY be transmitted to the authenticator. Therefore, a mechanism needs to be provided to transmit the AAA-Key from the authentication server to the authenticator that needs it. The specification of the AAA-key derivation, transport, and wrapping mechanisms is outside the scope of this document. Further details on AAA-Key Derivation are provided within [KEYFRAME].

7.14. Cleartext Passwords

This specification does not define a mechanism for cleartext password authentication. The omission is intentional. Use of cleartext passwords would allow the password to be captured by an attacker with access to a link over which EAP packets are transmitted.

Since protocols encapsulating EAP, such as RADIUS [RFC3579], may not provide confidentiality, EAP packets may be subsequently encapsulated for transport over the Internet where they may be captured by an attacker.

As a result, cleartext passwords cannot be securely used within EAP, except where encapsulated within a protected tunnel with server authentication. Some of the same risks apply to EAP methods without dictionary attack resistance, as defined in Section 7.2.1. For details, see Section 7.6.

7.15. Channel Binding

It is possible for a compromised or poorly implemented EAP authenticator to communicate incorrect information to the EAP peer and/or server. This may enable an authenticator to impersonate another authenticator or communicate incorrect information via out-of-band mechanisms (such as via a AAA or lower layer protocol).

Where EAP is used in pass-through mode, the EAP peer typically does not verify the identity of the pass-through authenticator, it only verifies that the pass-through authenticator is trusted by the EAP server. This creates a potential security vulnerability.

Section 4.3.7 of [RFC3579] describes how an EAP pass-through authenticator acting as a AAA client can be detected if it attempts to impersonate another authenticator (such by sending incorrect NAS-Identifier [RFC2865], NAS-IP-Address [RFC2865] or NAS-IPv6-Address

[RFC3162] attributes via the AAA protocol). However, it is possible for a pass-through authenticator acting as a AAA client to provide correct information to the AAA server while communicating misleading information to the EAP peer via a lower layer protocol.

For example, it is possible for a compromised authenticator to utilize another authenticator's Called-Station-Id or NAS-Identifier in communicating with the EAP peer via a lower layer protocol, or for a pass-through authenticator acting as a AAA client to provide an incorrect peer Calling-Station-Id [RFC2865][RFC3580] to the AAA server via the AAA protocol.

In order to address this vulnerability, EAP methods may support a protected exchange of channel properties such as endpoint identifiers, including (but not limited to): Called-Station-Id [RFC2865][RFC3580], Calling-Station-Id [RFC2865][RFC3580], NAS-Identifier [RFC2865], NAS-IP-Address [RFC2865], and NAS-IPv6-Address [RFC3162].

Using such a protected exchange, it is possible to match the channel properties provided by the authenticator via out-of-band mechanisms against those exchanged within the EAP method. Where discrepancies are found, these SHOULD be logged; additional actions MAY also be taken, such as denying access.

7.16. Protected Result Indications

Within EAP, Success and Failure packets are neither acknowledged nor integrity protected. Result indications improve resilience to loss of Success and Failure packets when EAP is run over lower layers which do not support retransmission or synchronization of the authentication state. In media such as IEEE 802.11, which provides for retransmission, as well as synchronization of authentication state via the 4-way handshake defined in [IEEE-802.11i], additional resilience is typically of marginal benefit.

Depending on the method and circumstances, result indications can be spoofable by an attacker. A method is said to provide protected result indications if it supports result indications, as well as the "integrity protection" and "replay protection" claims. A method supporting protected result indications MUST indicate which result indications are protected, and which are not.

Protected result indications are not required to protect against rogue authenticators. Within a mutually authenticating method, requiring that the server authenticate to the peer before the peer will accept a Success packet prevents an attacker from acting as a rogue authenticator.

However, it is possible for an attacker to forge a Success packet after the server has authenticated to the peer, but before the peer has authenticated to the server. If the peer were to accept the forged Success packet and attempt to access the network when it had not yet successfully authenticated to the server, a denial of service attack could be mounted against the peer. After such an attack, if the lower layer supports failure indications, the authenticator can synchronize state with the peer by providing a lower layer failure indication. See Section 7.12 for details.

If a server were to authenticate the peer and send a Success packet prior to determining whether the peer has authenticated the authenticator, an idle timeout can occur if the authenticator is not authenticated by the peer. Where supported by the lower layer, an authenticator sensing the absence of the peer can free resources.

In a method supporting result indications, a peer that has authenticated the server does not consider the authentication successful until it receives an indication that the server successfully authenticated it. Similarly, a server that has successfully authenticated the peer does not consider the authentication successful until it receives an indication that the peer has authenticated the server.

In order to avoid synchronization problems, prior to sending a success result indication, it is desirable for the sender to verify that sufficient authorization exists for granting access, though, as discussed below, this is not always possible.

While result indications may enable synchronization of the authentication result between the peer and server, this does not guarantee that the peer and authenticator will be synchronized in terms of their authorization or that timeouts will not occur. For example, the EAP server may not be aware of an authorization decision made by a AAA proxy; the AAA server may check authorization only after authentication has completed successfully, to discover that authorization cannot be granted, or the AAA server may grant access but the authenticator may be unable to provide it due to a temporary lack of resources. In these situations, synchronization may only be achieved via lower layer result indications.

Success indications may be explicit or implicit. For example, where a method supports error messages, an implicit success indication may be defined as the reception of a specific message without a preceding error message. Failures are typically indicated explicitly. As described in Section 4.2, a peer silently discards a Failure packet received at a point where the method does not explicitly permit this

to be sent. For example, a method providing its own error messages might require the peer to receive an error message prior to accepting a Failure packet.

Per-packet authentication, integrity, and replay protection of result indications protects against spoofing. Since protected result indications require use of a key for per-packet authentication and integrity protection, methods supporting protected result indications **MUST** also support the "key derivation", "mutual authentication", "integrity protection", and "replay protection" claims.

Protected result indications address some denial-of-service vulnerabilities due to spoofing of Success and Failure packets, though not all. EAP methods can typically provide protected result indications only in some circumstances. For example, errors can occur prior to key derivation, and so it may not be possible to protect all failure indications. It is also possible that result indications may not be supported in both directions or that synchronization may not be achieved in all modes of operation.

For example, within EAP-TLS [RFC2716], in the client authentication handshake, the server authenticates the peer, but does not receive a protected indication of whether the peer has authenticated it. In contrast, the peer authenticates the server and is aware of whether the server has authenticated it. In the session resumption handshake, the peer authenticates the server, but does not receive a protected indication of whether the server has authenticated it. In this mode, the server authenticates the peer and is aware of whether the peer has authenticated it.

8. Acknowledgements

This protocol derives much of its inspiration from Dave Carrel's AHA document, 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, Bill Arbaugh, Nick Petroni and Bryan Payne of the University of Maryland, Steve Bellovin of AT&T Research, Paul Funk of Funk Software, Pasi Eronen of Nokia, Joseph Salowey of Cisco, Paul Congdon of HP, and members of the EAP working group.

The use of Security Claims sections for EAP methods, as required by Section 7.2 and specified for each EAP method described in this document, was inspired by Glen Zorn through [EAP-EVAL].

9. References

9.1. 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.
- [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] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Overview and Architecture", IEEE Standard 802, 1990.
- [IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X, September 2001.

9.2. Informative References

- [RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC1510] Kohl, J. and B. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.
- [RFC1750] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.
- [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.
- [RFC2408] Maughan, D., Schneider, M. and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, 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.
- [RFC2865] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

- [RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.
- [RFC3162] Aboba, B., Zorn, G. and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.
- [RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.
- [RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.
- [RFC3580] Congdon, P., Aboba, B., Smith, A., Zorn, G. and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines", RFC 3580, September 2003.
- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, January 2004.
- [DECEPTION] Slatalla, M. and J. Quittner, "Masters of Deception", Harper-Collins, New York, 1995.
- [KRBATTACK] Wu, T., "A Real-World Analysis of Kerberos Password Security", Proceedings of the 1999 ISOC Network and Distributed System Security Symposium, <http://www.isoc.org/isoc/conferences/ndss/99/proceedings/papers/wu.pdf>.
- [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", Work in Progress, October 2002.
- [IKEv2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", Work in Progress, January 2004.
- [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-802.11] Institute of Electrical and Electronics Engineers, "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>.
- [KEYFRAME] Aboba, B., "EAP Key Management Framework", Work in Progress, October 2003.
- [SASLPREP] Zeilenga, K., "SASLprep: Stringprep profile for user names and passwords", Work in Progress, March 2004.
- [IEEE-802.11i] Institute of Electrical and Electronics Engineers, "Unapproved Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security", IEEE Draft 802.11i (work in progress), 2003.
- [DIAM-EAP] Eronen, P., Hiller, T. and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", Work in Progress, February 2004.
- [EAP-EVAL] Zorn, G., "Specifying Security Claims for EAP Authentication Types", Work in Progress, October 2002.

- [BINDING] Puthenkulam, J., "The Compound Authentication Binding Problem", Work in Progress, October 2003.
- [MITM] Asokan, N., Niemi, V. and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", IACR ePrint Archive Report 2002/163, October 2002, <<http://eprint.iacr.org/2002/163>>.
- [IEEE-802.11i-req] Stanley, D., "EAP Method Requirements for Wireless LANs", Work in Progress, February 2004.
- [PPTPv2] Schneier, B. and Mudge, "Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)", CQRE 99, Springer-Verlag, 1999, pp. 192-203.

Appendix A. 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.

- o The Terminology section (Section 1.2) has been expanded, defining more concepts and giving more exact definitions.
- o The concepts of Mutual Authentication, Key Derivation, and Result Indications are introduced and discussed throughout the document where appropriate.
- o 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 be used and how it may not be used is specified in detail in Section 2.1.
- o Also in Section 2, some requirements have been made explicit for the authenticator when acting in pass-through mode.
- o An EAP multiplexing model (Section 2.2) has been added to illustrate a typical implementation of EAP. There is no requirement that an implementation conform to this model, as long as the on-the-wire behavior is consistent with it.
- o 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.
- o In the description of the EAP Request and Response interaction (Section 4.1), both the behavior on receiving duplicate requests, and when packets should be silently discarded has been more exactly specified. The implementation notes in this section have been substantially expanded.
- o 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.
- o 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.

- o In Sections 5, 5.1, and 5.2, a requirement has been added such that fields with displayable messages should contain UTF-8 encoded ISO 10646 characters.
- o It is now required in Section 5.1 that if the Type-Data field of an Identity Request contains a NUL-character, only the part before the null is displayed. RFC 2284 prohibits the null termination of the Type-Data field of Identity messages. This rule has been relaxed for Identity Request messages and the Identity Request Type-Data field may now be null terminated.
- o In Section 5.5, support for OTP Extended Responses [RFC2243] has been added to EAP OTP.
- o An IANA Considerations section (Section 6) has been added, giving registration policies for the numbering spaces defined for EAP.
- o The Security Considerations (Section 7) have been greatly expanded, giving a much more comprehensive coverage of possible threats and other security considerations.
- o In Section 7.5, text has been added on method-specific behavior, providing guidance on how EAP method-specific integrity checks should be processed. Where possible, it is desirable for a method-specific MIC to be computed over the entire EAP packet, including the EAP layer header (Code, Identifier, Length) and EAP method layer header (Type, Type-Data).
- o In Section 7.14 the security risks involved in use of cleartext passwords with EAP are described.
- o In Section 7.15 text has been added relating to detection of rogue NAS behavior.

Authors' Addresses

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

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

EMail: jrv@umich.edu

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 Levkowitz
ipUnplugged AB
Arenavagen 33
Stockholm S-121 28
SWEDEN

Phone: +46 708 32 16 08
EMail: henrik@levkowitz.com

Full 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.

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.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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