

EAP Working Group
INTERNET-DRAFT
Category: Informational
<[draft-aboba-pppext-key-problem-07.txt](#)>
9 August 2003

Bernard Aboba
Dan Simon
Microsoft

EAP Key Management Framework

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC 2026](#).

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

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

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

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

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This document provides a framework for EAP key management, including a statement of applicability and guidelines for generation, transport and usage of EAP keying material. Algorithms for key derivation or mechanisms for key transport are not specified in this document. Rather, this document provides a framework within which algorithms and transport mechanisms can be discussed and evaluated.

Table of Contents

1.	Introduction	3
1.1	Requirements Language	3
1.2	Terminology	3
1.3	Conversation Overview	5
2.	EAP key hierarchy	9
2.1	EAP Invariants	9
2.2	Key Hierarchy	11
2.3	Exchanges	14
2.4	Security Relationships	18
3.	Security associations	19
3.1	EAP SA	19
3.2	AAA-Key SA	20
3.3	Unicast Secure Association SA	21
3.4	Multicast Secure Association SA	22
3.5	Key Naming	23
4.	Threat model	25
4.1	Security Assumptions	25
4.2	Security Requirements	26
5.	IANA Considerations	32
6.	Security Considerations	32
6.1	Key Strength	32
6.2	Key Wrap	32
6.3	Man-in-the-middle Attacks	33
6.4	Impersonation	33
7.	References	34
7.1	Normative References	34
7.2	Informative References	35
Appendix A	- Ciphersuite Keying Requirements	39
Appendix B	- TEK Hierarchy	40
Appendix C	- MSK and EMSK Hierarchy	41
Appendix D	- TSK Derivation	43
Appendix E	- AAA-Key Key Derivation	44
	Acknowledgments	44
	Author's Addresses	44
	Intellectual Property Statement	45
	Full Copyright Statement	45

1. Introduction

The Extensible Authentication Protocol (EAP), defined in [[RFC2284bis](#)], was designed to enable extensible authentication for network access in situations in which the IP protocol is not available. Originally developed for use with PPP [[RFC1661](#)], it has subsequently also been applied to IEEE 802 wired networks [[IEEE8021X](#)].

This document provides a framework for the generation, transport and usage of keying material generated by EAP authentication algorithms, known as "methods". Since in EAP keying material is generated by EAP methods, transported by AAA protocols, transformed into session keys by secure association protocols and used by link layer ciphersuites, it is necessary to describe each of these elements and provide a system-level security analysis.

The document is organized as follows:

[Section 1](#) provides an introduction and defines terminology.

[Section 2](#) describes the EAP key hierarchy and exchanges.

[Section 3](#) describes EAP security associations and key naming.

[Section 4](#) describes the threat model and security requirements.

[Section 5](#) describes the IANA considerations.

[Section 6](#) describes security considerations.

[Section 7](#) provides references.

Appendix A summarizes the keying requirements for link layer ciphersuites.

Appendix B provides an example transient EAP key (TEK) hierarchy.

Appendix C provides an example MSK and EMSK hierarchy.

Appendix D provides an example Transient Session Key (TSK) derivation.

Appendix E describes AAA-Key derivation, including fast handoff.

1.1. Requirements Language

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)].

1.2. Terminology

This document frequently uses the following terms:

authenticator

The end of the link initiating EAP authentication. Where no backend authentication server is present, the authenticator acts as the EAP server, terminating the EAP conversation with the peer.

Where a backend authentication server is present, the authenticator

may act as a pass-through for one or more authentication methods and for non-local users. This terminology is also used in [\[IEEE8021X\]](#), and has the same meaning in this document.

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 [\[IEEE8021X\]](#).

AAA-Token

The package within which keying material and one or more attributes is transported between the backend authentication server and the authenticator. The attributes provide the authenticator with usage context and key names suitable to bind the key to the appropriate context. For example, attributes might include the peer layer 2 address (Calling-Station-Id), the authenticator layer 2 address (Called-Station-Id) and IP address (NAS-IP-Address), the key name, etc. The format and wrapping of the AAA-Token, which is intended to be accessible only to the backend authentication server and authenticator, is defined by the AAA protocol.

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 sequence or tunnel. 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 sequence or tunnel. If executed correctly, binding serves to mitigate man-in-the-middle vulnerabilities.

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 exhaustive search for the secret state value.

EAP server

The entity which terminates EAP authentication with the peer is

known as the EAP server. Where pass-through is supported, the backend authentication server functions as the EAP server; where authentication occurs locally, the EAP server is the authenticator.

Key derivation

This refers to the ability of the EAP method to export a ciphersuite-independent Master Session Key (MSK), Extended Master Session Key (EMSK), and (optionally) an Initialization Vector (IV).

Key strength

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

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 one-way conversations, running in opposite directions do not provide mutual authentication as defined here.

peer The end of the link that responds to the authenticator. In [\[IEEE8021X\]](#), this end is known as the Supplicant.

1.3. Conversation Overview

Where EAP key derivation is supported, EAP authentication is typically a component of a three phase exchange:

Discovery phase (phase 0)

EAP authentication, key derivation and transport (phase 1)

Unicast and multicast secure association establishment (phase 2)

In the discovery phase (phase 0), the EAP peers locate each other and discover their capabilities. This can include an EAP peer locating an authenticator suitable for access to a particular network, or it could involve an EAP peer locating an authenticator behind a bridge with which it desires to establish a secure association. Typically the discovery phase takes place between the EAP peer and authenticator.

Once the EAP peer and authenticator discover each other, they authenticate using EAP (phase 1a). EAP enables deployment of new authentication methods without requiring development of new code on the authenticator. While the authenticator may implement some EAP methods locally and use those methods to authenticate local users, it may at the same time act as a pass-through for other users and methods, forwarding EAP packets back and forth between the backend authentication server and

the peer.

As described in [Section 2](#), in addition to supporting authentication, EAP methods may also support derivation of keying material for purposes including protection of the EAP conversation and subsequent data exchanges. EAP key derivation takes place between the EAP peer and EAP server, and methods supporting key derivation **MUST** also support mutual authentication. Where an authenticator server is present, it acts as the EAP server and transports derived keying material (known as the AAA-Key) to the authenticator (phase 1b). In 802.11 terminology, the first **32 octets of the AAA-Key is known as the Pairwise Master Key (PMK)**.

While EAP methods may be based on key management protocols, EAP itself is not a key management protocol. Thus, while EAP may provide for mutual authentication and derivation of keying material, it does not provide for the derivation or naming of transient session keys, the selection of traffic modes such as transport or tunnel mode, the secure negotiation of capabilities such as ciphersuites or filters, or support for key activation. As a result, where EAP is used for key derivation, a secure association protocol (phase 2) should be provided, supporting the creation and deletion of unicast (phase 2a) and multicast (phase 2b) security associations used for the protection of data.

The phases and the relationship between the parties is illustrated below.

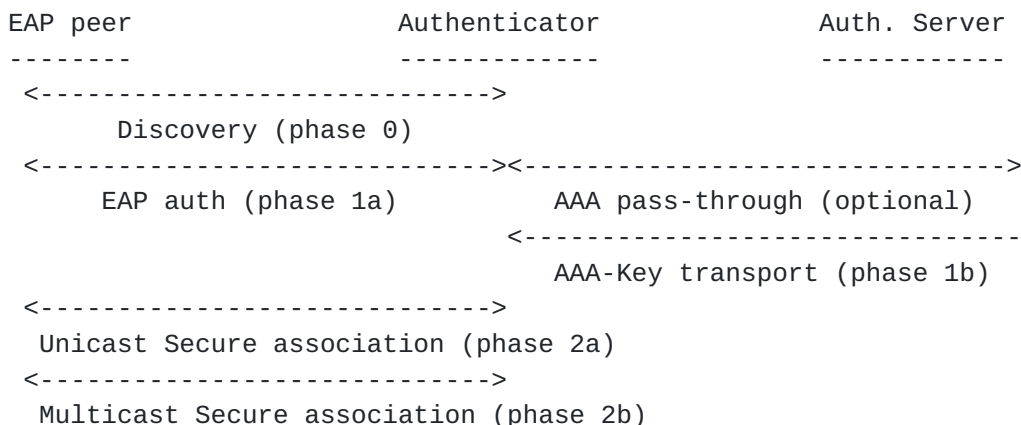


Figure 1 - Conversation Overview

1.3.1. Discovery Phase

In the peer discovery exchange (phase 0), the EAP peer and authenticator locate each other and discover each other's capabilities. For example, PPPoE [[RFC2516](#)] includes support for a Discovery Stage to allow a peer to identify the Ethernet MAC address of one or more authenticators and establish a PPPoE SESSION_ID. In [IEEE 80211], the

EAP peer (also known as the Station or STA) discovers the authenticator (Access Point or AP) and determines its capabilities using Beacon or Probe Request/Response frames. Since device discovery is handled outside of EAP, there is no need to provide this functionality within EAP.

Device discovery can occur manually or automatically. In EAP implementations running over PPP, the EAP peer might be configured with a phone book providing phone numbers of authenticators and associated capabilities such as supported rates, authentication protocols or ciphersuites.

Since device discovery can occur prior to authentication and key derivation, it may not be possible for the discovery phase to be protected using keying material derived during an authentication exchange. As a result, device discovery protocols may be insecure, leaving them vulnerable to spoofing unless the discovered parameters can subsequently be securely verified.

1.3.2. Authentication Phase

Once the EAP peer and authenticator discover each other, they authenticate using EAP (phase 1a). Typically, the peer desires access to the network, and the authenticators are Network Access Servers (NASes) providing that access. In such a situation, access to the network can be provided by any authenticator attaching to the desired network, and the EAP peer is typically willing to send data traffic through any authenticator that can demonstrate that it is authorized to provide access to the desired network.

An EAP authenticator may handle the authentication locally, or it may act as a pass-through to a backend authentication server. In the latter case the EAP exchange occurs between the EAP peer and a backend authenticator server, with the authenticator forwarding EAP packets between the two. The entity which terminates EAP authentication with the peer is known as the EAP server. Where pass-through is supported, the backend authentication server functions as the EAP server; where authentication occurs locally, the EAP server is the authenticator. Where a backend authentication server is present, at the successful completion of an authentication exchange, the AAA-Key is transported to the authenticator (phase 1b).

EAP may also be used when it is desired for two network devices (e.g. two switches or routers) to authenticate each other, or where two peers desire to authenticate each other and set up a secure association suitable for protecting data traffic.

EAP supports either one-way authentication (in which the peer authenticates to the EAP server), or mutual authentication (in which the peer and EAP server mutually authenticate). In either case, it can be assumed that the parties do not utilize the link to exchange data traffic unless their authentication requirements have been met. For example, a peer completing mutual authentication with an EAP server will not send data traffic over the link until the EAP server has authenticated successfully to the peer, and a secure association has been negotiated.

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

Successful completion of EAP authentication and key derivation by an EAP peer and EAP server does not necessarily imply that the peer is committed to joining the network associated with an EAP server. Rather, this commitment is implied by the creation of a security association between the EAP peer and authenticator, as part of the secure association protocol (phase 2). As a result, EAP may be used for "pre-authentication" in situations where it is necessary to pre-establish EAP security associations in order to decrease handoff or roaming latency.

1.3.3. Secure Association Phase

While EAP methods may be based on key management protocols, EAP itself is not a protocol for negotiation of security associations. While EAP methods supporting key derivation provide for mutual authentication and creation of EAP (phase 1) security associations, in order to preserve media independence, they typically do not support generation of transient session keys or negotiation of ciphersuites used in the protection of data. As a result, where EAP is used for key derivation, a secure association protocol (phase 2) should be provided, supporting the creation and deletion of phase 2 security associations used for the protection of data.

The secure association phase (phase 2) always occurs after the completion of EAP authentication (phase 1a) and key transport (phase 1b), and typically supports the following features:

- [1] The secure negotiation of capabilities. This includes usage modes, session parameters and ciphersuites, and required filters, including confirmation of the capabilities discovered during phase 0. By securely negotiating session parameters, the secure association protocol protects against spoofing during the discovery phase and ensures that the peer and authenticator are in agreement about how data is to be secured.

- [2] Generation of fresh transient session keys. This is typically accomplished via the exchange of nonces within the secure association protocol, and includes generation of both unicast (phase 2a) and multicast (phase 2b) session keys. While multicast traffic may only pass in one direction in certain cases (such as in IEEE 802.11 infrastructure mode, where only the Access Point sends multicast traffic), in other cases (such as IEEE 802.11 adhoc mode), both endpoints may send multicast traffic. By not using the AAA-Key directly to protect data, the secure association protocol protects against compromise of the AAA-Key, and by guaranteeing the freshness of transient session key, assures that session keys are not reused.
- [3] Key activation and deletion.
- [4] Mutual proof of possession of the keying material generated during EAP authentication (phase 1). By requiring a mutual proof of possession of the AAA-Key, the secure association protocol demonstrates that both the EAP peer and authenticator have been authenticated and authorized by the AAA server. Note that mutual proof of possession is not the same thing as mutual authentication. For example, as a result of a secure association protocol exchange, the EAP peer may not be able to confirm the identity of the authenticator.

2. EAP Key Hierarchy

2.1. EAP Invariants

The EAP key management framework assumes that certain basic characteristics, known as the "EAP Invariants" hold true for all implementations of EAP. These include:

- Media independence
- Method independence
- Ciphersuite independence

2.1.1. Media Independence

As described in [\[RFC2284bis\]](#), EAP authentication can run over multiple lower layers, including PPP [\[RFC1661\]](#) and IEEE 802 wired networks [\[IEEE8021X\]](#). Use with IEEE 802.11 wireless LANs is also contemplated [\[IEEE80211i\]](#). Since EAP methods cannot be assumed to have knowledge of the lower layer over which they are transported, EAP methods can function on any lower layer meeting the criteria outlined in [\[RFC2284bis\]](#), Section 3.1.

2.1.2. Method Independence

Supporting pass-through of authentication to the backend authentication server enables the authenticator to support any authentication method implemented on the backend authentication server and peer, not just locally implemented methods.

This implies that the authenticator need not implement code for each EAP method required by authenticating peers. In fact, the authenticator is not required to implement any EAP methods at all, nor cannot it be assumed to implement code specific to any EAP method.

This is useful where there is no single EAP method that is both mandatory-to-implement and offers acceptable security for the media in use. For example, the [[RFC2284bis](#)] mandatory-to-implement EAP method (MD5-Challenge) does not provide dictionary attack resistance, mutual authentication or key derivation, and as a result is not appropriate for use in wireless authentication.

2.1.3. Ciphersuite Independence

While EAP methods may negotiate the ciphersuite used in protection of the EAP conversation, the ciphersuite used for the protection of data is negotiated within the secure association protocol, out-of-band of EAP. The backend authentication server is not a party to this negotiation nor is it an intermediary in the data flow between the EAP peer and authenticator. The backend authentication server may not even have knowledge of the ciphersuites implemented by the peer and authenticator, or be aware of the ciphersuite negotiated between them, and therefore does not implement ciphersuite-specific code.

Since ciphersuite negotiation occurs in the secure association protocol, not in EAP, ciphersuite-specific key generation, if implemented within an EAP method, would potentially conflict with the transient session key derivation occurring in the secure association protocol. As a result, EAP methods generate keying material that is ciphersuite-independent. Additional advantages of ciphersuite-independence include:

Update requirements

If EAP methods were to specify how to derive transient session keys for each ciphersuite, they would need to be updated each time a new ciphersuite is developed. In addition, backend authentication servers might not be usable with all EAP-capable authenticators, since the backend authentication server would also need to be updated each time support for a new ciphersuite is added to the authenticator.

EAP method complexity

Requiring each EAP method to include ciphersuite-specific code for transient session key derivation would increase the complexity of each EAP method and would result in duplicated effort.

Ciphersuite negotiation

In practice, an EAP method may not have knowledge of the ciphersuite that has been negotiated between the peer and authenticator. In PPP, ciphersuite negotiation occurs in the Encryption Control Protocol (ECP) [[RFC1968](#)]. Since ECP negotiation occurs after authentication, unless an EAP method is utilized that supports ciphersuite negotiation, the peer, authenticator and backend authentication server may not be able to anticipate the negotiated ciphersuite and therefore this information cannot be provided to the EAP method. Since ciphersuite negotiation is assumed to occur out-of-band, there is no need for ciphersuite negotiation within EAP.

2.2. Key Hierarchy

The EAP keying hierarchy, illustrated in Figure 2, makes use of the following types of keys:

EAP Master key (MK)

A key derived between the EAP client and server during the EAP authentication process, and that is kept local to the EAP method and not exported or made available to a third party. Since the MK is a residue of a successful EAP authentication exchange, it is possible to shorten future EAP exchanges between an EAP peer and server by providing proof of MK possession, a technique known as "fast resume".

Master Session Key (MSK)

Keying material (at least 64 octets) that is derived between the EAP client and server and exported by the EAP method. Whenever a full EAP authentication is performed (not fast handoff), the MSK is chosen as the AAA-Key (see [Appendix E](#) for details).

AAA-Key

Where a backend authentication server is present, acting as an EAP server, keying material known as the AAA-Key is transported from the authentication server to the authenticator wrapped within the AAA-Token. The AAA-Key is used by the EAP peer and authenticator in the derivation of Transient Session Keys (TSKs) for the ciphersuite negotiated between the EAP peer and authenticator. As a result, the AAA-Key is typically known by all parties in the EAP exchange: the peer, authenticator and the authentication server (if present). AAA-Key derivation is discussed in [Appendix E](#).

Extended Master Session Key (EMSK)

Additional keying material (64 octets) derived between the EAP client and server that is exported by the EAP method. Unlike the MSK which is transported from the authentication server to the authenticator, the EMSK is known only to the EAP peer and server and is not provided to a third party. The EMSK therefore is not transported by the backend authentication server to the authenticator, although quantities derived from it may be used as the AAA-Key in situations in which EAP authentication is bypassed (e.g. fast handoff).

Currently the EMSK is reserved for future uses that are not defined yet. For example, it could be used to derive additional keying material for purposes such as fast handoff, man-in-the-middle vulnerability protection, etc.

Initialization Vector (IV)

A quantity of at least 64 octets, suitable for use in an initialization vector field, that is derived between the EAP client and server. Since the IV is a known value in methods such as EAP-TLS [[RFC2716](#)] it cannot be used in computation of any quantity that needs to remain secret, and is not used with any known ciphersuite. As a result, its use has been deprecated and EAP methods are not required to generate it.

Pairwise Master Key (PMK)

The AAA-Key is divided into two halves, the "Peer to Authenticator Encryption Key" (Enc-RECV-Key) and "Authenticator to Peer Encryption Key" (Enc-SEND-Key) (reception is defined from the point of view of the authenticator). Within [[IEEE80211i](#)] Octets 0-31 of the AAA-Key (Enc-RECV-Key) are known as the Pairwise Master Key (PMK). [[IEEE80211i](#)] ciphersuites derive their Transient Session Keys (TSKs) solely from the PMK, whereas the WEP ciphersuite, when used with [[IEEE8021X](#)], as noted in [[RFC3580](#)], derives its TSKs from both halves of the AAA-Key, the Enc-RECV-Key and the Enc-SEND-Key.

Transient EAP Keys (TEKs)

Session keys which are used to establish a protected channel between the EAP peer and server during the EAP authentication exchange. The TEKs are appropriate for use with the ciphersuite negotiated between EAP peer and server for use in protecting the EAP conversation. Note that the ciphersuite used to set up the protected channel between the EAP peer and server during EAP authentication is unrelated to the ciphersuite used to subsequently protect data sent between the EAP peer and authenticator. An example TEK key hierarchy is described in [Appendix C](#).

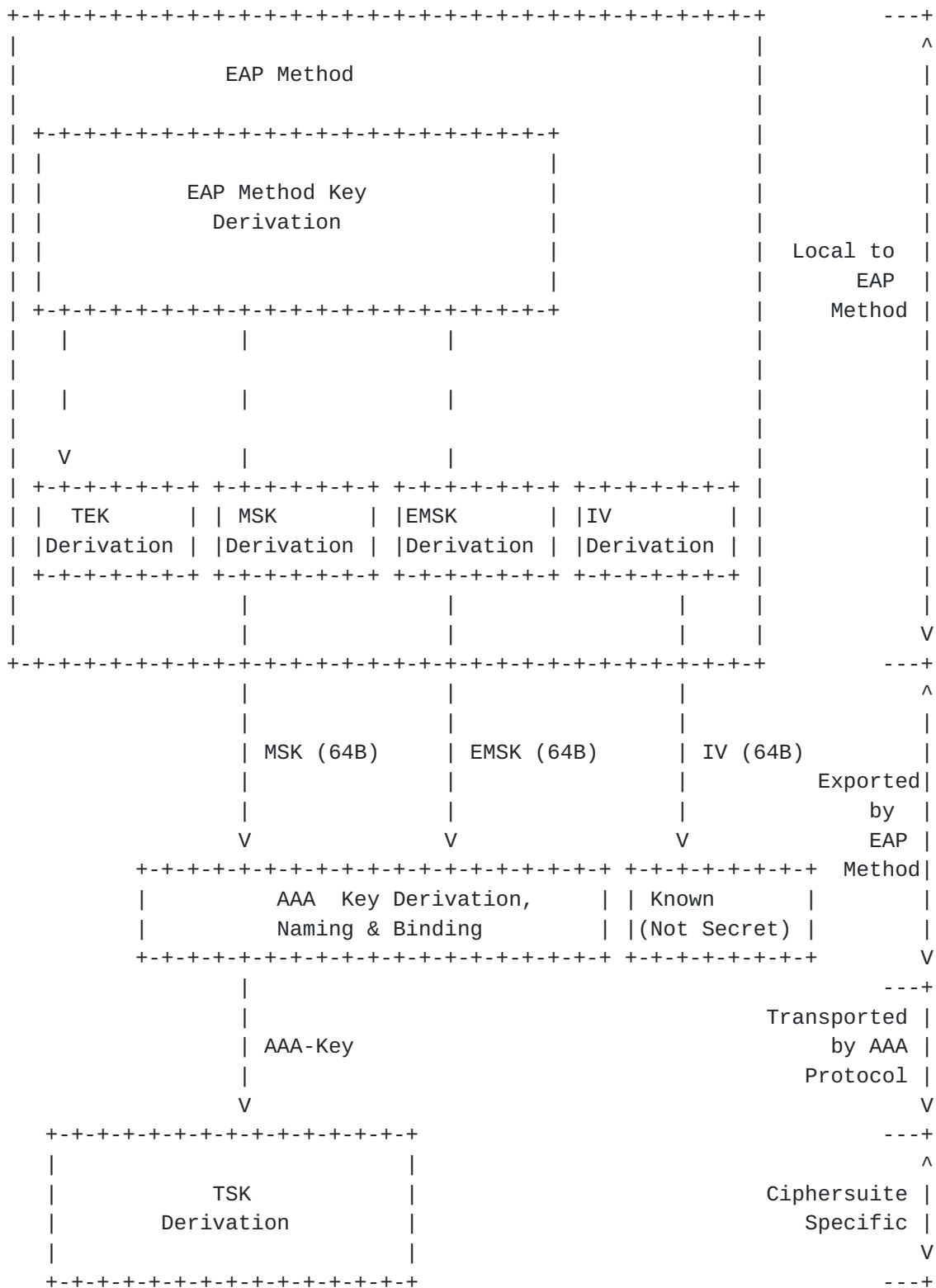


Figure 2 - EAP Key Hierarchy

Transient Session Keys (TSKs)

Session keys used to protect data which are appropriate for the ciphersuite negotiated between the EAP peer and authenticator. The TSKs are derived from the keying material included in the AAA-Token via the secure association protocol. In the case of IEEE 802.11, the role of the secure association protocol is handled by the 4-way handshake and group key derivation. An example TSK derivation is provided in [Appendix D](#).

[2.3.](#) Exchanges

EAP supports both a two party exchange between an EAP peer and an authenticator, as well as a three party exchange between an EAP peer, an authenticator and an EAP server.

Figure 3 illustrates the two party exchange. Here EAP is spoken between the peer and authenticator, encapsulated within a lower layer protocol, such as PPP, defined in [[RFC1661](#)] or IEEE 802, defined in [[IEEE802](#)].

Since the authenticator acts as an endpoint of the EAP conversation rather than a pass-through, EAP methods are implemented on the authenticator as well as the peer. If the EAP method negotiated between the EAP peer and authenticator supports mutual authentication and key derivation, the EAP Master Session Key (MSK) and Extended Master Session Key (EMSK) are derived on the EAP peer and authenticator and exported by the EAP method.

Where no backend authentication server is present, the MSK and EMSK are known only to the peer and authenticator and neither is transported to a third party. As demonstrated in [[RoamCERT](#)], despite the absence of a backend authentication server, such exchanges can support roaming between providers; it is even possible to support fast handoff without re-authentication. However, this is typically only possible where both the EAP peer and authenticator support certificate-based authentication, or where the user base is sufficiently small that EAP authentication can occur locally.

In order to protect the EAP conversation, the EAP method may negotiate a ciphersuite and derive Transient EAP Keys (TEKs) to provide keys for that ciphersuite in order to protect some or all of the EAP exchange. The TEKs are stored locally within the EAP method and are not exported.

Once EAP mutual authentication completes and is successful, the secure association protocol is run between the peer and authenticator. This derives fresh transient session keys (TSKs), provides for the secure negotiation of the ciphersuite used to protect data, and supports mutual proof of possession of the AAA-Key.

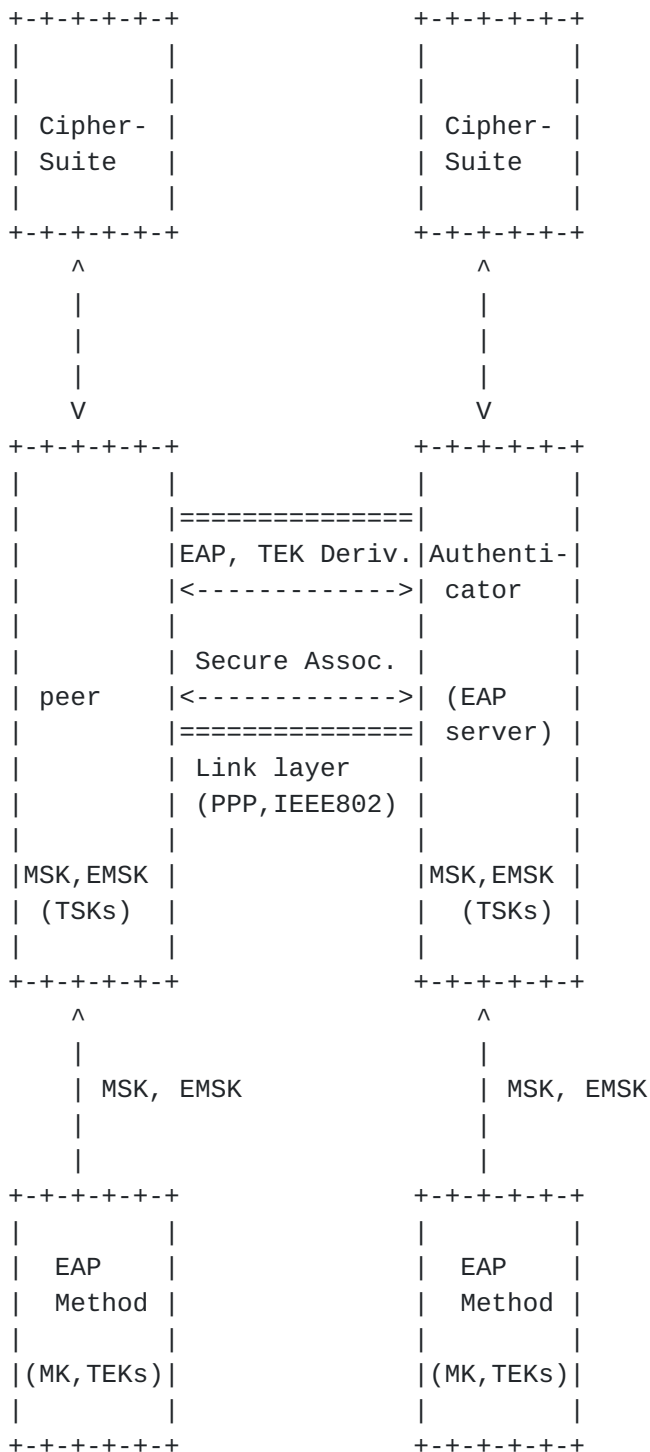


Figure 3 - Relationship between EAP peer and authenticator (acting as an EAP server), where no backend authentication server is present.

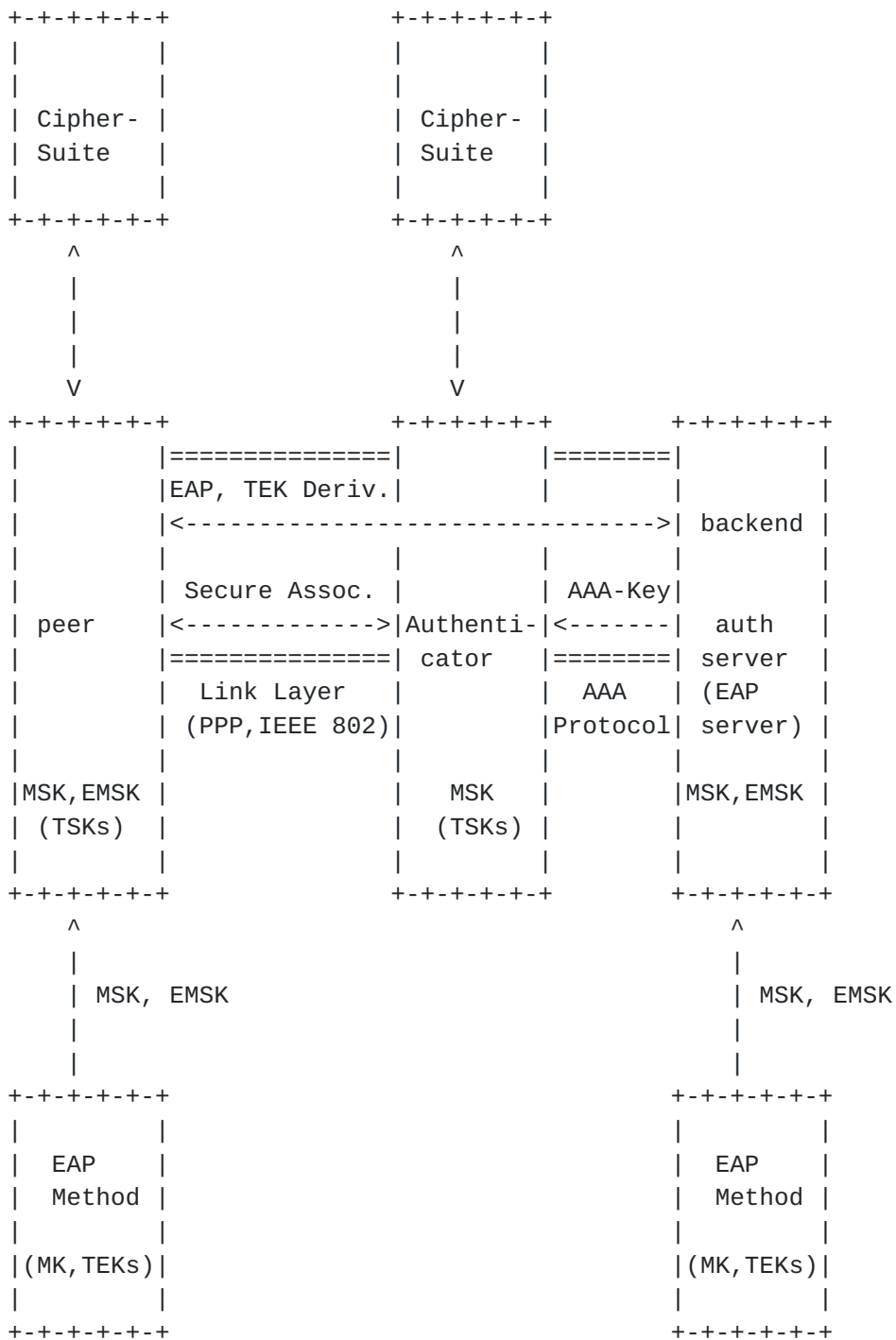


Figure 4 - Pass-through relationship between EAP peer, authenticator and backend authentication server.

Where these conditions cannot be met, a backend authentication server is typically required. In this exchange, as described in [[RFC3579](#)], the authenticator acts as a pass-through between the EAP peer and a backend authentication server. In this model, the authenticator delegates the access control decision to the backend authentication server, which acts as a Key Distribution Center (KDC), supplying keying material to both the EAP peer and authenticator.

Figure 4 illustrates the case where the authenticator acts as a pass-through. Here EAP is spoken between the peer and authenticator as before. The authenticator then encapsulates EAP packets within a AAA protocol such as RADIUS [[RFC3579](#)] or Diameter [[DiamEAP](#)], and forwards packets to and from the backend authentication server, which acts as the EAP server. Since the authenticator acts as a pass-through, EAP methods (as well as the derived EAP Master Key, and TEKs) reside only on the peer and backend authentication server.

On completion of a successful authentication, EAP methods on the EAP peer and EAP server export the Master Session Key (MSK) and Extended Master Session Key (EMSK). The backend authentication server then sends a message to the authenticator indicating that authentication has been successful, providing the AAA-Key within a protected package known as the AAA-Token. Along with the keying material, the AAA-Token contains attributes naming the enclosed keys and providing context.

The MSK and EMSK are used to derive the AAA-Key and key name which are enclosed within the AAA-Token, transported to the NAS by the AAA server, and used within the secure association protocol for derivation of Transient Session Keys (TSKs) required for the negotiated ciphersuite. The TSKs are known only to the peer and authenticator.

2.4. Security Relationships

Figure 5 illustrates the relationship between the peer, authenticator and backend authentication server. As noted in the figure, each party in the exchange mutually authenticates with each of the other parties, and derives a unique key. All parties in the diagram have access to the AAA-Key.

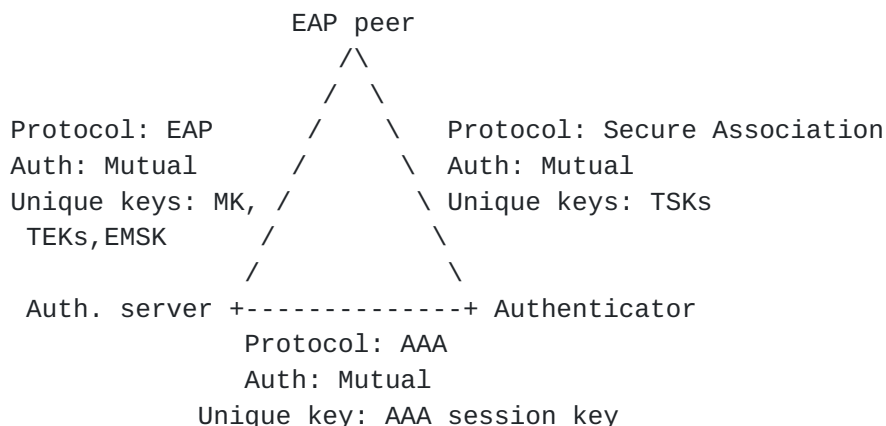


Figure 5: Three-party EAP key distribution

The EAP peer and backend authentication server mutually authenticate via the EAP method, and derive the MK, TEKS and EMSK which are known only to them. The TEKS are used to protect some or all of the EAP conversation between the peer and authenticator, so as to guard against modification or insertion of EAP packets by an attacker. The degree of protection afforded by the TEKS is determined by the EAP method; some methods may protect the entire EAP packet, including the EAP header, while other methods may only protect the contents of the Type-Data field, defined in [\[RFC2284bis\]](#).

Since EAP is spoken only between the EAP peer and server, if a backend authentication server is present then the EAP conversation does not provide mutual authentication between the peer and authenticator, only between the EAP peer and EAP server (backend authentication server). As a result, mutual authentication between the peer and authenticator only occurs where a secure association protocol is used, such as the unicast and group key derivation handshake supported in [\[IEEE80211i\]](#). This means that absent use of a secure association protocol, from the point of view of the peer, EAP mutual authentication only proves that the authenticator is trusted by the backend authentication server; the identity of the authenticator is not confirmed.

Utilizing the AAA protocol, the authenticator and backend authentication server mutually authenticate and derive session keys known only to them, used to provide per-packet integrity and replay protection,

authentication and confidentiality. The MSK is distributed by the backend authentication server to the authenticator over this channel, bound to attributes constraining its usage, as part of the AAA-Token. The binding of attributes to the MSK within a protected package is important so the authenticator receiving the AAA-Token can determine that it has not been compromised, and that the keying material has not been replayed, or mis-directed in some way.

3. Security Associations

The EAP model has four types of security associations (SAs):

- [1] An EAP SA. This is an SA between the EAP peer and the EAP server, created as the result of an EAP authentication exchange (phase 1a). This is a bi-directional SA; that is, both parties use the information in the SA for both sending and receiving.
- [2] A AAA-Key SA, known in [[IEEE80211i](#)] as a PMK SA. This is a bi-directional SA between the EAP peer and authenticator. The keying material for the AAA-Key SA (known as the AAA-Key) is derived on the EAP peer and server, and transported by the EAP server to the authenticator (phase 1b). The choice of keying material is proposed by the EAP peer and confirmed by the EAP authenticator during the unicast secure association protocol (phase 2a).
- [3] A unicast secure association SA. This is a bi-directional SA created as the result of a successful unicast secure association exchange (phase 2a). A unicast secure association SA is bound to a single EAP SA and a single AAA-Key SA.
- [4] A multicast secure association SA (phase 2b). This SA is created as the result of a successful multicast secure association exchange. This SA may be uni-directional (e.g. 802.11 group-key exchange) or bi-directional depending on the design of the multicast secure association protocol, and can be created either from the unicast secure association SA (phase 2a) or directly as the result of a multicast secure association exchange (phase 2b).

3.1. EAP SA

An EAP SA includes:

- the EAP peer and EAP server identities
- the EAP method type
- the EAP peer and server nonces
- the Transient EAP Keys (TEKs)
- the Master Session Key (MSK)
- the Extended Master Session Key (EMSK)

The EAP SA is not explicitly bound to a particular port on the EAP peer. An EAP peer with multiple ports may create an EAP SA on one port and then choose to use that SA to subsequently create a phase 2 SA on another port.

It cannot be assumed that the EAP SA expires after the EAP authentication and key derivation is complete. Some methods may be support "fast resume" by caching EAP SA state on the EAP peer and server.

EAP does not support SA lifetime negotiation or an SA "delete" operation, although some EAP methods may support this. Either the EAP peer or EAP server may delete an EAP SA at any time, and methods which allow an EAP SA to persist need to permit the EAP peer and server to recognize when they have gotten out of sync with respect to the EAP SA state.

For example, EAP TLS [[RFC 2716](#)] supports "fast resume" (TLS session resumption), which assumes that both the EAP peer and server cache EAP master keys (the TLS master secret). An EAP peer attempting a fast resume provides the session-id identifying the session that it wishes to resume. If the EAP server retains the master key corresponding to this session in its cache, then the "fast resume" can proceed; otherwise a full TLS exchange ensues.

An EAP peer may negotiate EAP SAs with one or more EAP servers as the result of pre-authentication or AAA load balancing and failover effects. For example, an EAP peer may pre-authenticate to one or more EAP servers, or may be directed to more than one EAP server as the result of an authentication server becoming unreachable. In general, EAP servers cannot be assumed to be synchronized with respect to EAP SA state, particularly since they may not exist within the same administrative domain. Since an EAP SA is typically created prior to secure association, the EAP SA is not bound to a particular target network.

[3.2.](#) AAA-Key SA

An AAA-Key SA includes:

- the EAP peer (Calling-Station-Id) identifier
- the EAP authenticator (Called-Station-Id) identifier
- the AAA-Key
- the AAA-Key maximum lifetime (if known)
- the advertised peer and authenticator capabilities

The AAA-Key SA is created as the result of the transport of the AAA-Token from the authentication server to the NAS/authenticator. The AAA-Key SA is distinct from the EAP SA in that it is bound to the EAP peer

and authenticator ports. This binding occurs during the unicast secure association protocol (phase 2a) when the EAP peer and authenticator prove possession of the AAA-Key, which is transported from the authentication server to the NAS/EAP authenticator after completion of EAP authentication.

Since the AAA-Key SA is bound to a particular authenticator port, a NAS/authenticator that operates on a shared use network will share the AAA-Key SA between multiple virtual NAS devices unless a separate port (as identified by the Called-Station-Id) is used for each virtual NAS. This would represent a security vulnerability.

In fast handoff, a single EAP SA may be used to establish multiple AAA-Key SAs (see [Appendix E](#) for details). Although a AAA-Key SA may not persist longer than the maximum SA lifetime negotiated for an EAP SA (for methods that support such a negotiation), if an EAP SA is deleted by an EAP peer or authenticator, this does not necessarily imply deletion of the child AAA-Key SA. For example, fast handoff keying material provided by an authentication server may continue to be cached by NASes/authenticators after the corresponding EAP SA has been deleted by the authentication server and/or peer.

3.3. Unicast Secure Association SA

The unicast secure association SA includes:

- the EAP peer (Calling-Station-Id) identifier
- the EAP authenticator (Called-Station-Id) identifier
- the unicast Transient Session Keys (TSKs)
- the unicast secure association peer nonce
- the unicast secure association authenticator nonce
- the negotiated unicast capabilities and unicast ciphersuite.

During the phase 2a exchange, the EAP peer and authenticator demonstrate mutual possession of the AAA-Key derived and transported in phase 1; securely negotiate the session capabilities (including unicast ciphersuites), and derive fresh unicast transient session keys. The AAA-Key SA (phase 1b) is therefore used to create the unicast secure association SA (phase 2a), and in the process the phase 2a unicast secure association SA is bound to ports on the EAP peer and authenticator. However in order for a phase 2a security association to be established, it is not necessary for the phase 1a exchange to be rerun each time. This enables the EAP exchange to be bypassed when fast handoff support is desired.

Since both peer and authenticator nonces are used in the creation of the unicast secure association SA, the transient session keys (TSKs) are guaranteed to be fresh, even if the AAA-Key is not. As a result one or

more unicast secure association SAs (phase 2a) may be derived from a single AAA-Key SA (phase 1b). The phase 2a security associations may utilize the same security parameters (e.g. mode, ciphersuite, etc.) or they may utilize different parameters.

A unicast secure association SA (phase 2a) may not persist longer than the maximum lifetime of its parent AAA-Key SA (if known). However, the deletion of a parent EAP or AAA-Key SA does not necessarily imply deletion of the corresponding unicast secure association SA. Similarly, the deletion of a unicast secure association protocol SA does not imply the deletion of the parent AAA-key SA or EAP SA. However, the failure to mutually prove possession of the AAA-Key during the unicast secure association protocol exchange (phase 2a) is grounds for removal of a AAA-Key SA by both parties.

An EAP peer may be able to negotiate multiple phase 2a SAs with a single EAP authenticator, or may be able to maintain multiple phase 2a SAs with multiple authenticators, based on a single EAP SA derived in phase 1a. For example, during a re-key of the secure association protocol SA, it is possible for two phase 2a SAs to exist during the period between when the new phase 2a SA parameters (such as the TSKs) are calculated and when they are installed. Except where explicitly specified by the semantics of the unicast secure association protocol, it should not be assumed that the installation of a new phase 2a SA necessarily implies deletion of the old phase 2a SA.

On some media (e.g. 802.11) a port on an EAP peer may only establish phase 2a and 2b SAs with a single port of an authenticator within a given Local Area Network (LAN). This implies that the successful negotiation of phase 2a and/or 2b SAs between an EAP peer port and a new authenticator port within a given LAN implies the deletion of existing phase 2a and 2b SAs with authenticators offering access to that Local Area Network (LAN). However, since a given IEEE 802.11 SSID may be comprised of multiple LANs, this does not imply an implicit binding of phase 2a and 2b SAs to an SSID.

3.4. Multicast Secure Association SA

The multicast secure association SA includes:

- the multicast Transient Session Keys
- the direction vector (for a uni-directional SA)
- the negotiated multicast capabilities and multicast ciphersuite

It is possible for more than one multicast secure association SA to be derived from a single unicast secure association SA. However, a multicast secure association SA is bound to a single EAP SA and a single AAA-Key SA.

During a re-key of the multicast secure association protocol SA, it is possible for two phase 2b SAs to exist during the period between when the new phase 2b SA parameters (such as the multicast TSKs) are calculated and when they are installed. Except where explicitly specified by the semantics of the multicast secure association protocol, it should not be assumed that the installation of a new phase 2b SA necessarily implies deletion of the old phase 2b SA.

A multicast secure association SA (phase 2b) may not persist longer than the maximum lifetime of its parent AAA-Key or unicast secure association SA. However, the deletion of a parent EAP, AAA-Key or unicast secure association SA does not necessarily imply deletion of the corresponding multicast secure association SA. For example, a unicast secure association SA may be rekeyed without implying a rekey of the multicast secure association SA.

Similarly, the deletion of a multicast secure association protocol SA does not imply the deletion of the parent EAP, AAA-Key or unicast secure association SA. However, the failure to mutually prove possession of the AAA-Key during the unicast secure association protocol exchange (phase 2a) is grounds for removal of the AAA-Key, unicast secure association and multicast secure association SAs.

3.5. Key Naming

In order to support the correct processing of phase 2 security associations, the secure association (phase 2) protocol supports the naming of phase 2 security associations and associated transient session keys, so that the correct set of transient session keys can be identified for processing a given packet. Explicit creation and deletion operations are also typically supported so that establishment and re-establishment of transient session keys can be synchronized between the parties.

In order to securely bind the EAP security association (phase 1) to its child phase 2 security association, the phase 2 secure association protocol allows the EAP peer and authenticator to mutually prove possession of the EAP (phase 1) keying material derived during the EAP exchange (phase 1). In order to avoid confusion in the case where an EAP peer has more than one EAP security association (phase 1) applicable to establishment of a given phase 2 security association, the secure association protocol (phase 2) supports key naming so that the appropriate phase 1 keying material can be utilized by both parties in the secure association protocol exchange.

As noted earlier, the discovery phase (phase 0) may be insecure so that in order to prevent spoofing of discovery packets, the secure association (phase 2) protocol should support the secure verification of

discovered capabilities, including ciphersuites and other security parameters. This is more scalable than attempting to configure the supported capabilities on each peer and authenticator and more secure than unprotected capabilities negotiation.

For example, a peer might be pre-configured with policy indicating the ciphersuite to be used in communicating with a given authenticator. Within PPP, the ciphersuite is negotiated within the Encryption Control Protocol (ECP), after EAP authentication is completed. Within [\[IEEE80211i\]](#), the AP ciphersuites are advertised in the Beacon and Probe Responses, and are securely verified during a 4-way exchange after EAP authentication has completed.

As part of the secure association protocol (phase 2), it is necessary to bind the Transient Session Keys (TSKs) to the keying material provided in the AAA-Token. This ensures that the EAP peer and authenticator are both clear about what key to use to provide mutual proof of possession. Keys within the EAP key hierarchy are named as follows:

EAP SA name

The EAP security association is negotiated between the EAP peer and EAP server, and is uniquely named as follows <EAP peer name, EAP server name, EAP Method Type, EAP peer nonce, EAP server nonce>. Here the EAP peer name and EAP server name are the identifiers securely exchanged within the EAP method. Since multiple EAP SAs may exist between an EAP peer and EAP server, the EAP peer nonce and EAP server nonce allow EAP SAs to be differentiated. The inclusion of the Method Type in the EAP SA name ensures that each EAP method has a distinct EAP SA space.

MK Name

The EAP Master Key, if supported by an EAP method, is named by the concatenation of the EAP SA name and a method-specific session-id.

MSK Name

The MSK is named by the concatenation of the EAP SA name, "MSK" and the authenticator name, since the MSK may only be bound to a single authenticator. While the AAA server has several potentially unique authenticator identifiers (including the NAS-Identifier, NAS-IP-Address, and NAS-IPv6-Address attributes), only the Called-Station-Id is guaranteed to be known by the EAP peer, EAP server and authenticator and as a result, this is used as the NAS name.

EMSK Name

The EMSK is named by the concatenation of the EAP SA name and "EMSK".

4. Threat Model

4.1. Security Assumptions

The security properties of the EAP exchange are dependent on each leg of the triangle: the selected EAP method, AAA protocol and the secure association protocol.

Assuming that the AAA protocol provides protection against rogue authenticators forging their identity, then the AAA-Token can be assumed to be sent to the correct authenticator, and where it is wrapped appropriately, it can be assumed to be immune to compromise by a snooping attacker.

Where an untrusted AAA intermediary is present, the AAA-Token must not be provided to the intermediary so as to avoid compromise of the AAA-Token. This can be avoided by use of re-direct as defined in [DiamBase].

When EAP is used for authentication on PPP or wired IEEE 802 networks, it is typically assumed that the link is physically secure, so that an attacker cannot gain access to the link, or insert a rogue device. EAP methods defined in [[RFC2284bis](#)] reflect this usage model. These include EAP MD5, as well as One-Time Password (OTP) and Generic Token Card. These methods support one-way authentication (from EAP peer to authenticator) but not mutual authentication or key derivation. As a result, these methods do not bind the initial authentication and subsequent data traffic, even when the the ciphersuite used to protect data supports per-packet authentication and integrity protection. As a result, EAP methods not supporting mutual authentication are vulnerable to session hijacking as well as attacks by rogue devices.

On wireless networks such as IEEE 802.11 [[IEEE80211](#)], these attacks become easy to mount, since any attacker within range can access the wireless medium, or act as an access point. As a result, new ciphersuites have been proposed for use with wireless LANs [[IEEE80211i](#)] which provide per-packet authentication, integrity and replay protection. In addition, mutual authentication and key derivation, provided by methods such as EAP TLS [[RFC2716](#)] are required [[IEEE80211i](#)], so as to address the threat of rogue devices, and provide keying material to bind the initial authentication to subsequent data traffic.

If the selected EAP method does not support mutual authentication, then the peer will be vulnerable to attack by rogue authenticators and backend authentication servers. If the EAP method does not derive keys, then TSKs will not be available for use with a negotiated ciphersuite, and there will be no binding between the initial EAP authentication and subsequent data traffic, leaving the session vulnerable to hijack.

If the authenticator and backend authentication server do not mutually authenticate, then the peer will be vulnerable to rogue backend authentication servers, authenticators, or both. If there is no per-packet authentication, integrity and replay protection between the authenticator and backend authentication server, then an attacker can spoof or modify packets in transit. If the backend authentication server does not protect against authenticator masquerade, or provide the proper binding of the AAA-Key to the session within the AAA-Token, then one or more AAA-Keys may be sent to an unauthorized party, and an attacker may be able to gain access to the network. If the AAA-Token is provided to an untrusted AAA intermediary, then that intermediary may be able to modify the AAA-Key, or the attributes associated with it, as described in [[RFC2607](#)].

If the secure association protocol does not provide mutual proof of possession of the AAA-Key material, then the peer will not have assurance that it is connected to the correct authenticator, only that the authenticator and backend authentication server share a trust relationship (since AAA protocols support mutual authentication). This distinction can become important when multiple authenticators receive AAA-Keys from the backend authentication server, such as where fast handoff is supported. If the TSK derivation does not provide for protected ciphersuite and capabilities negotiation, then downgrade attacks are possible.

[4.2.](#) Security Requirements

This section describes the security requirements for EAP methods, AAA protocols, secure association protocols and Ciphersuites. These requirements **MUST** be met by specifications requesting publication as an RFC. Based on these requirements, the security properties of EAP exchanges are analyzed.

[4.2.1.](#) EAP method requirements

Key usage

Key material exported by EAP methods **MUST NOT** be used directly to protect data.

Mutual authentication

EAP Methods deriving keys **MUST** provide for mutual authentication between the EAP peer and EAP Server.

State synchronization

EAP peers, authenticators and authentication servers **MUST** be prepared for situations in which one or more of the parties have discarded an EAP SA (phase 1), which is still valid on another party.

TEK derivation

Methods deriving keys MUST specify how Transient EAP Keys (TEKs) are derived. TEKs MUST remain local to the EAP method and MUST NOT be provided to third parties.

MSK and EMSK

EAP methods supporting key derivation MUST export two quantities of at least 64 octets each, known as the Master Session Key (MSK), and the Extended Master Session Key (EMSK).

Cryptographic separation

Methods supporting key derivation MUST demonstrate cryptographic separation between the TEK, 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 TEKs, MSK or EMSK MUST NOT be able to recover the other quantities with a level of effort less than brute force. Since Transient Session Keys (TSKs) are derived from the MSK, if branch independence holds, then it is also true that the TSKs are cryptographically separate from the EMSK and TEKs.

EMSK reservation

While the EMSK is exported by the EAP method, its use is reserved, and as a result it MUST remain known only to the EAP peer and server and MUST NOT be provided to third parties. Since the EMSK is the only keying material exported by an EAP method that is neither provided to a third party nor a known quantity, it is attractive for use in future applications such as fast handoff or man-in-the-middle detection. Given its potential future uses, damage due to EMSK compromise is second only in effect to compromise of the MK, yielding an attacker the ability to access the network at will, and to decrypt past and future data traffic.

TEK derivation

In order to establish a protected channel between the EAP peer and server as part of the EAP exchange, a ciphersuite needs to be negotiated and suitable keys need to be provided (known as the transient EAP keys). The ciphersuite used to protect the EAP exchange between the peer and server is distinct from the ciphersuite negotiated between the peer and authenticator, used to protect data. Where a protected channel is established within the EAP method, the method specification MUST specify the mechanism by which the EAP ciphersuite is negotiated, as well as the algorithms for derivation of TEKs.

Ciphersuite Independence

Keying material exported by EAP methods MUST be independent of the ciphersuite negotiated to protect data.

Key Strength

The strength of Transient Session Keys (TSKs) and Transient EAP Keys (TEKs) used to protect data is ultimately dependent on the strength of keys generated by the EAP method. If EAP method does not produce keying material of sufficient strength, then the TSKs and TEKs may be subject to brute force attack. EAP methods supporting key derivation MUST be capable of generating an MSK and EMSK, each with an effective key strength of at least 128 bits. More details on key strength are provided in [Section 6.1](#).

Resilience against compromise

In order to protect against compromise of the AAA-Token, EAP methods MUST demonstrate that compromise of the MSK or EMSK of a given EAP security association does not enable compromise of subsequent or prior MSKs or EMSKs derived between the EAP peer and EAP server.

Freshness

In order to support key naming and assure freshness of TSKs even in cases where one party may not have a high quality random number generator, EAP methods generating keys MUST support a two-nonce exchange in the derivation of the MSK and EMSK, using nonces of at least 128-bits.

Known-good algorithms

The development and validation of key derivation algorithms is difficult, and as a result EAP methods SHOULD reuse existing key derivation algorithms, rather than inventing new ones. EAP methods requesting publication as an RFC MUST provide citations to literature justifying the security of the chosen algorithms. EAP methods SHOULD utilize well established and analyzed mechanisms for MSK, EMSK, TSK and TEK derivation.

[4.2.2.](#) AAA Protocol Requirements

AAA protocols suitable for use in transporting EAP MUST provide the following facilities:

Security services

AAA protocols used for transport of EAP keying material MUST implement and SHOULD use per-packet integrity and authentication, replay protection and confidentiality. These requirements are met by Diameter EAP [[DiamEAP](#)], as well as RADIUS over IPsec [[RFC3579](#)].

Session Keys

AAA protocols used for transport of EAP keying material MUST implement and SHOULD use session keys, as in Diameter EAP [[DiamEAP](#)] and RADIUS over IPsec [[RFC3579](#)], rather than using a

static key, as originally defined in RADIUS [[RFC2865](#)].

Mutual authentication

AAA protocols used for transport of EAP keying material MUST provide for mutual authentication between the authenticator and backend authentication server. These requirements are met by Diameter EAP [[DiamEAP](#)] as well as by RADIUS/EAP [[RFC3579](#)].

Forgery protection

AAA protocols used for transport of EAP keying material SHOULD provide protection against rogue authenticators masquerading as other authenticators. This can be accomplished, for example, by requiring that AAA agents check the source address of packets against the origin attributes (Origin-Host AVP in Diameter, NAS-IP-Address, NAS-IPv6-Address, NAS-Identifier in RADIUS). For details, see [Section 4.3.7 of \[RFC3579\]](#).

Key transport

Since EAP methods do not export Transient Session Keys (TSKs) in order to maintain media and ciphersuite independence, the AAA server MUST NOT transport TSKs from the backend authentication server to authenticator.

Key transport specification

In order to enable backend authentication servers to provide keying material to the authenticator in a well defined format, AAA protocols suitable for use with EAP MUST define the format and wrapping of the AAA-Token.

EMSK transport

Since the EMSK is a secret known only to the backend authentication server and peer, the AAA-Token MUST NOT transport the EMSK from the backend authentication server to the authenticator.

AAA-Token protection

To ensure against compromise, the AAA-Token MUST be integrity protected, authenticated, replay protected and encrypted in transit, using well-established cryptographic algorithms.

Session Keys

The AAA-Token SHOULD be protected with session keys as in Diameter [[DiamBASE](#)] or RADIUS over IPsec [[RFC3579](#)] rather than static keys, as in [[RFC2548](#)].

Key naming

In order to ensure against confusion between the appropriate keying material to be used in a given secure association protocol exchange, the AAA-Token SHOULD include explicit key names and

context appropriate for informing the authenticator how the keying material is to be used.

Key Compromise

Where untrusted intermediaries are present, the AAA-Token SHOULD NOT be provided to the intermediaries. In Diameter, handling of keys by intermediaries can be avoided using Redirect functionality [[DiamBASE](#)].

4.2.3. Secure Association Protocol Requirements

The Secure Association Protocol supports the following:

Mutual proof of possession

The peer and authenticator MUST each demonstrate possession of the keying material transported between the AAA server and authenticator (AAA-Key).

Key Naming

The Secure Association Protocol MUST explicitly name the keys used in the proof of possession exchange, so as to prevent confusion when more than one set of keying material could potentially be used as the basis for the exchange.

Creation and Deletion

In order to support the correct processing of phase 2 security associations, the secure association (phase 2) protocol MUST support the naming of phase 2 security associations and associated transient session keys, so that the correct set of transient session keys can be identified for processing a given packet. The phase 2 secure association protocol also MUST support transient session key activation and SHOULD support deletion, so that establishment and re-establishment of transient session keys can be synchronized between the parties.

Integrity and Replay Protection

The Secure Association Protocol MUST support integrity and replay protection of all messages.

Direct operation

Since the phase 2 secure association protocol is concerned with the establishment of security associations between the EAP peer and authenticator, including the derivation of transient session keys, only those parties have "a need to know" the transient session keys. The secure association protocol MUST operate directly between the peer and authenticator, and MUST NOT be passed-through to the backend authentication server, or include additional parties.

Derivation of transient session keys

The secure association protocol negotiation **MUST** support derivation of unicast and multicast transient session keys suitable for use with the negotiated ciphersuite.

TSK freshness

The secure association (phase 2) protocol **MUST** support the derivation of fresh unicast and multicast transient session keys, even when the keying material provided by the AAA server is not fresh. This is typically supported by including an exchange of nonces within the secure association protocol.

Bi-directional operation

While some ciphersuites only require a single set of transient session keys to protect traffic in both directions, other ciphersuites require a unique set of transient session keys in each direction. The phase 2 secure association protocol **SHOULD** provide for the derivation of unicast and multicast keys in each direction, so as not to require two separate phase 2 exchanges in order to create a bi-directional phase 2 security association.

Secure capabilities negotiation

The Secure Association Protocol **MUST** support secure capabilities negotiation. This includes security parameters such as the security association identifier (SAID) and ciphersuites. It also includes confirmation of the capabilities discovered during the discovery phase (phase 0), so as to ensure that the announced capabilities have not been forged.

4.2.4. Ciphersuite Requirements

Ciphersuites suitable for keying by EAP methods **MUST** provide the following facilities:

TSK derivation

In order to allow a ciphersuite to be usable within the EAP keying framework, a specification **MUST** be provided describing how transient session keys suitable for use with the ciphersuite are derived from the AAA-Key.

EAP method independence

Algorithms for deriving transient session keys from the AAA-Key **MUST NOT** depend on the EAP method. However, algorithms for deriving TEKS **MAY** be specific to the EAP method.

Cryptographic separation

The TSKs derived from the AAA-Key **MUST** be cryptographically separate from each other. Similarly, TEKS **MUST** be

cryptographically separate from each other. In addition, the TSKs MUST be cryptographically separate from the TEKs.

5. IANA Considerations

This specification does not create any new registries, or define any new EAP codes or types.

6. Security Considerations

6.1. Key Strength

In order to guard against brute force attacks, EAP methods deriving keys need to be capable of generating keys with an appropriate effective symmetric key strength. In order to ensure that key generation is not the weakest link, it is necessary for EAP methods utilizing public key cryptography to choose a public key that has a cryptographic strength meeting the symmetric key strength requirement.

As noted in Section 5 of [[KeyLen](#)], this results in the following required RSA or DH module and DSA subgroup size in bits, for a given level of attack resistance in bits:

Attack Resistance (bits)	RSA or DH Modulus size (bits)	DSA subgroup size (bits)
-----	-----	-----
70	947	128
80	1228	145
90	1553	153
100	1926	184
150	4575	279
200	8719	373
250	14596	475

6.2. Key Wrap

As described in [[RFC3579](#)], [Section 4.3](#), known problems exist in the key wrap specified in [[RFC2548](#)]. Where the same RADIUS shared secret is used by a PAP authenticator and an EAP authenticator, there is a vulnerability to known plaintext attack. Since RADIUS uses the shared secret for multiple purposes, including per-packet authentication, attribute hiding, considerable information is exposed about the shared secret with each packet. This exposes the shared secret to dictionary attacks. MD5 is used both to compute the RADIUS Response Authenticator and the Message-Authenticator attribute, and some concerns exist relating to the security of this hash [[MD5Attack](#)]. As discussed in [[RFC3579](#)], [Section 4.2](#), these and other RADIUS vulnerabilities may be addressed by running RADIUS over IPsec.

Where an untrusted AAA intermediary is present (such as a RADIUS proxy or a Diameter agent), and data object security is not used, the MSK may be recovered by an attacker in control of the untrusted intermediary. Possession of the MSK enables decryption of data traffic sent between the peer and a specific authenticator; however where Perfect Forward Secrecy (PFS) is implemented, compromise of the MSK does enable an attacker to impersonate the peer to another authenticator, since that requires possession of the MK or EMSK, which are not transported by the AAA protocol. This vulnerability may be mitigated by implementation of redirect functionality, as provided in [\[DiamBASE\]](#).

6.3. Man-in-the-middle Attacks

As described in [\[MiTM\]](#), EAP method sequences and compound authentication mechanisms may be subject to man-in-the-middle attacks. When such attacks are successfully carried out, the attacker acts as an intermediary between a victim and a legitimate authenticator. This allows the attacker to authenticate successfully to the authenticator, as well as to obtain access to the network.

In order to prevent these attacks, [\[MiTM\]](#) recommends derivation of a compound key by which the EAP peer and authenticator can prove that they have participated in the entire EAP exchange. Since the compound key must not be known to an attacker posing as an authenticator, and yet must be derived from quantities that are exported by EAP methods, it may be desirable to derive the compound key from a portion of the EMSK. In order to provide proper key hygiene, it is recommended that the compound key used for man-in-the-middle protection be cryptographically separate from other keys derived from the EMSK, such as fast handoff keys, discussed in [Appendix E](#).

6.4. Impersonation

Both the RADIUS and Diameter protocols are potentially vulnerable to impersonation by a rogue authenticator.

When RADIUS requests are forwarded by a proxy, the NAS-IP-Address or NAS-IPv6-Address attributes may not correspond to the source address. Since the NAS-Identifier attribute need not contain an FQDN, it also may not correspond to the source address, even indirectly. [\[RFC2865\]](#) [Section 3](#) states:

A RADIUS server MUST use the source IP address of the RADIUS UDP packet to decide which shared secret to use, so that RADIUS requests can be proxied.

This implies that it is possible for a rogue authenticator to forge NAS-IP-Address, NAS-IPv6-Address or NAS-Identifier attributes within a

RADIUS Access-Request in order to impersonate another authenticator. Among other things, this can result in messages (and MSKs) being sent to the wrong authenticator. Since the rogue authenticator is authenticated by the RADIUS proxy or server purely based on the source address, other mechanisms are required to detect the forgery. In addition, it is possible for attributes such as the Called-Station-Id and Calling-Station-Id to be forged as well.

As recommended in [[RFC3579](#)], this vulnerability can be mitigated by having RADIUS proxies check authenticator identification attributes against the source address.

To allow verification of session parameters such as the Called-Station-Id and Calling-Station-Id, these can be sent by the EAP peer to the server, protected by the TEKS. The RADIUS server can then check the parameters sent by the EAP peer against those claimed by the authenticator. If a discrepancy is found, an error can be logged.

While [[DiamBASE](#)] requires use of the Route-Record AVP, this utilizes FQDNs, so that impersonation detection requires DNS A/AAAA and PTR RRs to be properly configured. As a result, it appears that Diameter is as vulnerable to this attack as RADIUS, if not more so. To address this vulnerability, it is necessary to allow the backend authentication server to communicate with the authenticator directly, such as via the redirect functionality supported in [[DiamBase](#)].

[7. References](#)

[7.1. Normative References](#)

- [RFC1661] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51, [RFC 1661](#), July 1994.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2434] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 2434](#), October 1998.
- [RFC2284bis] Blunk, L., et al. "Extensible Authentication Protocol (EAP)", Internet draft (work in progress), [draft-ietf-eap-rfc2284bis-04.txt](#), June 2003.
- [IEEE802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, ANSI/IEEE Std 802, 1990.

7.2. Informative References

- [RFC793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), September 1981.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", [RFC 1321](#), April 1992.
- [RFC1968] Meyer, G., "The PPP Encryption Protocol (ECP)", [RFC 1968](#), June 1996.
- [RFC2104] Krawczyk, et al., "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), February 1997.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", [RFC 2246](#), November 1998.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", [RFC 2409](#), November 1998.
- [RFC2419] Sklower, K. and G. Meyer, "The PPP DES Encryption Protocol, Version 2 (DESE-bis)", [RFC 2419](#), September 1998.
- [RFC2420] Hummert, K., "The PPP Triple-DES Encryption Protocol (3DESE)", [RFC 2420](#), September 1998.
- [RFC2434] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 2434](#), October 1998.
- [RFC2516] Mamakos, L., "A Method for Transmitting PPP Over Ethernet (PPPoE)", [RFC 2516](#), February 1999.
- [RFC2548] Zorn, G., "Microsoft Vendor-Specific RADIUS Attributes", [RFC 2548](#), March 1999.
- [RFC2607] Aboba, B. and J. Vollbrecht, "Proxy Chaining and Policy Implementation in Roaming", [RFC 2607](#), June 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] R. Stewart et al., "Stream Control Transmission Protocol", [RFC 2960](#), October 2000.
- [RFC3078] Pall, G. and G. Zorn, "Microsoft Point-to-Point Encryption (MPPE) [RFC 3078](#), March 2001.
- [RFC3079] Zorn, G. "Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)," [RFC 3079](#), March 2001.
- [RFC3394] R. Housley, "Advance Encryption Standard (AES) Key Wrap Algorithm", [RFC 3394](#), September 2002.
- [RFC3580] Congdon, P., et al., "IEEE 802.1X RADIUS Usage Guidelines", [RFC 3580](#), August 2003.
- [FIPSDDES] National Bureau of Standards, "Data Encryption Standard", FIPS PUB 46 (January 1977).
- [DESMODES] National Bureau of Standards, "DES Modes of Operation", FIPS PUB 81 (December 1980).
- [FIPS197] FIPS PUB 197, Advanced Encryption Standard (AES), 2001 November 26H.
- [SHA] National Institute of Standards and Technology (NIST), "Announcing the Secure Hash Standard," FIPS 180-1, U.S. Department of Commerce, 04/1995
- [IEEE80211] Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1997, 1997.
- [IEEE8021X] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std 802.1X-2001, June 2002.
- [IEEE80211f] IEEE 802.11F, "Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation", July 2003.
- [IEEE80211i] IEEE Draft 802.11i/D5.0, "Draft Supplement to STANDARD FOR Telecommunications and Information Exchange between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer

(PHY) specifications: Specification for Enhanced Security", August 2003.

- [EAPAPI] Microsoft Developer Network, "Windows 2000 EAP API", August 2000, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/eap/eapport_0fj9.asp
- [RFC3579] Aboba, B. and P. Calhoun, "RADIUS Support For Extensible Authentication Protocol (EAP)", [RFC 3579](#), August 2003.
- [RoamCERT] Aboba, B., "Certificate-Based Roaming", Internet draft (work in progress), [draft-ietf-roamops-cert-02.txt](#), April 1999.
- [DiamBASE] Calhoun, P., et al., "Diameter Base Protocol", Internet draft (work in progress), [draft-ietf-aaa-diameter-17.txt](#), December 2002.
- [DiamEAP] Eronen, P., et al., "Diameter Extensible Authentication Protocol (EAP) Application", Internet draft (work in progress), [draft-ietf-aaa-eap-02.txt](#), June 2003.
- [Handoff] Arbaugh, B. and B. Aboba, "Experimental Handoff Extension to RADIUS", Internet draft (work in progress), [draft-irtf-aaaarch-handoff-02.txt](#), June 2003.
- [IEEE-02-758] Mishra, A., Shin, M., Arbaugh, W., Lee, I., Jang, K., "Proactive Caching Strategies for IAPP Latency Improvement during 802.11 Handoff", IEEE 802.11 Working Group, IEEE-02-758r1-F, November 2002.
- [IEEE-03-084] Mishra, A., Shin, M., Arbaugh, W., Lee, I., Jang, K., "Proactive Key Distribution to support fast and secure roaming", IEEE 802.11 Working Group, IEEE-03-084r1-I, <http://www.ieee802.org/11/Documents/DocumentHolder/3-084.zip>, January 2003.
- [IEEE-03-155] Aboba, B., "Fast Handoff Issues", IEEE 802.11 Working Group, IEEE-03-155r0-I, <http://www.ieee802.org/11/Documents/DocumentHolder/3-155.zip>, March 2003.
- [KeyLen] Orman, H., and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", Internet draft (work in progress), [draft-orman-public-key-lengths-05.txt](#), December 2001.

- [8021XHandoff] Pack, S. and Y. Choi, "Pre-Authenticated Fast Handoff in a Public Wireless LAN Based on IEEE 802.1X Model", School of Computer Science and Engineering, Seoul National University, Seoul, Korea, 2002.
- [MD5Attack] Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes Vol.2 No.2, Summer 1996.
- [MiTM] Puthenkulam, J., et al, "The Compound Authentication Binding Problem", Internet draft (work in progress), [draft-puthekulam-eap-binding-03.txt](#), June 2003.

Appendix A - Ciphersuite Keying Requirements

To date, PPP and IEEE 802.11 ciphersuites are suitable for keying by EAP. This Appendix describes the keying requirements of common PPP and **802.11 ciphersuites**.

PPP ciphersuites include DESEbis [[RFC2419](#)], 3DES [[RFC2420](#)], and MPPE [[RFC3078](#)]. The DES algorithm is described in [[FIPSDDES](#)], and DES modes (such as CBC, used in [[RFC2419](#)] and DES-EDE3-CBC, used in [[RFC2420](#)]) are described in [[DESMODES](#)]. For PPP DESEbis, a single 56-bit encryption key is required, used in both directions. For PPP 3DES, a 168-bit encryption key is needed, used in both directions. As described in [[RFC2419](#)] for DESEbis and [[RFC2420](#)] for 3DES, the IV, which is different in each direction, is "deduced from an explicit 64-bit nonce, which is exchanged in the clear during the [ECP] negotiation phase." There is therefore no need for the IV to be provided by EAP.

For MPPE, 40-bit, 56-bit or 128-bit encryption keys are required in each direction, as described in [[RFC3078](#)]. No initialization vector is required.

While these PPP ciphersuites provide encryption, they do not provide per-packet authentication or integrity protection, so an authentication key is not required in either direction.

Within [[IEEE80211](#)], Transient Session Keys (TSKs) are required both for unicast traffic as well as for multicast traffic, and therefore separate key hierarchies are required for unicast keys and multicast keys. IEEE **802.11 ciphersuites include WEP-40, described in [[IEEE80211](#)], which** requires a 40-bit encryption key, the same in either direction; and WEP-128, which requires a 104-bit encryption key, the same in either direction. These ciphersuites also do not support per-packet authentication and integrity protection. In addition to these unicast keys, authentication and encryption keys are required to wrap the multicast encryption key.

Recently, new ciphersuites have been proposed for use with IEEE 802.11 that provide per-packet authentication and integrity protection as well as encryption [[IEEE80211i](#)]. These include TKIP, which requires a single 128-bit encryption key and a 128-bit authentication key (used in both directions); AES CCMP, which requires a single 128-bit key (used in both directions) in order to authenticate and encrypt data; and WRAP, which requires a single 128-bit key (used in both directions).

As with WEP, authentication and encryption keys are also required to wrap the multicast encryption (and possibly, authentication) keys.

Appendix B - TEK Hierarchy

Figure B-1 illustrates the TEK key hierarchy for EAP-TLS [RFC2716], which is based on the TLS key hierarchy described in [RFC2246]. The TLS-negotiated ciphersuite is used to set up a protected channel for use in protecting the EAP conversation, keyed by the derived TEKs. The TEK derivation proceeds as follows:

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

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

Where:

```
TLS-PRF-X =    TLS pseudo-random function defined in [RFC2246],
                computed to X octets.
```

```
master_secret = TLS term for the MK.
```

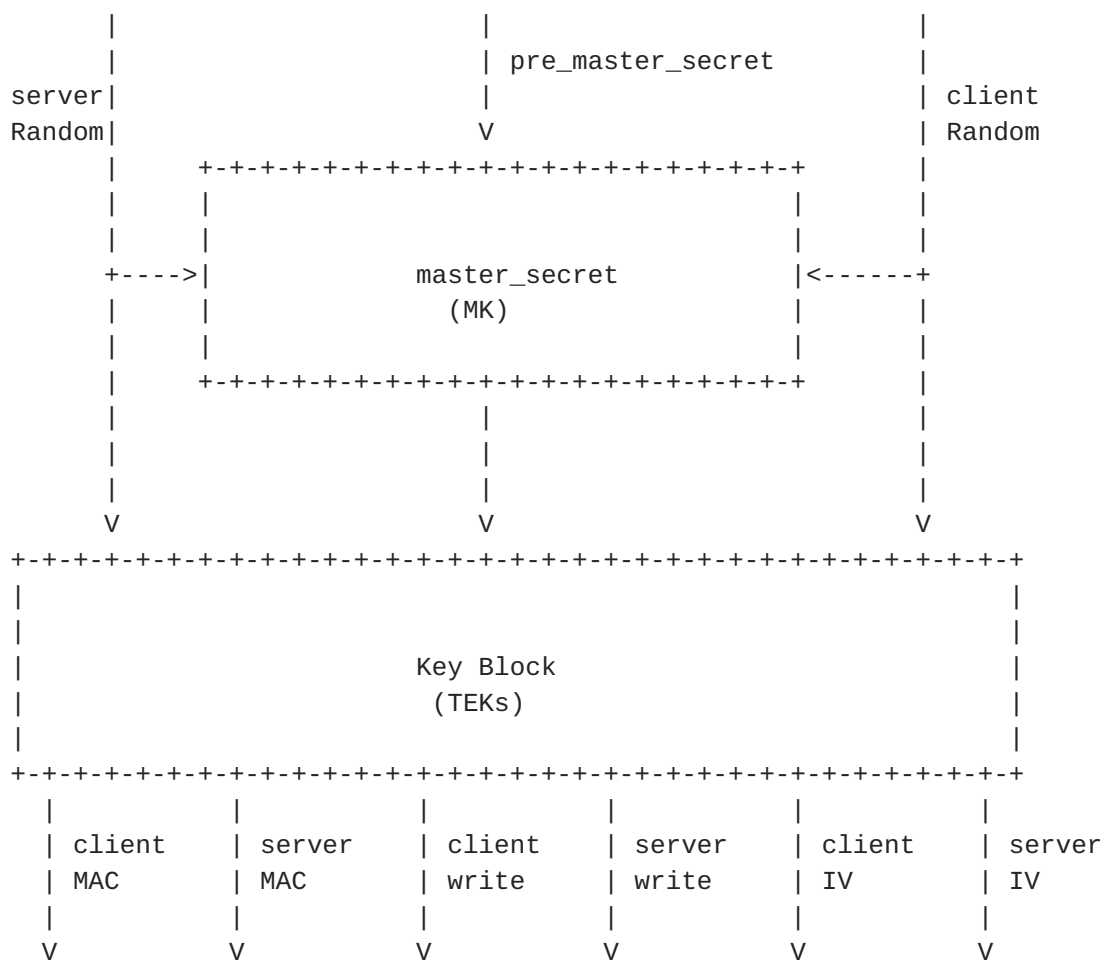


Figure B-1 - TLS [RFC2246] Key Hierarchy

Appendix C - MSK and EMSK Hierarchy

In EAP-TLS [[RFC2716](#)], the MSK is divided into two halves, corresponding to the "Peer to Authenticator Encryption Key" (Enc-RECV-Key, 32 octets, also known as the PMK) and "Authenticator to Peer Encryption Key" (Enc-SEND-Key, 32 octets). In [[RFC2548](#)], the Enc-RECV-Key (the PMK) is transported in the MS-MPPE-Recv-Key attribute, and the Enc-SEND-Key is transported in the MS-MPPE-Send-Key attribute.

The EMSK is also divided into two halves, corresponding to the "Peer to Authenticator Authentication Key" (Auth-RECV-Key, 32 octets) and "Authenticator to Peer Authentication Key" (Auth-SEND-Key, 32 octets). The IV is a 64 octet quantity that is a known value; octets 0-31 are known as the "Peer to Authenticator IV" or RECV-IV, and Octets 32-63 are known as the "Authenticator to Peer IV", or SEND-IV.

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

As described in [[RFC2716](#)], the formula for the derivation of the MSK, EMSK and IV from the MK is as follows:

MSK = TLS-PRF-64(MK, "client EAP encryption",
client.random || server.random)

EMSK = second 64 octets of:
TLS-PRF-128(MK, "client EAP encryption",
client.random || server.random)

IV = TLS-PRF-64("", "client EAP encryption",
client.random || server.random)

AAA-Key(0,31) = Peer to Authenticator Encryption Key (Enc-RECV-Key)
(MS-MPPE-Recv-Key in [[RFC2548](#)]). Also known as the
PMK.

AAA-Key(32,63)= Authenticator to Peer Encryption Key (Enc-SEND-Key)
(MS-MPPE-Send-Key in [[RFC2548](#)])

EMSK(0,31) = Peer to Authenticator Authentication Key (Auth-RECV-Key)

EMSK(32,63) = Authenticator to Peer Authentication Key (Auth-Send-Key)

IV(0,31) = Peer to Authenticator Initialization Vector (RECV-IV)

IV(32,63) = Authenticator to Peer Initialization vector (SEND-IV)

Where:

AAA-Key(W,Z) = Octets W through Z includes of the AAA-Key.

IV(W,Z) = Octets W through Z inclusive of the IV.

MSK(W,Z) = Octets W through Z inclusive of the MSK.
 EMSK(W,Z) = Octets W through Z inclusive of the EMSK.
 MK = TLS master_secret
 TLS-PRF-X = TLS PRF function defined in [RFC2246] computed to X octets
 client.random = Nonce generated by the TLS client.
 server.random = Nonce generated by the TLS server.

Figure C-1 describes the process by which the MSK, EMSK, IV and ultimately the TSKs, are derived from the MK. Note that in [RFC2716], the MK is referred to as the "TLS Master Secret".

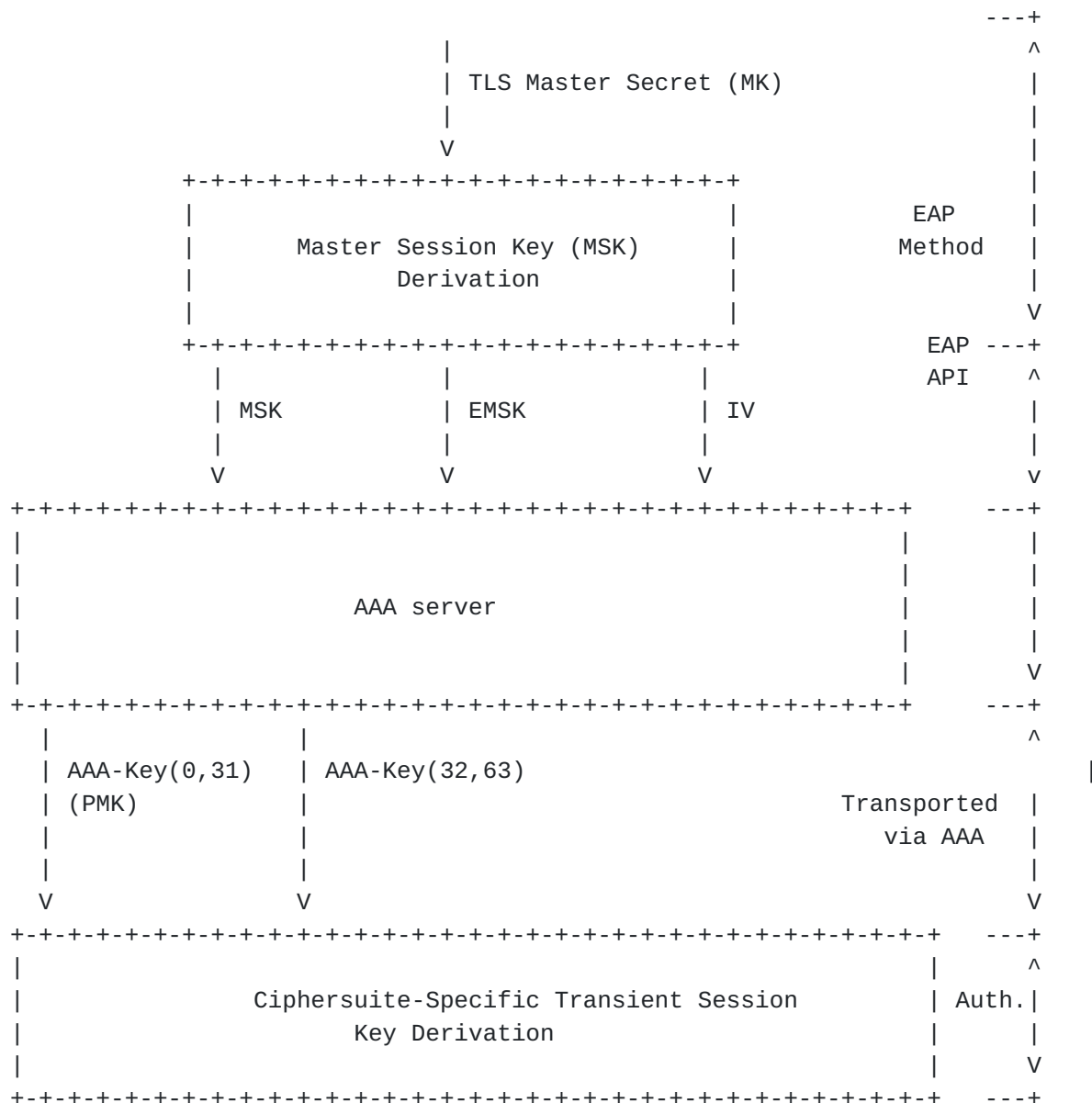


Figure C-1 - EAP TLS [RFC2716] Key hierarchy

Appendix D - Transient Session Key (TSK) Derivation

Within IEEE 802.11 RSN, the Pairwise Transient Key (PTK), a transient session key used to protect unicast traffic, is derived from the PMK (octets 0-31 of the MSK), known in [[RFC2716](#)] as the Peer to Authenticator Encryption Key. In [[IEEE80211i](#)], the PTK is derived from the PMK via the following formula:

$$\text{PTK} = \text{EAPOL-PRF-X}(\text{PMK}, \text{"Pairwise key expansion"}, \text{Min}(\text{AA}, \text{SA}) \parallel \text{Max}(\text{AA}, \text{SA}) \parallel \text{Min}(\text{ANonce}, \text{SNonce}) \parallel \text{Max}(\text{ANonce}, \text{SNonce}))$$

Where:

PMK	= AAA-Key(0,31)
SA	= Station MAC address (Calling-Station-Id)
AA	= Access Point MAC address (Called-Station-Id)
ANonce	= Access Point Nonce
SNonce	= Station Nonce
EAPOL-PRF-X	= Pseudo-Random Function based on HMAC-SHA1, generating a PTK of size X octets.

TKIP uses $X = 64$, while CCMP, WRAP, and WEP use $X = 48$.

The EAPOL-Key Confirmation Key (KCK) is used to provide data origin authenticity in the TSK derivation. It utilizes the first 128 bits (bits 0-127) of the PTK. The EAPOL-Key Encryption Key (KEK) provides confidentiality in the TSK derivation. It utilizes bits 128-255 of the PTK. Bits 256-383 of the PTK are used by Temporal Key 1, and Bits 384-511 are used by Temporal Key 2. Usage of TK1 and TK2 is ciphersuite specific. Details are available in [[IEEE80211i](#)].

Appendix E - AAA-Key Derivation

As discussed in [[Handoff](#)], [[IEEE-02-758](#)], [[IEEE-03-084](#)], and [[8021XHandoff](#)], keying material may be required for use in fast handoff between IEEE 802.11 authenticators. Where the backend authentication server provides keying material to multiple authenticators in order to facilitate fast handoff, it is highly desirable for the keying material used on different authenticators to be cryptographically separate, so that if one authenticator is compromised, it does not lead to the compromise of other authenticators. Where keying material is provided by the backend authentication server, a key hierarchy derived from the EMSK, as suggested in [[IEEE-03-155](#)] can be used to provide cryptographically separate keying material for use in fast handoff:

AAA-Key-A = MSK(0,63)

AAA-Key-B = PRF(EMSK(0,63),AAA-Key-A,B-Called-Station-Id,Calling-Station-Id)

AAA-Key-E = PRF(EMSK(0,63),AAA-Key-A,E-Called-Station-Id,Calling-Station-Id)

Where:

Calling-Station-Id = STA MAC address

B-Called-Station-Id = AP B MAC address

E-Called-Station-Id = AP E MAC address

Here AAA-Key-A is the AAA-Key derived during the initial EAP authentication between the peer and authenticator A. Based on this initial EAP authentication, the EMSK is also derived, which can be used to derive AAA-Keys for fast authentication between the EAP peer and authenticators B and E. Since the EMSK is cryptographically separate from the MSK, each of these AAA-Keys is cryptographically separate from each other, and are guaranteed to be unique between the EAP peer (also known as the STA) and the authenticator (also known as the AP).

Acknowledgments

Thanks to Arun Ayyagari, Ashwin Palekar, and Tim Moore of Microsoft, Dorothy Stanley of Agere, Bob Moskowitz of TruSecure, and Russ Housley of Vigil Security for useful feedback.

Author Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: bernarda@microsoft.com
Phone: +1 425 706 6605
Fax: +1 425 936 7329

Dan Simon
Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: dansimon@microsoft.com
Phone: +1 425 706 6711
Fax: +1 425 936 7329

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in [BCP-11](#). Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

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

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.
This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English. The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its

successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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."

Open issues

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

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

Expiration Date

This memo is filed as <[draft-aboba-pppext-key-problem-07.txt](#)>, and expires February 22, 2004.

