EAP Working Group
INTERNET-DRAFT
Category: Standards Track
<draft-ietf-eap-keying-09.txt>
8 January 2006

Bernard Aboba
Dan Simon
Microsoft
J. Arkko
Ericsson
P. Eronen
Nokia
H. Levkowetz, Ed.
ipUnplugged

Extensible Authentication Protocol (EAP) Key Management Framework

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

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

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

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

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

This Internet-Draft will expire on August 22, 2006.

Copyright Notice

Copyright (C) The Internet Society 2006.

Abstract

The Extensible Authentication Protocol (EAP), defined in [RFC3748], enables extensible network access authentication. This document provides a framework for the transport and usage of keying material generated by EAP authentication algorithms, known as "methods". It also specifies the EAP key hierarchy.

Table of Contents

1. Introduction 3
1.1 Requirements Language 3
1.2 Terminology 3
1.3 Overview 5
1.4 EAP Invariants 9
2. Lower Layer Operation 12
2.1 Overview 12
2.2 Layering 13
2.3 Transient Session Keys 15
2.4 Key Scope 18
3. Key Management 22
3.1 Secure Association Protocol 22
3.2 Parent-Child Relationships 25
3.3 Local Key Lifetimes 26
3.4 Exported and Calculated Key Lifetimes 26
3.5 Key Cache Synchronization 28
3.6 Key Strength 28
3.7 Key Wrap 29
4. Handoff Vulnerabilities 30
4.1 Authorization 30
4.2 Correctness 31
5. Security Considerations 34
5.1 Security Terminology 35
5.2 Threat Model 35
5.3 Authenticator Compromise 36
5.4 Spoofing 37
5.5 Downgrade Attacks 37
5.6 Unauthorized Disclosure 38
5.7 Replay Protection 40
5.8 Key Freshness 40
5.9 Elevation of Privilege 41
5.10 Man-in-the-Middle Attacks 42
5.11 Denial of Service Attacks 43
5.12 Impersonation 43
5.13 Channel Binding 44
6. IANA Considerations 45
7. References 45
7.1 Normative References 45
7.2 Informative References 46
Acknowledgments 50
Author's Addresses 50
Appendix A - EAP-TLS Key Hierarchy 52
Appendix B - Exported Parameters in Existing Methods 53
Intellectual Property Statement 55
Disclaimer of Validity 56
Copyright Statement 56

1. Introduction

The Extensible Authentication Protocol (EAP), defined in [RFC3748], 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 [IEEE-802.1X].

This document provides a framework for the transport and usage of keying material generated by EAP authentication algorithms, known as "methods". In EAP, keying material is generated by EAP methods. Part of this keying material may be used by EAP methods themselves and part of this material may be exported. The exported keying material may be transported by AAA protocols or used by Secure Association Protocols in the generation or transport of session keys which are used by lower layer ciphersuites. This document describes each of these elements and provides a system-level security analysis. It also specifies the EAP key hierarchy.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [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 authenticator 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 [RFC4072]. In this document, the terms "AAA server" and "backend authentication server" are used interchangeably.

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.

security association
A set of policies and cryptographic state used to protect information. Elements of a security association may include cryptographic keys, negotiated ciphersuites and other parameters, counters, sequence spaces, authorization attributes, etc.

Long Term Credential
EAP methods frequently make use of long term secrets in order to enable authentication between the peer and server. In the case of a method based on pre-shared key authentication, the long term credential is the pre-shared key. In the case of a public-key based method, the long term credential is the corresponding private key.

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

Extended Master Session Key (EMSK)
Additional keying material derived between the peer and server that is exported by the EAP method. The EMSK is at least 64 octets in length, and is never shared with a third party.

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

Pairwise Master Key (PMK)
Lower layers use MSK in lower-layer dependent manner. For instance, in [IEEE-802.11i] Octets 0-31 of the MSK are known as the

Pairwise Master Key (PMK). In [IEEE-802.11i] the TKIP and AES CCMP ciphersuites derive their Transient Session Keys (TSKs) solely from the PMK, whereas the WEP ciphersuite as noted in [RFC3580], derives its TSKs from both halves of the MSK. In [802.16e], the MSK is truncated to 40 octets for PMK and 20 octets for PMK2.

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. The TEKs are stored locally by the EAP method and are not exported. 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 A.

Transient Session Keys (TSKs)
Session keys used to protect data exchanged after EAP authentication has successfully completed, using the ciphersuite negotiated between the EAP peer and authenticator.

AAA-Key
The term AAA-Key is synonymous with MSK.

1.3. Overview

EAP, defined in [RFC3748], is a two-party protocol spoken between the EAP peer and server. Within EAP, keying material is generated by EAP methods. Part of this keying material may be used by EAP methods themselves and part of this material may be exported. In addition to export of keying material, EAP methods may also export associated parameters, and may import and export Channel Bindings from the lower layer.

As illustrated in Figure 1, the EAP method key derivation has at the root the long term credential utilized by the selected EAP method. If authentication is based on a pre-shared key, the parties store the EAP method to be used and the pre-shared key. The EAP server also stores the peer's identity as well as other information associated with it. This information may be used to determine whether access to some service should be granted. The peer stores information necessary to choose which secret to use for which service.

If authentication is based on proof of possession of the private key corresponding to the public key contained within a certificate, the parties store the EAP method to be used and the trust anchors used to

validate the certificates. The EAP server also stores the peer's identity and the peer stores information necessary to choose which certificate to use for which service.

Based on the long term credential established between the peer and the server, EAP methods derive two types of keys:

[1] Keys calculated locally by the EAP method but not exported by the EAP method, such as the TEKs.
[2] Keying material exported by the EAP method: MSK, EMSK, IV.

As noted in [RFC3748] Section 7.10, EAP methods generating keys are required to calculate and export the MSK and EMSK, which must be at least 64 octets in length. EAP methods also may export the IV; however, the use of the IV is deprecated.

EAP methods also MAY export method-specific peer and server identifiers (peer-ID and server-ID), a method-specific EAP conversation identifier known as the Method-ID, and the lifetime of the exported keys, known as the Key-Lifetime. EAP methods MAY also support the import and export of Channel Bindings. New EAP method specifications MUST define the Peer-ID, Server-ID and Method-ID. The combination of the Peer-ID and Server-ID uniquely specifies the endpoints of the EAP method exchange.

Peer-ID

As described in [RFC3748] Section 7.3, the identity provided in the EAP-Response/Identity, may be different from the peer identity authenticated by the EAP method. Where the EAP method authenticates the peer identity, that identity is exported by the method as the Peer-ID. A suitable EAP peer name may not always be available. Where an EAP method does not define a method-specific peer identity, the Peer-ID is the null string. The Peer-ID for existing EAP methods is defined in Appendix B.

Server-ID

Where the EAP method authenticates the server identity, that identity is exported by the method as the Server-ID. A suitable EAP server name may not always be available. Where an EAP method does not define a method-specific peer identity, the Server-ID is the null string. The Server-ID for existing EAP methods is defined in Appendix B.

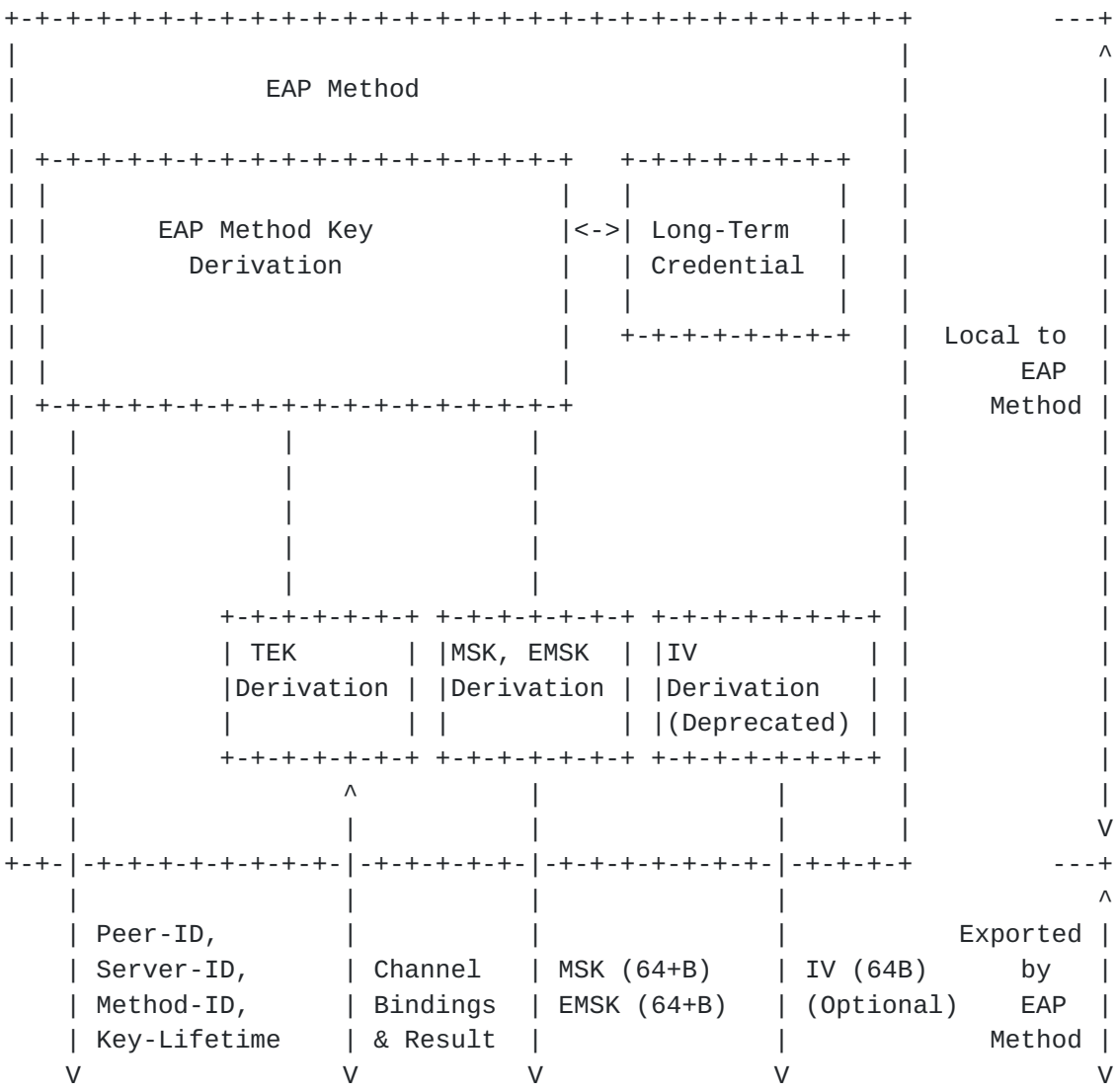


Figure 1: EAP Method Parameter Import/Export

Method-ID

EAP method specifications deriving keys MUST specify a temporally unique method identifier known as the Method-ID. The EAP Method-ID uniquely identifies an EAP session of a given Type between an EAP peer and server. The Method-ID is typically constructed from nonces or counters used within the EAP method exchange. The Method-ID for existing EAP methods is defined in Appendix B.

Session-ID

The Session-ID uniquely identifies an EAP session between an EAP peer (as identified by the Peer-ID) and server (as identified by the Server-ID). The EAP Session-ID consists of the concatenation of the

Expanded EAP Type Code (including the Type, Vendor-ID and Vendor-Type fields defined in [RFC3748] Section 5.7) and the Method-ID. The inclusion of the Expanded Type Code in the EAP Session-Id ensures that each EAP method has a distinct Session-ID space. Since an EAP session is not bound to a particular authenticator or specific ports on the peer and authenticator, the authenticator port or identity are not included in the Session-Id.

Key-Lifetime

While EAP itself does not support key lifetime negotiation, it is possible to specify methods that do. However, systems that rely on such negotiation for exported keys would only function with these methods. As a result, it is NOT RECOMMENDED to use this approach as the sole way to determine key lifetimes.

Channel Bindings

Channel Bindings include lower layer parameters that are verified for consistency between the EAP peer and server. In order to avoid introducing media dependencies, EAP methods that transport Channel Binding data MUST treat this data as opaque octets. Typically the EAP method imports Channel Bindings from the lower layer on the peer, and transmits them securely to the EAP server, which exports them to the lower layer. However, transport may occur from EAP server to peer, or may be bi-directional. On the side of the exchange (peer or server) where Channel Bindings are verified, the lower layer passes the result of the verification (TRUE or FALSE) up to the EAP method.

1.3.1. Key Naming

Each key created within the EAP key management framework has a name (a unique identifier), as well as a scope (the parties to whom the key is available). The scope of exported parameters is defined by the EAP peer name (if securely exchanged within the method) and the EAP server name (also only if securely exchanged). Where a peer or server name is missing the null string is used.

MSK and EMSK Names

These parameters are exported by the EAP peer and EAP server, and can be referred to using the EAP Session-ID and a binary or textual indication of the parameter being referred to.

PMK Name

This document does not specify a naming scheme for the PMK. The PMK is only identified by the key from which it is derived.

Note: IEEE 802.11i names the PMKID for the purposes of being able

to refer to it in the Secure Association protocol; this naming is based on a hash of the PMK itself as well as some other parameters (see Section 8.5.1.2 [IEEE-802.11i]).

TEK Name
The TEKs may or may not be named. Their naming is specified in the EAP method.

TSK Name
The TSKs are typically named. Their naming is specified in the lower layer so that the correct set of transient session keys can be identified for processing a given packet.

1.4. EAP Invariants

Certain basic characteristics, known as "EAP Invariants", hold true for EAP implementations on all media:

Mode independence
Media independence
Method independence
Ciphersuite independence

1.4.1. Mode Independence

EAP is typically deployed in order to support extensible network access authentication in situations where a peer desires network access via one or more authenticators. Where authenticators are deployed standalone, the EAP conversation occurs between the peer and authenticator, and the authenticator must locally implement an EAP method acceptable to the peer. However, one of the advantages of EAP is that it 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. This is accomplished by encapsulating EAP packets within the Authentication, Authorization and Accounting (AAA) protocol, spoken between the authenticator and backend authentication server. AAA protocols supporting EAP include RADIUS [RFC3579] and Diameter [RFC4072].

It is a fundamental property of EAP that at the EAP method layer, the conversation between the EAP peer and server is unaffected by whether the EAP authenticator is operating in "pass-through" mode. EAP methods operate identically in all aspects, including key derivation

and parameter import/export, regardless of whether the authenticator is operating as a pass-through or not.

The successful completion of an EAP method that supports key derivation results in the export of keying material on the EAP peer and server. Even though the EAP peer or server may import Channel-Bindings that may include the identity of the EAP authenticator, this information is treated as opaque octets. As a result, within EAP the only relevant identities are the Peer-ID and Server-ID. Channel Bindings are only interpreted by the lower layer.

Within EAP, the primary function of the AAA protocol is to maintain the principle of Mode Independence, so that as far as the EAP peer is concerned, its conversation with the EAP authenticator, and all consequences of that conversation, are identical, regardless of the authenticator mode of operation.

1.4.2. Media Independence

One of the goals of EAP is to allow EAP methods to function on any lower layer meeting the criteria outlined in [RFC3748], Section 3.1. For example, as described in [RFC3748], EAP authentication can be run over PPP [RFC1661], IEEE 802 wired networks [IEEE-802.1X], and IEEE 802.11 wireless LANs [IEEE-802.11i].

In order to maintain media independence, it is necessary for EAP to avoid consideration of media-specific elements. For example, EAP methods cannot be assumed to have knowledge of the lower layer over which they are transported, and cannot be restricted to identifiers associated with a particular usage environment (e.g. MAC addresses).

Note that media independence may be retained within EAP methods that support Channel-Bindings or method-specific identification. An EAP method need not be aware of the content of an identifier in order to use it. This enables an EAP method to use media-specific identifiers such as MAC addresses without compromising media independence. Channel-Bindings are treated as opaque octets by EAP methods, so that handling them does not require media-specific knowledge.

1.4.3. Method Independence

By enabling pass-through, authenticators can support any method implemented on the peer and server, not just locally implemented methods. This allows the authenticator to avoid implementing code for each EAP method required by peers. In fact, since a pass-through authenticator is not required to implement any EAP methods at all, it cannot be assumed to support any EAP method-specific code.

As a result, as noted in [RFC3748], authenticators must by default be capable of supporting 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 [RFC3748] 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 LAN authentication [RFC4017]. However, despite this it is possible for the peer and authenticator to interoperate as long as a suitable EAP method is supported on the EAP server.

1.4.4. Ciphersuite Independence

Ciphersuite Independence is a requirement for Media Independence. Since lower layer ciphersuites vary between media, media independence requires that EAP keying material needs to be large enough (with sufficient entropy) to handle any ciphersuite.

While EAP methods may negotiate the ciphersuite used in protection of the EAP conversation, the ciphersuite used for the protection of the data exchanged after EAP authentication has completed is negotiated between the peer and authenticator within the lower layer, outside of EAP.

For example, within PPP, the ciphersuite is negotiated within the Encryption Control Protocol (ECP) defined in [RFC1968], after EAP authentication is completed. Within [IEEE-802.11i], the AP ciphersuites are advertised in the Beacon and Probe Responses prior to EAP authentication, and are securely verified during a 4-way handshake exchange.

Since the ciphersuites used to protect data depend on the lower layer, requiring EAP methods have knowledge of lower layer ciphersuites would compromise the principle of Media Independence. Since ciphersuite negotiation occurs in the lower layer, there is no need for ciphersuite negotiation within EAP, and EAP methods generate keying material that is ciphersuite-independent.

Algorithms for deriving TSKs MUST NOT depend on the EAP method, although algorithms for TEK derivation MAY be specific to the EAP method.

In order to allow a ciphersuite to be usable within the EAP keying framework, a specification MUST be provided describing how TSKs suitable for use with the ciphersuite are derived from exported EAP keying parameters.

Advantages of ciphersuite-independence include:

Reduced 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.

Reduced EAP method complexity

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

Simplified configuration

The ciphersuite is negotiated between the peer and authenticator outside of EAP. Where the authenticator operates in "pass-through" mode, the EAP server is not a party to this negotiation, nor is it involved in the data flow between the EAP peer and authenticator. As a result, the EAP server may not have knowledge of the ciphersuites and negotiation policies implemented by the peer and authenticator, or be aware of the ciphersuite negotiated between them. For example, since ECP negotiation occurs after authentication, when run over PPP, the EAP peer and server may not anticipate the negotiated ciphersuite and therefore this information cannot be provided to the EAP method.

2. Lower Layer Operation

2.1. Overview

Where EAP key derivation is supported, the conversation typically takes place in three phases:

```
Phase 0: Discovery
Phase 1: Authentication
         1a: EAP authentication
         1b: AAA Key Transport (optional)
Phase 2: Secure Association Establishment
         2a: Unicast Secure Association
         2b: Multicast Secure Association (optional)
```

Of these phases, Phase 0, 1b and Phase 2 are handled by a lower layer. In the discovery phase (phase 0), peers locate authenticators and discover their capabilities. A peer may locate an authenticator providing access to a particular network, or a peer may locate an authenticator behind a bridge with which it desires to establish a Secure Association. Discovery can occur manually or

automatically, depending on the lower layer over which EAP runs.

The authentication phase (phase 1) may begin once the peer and authenticator discover each other. This phase, if it occurs, always includes EAP authentication (phase 1a). Where the chosen EAP method supports key derivation, in phase 1a EAP keying material is derived on both the peer and the EAP server.

An additional step (phase 1b) is required in deployments which include a backend authentication server, in order to transport keying material from the backend authentication server to the authenticator. In order to obey the principle of Mode Independence, where a backend server is present, all keying material which us required by the lower layer needs to be transported from the EAP server to the authenticator. Since existing TSK derivation techniques depend solely on the MSK, in existing implementations, this is the only keying material replicated in the AAA key transport phase 1b.

Successful completion of EAP authentication and key derivation by a 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).

The Secure Association exchange (phase 2) occurs between the peer and authenticator in order to manage the creation and deletion of unicast (phase 2a) and multicast (phase 2b) security associations between the peer and authenticator. The conversation between the parties is shown in Figure 2.

2.2. Layering

In completion of EAP authentication, EAP methods on the peer and EAP server export the Master Session Key (MSK), Extended Master Session Key (EMSK), Initialization Vector (IV), Peer-ID, Server-ID, Session-ID and Key-Lifetime. As illustrated in Figure 3, EAP methods export keying material and parameters to the EAP peer or authenticator layers.

The EAP peer and authenticator layers MUST NOT modify or cache keying material or parameters (including Channel Bindings) passing in either direction between the EAP method layer and the EAP layer. The EAP layer also MUST NOT cache keying material or parameters (including Channel Bindings) passed to it by the EAP peer/authenticator layer or the lower layer.

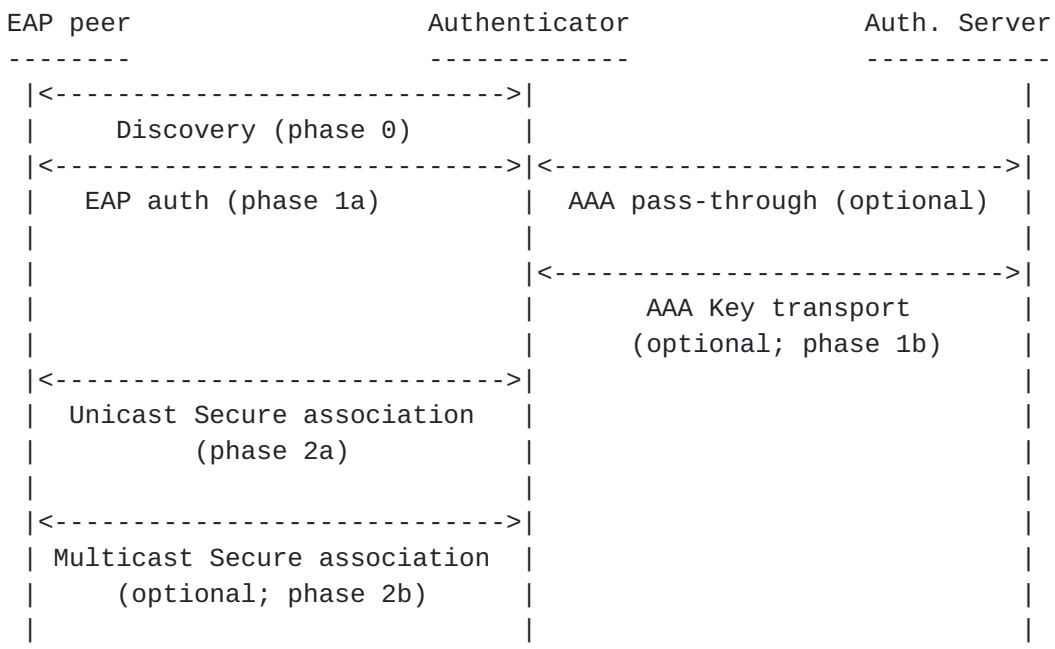


Figure 2: Conversation Overview

Based on the Method-ID exported by the EAP method, the EAP layer forms the EAP Session-ID by concatenating the EAP Expanded Type with the Method-ID. Together with the MSK, IV (deprecated), Peer-ID, Server-ID, and Key-Lifetime, the EAP layer passes the Session-ID down to the lower layer. The Method-ID is exported by EAP methods rather than the Session-ID so as to prevent EAP methods from writing into each other's Session- ID space.

The EMSK MUST NOT be provided to the lower layer, nor is it permitted to pass any quantity to the lower layer from which the EMSK could be computed without breaking some cryptographic assumption, such as inverting a one-way function. As noted in [RFC3748] Section 7.10:

> 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.)

In order to preserve the security of keys derived within EAP methods, lower layers other than AAA MUST NOT export keys passed down by EAP methods. This implies that EAP keying material or parameters passed down to a lower layer are for the exclusive use of that lower layer and MUST NOT be used within another lower layer. This prevents compromise of one lower layer from compromising other applications using EAP keying parameters.

EAP keying material and parameters provided to a lower layer other than AAA MUST NOT be transported to another entity. For example, EAP keying material and parameters passed down to the EAP peer lower layer MUST NOT leave the peer; EAP keying material and parameters passed down or transported to the EAP authenticator lower layer MUST NOT leave the authenticator.

The exception to the "no sharing" rule is the AAA layer. On EAP server, keying material requested by and passed down to the AAA layer may be replicated to the AAA layer on the authenticator. On the authenticator, the AAA layer may provide the replicated keying material to the lower layer over which the EAP authentication conversation took place. This enables "mode independence" to be maintained.

As illustrated in Figure 4, a AAA client receiving transported EAP keying material and parameters passes them to the EAP authenticator and EAP layers, which then provide them to the authenticator lower layer using the same mechanisms that would be used if the EAP peer and authenticator were conducting a stand-alone conversation. The resulting key state in the lower layer is indistinguishable between the standalone and pass-through cases, as required by the principle of mode independence.

2.3. Transient Session Keys

Where explicitly supported by the lower layer, lower layers MAY cache the exported EAP keying material and parameters and/or TSKs. The structure of this key cache is defined by the lower layer. So as to enable interoperability, new lower layer specifications MUST describe EAP key caching behavior. Unless explicitly specified by the lower layer, the EAP peer, server and authenticator MUST assume that peers and authenticators do not cache exported EAP keying parameters or TSKs. Existing EAP lower layers handle the caching of EAP keying material and the generation of transient session keys in different ways:

PPP PPP, defined in [RFC1661] does not support caching of EAP keying material or parameters. PPP ciphersuites derive their TSKs directly from the MSK, as described in [RFC2716]. This method is NOT RECOMMENDED, since were PPP to support caching, this could result in stale TSKs. As a result, once the PPP session is terminated, EAP keying material and parameters MUST be discarded. Since caching of EAP keying material is not permitted, within PPP there is no way to handle TSK rekey without EAP re-authentication. Perfect Forward Secrecy (PFS) is only possible within PPP if the negotiated EAP method supports this.

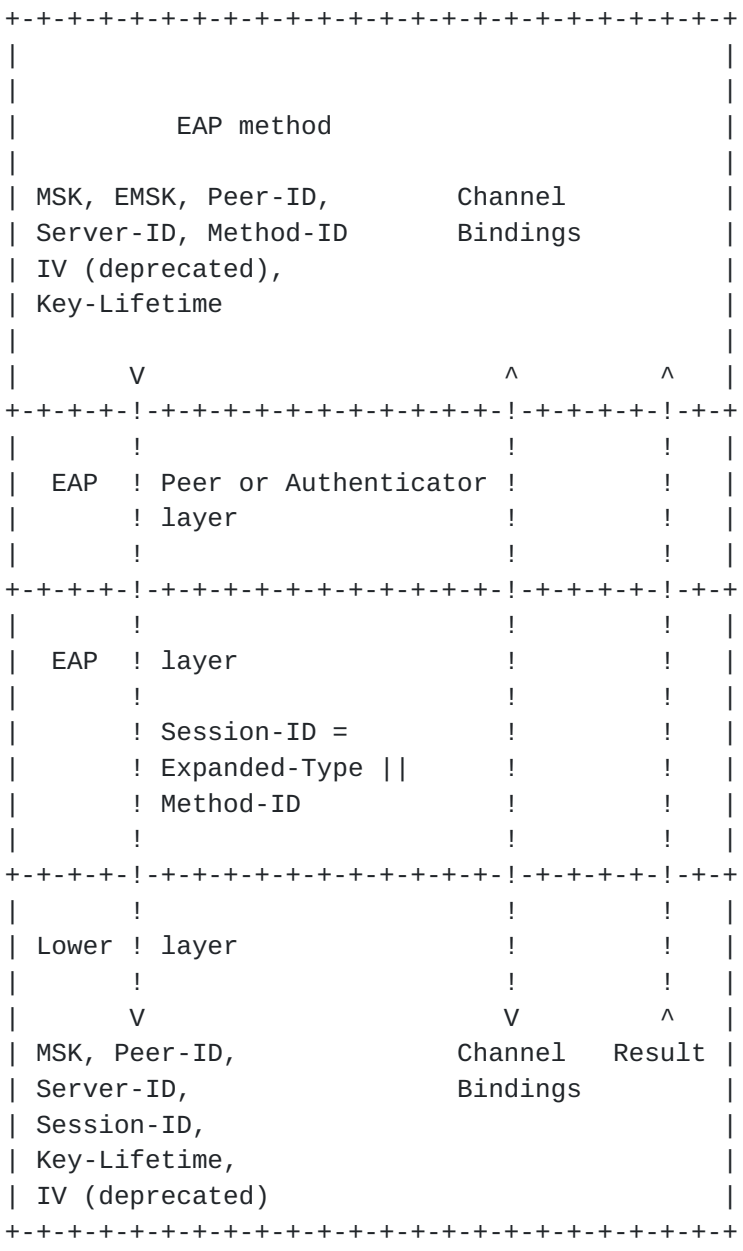


Figure 3: Flow of EAP parameters

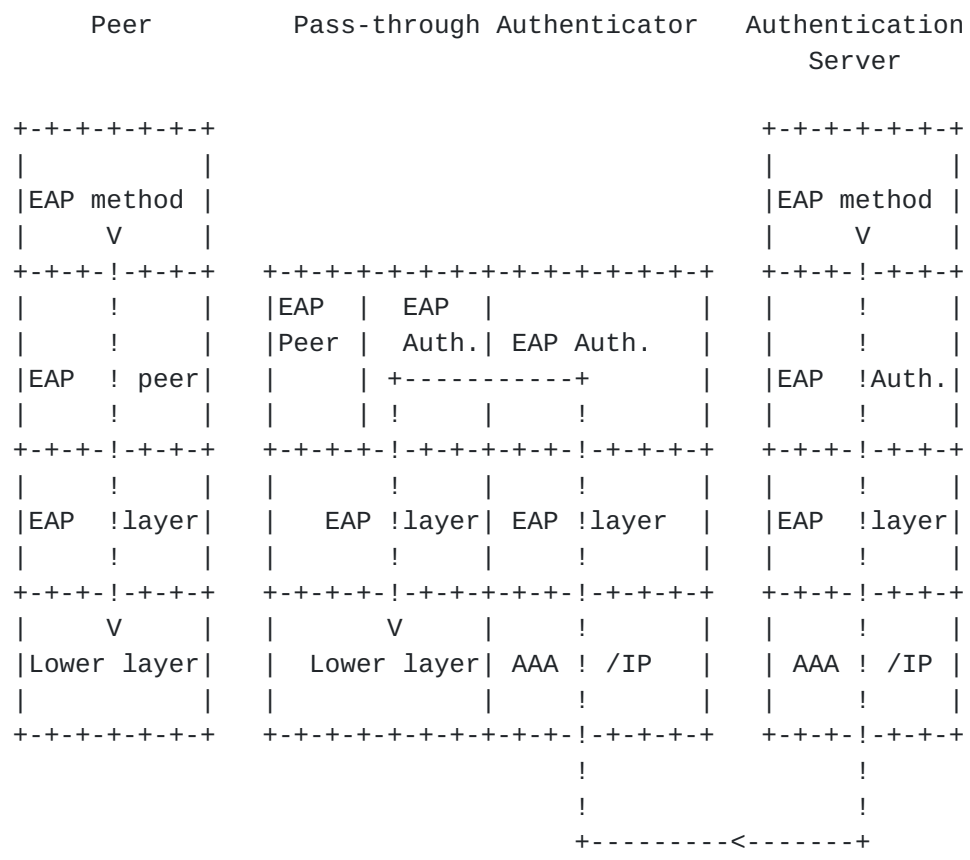


Figure 4: Flow of EAP Keying Material and Parameters

IKEv2

IKEv2, defined in [IKEv2] only uses EAP keying material for authentication purposes and not key derivation. As a result, the keying material derived within IKEv2 is independent of the EAP keying material and rekey of IPsec SAs can be handled without requiring EAP re-authentiation. Since generation of keying material is independent of EAP, within IKEv2 it is possible to negotiate PFS, regardless of the EAP method that is used. IKEv2 does not cache EAP keying material or parameters, nor does it utilize the EAP Key-Lifetime parameter to determine the lifetime of IPsec SAs. As a result, once IKEv2 authentication completes it is assumed that EAP keying material and parameters are discarded.

IEEE 802.11i

IEEE 802.11i enables caching of the MSK, but not the EMSK, IV, Peer-ID, Server-ID, or Session-ID. More details about the structure of the cache are available in [IEEE-802.11i]. In IEEE 802.11i, TSKs are derived from the MSK using the 4-way handshake, which includes a nonce exchange. This guarantees TSK freshness even if the MSK is reused. The 4-way handshake also enables TSK rekey without EAP re-authentication. PFS is only possible within

IEEE 802.11i if the negotiated EAP method supports this.

IEEE 802.1X-2004
IEEE 802.1X-2004, defined in [IEEE-802.1X-2004] does not support caching of EAP keying material or parameters. Once EAP authentication completes, it is assumed that EAP keying material and parameters are discarded.

IEEE 802.16e
IEEE 802.16e, defined in [IEEE-802.16e] supports caching of the MSK, but not the EMSK, IV, Peer-ID, Server-ID or Session-ID. In IEEE 802.16e, TSKs are generated by the authenticator without any contribution by the peer. The TSKs are encrypted, authenticated and integrity protected using the MSK. As a result, TSK rekey is possible without EAP re-authentication. PFS is not possible even if the negotiated EAP method supports it.

AAA Existing AAA implementations supporting RADIUS/EAP [RFC3579] or Diameter EAP [RFC4072] do not support caching of EAP keying material or parameters. In existing AAA client, proxy and server implementations, exported EAP keying material (MSK, EMSK and IV) as well as parameters and derived keys are not cached and MUST be presumed lost after the AAA exchange completes.

In order to avoid key reuse, the AAA layer MUST delete transported keys once they are sent. The AAA layer MUST NOT retain keys that it has previously sent. For example, a AAA layer that has transported the MSK MUST delete it, and keys MUST NOT be derived from the MSK from that point forward.

2.4. Key Scope

It should be understood that an EAP authenticator or peer:

[a] may contain one or more physical or logical ports;
[b] may advertise itself as one or more "virtual" authenticators or peers;
[c] may utilize multiple CPUs;
[d] may support clustering services for load balancing or failover.

The issues that arise from this are discussed below.

2.4.1. Multiple Ports

Both the EAP peer and authenticator may have more than one physical or logical port. A peer may simultaneously access the network via multiple authenticators, or via multiple physical or logical ports on a given authenticator. Similarly, an authenticator may offer network

access to multiple peers, each via a separate physical or logical port. The situation is illustrated in Figure 5.

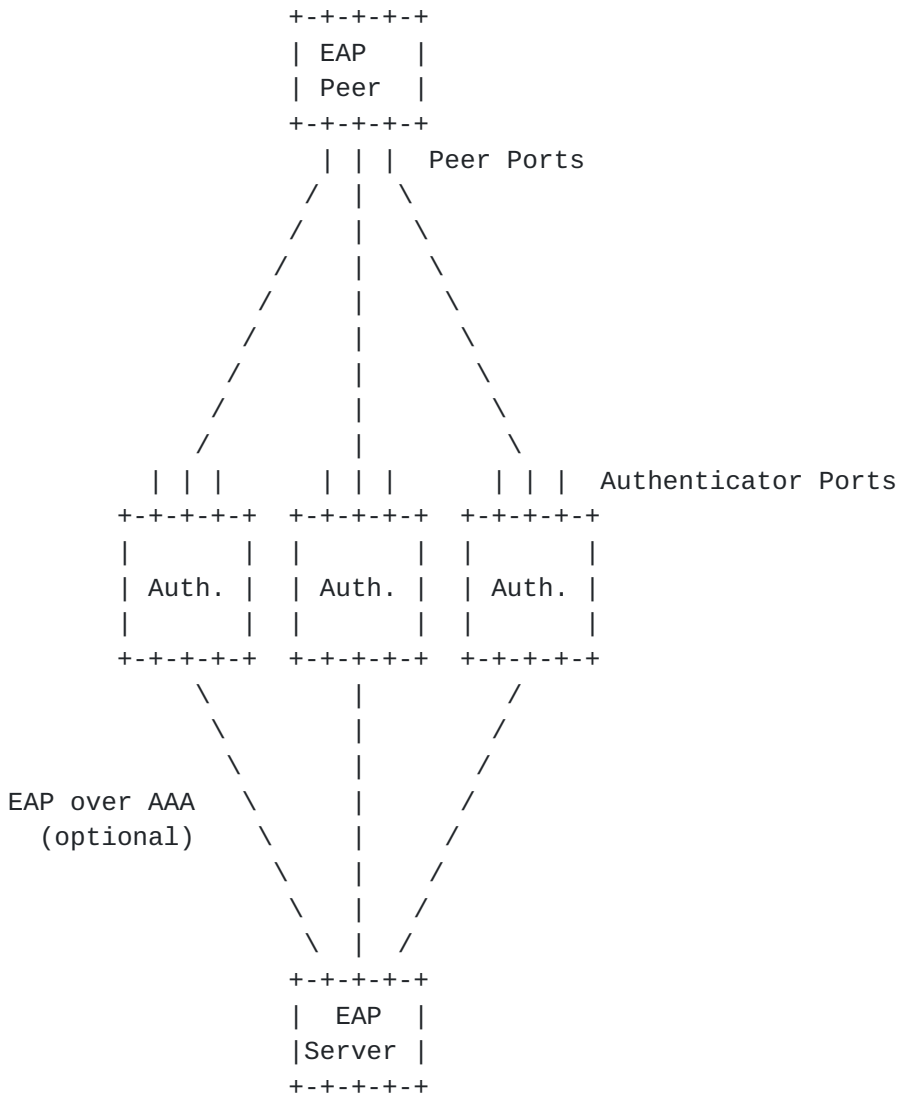


Figure 5: Relationship between EAP peer, authenticator and server

Absent explicit specification within the lower layer, EAP keying material and parameters are not bound to a specific peer or authenticator port. Where the peer and authenticator identify themselves within the lower layer using a port identifier such as a link layer address, this creates a problem, because it may not be obvious to the peer which authenticator ports are associated with which authenticators. Similarly, it may not be obvious to the authenticator which peer ports are associated with which peers. As a result, the peer and authenticator may not be able to determine the scope of the EAP keying material. This is particularly problematic for lower layers where key caching is supported.

For example, where the EAP peer cannot identify the EAP authenticator, it will be unable to determine whether EAP keying material has been shared outside of its authorized scope, and therefore needs to be considered compromised. There is also a practical problem because the EAP peer will be unable to utilize the EAP authenticator key cache in an efficient way.

The solution to this problem is for lower layers to identify EAP peers and authenticators unambiguously, without incorporating implicit assumptions about peer and authenticator architectures. Use of port identifiers is NOT RECOMMENDED where peers and authenticators may support multiple ports.

In order to further limit the key scope the following measures are suggested:

[a] The lower layer MAY specify additional restrictions on key usage, such as limiting the use of EAP keying material and parameters on the EAP peer to the port over which on the EAP conversation was conducted.

[b] The backend authentication server and authenticator MAY implement additional attributes in order to further restrict the scope of EAP keying material. For example, in 802.11, the backend authentication server may provide the authenticator with a list of authorized Called or Calling-Station-Ids and/or SSIDs for which EAP keying material is valid.

[c] Where the backend authentication server provides attributes restricting the key scope, it is RECOMMENDED that restrictions be securely communicated by the authenticator to the peer. This can be accomplished using the Secure Association Protocol, but also can be accomplished via the EAP method or the lower layer.

2.4.2. Authenticator Architecture

The EAP method conversation is between the EAP peer and server, as identified by the Peer-ID and Server-ID. The authenticator identity, if considered at all by the EAP method, is treated as an opaque blob for the purposes of Channel bindings. However, the Secure Association Protocol conversation is between the peer and the authenticator, and therefore the authenticator and peer identities are relevant to that exchange, and define the scope of use of the EAP keying material passed down to the lower layer.

Since an authenticator may have many ports, the authenticator identifier used within the Secure Association Protocol exchange SHOULD be distinct from any port identifier (e.g. MAC address).

Similarly, where a peer may have multiple ports, and sharing of EAP keying material and parameters between peer ports of the same link type is allowed, the peer identifier used within the Secure Association Protocol exchange SHOULD also be distinct from any port identifier.

While EAP Keying Material passed down to the lower layer is not intrinsically bound to particular authenticator and peer ports, Transient Session Keys MAY be bound to particular authenticator and peer ports by the Secure Association Protocol. However, a lower layer MAY also permit TSKs to be used on multiple peer and/or authenticator ports, providing that TSK freshness is guaranteed (such as by keeping replay counter state within the authenticator).

This specification does not impose constraints on the architecture of the EAP authenticator or peer. Any of the authenticator architectures described in [RFC4118] can be used. For example, it is possible for multiple base stations and a "controller" (e.g. WLAN switch) to comprise a single EAP authenticator. In such a situation, the "base station identity" is irrelevant to the EAP method conversation, except perhaps as an opaque blob to be used in Channel Bindings. Many base stations can share the same authenticator identity.

AAA protocols such as RADIUS [RFC3579] and Diameter [RFC4072] provide a mechanism for the identification of AAA clients; since the EAP authenticator and AAA client are always co-resident, this mechanism is applicable to the identification of EAP authenticators.

RADIUS [RFC2865] requires that an Access-Request packet contain one or more of the NAS-Identifier, NAS-IP-Address and NAS-IPv6-Address attributes. Since a NAS may have more than one IP address, the NAS-Identifier attribute is RECOMMENDED for the unambiguous identification of the EAP authenticator.

From the point of view of the AAA server, EAP keying material and parameters are transported to the EAP authenticator identified by the NAS-Identifier attribute. Since an EAP authenticator MUST NOT share EAP keying material or parameters with another party, if the EAP peer or AAA server detects use of EAP keying material and parameters outside the scope defined by the NAS-Identifier, the keying material MUST be considered compromised.

2.4.3. Virtual Authenticators

When a single physical authenticator advertises itself as multiple "virtual authenticators", the EAP peer and authenticator also may not be able to agree on the scope of the EAP keying material, creating a

security vulnerability. For example, the peer may assume that the "virtual authenticators" are distinct and do not share a key cache, whereas, depending on the architecture of the physical authenticator, a shared key cache may or may not be implemented.

Where EAP keying material is shared between "virtual authenticators" an attacker acting as a peer could authenticate with the "Guest" "virtual authenticator" and derive EAP keying material. If the virtual authenticators share a key cache, then the peer can utilize the EAP keying material derived for the "Guest" network to obtain access to the "Corporate Intranet" virtual authenticator.

Several measures are recommended to address these issues:

[d] Authenticators are REQUIRED to cache associated authorizations along with EAP keying material and parameters and to apply authorizations consistently. This ensures that an attacker cannot obtain elevated privileges even where the key cache is shared between "virtual authenticators".

[e] It is RECOMMENDED that physical authenticators maintain separate key caches for each "virtual authenticator".

[f] It is RECOMMENDED that each "virtual authenticator" identify itself distinctly to the backend authentication server, such as by utilizing a distinct NAS-Identifier attribute. This enables the backend authentication server to utilize a separate credential to authenticate each "virtual authenticator".

3. Key Management

EAP as defined in [RFC3748] supports key derivation, but not key management. While EAP methods may derive keying material, EAP does not provide for the management of exported or derived keys. For example, EAP does not support negotiation of the key lifetime of exported or derived keys, nor does it support re-key. Although EAP methods may support "fast reconnect" as defined in [RFC3748] Section 7.2.1, re-key of exported keys cannot occur without re-authentication. In order to provide method independence, key management of exported or derived keys SHOULD NOT be provided within EAP methods.

3.1. Secure Association Protocol

Since neither EAP nor EAP methods provide key management support, it is RECOMMENDED that key management facilities be provided within the Secure Association Protocol. This includes:

[a] Entity Naming. A basic feature of a Secure Association Protocol is the explicit naming of the parties engaged in the exchange. Without explicit identification, the parties engaged in the exchange are not identified and the scope of the EAP keying parameters negotiated during the EAP exchange is undefined. As shown in Figure 5, both the peer and authenticator may have more than one physical or virtual port, and as a result SHOULD identify themselves in a manner that is independent of their attached ports.

[b] Mutual proof of possession of EAP keying material. During the Secure Association Protocol the EAP peer and authenticator MUST demonstrate possession of the keying material transported between the backend authentication server and authenticator (e.g. MSK), in order to demonstrate that the peer and authenticator have been authorized. Since mutual proof of possession is not the same as mutual authentication, the peer cannot verify authenticator assertions (including the authenticator identity) as a result of this exchange.

[c] Secure capabilities negotiation. In order to protect against spoofing during the discovery phase, ensure selection of the "best" ciphersuite, and protect against forging of negotiated security parameters, the Secure Association Protocol MUST support secure capabilities negotiation. This includes the secure negotiation of usage modes, session parameters (such as security association identifiers (SAIDs) and key lifetimes), ciphersuites and required filters, including confirmation of security-relevant capabilities discovered during phase 0. As part of secure capabilities negotiation, the Secure Association Protocol MUST support integrity and replay protection of all messages.

[d] Key naming and selection. Where key caching is supported, it may be possible for the EAP peer and authenticator to share more than one key of a given type. As a result, 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. Use of the key naming mechanism described in this document is RECOMMENDED.

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.

[e] Generation of fresh transient session keys (TSKs). Where the lower layer supports caching of exported EAP keying material, the EAP peer lower layer may initiate a new session using keying material that was derived in a previous session. Were the TSKs to be derived from a portion of the exported EAP keying material, this would result in reuse of the session keys which could expose the underlying ciphersuite to attack.

In lower layers where caching of EAP keying material is supported, the Secure Association Protocol phase is REQUIRED, and MUST support the derivation of fresh unicast and multicast TSKs, even when the keying material provided by the backend authentication server is not fresh. This is typically supported via the exchange of nonces or counters, which are then mixed with the exported keying material in order to generate fresh unicast (phase 2a) and possibly multicast (phase 2b) session keys. By not using EAP keying material directly to protect data, the Secure Association Protocol protects it against compromise.

[f] Key lifetime management. This includes explicit key lifetime negotiation or seamless re-key. EAP does not support negotiation of key lifetimes, nor does it support re-key without re-authentication. As a result, the Secure Association Protocol may handle re-key and determination of the key lifetime. Where key caching is supported, secure negotiation of key lifetimes is RECOMMENDED. Lower layers that support re-key, but not key caching, may not require key lifetime negotiation. To take an example from IKE, the difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and re-keying the SA when necessary.

[g] Key resynchronization. It is possible for the peer or authenticator to reboot or reclaim resources, clearing portions or all of the key cache. Therefore, key lifetime negotiation cannot guarantee that the key cache will remain synchronized, and the peer may not be able to determine before attempting to use a key whether it exists within the authenticator cache. It is therefore RECOMMENDED for the Secure Association Protocol to provide a mechanism for key state resynchronization. Since in this situation one or more of the parties initially do not possess a key with which to protect the resynchronization exchange, securing this mechanism may be difficult.

[h] Key scope synchronization. Since the Discovery phase is handled out-of-band, EAP does not provide a mechanism by which the peer can

determine the authenticator identity. As a result, where the authenticator has multiple ports and key caching is supported, the EAP peer may not be able to determine the scope of validity of the exported EAP keying material. Similarly, where the EAP peer has multiple ports, the authenticator may not be able to determine whether a peer has authorization to use a particular key. To allow key scope determination, the Secure Association Protocol SHOULD provide a mechanism by which the peer can determine the scope of the key cache on each authenticator, and by which the authenticator can determine the scope of the key cache on a peer. This includes negotiation of restrictions on key usage.

[i] 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.

[j] 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.

3.2. Parent-Child Relationships

When keying material exported by EAP methods expires, all keying material derived from the exported keying material expires, including the TSKs.

When an EAP re-authentication takes place, new keying material is derived and exported by the EAP method, which eventually results in replacement of calculated keys, including the TSKs.

As a result, while the lifetime of calculated keys can be less than or equal that of the exported keys they are derived from, it cannot be greater. For example, TSK re-key may occur prior to EAP re-authentication.

Failure to mutually prove possession of keying material during the Secure Association Protocol exchange need not be grounds for deletion of the keying material by both parties; rate-limiting Secure

Association Protocol exchanges could be used to prevent a brute force attack.

3.3. Local Key Lifetimes

The Transient EAP Keys (TEKs) are session keys used to protect the EAP conversation. The TEKs are internal to the EAP method and are not exported. TEKs are typically created during an EAP conversation, used until the end of the conversation and then discarded. However, methods may re-key TEKs during a conversation.

When using TEKs within an EAP conversation or across conversations, it is necessary to ensure that replay protection and key separation requirements are fulfilled. For instance, if a replay counter is used, TEK re-key MUST occur prior to wrapping of the counter. Similarly, TSKs MUST remain cryptographically separate from TEKs despite TEK re-keying or caching. This prevents TEK compromise from leading directly to compromise of the TSKs and vice versa.

EAP methods may cache local keying material which may persist for multiple EAP conversations when fast reconnect is used [RFC 3748]. For example, EAP methods based on TLS (such as EAP-TLS [RFC2716]) derive and cache the TLS Master Secret, typically for substantial time periods. The lifetime of other local keying material calculated within the EAP method is defined by the method. Note that in general, when using fast reconnect, there is no guarantee to that the original long-term credentials are still in the possession of the peer. For instance, a card hold holding the private key for EAP-TLS may have been removed. EAP servers SHOULD also verify that the long-term credentials are still valid, such as by checking that certificate used in the original authentication has not yet expired.

3.4. Exported and Calculated Key Lifetimes

All EAP methods generating keys are required to generate the MSK and EMSK, and may optionally generate the IV. However, EAP, defined in [RFC3748], does not support the negotiation of lifetimes for exported keying material such as the MSK, EMSK and IV.

Several mechanisms exist for managing key lifetimes:

[a] AAA attributes. AAA protocols such as RADIUS [RFC2865] and Diameter [RFC4072] support the Session-Timeout attribute. The Session-Timeout value represents the maximum lifetime of the exported keys, and all keys calculated from it, on the authenticator. Since existing backend authentication servers do not cache keys exported by EAP methods, or keys calculated from exported keys, the value of the Session-Timeout attribute has no

bearing on the key lifetime within the backend authentication server.

On the authenticator, where EAP is used for authentication, the Session-Timeout value represents the maximum session time prior to re-authentication, as described in [RFC3580]. Where EAP is used for pre-authentication, the session may not start until some future time, or may never occur. Nevertheless, the Session-Timeout value represents the time after which transported EAP keying material, and all keys calculated from it, will have expired on the authenticator. If the session subsequently starts, re-authentication will be initiated once the Session-Time has expired. If the session never started, or started and ended, by default keys transported by AAA and all keys calculated from them will be expired by the authenticator prior to the future time indicated by Session-Timeout.

Since the TSK lifetime is often determined by authenticator resources, the backend authentication server has no insight into the TSK derivation process, and by the principle of ciphersuite independence, it is not appropriate for the backend authentication server to manage any aspect of the TSK derivation process, including the TSK lifetime.

[b] Lower layer mechanisms. While AAA attributes can communicate the maximum exported key lifetime, this only serves to synchronize the key lifetime between the backend authentication server and the authenticator. Lower layer mechanisms such as the Secure Association Protocol can then be used to enable the lifetime of exported and calculated keys to be negotiated between the peer and authenticator.

Where TSKs are established as the result of a Secure Association Protocol exchange, it is RECOMMENDED that the Secure Association Protocol include support for TSK resynchronization. Where the TSK is taken from the MSK, there is no need to manage the TSK lifetime as a separate parameter, since the TSK lifetime and MSK lifetime are identical.

[c] System defaults. Where the EAP method does not support the negotiation of the exported key lifetime, and a key lifetime negotiation mechanism is not provided by the lower lower, there may be no way for the peer to learn the exported key lifetime. In this case it is RECOMMENDED that the peer assume a default value of the exported key lifetime; 8 hours is recommended. Similarly, the lifetime of calculated keys can also be managed as a system parameter on the authenticator.

[d] Method specific negotiation within EAP. While EAP itself does not support lifetime negotiation, it would be possible to specify methods that do. However, systems that rely on such negotiation for exported keys would only function with these methods. As a result, it is NOT RECOMMENDED to use this approach as the sole way to determine key lifetimes.

3.5. Key cache synchronization

Issues arise when attempting to synchronize the key cache on the peer and authenticator. Lifetime negotiation alone cannot guarantee key cache synchronization.

One problem is that the AAA protocol cannot guarantee synchronization of key lifetimes between the peer and authenticator. Where the Secure Association Protocol is not run immediately after EAP authentication, the exported and calculated key lifetimes will not be known by the peer during the hiatus. Where EAP pre-authentication occurs, this can leave the peer uncertain whether a subsequent attempt to use the exported keys will prove successful.

However, even where the Secure Association Protocol is run immediately after EAP, it is still possible for the authenticator to reclaim resources if the created key state is not immediately utilized.

The lower layer may utilize Discovery mechanisms to assist in this. For example, the authenticator manages the key cache by deleting the oldest key first (LIFO), the relative creation time of the last key to be deleted could be advertised with the Discovery phase, enabling the peer to determine whether a given key had been expired from the authenticator key cache prematurely.

3.6. 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 RECOMMENDED that EAP methods utilizing public key cryptography choose a public key that has a cryptographic strength meeting the symmetric key strength requirement.

As noted in [RFC3766] Section 5, 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

3.7. 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 4.3, the security vulnerabilities of RADIUS are extensive, and therefore development of an alternative key wrap technique based on the RADIUS shared secret would not substantially improve security. As a result, [RFC3759] Section 4.2 recommends running RADIUS over IPsec. The same approach is taken in Diameter EAP [RFC4072], which defines cleartext key attributes, to be protected by IPsec or TLS.

Where an untrusted AAA intermediary is present (such as a RADIUS proxy or a Diameter agent), and data object security is not used, transported keying material may be recovered by an attacker in control of the untrusted intermediary. Possession of transported keying material enables decryption of data traffic sent between the peer and a specific authenticator. However, as long as EAP keying material or keys derived from it is only utilized by a single authenticator, compromise of the transported keying material does not enable an attacker to impersonate the peer to another authenticator. Vulnerability to an untrusted AAA intermediary can be mitigated by implementation of redirect functionality, as described in [RFC3588] and [RFC4072].

4. Handoff Vulnerabilities

With EAP, a number of mechanisms are be utilized in order to reduce the latency of handoff between authenticators. One such mechanism is EAP pre-authentication, in which EAP is utilized to pre-establish EAP keying material on an authenticator prior to arrival of the peer. Another such mechanism is key caching, in which an EAP peer can re-attach to an authenticator without having to re-authenticate using EAP. Yet another mechanism is context transfer, such as is defined in [IEEE-802.11F] (now deprecated) and [CTP]. These mechanisms introduce new security vulnerabilities, as discussed in the sections that follow.

4.1. Authorization

In a typical network access scenario (dial-in, wireless LAN, etc.) access control mechanisms are typically applied. These mechanisms include user authentication as well as authorization for the offered service.

As a part of the authentication process, the backend authentication server determines the user's authorization profile. The user authorizations are transmitted by the backend authentication server to the EAP authenticator (also known as the Network Access Server or authenticator) and with the transported EAP keying material, in Phase 1b of the EAP conversation. Typically, the profile is determined based on the user identity, but a certificate presented by the user may also provide authorization information.

The backend authentication server is responsible for making a user authorization decision, answering the following questions:

[a] Is this a legitimate user for this particular network?

[b] Is this user allowed the type of access he or she is requesting?

[c] Are there any specific parameters (mandatory tunneling, bandwidth, filters, and so on) that the access network should be aware of for this user?

[d] Is this user within the subscription rules regarding time of day?

[e] Is this user within his limits for concurrent sessions?

[f] Are there any fraud, credit limit, or other concerns that indicate that access should be denied?

While the authorization decision is in principle simple, the process is complicated by the distributed nature of the decision making. Where brokering entities or proxies are involved, all of the AAA entities in the chain from the authenticator to the home backend authentication server are involved in the decision. For instance, a broker can disallow access even if the home backend authentication server would allow it, or a proxy can add authorizations (e.g., bandwidth limits).

Decisions can be based on static policy definitions and profiles as well as dynamic state (e.g. time of day or limits on the number of concurrent sessions). In addition to the Accept/Reject decision made by the AAA chain, parameters or constraints can be communicated to the authenticator.

The criteria for Accept/Reject decisions or the reasons for choosing particular authorizations are typically not communicated to the authenticator, only the final result. As a result, the authenticator has no way to know what the decision was based on. Was a set of authorization parameters sent because this service is always provided to the user, or was the decision based on the time/day and the capabilities of the requesting authenticator device?

4.2. Correctness

When the AAA exchange is bypassed via use of techniques such as key caching, this creates challenges in ensuring that authorization is properly handled. These include:

[a] Consistent application of session time limits. Bypassing AAA should not automatically increase the available session time, allowing a user to endlessly extend their network access by changing the point of attachment.

[b] Avoidance of privilege elevation. Bypassing AAA should not result in a user being granted access to services which they are not entitled to.

[c] Consideration of dynamic state. In situations in which dynamic state is involved in the access decision (day/time, simultaneous session limit) it should be possible to take this state into account either before or after access is granted. Note that consideration of network-wide state such as simultaneous session limits can typically only be taken into account by the backend authentication server.

[d] Encoding of restrictions. Since a authenticator may not be aware of the criteria considered by a backend authentication server when

allowing access, in order to ensure consistent authorization during a fast handoff it may be necessary to explicitly encode the restrictions within the authorizations provided by the backend authentication server.

[e] State validity. The introduction of fast handoff should not render the authentication server incapable of keeping track of network-wide state.

A handoff mechanism capable of addressing these concerns is said to be "correct". One condition for correctness is as follows: For a handoff to be "correct" it MUST establish on the new device the same context as would have been created had the new device completed a AAA conversation with the backend authentication server.

A properly designed handoff scheme will only succeed if it is "correct" in this way. If a successful handoff would establish "incorrect" state, it is preferable for it to fail, in order to avoid creation of incorrect context.

Some backend authentication server and authenticator configurations are incapable of meeting this definition of "correctness". For example, if the old and new device differ in their capabilities, it may be difficult to meet this definition of correctness in a handoff mechanism that bypasses AAA. Backend authentication servers often perform conditional evaluation, in which the authorizations returned in an Access-Accept message are contingent on the authenticator or on dynamic state such as the time of day or number of simultaneous sessions. For example, in a heterogeneous deployment, the backend authentication server might return different authorizations depending on the authenticator making the request, in order to make sure that the requested service is consistent with the authenticator capabilities.

If differences between the new and old device would result in the backend authentication server sending a different set of messages to the new device than were sent to the old device, then if the handoff mechanism bypasses AAA, then the handoff cannot be carried out correctly.

For example, if some authenticator devices within a deployment support dynamic VLANs while others do not, then attributes present in the Access-Request (such as the authenticator-IP-Address, authenticator-Identifier, Vendor-Identifier, etc.) could be examined to determine when VLAN attributes will be returned, as described in [RFC3580]. VLAN support is defined in [IEEE-802.1Q]. If a handoff bypassing the backend authentication server were to occur between a authenticator supporting dynamic VLANs and another authenticator

which does not, then a guest user with access restricted to a guest VLAN could be given unrestricted access to the network.

Similarly, in a network where access is restricted based on the day and time, Service Set Identifier (SSID), Calling-Station-Id or other factors, unless the restrictions are encoded within the authorizations, or a partial AAA conversation is included, then a handoff could result in the user bypassing the restrictions.

In practice, these considerations limit the situations in which fast handoff mechanisms bypassing AAA can be expected to be successful. Where the deployed devices implement the same set of services, it may be possible to do successful handoffs within such mechanisms. However, where the supported services differ between devices, the handoff may not succeed. For example, [RFC2865] section 1.1 states:

> "A authenticator that does not implement a given service MUST NOT implement the RADIUS attributes for that service. For example, a authenticator that is unable to offer ARAP service MUST NOT implement the RADIUS attributes for ARAP. A authenticator MUST treat a RADIUS access-accept authorizing an unavailable service as an access-reject instead."

Note that this behavior only applies to attributes that are known, but not implemented. For attributes that are unknown, [RFC2865] Section 5 states:

> "A RADIUS server MAY ignore Attributes with an unknown Type. A RADIUS client MAY ignore Attributes with an unknown Type."

In order to perform a correct handoff, if a new device is provided with RADIUS context for a known but unavailable service, then it MUST process this context the same way it would handle a RADIUS Access-Accept requesting an unavailable service. This MUST cause the handoff to fail. However, if a new device is provided with RADIUS context that indicates an unknown attribute, then this attribute MAY be ignored.

Although it may seem somewhat counter-intuitive, failure is indeed the "correct" result where a known but unsupported service is requested. Presumably a correctly configured backend authentication server would not request that a device carry out a service that it does not implement. This implies that if the new device were to complete a AAA conversation that it would be likely to receive different service instructions. In such a case, failure of the handoff is the desired result. This will cause the new device to go back to the AAA server in order to receive the appropriate service definition.

In practice, this implies that handoff mechanisms which bypass AAA are most likely to be successful within a homogeneous device deployment within a single administrative domain. For example, it would not be advisable to carry out a fast handoff bypassing AAA between a authenticator providing confidentiality and another authenticator that does not support this service. The correct result of such a handoff would be a failure, since if the handoff were blindly carried out, then the user would be moved from a secure to an insecure channel without permission from the backend authentication server. Thus the definition of a "known but unsupported service" MUST encompass requests for unavailable security services. This includes vendor-specific attributes related to security, such as those described in [RFC2548].

5. Security Considerations

In order to analyze whether the EAP conversation achieves its security goals, it is first necessary to state those goals as well as the underlying security assumptions.

The overall goal of the EAP conversation is to derive fresh session keys between the EAP peer and authenticator that are known only to those parties, and for both the EAP peer and authenticator to demonstrate that they are authorized to perform their roles either by each other or by a trusted third party (the backend authentication server).

The principals of the authentication phase are the EAP peer and server. Completion of an EAP method exchange supporting key derivation results in the derivation of EAP keying material (MSK, EMSK, TEKs) known only to the EAP peer (identified by the Peer-ID) and server (identified by the Server-ID). Both the EAP peer and EAP server know the exported keying material to be fresh.

The principals of the AAA Key transport exchange are the EAP authenticator and the EAP server. Completion of the AAA exchange results in the transport of EAP keying material from the EAP server (identified by the Server-ID) to the EAP authenticator (identified by the NAS-Identifier) without disclosure to any other party. Both the EAP server and EAP authenticator know this keying material to be fresh.

The principals of the Secure Association Protocol are the EAP peer (identified by the Peer-ID) and authenticator (identified by the NAS-Identifier). Completion of the Secure Association Protocol results in the derivation of TSKs known only to the EAP peer and authenticator. Both the EAP peer and authenticator know the TSKs to be fresh.

5.1. Terminology

"Cryptographic binding", "Cryptographic separation", "Key strength" and "Mutual authentication" are defined in [RFC3748] and are used with the same meaning here.

5.2. Threat Model

The EAP threat model is described in [RFC3748] Section 7.1. The security properties of EAP methods (known as "security claims", described in [RFC3784] Section 7.2.1), address these threats. EAP method requirements for applications such as Wireless LAN authentication are described in [RFC4017]. The RADIUS threat model is described in [RFC3579] Section 4.1, and responses to these threats are described in [RFC3579] Sections 4.2 and 4.3.

However, in addition to threats against EAP and AAA, there are other system-level threats worth discussing. These include:

[1] An attacker may compromise or steal an EAP authenticator, in an attempt to gain access to other EAP authenticators or obtain long-term secrets.

[2] An attacker may compromise an EAP authenticator in an effort to commit fraud. For example, a compromised 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.

[3] An attacker may try to modify or spoof packets, including Discovery or Secure Association Protocol frames, EAP or AAA packets.

[4] An attacker may attempt a downgrade attack in order to exploit known weaknesses in an authentication method or cryptographic transform.

[5] An attacker may attempt to induce an EAP peer, authenticator or server to disclose keying material to an unauthorized party, or utilize keying material outside the context that it was intended for.

[6] An attacker may replay packets.

[7] An attacker may cause an EAP peer, authenticator or server to reuse an stale key. Use of stale keys may also occur unintentionally. For example, a poorly implemented backend authentication server may provide stale keying material to an authenticator, or a poorly

implemented authenticator may reuse nonces.

[8] An authenticated attacker may attempt to obtain elevated privilege in order to access information that it does not have rights to.

In order to address these threats, [Housley] provides a description of mandatory system security properties. Issues relating system security requirements are discussed in the sections that follow.

5.3. Authenticator Compromise

In the event that an authenticator is compromised or stolen, an attacker may gain access to the network via that authenticator, or may obtain the credentials required for that authenticator/AAA client to communicate with one or more backend authentication servers. However, this should not allow the attacker to compromise other authenticators or the backend authentication server, or obtain long-term user credentials.

The implications of this requirement are many, but some of the more important are as follows:

No Key Sharing
: An EAP authenticator MUST NOT share any keying material with another EAP authenticator, since if one EAP authenticator were compromised, this would enable the compromise of keying material on another authenticator. In order to be able to determine whether keying material has been shared, it is necessary for the identity of the EAP authenticator to be defined and understood by all parties that communicate with it.

No AAA Credential Sharing
: AAA credentials (such as RADIUS shared secrets, IPsec pre-shared keys or certificates) MUST NOT be shared between AAA clients, since if one AAA client were compromised, this would enable an attacker to impersonate other AAA clients to the backend authentication server, or even to impersonate a backend authentication server to other AAA clients.

No Compromise of Long-Term Credentials
: An attacker obtaining TSKs, TEKs or EAP keying material such as the MSK MUST NOT be able to obtain long-term user credentials such as pre-shared keys, passwords or private-keys without breaking a fundamental cryptographic assumption.

5.4. Spoofing

The use of per-packet authentication and integrity protection provides protection against spoofing attacks. Diameter [RFC3588] provides support for per-packet authentication and integrity protection via use of IPsec or TLS. RADIUS/EAP [RFC3579] provides for per-packet authentication and integrity protection via use of the Message-Authenticator attribute.

[RFC3748] Section 7.2.1 describes the "integrity protection" security claim and [RFC4017] requires use of EAP methods supporting this claim.

In order to prevent forgery of Secure Association Protocol frames, per-frame authentication and integrity protection is RECOMMENDED on all messages. [IEEE-802.11i] supports per-frame integrity protection and authentication on all messages within the 4-way handshake except the first message. An attack leveraging this ommission is described in [Analysis].

5.5. Downgrade Attacks

The ability to negotiate the use of a particular cryptographic algorithm provides resilience against compromise of a particular cryptographic algorithm. This is usually accomplished by including an algorithm identifier in the protocol, and by specifying the algorithm requirements in the protocol specification. In order to prevent downgrade attacks, secure confirmation of the "best" ciphersuite is required.

[RFC3748] Section 7.2.1 describes the "protected ciphersuite negotiation" security claim that 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. [RFC4017] requires EAP methods satisfying this security claim.

Diameter [RFC3588] provides support for cryptographic algorithm negotiation via use of IPsec or TLS. RADIUS [RFC3579] does not support the negotiation of cryptographic algorithms, and relies on MD5 for integrity protection, authentication and confidentiality, despite known weaknesses in the algorithm [MD5Attack]. This issue can be addressed via use of RADIUS over IPsec, as described in [RFC3579] Section 4.2.

As a result, EAP methods and AAA protocols are capable of addressing downgrade attacks. To ensure against downgrade attacks within lower layer protocols, algorithm independence is REQUIRED with lower layers using EAP for key derivation. For interoperability, at least one

suite of mandatory-to-implement algorithm MUST be selected. Lower layer protocols supporting EAP for key derivation SHOULD also support secure ciphersuite negotiation. As described in [RFC1968], PPP ECP does not provide support for secure ciphersuite negotiation. However, [IEEE-802.11i] does support secure ciphersuite negotiation.

5.6. Unauthorized Disclosure

While preserving algorithm independence, confidentiality of all keying material MUST be maintained. To prevent unauthorized disclose of keys, each party in the EAP conversation MUST be authenticated to the other parties with whom it communicates. Keying material MUST be bound to the appropriate context.

[RFC3748] Section 7.2.1 describes the "mutual authentication" and "dictionary attack resistance" claims, and [RFC4017] requires EAP methods satisfying these claims. EAP methods complying with [RFC4017] therefore provide for mutual authentication between the EAP peer and server. Binding of EAP keying material (MSK, EMSK) to the appropriate context is provided by the Peer-ID and Server-ID which are exported along with the keying material.

Diameter [RFC3588] provides for per-packet authentication and integrity protection via IPsec or TLS, and RADIUS/EAP [RFC3579] also provides for per-packet authentication and integrity protection. Where the NAS/authenticator and backend authentication server communicate directly and credible keywrap is used (see Section 3.7), this ensures that the AAA Key Transport phase achieves its security objectives: mutually authenticating the AAA client/authenticator and backend authentication server and providing EAP keying material to the EAP authenticator and to no other party.

As noted in Section 3.1, the Secure Association Protocol does not by itself provide for mutual authentication between the EAP peer and authenticator, even if mutual possession of EAP keying material is proven. However, where the NAS/authenticator and backend authentication server communicate directly, the backend authentication server can verify the correspondence between NAS identification attributes, the source address of packets sent by the NAS, and the AAA credentials. As long as the NAS has not shared its AAA credentials with another NAS, this allows the backend authentication server to authenticate the NAS. Using Channel Bindings, the EAP peer can then determine whether the NAS/authenticator has provided the same identifying information to the EAP peer and backend authentication server.

Peer and authenticator authorization MUST be performed. Authorization is REQUIRED whenever a peer associates with a new

authenticator. Authorization checking prevents an elevation of privilege attack, and ensures that an unauthorized authenticator is detected. Authorizations SHOULD be synchronized between the EAP peer, server, authenticator. Once the EAP conversation exchanges are complete, all of these parties should hold the same view of the authorizations associated the other parties. If peer authorization is restricted, then the peer SHOULD be made aware of the restriction.

The AAA exchange provides the EAP authenticator with authorizations relating to the EAP peer. However, neither the EAP nor AAA exchanges provides authorizations to the EAP peer. In order to ensure that all parties hold the same view of the authorizations it is RECOMMENDED that the Secure Association Protocol enable communication of authorizations between the EAP authenticator and peer.

In order to enable key binding and authorization of all parties, it is RECOMMENDED that the parties use a set of identities that are consistent between the conversation phases. RADIUS [RFC2865] and Diameter NASREQ [RFC4005] require that the NAS/EAP authenticator identify itself by including one or more identification attributes within an Access-Request packet (NAS-Identifier, NAS-IP-Address, NAS-IPv6-Address).

Since the backend authentication server provides EAP keying material for use by the EAP authenticator as identified by these attributes, where an EAP authenticator may have multiple ports, it is RECOMMENDED for the EAP authenticator to identify itself using NAS identification attributes during the Secure Association Protocol exchange with the EAP peer. This enables the EAP peer to determine whether EAP keying material has been shared between EAP authenticators as well as to confirm with the backend authentication server that an EAP authenticator proving possession of EAP keying material during the Secure Association Protocol was authorized to obtain it. Typically, the NAS-Identifier attribute is most convenient for this purpose, since a NAS/authenticator may have multiple IP addresses.

Similarly, the backend authentication server authorizes the EAP authenticator to provide access to the EAP peer identified by the Peer-ID, securely verified during the EAP authentication exchange. In order to determine whether EAP keying material has been shared between EAP peers, where the EAP peer has multiple ports it is RECOMMENDED for the EAP peer to identify itself using the Peer-ID during the Secure Association Protocol exchange with the EAP authenticator.

5.7. Replay Protection

Replay protection allows a protocol message recipient to discard any message that was recorded during a previous legitimate dialogue and presented as though it belonged to the current dialogue.

[RFC3748] Section 7.2.1 describes the "replay protection" security claim and [RFC4017] requires use of EAP methods supporting this claim.

Diameter [RFC3588] provides support for replay protection via use of IPsec or TLS. RADIUS/EAP [RFC3579] protects against replay of keying material via the Request Authenticator. However, some RADIUS packets are not replay protected. In Accounting, Disconnect and CoA-Request packets the Request Authenticator contains a keyed MAC rather than a Nonce. The Response Authenticator in Accounting, Disconnect and CoA Response packets also contains a keyed MAC whose calculation does not depend on a Nonce in either the Request or Response packets. Therefore unless an Event-Timestamp attribute is included or IPsec is used, the recipient may not be able to determine whether these packets have been replayed.

In order to prevent replay of Secure Association Protocol frames, replay protection is REQUIRED on all messages. [IEEE-802.11i] supports replay protection on all messages within the 4-way handshake.

5.8. Key Freshness

A session key should be considered compromised if it remains in use too long. As noted in [Housley], session keys MUST be strong and fresh, while preserving algorithm independence. A fresh cryptographic key is one that is generated specifically for the intended use. Each session deserves an independent session key; disclosure of one session key MUST NOT aid the attacker in discovering any other session keys.

Fresh keys are required even when a long replay counter (that is, one that "will never wrap") is used to ensure that loss of state does not cause the same counter value to be used more than once with the same session key.

EAP, AAA and the lower layer each bear responsibility for ensuring the use of fresh, strong session keys:

EAP EAP methods need to ensure the freshness and strength of EAP keying material provided as an input to session key derivation. [RFC3748] Section 7.10 states that "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." The contribution of nonces enables the EAP peer and server to ensure that exported EAP keying material is fresh.

[RFC3748] Section 7.2.1 describes the "key strength" and "session independence" security claims, and and [RFC4017] requires use of EAP methods supporting these claims as well as being capable of providing an equivalent key strength of 128 bits or greater.

AAA The AAA protocol needs to ensure that transported keying material is fresh and is not utilized outside its recommended lifetime. Replay protection is necessary for key freshness, but an attacker can deliver a stale (and therefore potentially compromised) key in a replay-protected message, so replay protection is not sufficient.

The EAP Session-ID, derived from the EAP Type and Method-ID (based on the nonces contributed by the peer and server) enables the EAP peer, authenticator and server to distinguish EAP conversations. However, unless the authenticator keeps track of EAP Session-IDs, the authenticator cannot use the Session-ID to guarantee the freshness of EAP keying material.

As described in [RFC3580] Section 3.17, When sent in an Access-Accept along with a Termination-Action value of RADIUS-Request, the Session-Timeout attribute specifies the maximum number of seconds of service provided prior to re-authentication. [IEEE-802.11i] also utilizes the Session-Timeout attribute to limit the maximum time that EAP keying material may be cache. Therefore the use of the Session-Timeout attribute enables the backend authentication server to limit the exposure of EAP keying material.

Lower Layer
The lower layer Secure Association Protocol MUST generate a fresh session key for each session, even if the keying material and parameters provided by EAP methods are cached, or the peer or authenticator lack a high entropy random number generator. A RECOMMENDED method is for the peer and authenticator to each provide a nonce or counter used in session key derivation. If a nonce is used, it is RECOMMENDED that it be at least 128 bits.

5.9. Elevation of Privilege

Parties MUST NOT have access to keying material that is not needed to perform their own role. A party has access to a particular key if it has access to all of the secret information needed to derive it. If

a post-EAP handshake is used to establish session keys, the post-EAP handshake MUST specify the scope for session keys.

Transported EAP keying material is permitted to be accessed by the EAP peer, authenticator and server. The EAP peer and server derive the transported keying material during the process of mutually authenticating each other using the selected EAP method. During the Secure Association Protocol, the EAP peer utilizes the transported EAP keying material to demonstrate to the authenticator that it is the same party that authenticated to the EAP server and was authorized by it. The EAP authenticator utilizes the transported EAP keying material to prove to the peer not only that the EAP conversation was transported through it (this could be demonstrated by a man-in-the-middle), but that it was uniquely authorized by the EAP server to provide the peer with access to the network. Unique authorization can only be demonstrated if the EAP authenticator does not share the transported keying material with a party other than the EAP peer and server.

TSKs are permitted to be accessed only by the EAP peer and authenticator. Since the TSKs can be determined from the transported EAP keying material and the cleartext of the Secure Association Protocol exchange, the backend authentication server will have access to the TSKs unless it deletes the transported EAP keying material after sending it.

5.10. Man-in-the-middle Attacks

As described in [I-D.puthenkulam-eap-binding], 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, [I-D.puthenkulam-eap-binding] recommends derivation of a compound key by which the EAP peer and server 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.

5.11. Denial of Service Attacks

Key caching may result in vulnerability to denial of service attacks. For example, EAP methods that create persistent state may be vulnerable to denial of service attacks on the EAP server by a rogue EAP peer.

To address this vulnerability, EAP methods creating persistent state may wish to limit the persistent state created by an EAP peer. For example, for each peer an EAP server may choose to limit persistent state to a few EAP conversations, distinguished by the EAP Session-ID. This prevents a rogue peer from denying access to other peers.

Similarly, to conserve resources an authenticator may choose to limit the persistent state corresponding to each peer. This can be accomplished by limiting each peer to persistent sttate corresponding to a few EAP converations, distinguished by the EAP Session-ID.

Depending on the media, creation of new TSKs may or may not imply deletion of previously derived TSKs. Where there is no implied deletion, the authenticator may choose to limit the number of TSKs and associated state that can be stored for each peer.

5.12. Impersonation

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

While AAA protocols such as RADIUS [RFC2865] or Diameter [RFC3588] support mutual authentication between the authenticator (known as the AAA client) and the backend authentication server (known as the backend authentication server), the security mechanisms vary according to the AAA protocol.

In RADIUS, the shared secret used for authentication is determined by the source address of the RADIUS packet. As noted in [RFC3579] Section 4.3.7, it is highly desirable that the source address be checked against one or more NAS identification attributes so as to detect and prevent impersonation attacks.

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 transorted keying material) 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] Section 4.3.7, this vulnerability can be mitigated by having RADIUS proxies check NAS identification attributes against the source address.

While [RFC3588] 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 [RFC3588].

5.13. 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 AAA or the lower layer).

Where EAP is used in pass-through mode, the EAP peer does not verify the identity of the pass-through authenticator. Within the Secure Association Protocol, the EAP peer and authenticator only demonstrate mutual possession of the transported EAP keying material. This creates a potential security vulnerability, described in [RFC3748] Section 7.15.

[RFC3579] Section 4.3.7 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 Called-Station-ID [RFC2865], 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 backend authentication server while communicating misleading information to

the EAP peer via the lower layer.

For example, a compromised authenticator can utilize another authenticator's Called-Station-Id or NAS-Identifier in communicating with the EAP peer via the lower layer, 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.

As noted in [RFC3748] Section 7.15, this vulnerability can be addressed by EAP methods that 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. For example, see [I-D.arkko-eap-service-identity-auth].

It is also possible to achieve Channel Bindings without transporting data over EAP. For example, see [draft-ohba-eap-aaakey-binding]. In this approach the authenticator informs the backend server about the Channel Binding parameters using AAA, and the backend server calculates transported keying material based on this parameter set, making it impossible for the peer and authenticator to complete the Secure Association Protocol if there was a mismatch in the parameters.

The main difference between these approaches is that Channel Binding support within an EAP method may require upgrading or changing the EAP method, impacting both the peer and the server. Where Channel Bindings are implemented in AAA, the peer, authenticator and the backend server need to be upgraded, but the EAP method need not be modified.

6. IANA Considerations

This document does not create any new name spaces nor does it allocate any protocol parameters.

7. References

7.1. Normative References

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

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

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

7.2. Informative References

[Analysis] He, C. and J. Mitchell, "Analysis of the 802.11i 4-Way Handshake", Proceedings of the 2004 ACM Workshop on Wireless Security, pp. 43-50, ISBN: 1-58113-925-X.

[CTP] Loughney, J., Nakhjiri, M., Perkins, C. and R. Koodli, "Context Transfer Protocol", draft-ietf-seamoby-ctp-11.txt, Internet draft (work in progress), August 2004.

[DESMODES] National Institute of Standards and Technology, "DES Modes of Operation", FIPS PUB 81, December 1980, <http://www.itl.nist.gov/fipspubs/fip81.htm>.

[FIPSDES] National Institute of Standards and Technology, "Data Encryption Standard", FIPS PUB 46, January 1977.

[Housley] Housley, R. and B. Aboba, "AAA Key Management", draft-housley-aaa-key-mgmt-01.txt, Internet draft (work in progress), November 2005.

[IEEE-802] Institute of Electrical and Electronics Engineers, "IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture", ANSI/IEEE Standard 802, 1990.

[IEEE-802.11] Institute of Electrical and Electronics Engineers, "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 IEEE Standard 802.11-2003, 2003.

[IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004, December 2004.

[IEEE-802.1Q]
Institute of Electrical and Electronics Engineers, "IEEE Standards for Local and Metropolitan Area Networks: Draft Standard for Virtual Bridged Local Area Networks", IEEE Standard 802.1Q/D8, January 1998.

[IEEE-802.11i]
Institute of Electrical and Electronics Engineers, "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", IEEE 802.11i, December 2004.

[IEEE-802.11F]
Institute of Electrical and Electronics Engineers, "Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation", IEEE 802.11F, July 2003 (now deprecated).

[IEEE-02-758]
Mishra, A., Shin, M., Arbaugh, W., Lee, I. and K. Jang, "Proactive Caching Strategies for IAPP Latency Improvement during 802.11 Handoff", IEEE 802.11 Working Group, IEEE-02-758r1-F Draft 802.11I/D5.0, November 2002.

[IEEE-03-084]
Mishra, A., Shin, M., Arbaugh, W., Lee, I. and K. Jang, "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.

[I-D.ietf-roamops-cert]
Aboba, B., "Certificate-Based Roaming", draft-ietf-roamops-cert-02 (work in progress), April 1999.

[I-D.puthenkulam-eap-binding]
Puthenkulam, J., "The Compound Authentication Binding Problem", draft-puthenkulam-eap-binding-04 (work in progress), October 2003.

[I-D.arkko-pppext-eap-aka]
Arkko, J. and H. Haverinen, "EAP AKA Authentication", draft-arkko-pppext-eap-aka-15.txt (work in progress), December 2004.

[I-D.arkko-eap-service-identity-auth]
Arkko, J. and P. Eronen, "Authenticated Service Information for the Extensible Authentication Protocol (EAP)", draft-arkko-eap-service-identity-auth-02.txt (work in progress), May 2005.

[I-D.ohba-eap-aaakey-binding]
Ohba, Y., "AAA-Key Derivation with Channel Binding", draft-ohba-eap-aaakey-binding-00.txt (work in progress), May 2005.

[IKEv2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", draft-ietf-ipsec-ikev2-17 (work in progress), September 2004.

[MD5Attack]
Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes, Vol.2 No.2, 1996.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[RFC1968] Meyer, G. and K. Fox, "The PPP Encryption Control Protocol (ECP)", RFC 1968, June 1996.

[RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2246] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, 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] Kummert, H., "The PPP Triple-DES Encryption Protocol (3DESE)", RFC 2420, September 1998.

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D. and R. Wheeler, "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.

[RFC3078] Pall, G. and G. Zorn, "Microsoft Point-To-Point Encryption (MPPE) Protocol", RFC 3078, March 2001.

[RFC3079] Zorn, G., "Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)", RFC 3079, March 2001.

[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.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G. and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

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

[RFC4005] Calhoun, P., Zorn, G., Spence, D. and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005.

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

[RFC4072] Eronen, P., Hiller, T. and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC4118] Yang, L., Zerfos, P. and E. Sadot, "Architecture Taxonomy for Control and Provisioning of Wireless Access Points (CAPWAP)", RFC 4118, June 2005.

[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.

Acknowledgments

Thanks to Arun Ayyagari, Ashwin Palekar, and Tim Moore of Microsoft, Dorothy Stanley of Agere, Bob Moskowitz of TruSecure, Jesse Walker of Intel, Joe Salowey of Cisco 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

Jari Arkko
Ericsson
Jorvas 02420
Finland

Phone:
EMail: jari.arkko@ericsson.com

Pasi Eronen
Nokia Research Center

P.O. Box 407
FIN-00045 Nokia Group
Finland

EMail: pasi.eronen@nokia.com

Henrik Levkowetz (editor)
ipUnplugged AB
Arenavagen 27
Stockholm S-121 28
SWEDEN

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

Appendix A - EAP-TLS Key Hierarchy

EAP-TLS [RFC 2716] was documented prior to the development of EAP key management terminology [RFC3748], and therefore does not explicitly define the MSK and EMSK.

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

[RFC2716] specifies that the MSK is divided into two halves, corresponding to the "Peer to Authenticator Encryption Key" (Enc-RECV-Key, 32 octets) and "Authenticator to Peer Encryption Key" (Enc-SEND-Key, 32 octets). In [RFC2548], the Enc-RECV-Key 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.

The key derivation scheme MUST be interpreted as follows:

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

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

```
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.
TMS           = 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 A-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.
```

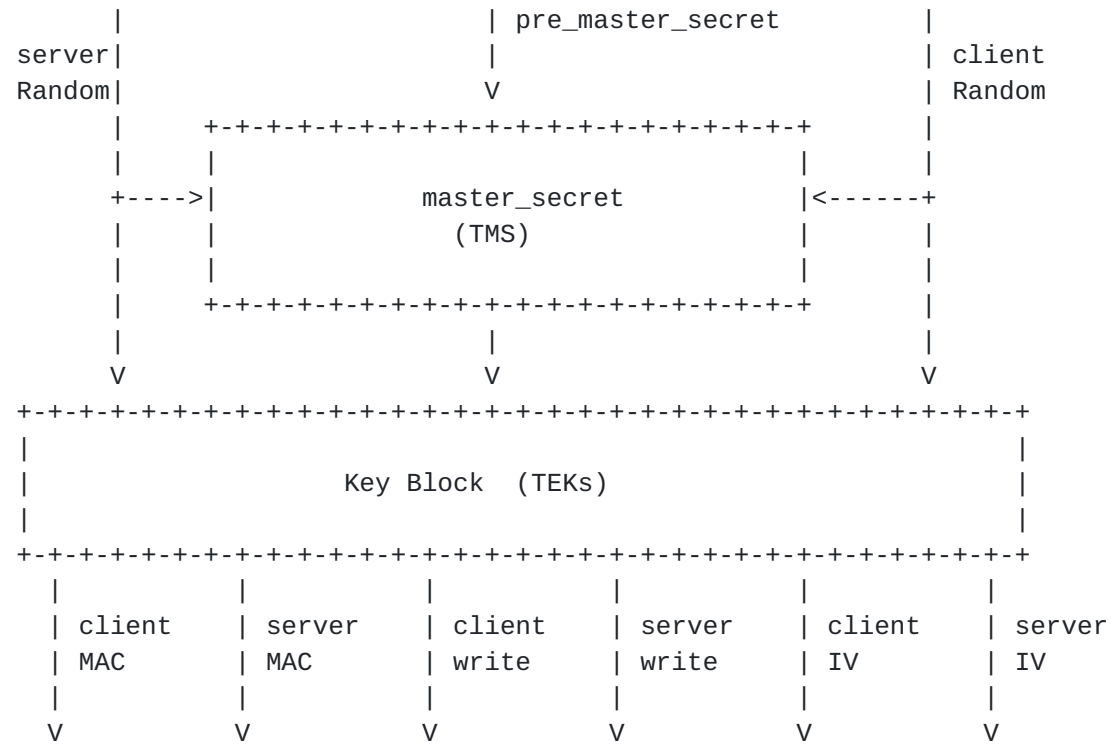


Figure A-1 - TLS [RFC2246] Key Hierarchy

Appendix B - Exported Parameters in Existing Methods

This Appendix specifies Method-ID, Peer-ID, Server-ID and Key-Lifetime for EAP methods that have been published prior to this specification. Future EAP method specifications MUST include a definition of the Method-ID, Peer-ID, and Server-ID (could be the empty string) and MAY also define the Key-Lifetime (assumed to be indeterminate if not described).

EAP-Identity

The EAP-Identity method does not derive keys, and therefore does not define the Key-Lifetime or Method-ID. The Peer-ID exported by the Identity method is determined by the octets included within the EAP- Response/Identity. The Server-ID is the empty string (zero length).

EAP-Notification

The EAP-Notification method does not derive keys and therefore does not define the Key-Lifetime and Method-ID. The Peer-ID and Server-ID are the empty string (zero length).

EAP-GTC

The EAP-GTC method does not derive keys and therefore does not define the Key-Lifetime and Method-ID. The Peer-ID and Server-ID are the empty string.

EAP-OTP

The EAP-OTP method does not derive keys and therefore does not define the Key-Lifetime and Method-ID. The Peer-ID and Server-ID are the empty string.

EAP-TLS

The EAP-TLS Method-Id is the concatenation of the peer and server nonces.

The Peer-ID and Server-ID are the contents of the altSubjectName in the peer and server certificates.

EAP-TLS does not negotiate a Key-Lifetime.

EAP-AKA

The EAP-AKA Method-Id is the contents of the RAND field from the

AT_RAND attribute, followed by the contents of the AUTN field in the AT_AUTN attribute.

The Peer-ID is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length octets from the beginning, however. Note that the contents are used as they are transmitted, regardless of whether the transmitted identity was a permanent, pseudonym, or fast re-authentication identity. The Server-ID is an empty string. EAP-AKA does not negotiate a key lifetime.

EAP-SIM

The Method-Id is the contents of the RAND field from the AT_RAND attribute, followed by the contents of the NONCE_MT field in the AT_NONCE_MT attribute.

The Peer-ID is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length octets from the beginning, however. Note that the contents are used as they are transmitted, regardless of whether the transmitted identity was a permanent, pseudonym, or fast re-authentication identity. The Server-ID is an empty string. EAP-SIM does not negotiate a key lifetime.

Intellectual Property Statement

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

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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

Open Issues

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

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