EAP Working Group
Internet Draft
Updates: 3748
Category: Standards Track
Expires: May 11, 2008

Bernard Aboba
Dan Simon
Microsoft Corporation
P. Eronen
Nokia

11 November 2007

**Extensible Authentication Protocol (EAP) Key Management Framework**
**draft-ietf-eap-keying-22.txt**



## Copyright Notice



## Abstract

The Extensible Authentication Protocol (EAP), defined in RFC 3748, enables extensible network access authentication. This document specifies the EAP key hierarchy and provides a framework for the transport and usage of keying material and parameters generated by EAP authentication algorithms, known as "methods". It also provides a detailed system-level security analysis, describing the conditions under which the key management guidelines described in RFC 4962 can be satisfied.

Table of Contents

## 1. Introduction

The Extensible Authentication Protocol (EAP), defined in [RFC3748], was designed to enable extensible authentication for network access in situations in which the Internet Protocol (IP) protocol is not available. Originally developed for use with Point-to-Point Protocol (PPP) [RFC1661], it has subsequently also been applied to IEEE 802 wired networks [IEEE-802.1X], IKEv2 [RFC4306] and wireless networks such as [IEEE-802.11] and [IEEE-802.16e].

EAP is a two-party protocol spoken between the EAP peer and server. Within EAP, keying material is generated by EAP authentication algorithms, known as "methods". Part of this keying material can be used by EAP methods themselves and part of this material can be exported. In addition to export of keying material, EAP methods can also export associated parameters such as authenticated peer and server identities and a unique EAP conversation identifier, and can import and export lower layer parameters known as "channel binding parameters", or simply "channel bindings".

This document specifies the EAP key hierarchy and provides a framework for the transport and usage of keying material and parameters generated by EAP methods. It also provides a detailed security analysis, describing the conditions under which the requirements described in "Guidance for Authentication, Authorization and Accounting (AAA) Key Management" [RFC4962] can be satisfied.

### 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 [RFC2119].

### 1.2. Terminology

The terms "Cryptographic binding", "Cryptographic separation", "Key strength" and "Mutual authentication" are defined in [RFC3748] and are used with the same meaning in this document, which also frequently uses the following terms:

4-Way Handshake
: A pairwise Authentication and Key Management Protocol (AKMP) defined in [IEEE-802.11], which confirms mutual possession of a Pairwise Master Key by two parties and distributes a Group Key.

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.

AAA-Key
: The term AAA-Key is synonymous with Master Session Key (MSK). Since multiple keys can be transported by AAA, the term is potentially confusing and is not used in this document.

authenticator
: The entity initiating EAP authentication.

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].

Channel Binding
: A secure mechanism for ensuring that a subset of the parameters transmitted by the authenticator (such as authenticator identifiers and properties) are agreed upon by the EAP peer and server. It is expected that the parameters are also securely agreed upon by the EAP peer and authenticator via the lower layer if the authenticator advertised the parameters.

Derived Keying Material
: Keys derived from EAP keying material, such as Transient Session Keys (TSKs).

EAP Keying Material
: Keys derived by an EAP method; this includes exported keying material (MSK, EMSK, IV) as well as local keying material such as Transient EAP Keys (TEKs).

EAP pre-authentication
: The use of EAP to pre-establish EAP keying material on an authenticator prior to arrival of the peer at the access network managed by that authenticator.

EAP re-authentication
: EAP authentication between an EAP peer and a server with whom the EAP peer shares valid unexpired EAP keying material.

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.

Exported keying material
: The EAP Master Session Key (MSK), Extended Master Session Key (EMSK), and Initialization Vector (IV).

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. The EMSK MUST be at least as long as the MSK in size.

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 [I-D.simon-emu-rfc2716bis], 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 it is OPTIONAL for EAP methods to generate it. However, when it is generated it MUST be unpredictable.

Keying Material
: Unless otherwise qualified, the term "keying material" refers to EAP keying material as well as derived keying material.

Key Scope
: The parties to whom a key is available.

Key Wrap
: The encryption of one symmetric cryptographic key in another. The algorithm used for the encryption is called a key wrap algorithm or a key encryption algorithm. The key used in the encryption process is called a key-encryption key (KEK).

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.

Lower Layer
: The lower layer is responsible for carrying EAP frames between the peer and authenticator.

Lower Layer Identity
: A name used to identify the EAP peer and authenticator within the lower layer.

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.

Network Access Server (NAS)
: A device that provides an access service for a user to a network.

Pairwise Master Key (PMK)
: Lower layers use the MSK in lower-layer dependent manner. For instance, in IEEE 802.11 [IEEE-802.11] Octets 0-31 of the MSK are known as the Pairwise Master Key (PMK); 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 20 octets for PMK and 20 octets for PMK2.

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

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

Secure Association Protocol
: An exchange that occurs between the EAP peer and authenticator in order to manage security associations derived from EAP exchanges. The protocol establishes unicast and (optionally) multicast security associations, which include symmetric keys and a context for the use of the keys. An example of a Secure Association Protocol is the 4-way handshake defined within [IEEE-802.11].

Session-Id
: The EAP Session-Id uniquely identifies an EAP authentication exchange between an EAP peer (as identified by the Peer-Id(s)) and server (as identified by the Server-Id(s)). For more information, see Section 1.4.

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.

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

## 1.3. 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 Protocol
         2a: Unicast Secure Association
         2b: Multicast Secure Association (optional)
```

Of these phases, Phase 0, 1b and Phase 2 are handled external to EAP. Phases 0 and 2 are handled by the lower layer protocol and phase 1b is typically handled by a AAA protocol.

In the discovery phase (phase 0), peers locate authenticators and discover their capabilities. A peer can locate an authenticator providing access to a particular network, or a peer can 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) can 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 needed 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 (see Section 1.6.1), where a backend server is present, all keying material needed by the lower layer is transported from the EAP server to the authenticator. Since existing TSK derivation and transport 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 Protocol 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 1.

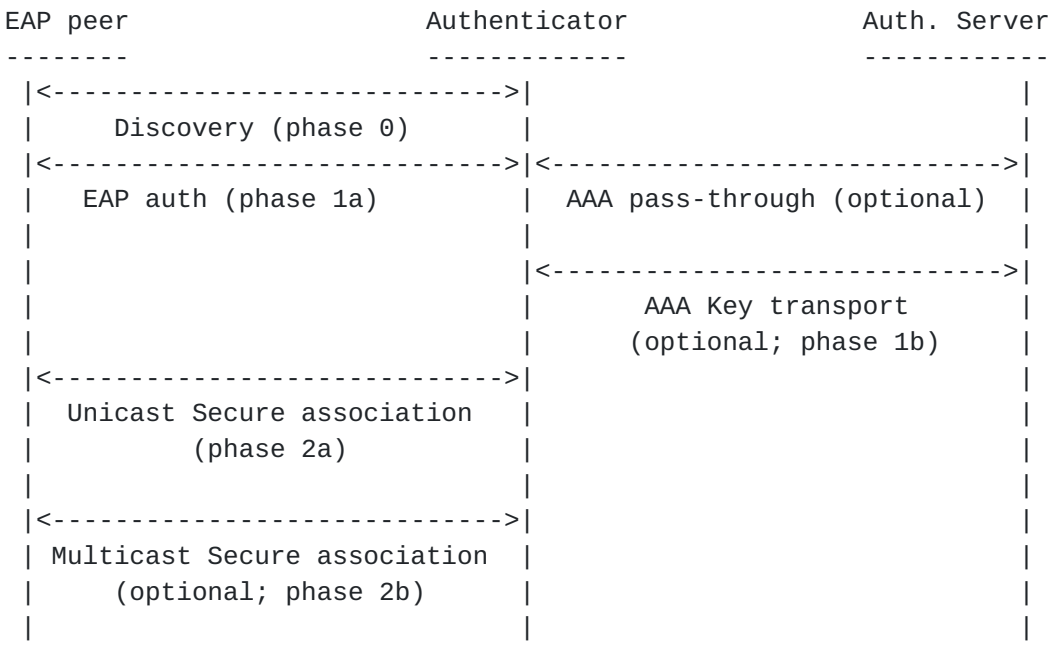


Figure 1: Conversation Overview

### 1.3.1. Examples

Existing EAP lower layers implement phase 0, 2a and 2b in different ways:

PPP The Point-to-Point Protocol (PPP), defined in [RFC1661] does not support discovery, nor does it include a Secure Association Protocol.

PPPoE
PPP over Ethernet (PPPoE), defined in [RFC2516], includes support for a Discovery stage (phase 0). In this step, the EAP peer sends a PPPoE Active Discovery Initiation (PADI) packet to the broadcast address, indicating the service it is requesting. The Access Concentrator replies with a PPPoE Active Discovery Offer (PADO)

packet containing its name, the service name and an indication of the services offered by the concentrator. The discovery phase is not secured. PPPoE, like PPP, does not include a Secure Association Protocol.

IKEv2
: Internet Key Exchange v2 (IKEv2), defined in [RFC4306], includes support for EAP and handles the establishment of unicast security associations (phase 2a). However, the establishment of multicast security associations (phase 2b) typically does not involve EAP and needs to be handled by a group key management protocol such as GDOI [RFC3547], GSAKMP [RFC4535], MIKEY [RFC3830], or GKDP [GKDP]. Several mechanisms have been proposed for discovery of IPsec security gateways. [RFC2230] discusses the use of Key eXchange (KX) Resource Records (RRs) for IPsec gateway discovery; while KX RRs are supported by many Domain Name Service (DNS) server implementations, they have not yet been widely deployed. Alternatively, DNS SRV RRs [RFC2782] can be used for this purpose. Where DNS is used for gateway location, DNS security mechanisms such as DNSSEC ([RFC4033], [RFC4035]), TSIG [RFC2845], and Simple Secure Dynamic Update [RFC3007] are available.

IEEE 802.11
: IEEE 802.11, defined in [IEEE-802.11], handles discovery via the Beacon and Probe Request/Response mechanisms. IEEE 802.11 access points periodically announce their Service Set Identifiers (SSIDs) as well as capabilities using Beacon frames. Stations can query for access points by sending a Probe Request to the broadcast address. Neither Beacon nor Probe Request/Response frames are secured. The 4-way handshake defined in [IEEE-802.11] enables the derivation of unicast (phase 2a) and multicast/broadcast (phase 2b) secure associations. Since the group key exchange transports a group key from the access point to the station, two 4-way handshakes can be needed in order to support peer-to-peer communications. A proof of the security of the IEEE 802.11 4-way handshake when used with EAP-TLS is provided in [He].

IEEE 802.1X
: IEEE 802.1X-2004, defined in [IEEE-802.1X] does not support discovery (phase 0), nor does it provide for derivation of unicast or multicast secure associations.

## 1.4. EAP Key Hierarchy

As illustrated in Figure 2, 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 additional information. This information is typically used outside of the EAP method to determine whether to grant access to a service. 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, methods derive two types of EAP keying material:

(a) Keying material calculated locally by the EAP method but not exported, such as the Transient EAP Keys (TEKs).
(b) Keying material exported by the EAP method: Master Session Key (MSK), Extended Master Session Key (EMSK), Initialization Vector (IV).

As noted in [RFC3748] Section 7.10:

> In order to provide keying material for use in a subsequently negotiated ciphersuite, an EAP method supporting key derivation MUST export a Master Session Key (MSK) of at least 64 octets, and an Extended Master Session Key (EMSK) of at least 64 octets.

EAP methods also MAY export the IV; however, the use of the IV is deprecated. The EMSK MUST NOT be provided to an entity outside the EAP server or peer, nor is it permitted to pass any quantity to an entity outside the EAP server or peer from which the EMSK could be computed without breaking some cryptographic assumption, such as inverting a one-way function.

EAP methods supporting key derivation and mutual authentication SHOULD export a method-specific EAP conversation identifier known as the Session-Id, as well as one or more method-specific peer identifiers (Peer-Id(s)) and MAY export one or more method-specific server identifiers (Server-Id(s)). EAP methods MAY also support the import and export of channel binding parameters. EAP method specifications developed after the publication of this document MUST define the Peer-Id, Server-Id and Session-Id. The Peer-Id(s) and Server-Id(s), when provided, identify the entities involved in generating EAP keying material. For existing EAP methods the Peer-Id, Server-Id and Session-Id are defined in Appendix A.

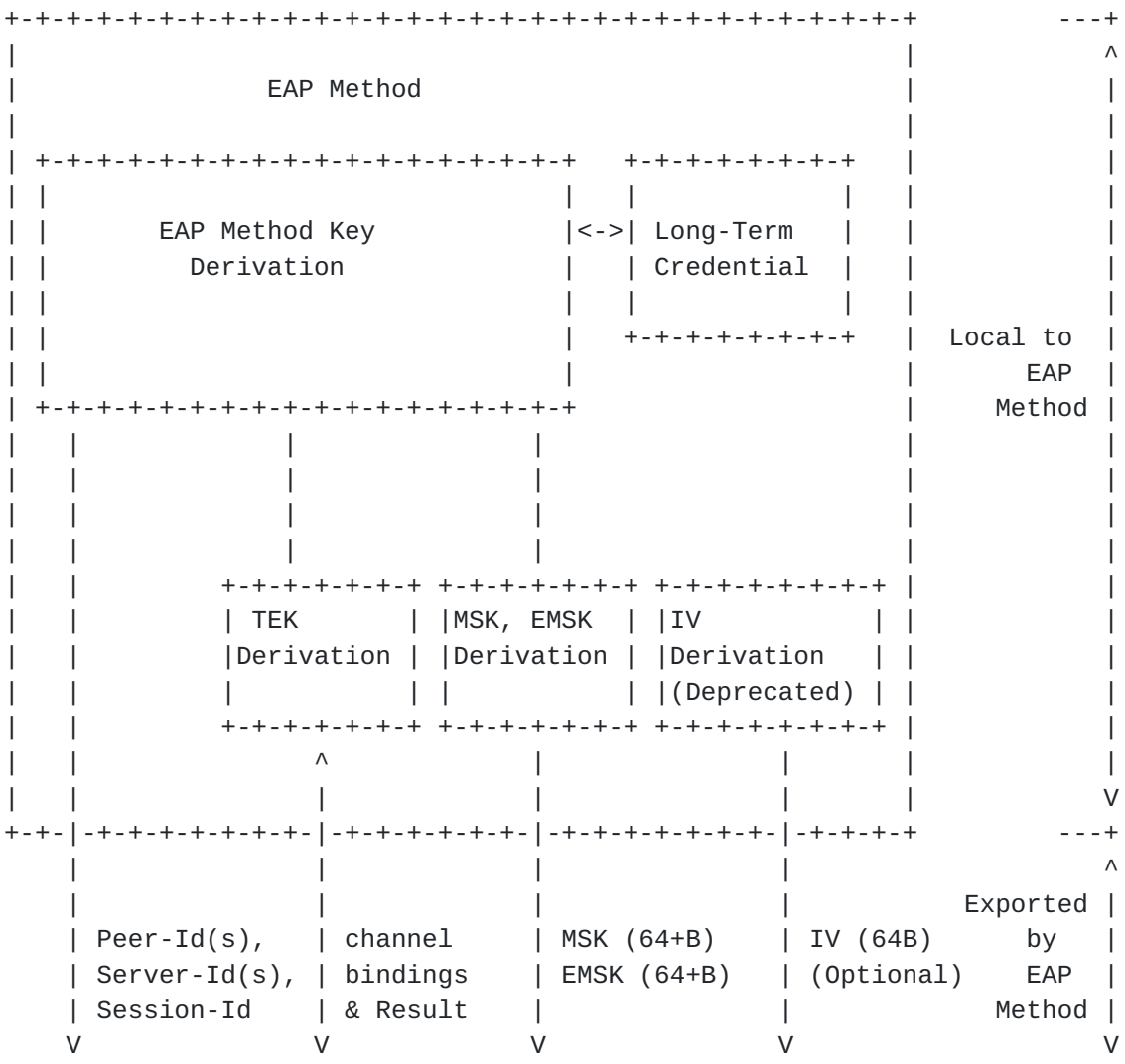


Figure 2: EAP Method Parameter Import/Export

Peer-Id

If an EAP method that generates keys authenticates one or more method-specific peer identities, those identities are exported by the method as the Peer-Id(s). It is possible for more than one Peer-Id to be exported by an EAP method. Not all EAP methods provide a method-specific peer identity; where this is not defined, the Peer-Id is the null string. In EAP methods that do not support key generation, the Peer-Id MUST be the null string. Where an EAP method that derives keys does not provide a Peer-Id, the EAP server will not authenticate the identity of the EAP peer with which it derived keying material.

Server-Id

If an EAP method that generates keys authenticates one or more method-specific server identities, those identities are exported by the method as the Server-Id(s). It is possible for more than one Server-Id to be exported by an EAP method. Not all EAP methods provide a method-specific server identity; where this is not defined, the Server-Id is the null string. If the EAP method does not generate keying material, the Server-Id MUST be the null string. Where an EAP method that derives keys does not provide a Server-Id, the EAP peer will not authenticate the identity of the EAP server with which it derived EAP keying material.

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). Where non-expanded EAP Type Codes are used (EAP Type Code not equal to 254), the EAP Session-Id is the concatenation of the single octet EAP Type Code and a temporally unique identifier obtained from the method (known as the Method-Id). Where expanded EAP Type Codes are used, the EAP Session-Id consists of the Expanded Type Code (including the Type, Vendor-Id and Vendor-Type fields defined in [RFC3748] Section 5.7) concatenated with a temporally unique identifier obtained from the method (Method-Id). The Method-Id is typically constructed from nonces or counters used within the EAP method exchange. The inclusion of the Type Code or 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.

Channel Binding

Channel Binding is the process by which lower layer parameters are verified for consistency between the EAP peer and server. In order to avoid introducing media dependencies, EAP methods that transport channel binding parameters MUST treat this data as opaque octets. See Section 5.3.3 for further discussion.

### 1.4.1. 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 keying material and TEKs is defined by the authenticated method-specific peer identities (Peer-Id(s)) and the authenticated server identities (Server-Id(s)), where available.

MSK and EMSK Names

The MSK and EMSK are exported by the EAP peer and EAP server, and MUST be named using the EAP Session-Id and a binary or textual indication of the EAP keying material being referred to.

PMK Name

This document does not specify a naming scheme for the Pairwise Master Key (PMK). The PMK is only identified by the name of the key from which it is derived.

Note: IEEE 802.11 names the PMK for the purposes of being able to refer to it in the Secure Association Protocol; the PMK name (known as the PMKID) is based on a hash of the PMK itself as well as some other parameters (see [IEEE-802.11] Section 8.5.1.2).

TEK Name

Transient EAP Keys (TEKs) MAY be named; their naming is specified in the EAP method specification.

TSK Name

Transient Session Keys (TSKs) are typically named. Their naming is specified in the lower layer so that the correct set of TSKs can be identified for processing a given packet.

## 1.5. Security Goals

The 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).

Completion of an EAP method exchange (Phase 1a) 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(s)) and EAP server (identified by the Server-Id(s)). Both the EAP peer and EAP server know this keying material to be fresh. The Peer-Id and Server-Id are discussed in Sections 1.4, 2.4 and 2.5 as well as in Appendix A. Key freshness is discussed in Sections 3.4, 3.5 and 5.7.

Completion of the AAA exchange (Phase 1b) results in the transport of keying material from the EAP server (identified by the Server-Id(s)) 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. Disclosure issues are discussed in Sections 3.8 and 5.3; security properties of AAA protocols are discussed in Sections 5.1-5.9.

The backend authentication server is trusted to transport keying material only to the authenticator that was established with the peer, and it is trusted to transport that keying material to no other parties. In many systems, EAP keying material established by the EAP peer and EAP server are combined with publicly available data to derive other keys. The backend authentication server is trusted to refrain from deriving these same keys or acting as a man-in-the-middle even though it has access to the keying material that is needed to do so.

The authenticator is also a trusted party. The authenticator is trusted not to distribute keying material provided by the backend authentication server to any other parties. If the authenticator uses a key derivation function to derive additional keying material, the authenticator is trusted to distribute the derived keying material only to the appropriate party that is known to the peer, and no other party. When this approach is used, care must be taken to ensure that the resulting key management system meets all of the principles in [RFC4962], confirming that keys used to protect data are to be known only by the peer and authenticator.

Completion of the Secure Association Protocol (Phase 2) results in the derivation or transport of Transient Session Keys (TSKs) known only to the EAP peer (identified by the Peer-Id(s)) and authenticator (identified by the NAS-Identifier). Both the EAP peer and authenticator know the TSKs to be fresh. Both the EAP peer and authenticator demonstrate that they are authorized to perform their roles. Authorization issues are discussed in Sections 4.3.2 and 5.5; security properties of Secure Association Protocols are discussed in Section 3.1.

## 1.6. EAP Invariants

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

Mode independence
Media independence
Method independence
Ciphersuite independence

### 1.6.1. Mode Independence

EAP is typically deployed 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 locally implements one or more

EAP methods. However, when utilized in "pass-through" mode, EAP enables deployment of new authentication methods without requiring development of new code on the authenticator.

While the authenticator can implement some EAP methods locally and use those methods to authenticate local users, it can 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 EAP keying material and parameters on the EAP peer and server. Even though the EAP peer or server can import channel binding parameters that can 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(s) and Server-Id(s). Channel binding parameters are only interpreted by the lower layer.

Within EAP, the primary function of the AAA protocol is to maintain the principle of mode independence. 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.

### 1.6.2. 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 wireless networks such as 802.11 [IEEE-802.11] and 802.16 [IEEE-802.16e].

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 can be retained within EAP methods that support Channel Binding 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 binding parameters are treated as opaque octets by EAP methods, so that handling them does not require media-specific knowledge.

### 1.6.3. 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 having to implement the EAP methods configured for use by peers. In fact, since a pass-through authenticator need not implement any EAP methods at all, it cannot be assumed to support any EAP method-specific code. As noted in [RFC3748] Section 2.3:

> Compliant pass-through authenticator implementations MUST by default forward EAP packets of any Type.

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 Local Area Network (WLAN) authentication [RFC4017]. However, despite this it is possible for the peer and authenticator to interoperate as long as a suitable EAP method is supported both on the EAP peer and server.

### 1.6.4. Ciphersuite Independence

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

While EAP methods can 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-802.11], 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 that EAP methods have knowledge of lower layer ciphersuites would compromise the principle of Media Independence. As a result, methods export EAP keying material that is ciphersuite-independent. Since ciphersuite negotiation occurs in the lower layer, there is no need for lower layer ciphersuite negotiation within EAP.

In order to allow a ciphersuite to be usable within the EAP keying framework, the ciphersuite specification needs to describe how TSKs suitable for use with the ciphersuite are derived from exported EAP keying material. To maintain Method Independence, algorithms for deriving TSKs MUST NOT depend on the EAP method, although algorithms for TEK derivation MAY be specific to the EAP method.

Advantages of ciphersuite-independence include:

Reduced update requirements
: Ciphersuite independence enables EAP methods to be used with new ciphersuites without requiring the methods to be updated. 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
: Ciphersuite independence enables EAP methods to avoid having to include ciphersuite-specific code. 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
: Ciphersuite independence enables EAP method implementations on the peer and server to avoid having to configure ciphersuite-specific parameters. 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 does not have knowledge of the ciphersuites and negotiation policies implemented by the peer and authenticator, nor is it aware of the ciphersuite negotiated between them. For example, since Encryption Control Protocol (ECP) negotiation occurs after authentication, when run over PPP, the EAP peer and server cannot anticipate the negotiated ciphersuite and therefore this information cannot be provided to the EAP method.

## 2. Lower Layer Operation

On completion of EAP authentication, EAP keying material and parameters exported by the EAP method are provided to the lower layer and AAA layer (if present). These include the Master Session Key (MSK), Extended Master Session Key (EMSK), Peer-Id(s), Server-Id(s) and Session-Id. The Initialization Vector (IV) is deprecated, but might be provided.

In order to preserve the security of EAP keying material derived within methods, lower layers MUST NOT export keys passed down by EAP methods. This implies that EAP keying material passed down to a lower layer is 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 material.

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

On the EAP server, keying material and parameters requested by and passed down to the AAA layer MAY be replicated to the AAA layer on the authenticator (with the exception of the EMSK). On the authenticator, the AAA layer provides the replicated keying material and parameters to the lower layer over which the EAP authentication conversation took place. This enables mode independence to be maintained.

The EAP layer as well as the 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 lower layer or AAA layer.

## 2.1. Transient Session Keys

Where explicitly supported by the lower layer, lower layers MAY cache keying material, including exported EAP keying material 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 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 keying material. Existing EAP lower layers and AAA layers handle the generation of transient session keys and caching of EAP keying material in different ways:

IEEE 802.1X-2004
: When used with wired networks, IEEE 802.1X-2004 [IEEE-802.1X] does not support link layer ciphersuites and a result, it does not provide for generation of TSKs, or caching of EAP keying material and parameters. Once EAP authentication completes, it is assumed that EAP keying material and parameters are discarded; on IEEE 802 wired networks there is no subsequent Secure Association Protocol exchange. Perfect Forward Secrecy (PFS) is only possible if the negotiated EAP method supports this.

PPP
: PPP, defined in [RFC1661], does not include support for a Secure Association Protocol; nor does it support caching of EAP keying material or parameters. PPP ciphersuites derive their TSKs directly from the MSK, as described in [RFC2716]. This is NOT RECOMMENDED, since if PPP were to support caching of EAP keying material, this could result in TSK reuse. 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 re-key without EAP re-authentication. Perfect Forward Secrecy (PFS) is only possible if the negotiated EAP method supports this.

IKEv2
: IKEv2, defined in [RFC4306], only uses the MSK for authentication purposes and not key derivation. The EMSK, IV, Peer-Id, Server-Id or Session-Id are not used. As a result, the TSKs derived by IKEv2 are cryptographically independent of the EAP keying material and re-key of IPsec SAs can be handled without requiring EAP re-authentication. Within IKEv2 it is possible to negotiate PFS, regardless of which EAP method is negotiated. IKEv2 as specified in [RFC4306] does not cache EAP keying material or parameters; once IKEv2 authentication completes it is assumed that EAP keying material and parameters are discarded. The Session-Timeout attribute is therefore interpreted as a limit on the VPN session time, rather than an indication of the MSK key lifetime.

IEEE 802.11
IEEE 802.11 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-802.11]. In IEEE 802.11, TSKs are derived from the MSK using a Secure Association Protocol known as 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 re-key without EAP re-authentication. PFS is only possible within IEEE 802.11 if caching is not enabled and the negotiated EAP method supports PFS.

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. IEEE 802.16e support a Secure Association Protocol in which TSKs are chosen by the authenticator without any contribution by the peer. The TSKs are encrypted, authenticated and integrity protected using the MSK and are transported from the authenticator to the peer. TSK re-key is possible without EAP re-authentication. PFS is not possible even if the negotiated EAP method supports it.

AAA Existing implementations and specifications for RADIUS/EAP [RFC3579] or Diameter EAP [RFC4072] do not support caching of 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.

## 2.2. Authenticator and Peer Architecture

This specification does not impose constraints on the architecture of the EAP authenticator or peer. For example, any of the authenticator architectures described in [RFC4118] can be used. As a result, lower layers need to identify EAP peers and authenticators unambiguously, without incorporating implicit assumptions about peer and authenticator architectures.

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 Binding. Many base stations can

share the same authenticator identity. An EAP authenticator or peer:

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

Both the EAP peer and authenticator can have more than one physical or logical port. A peer can simultaneously access the network via multiple authenticators, or via multiple physical or logical ports on a given authenticator. Similarly, an authenticator can offer network access to multiple peers, each via a separate physical or logical port. When a single physical authenticator advertises itself as multiple virtual authenticators, it is possible for a single physical port to belong to multiple virtual authenticators.

An authenticator can be configured to communicate with more than one EAP server, each of which is configured to communicate with a subset of the authenticators. The situation is illustrated in Figure 3.

## 2.3. Authenticator Identification

The EAP method conversation is between the EAP peer and server. The authenticator identity, if considered at all by the EAP method, is treated as an opaque blob for the purpose of Channel Binding (see Section 5.3.3). However, the authenticator identity is important in two other exchanges - the AAA protocol exchange and the Secure Association Protocol conversation.

The AAA conversation is between the EAP authenticator and the backend authentication server. From the point of view of the backend authentication server, 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 backend authentication server detects use of EAP keying material and parameters outside the scope defined by the NAS-Identifier, the keying material MUST be considered compromised.

The Secure Association Protocol conversation is between the peer and the authenticator. For lower layers which support key caching it is particularly important for the EAP peer, authenticator and backend server to have a consistent view of the usage scope of the transported keying material. In order to enable this, it is RECOMMENDED that the Secure Association Protocol explicitly communicate the usage scope of the EAP keying material passed down to the lower layer, rather than implicitly assuming that this is defined

by the authenticator and peer endpoint addresses.

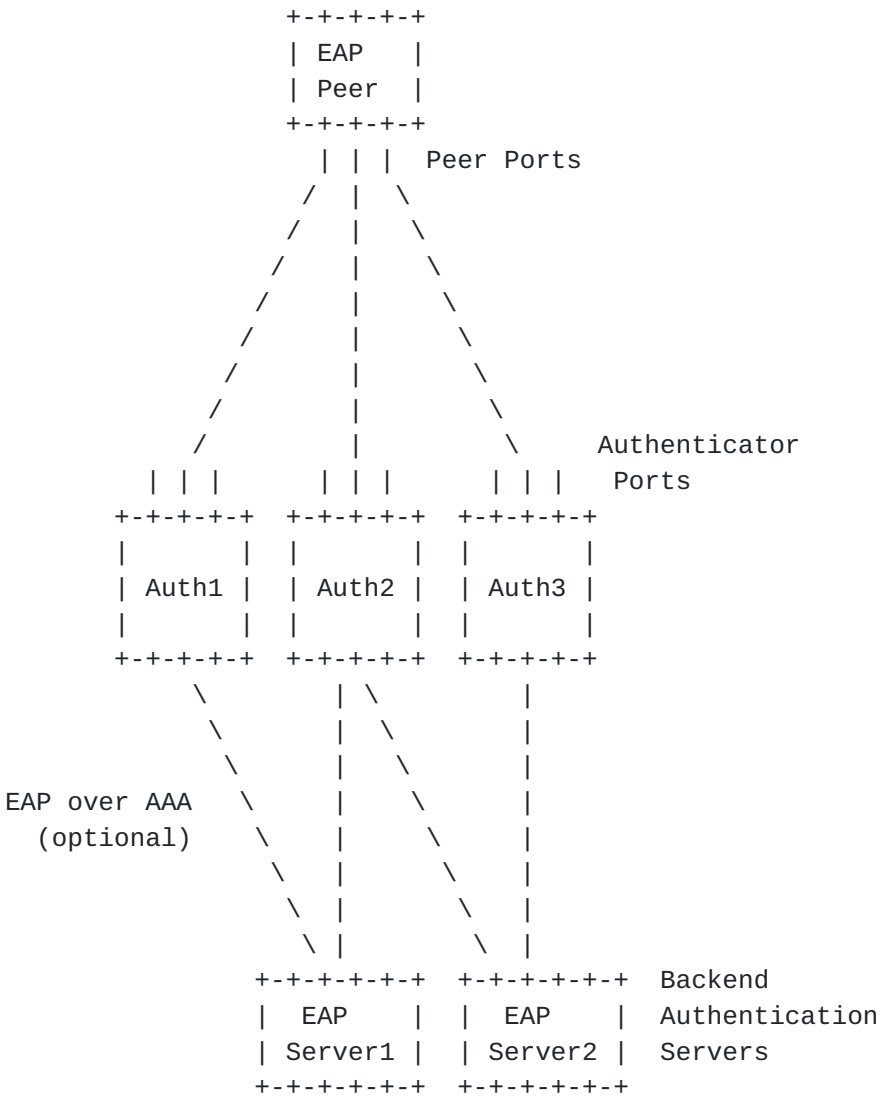


Figure 3: Relationship between EAP peer, authenticator and server

Since an authenticator can have multiple ports, the scope of the authenticator key cache cannot be described by a single endpoint address. Similarly, where a peer can have multiple ports and sharing of EAP keying material and parameters between peer ports of the same link type is allowed, the extent of the peer key cache cannot be communicated by using a single endpoint address. Instead, it is RECOMMENDED that the EAP peer and authenticator consistently identify themselves utilizing explicit identifiers, rather than endpoint addresses or port identifiers.

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 MUST be 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 can have more than one IP address, the NAS-Identifier attribute is RECOMMENDED for explicit identification of the authenticator, both within the AAA protocol exchange and the Secure Association Protocol conversation.

Problems which can arise where the peer and authenticator implicitly identify themselves using endpoint addresses include the following:

(a) It is possible that the peer will not be able to determine which authenticator ports are associated with which authenticators. As a result, the EAP peer will be unable to utilize the authenticator key cache in an efficient way, and will also be unable to determine whether EAP keying material has been shared outside its authorized scope, and therefore needs to be considered compromised.

(b) It is possible that the authenticator will not be able to determine which peer ports are associated with which peers, preventing the peer from communicating with it utilizing multiple peer ports.

(c) It is possible that the peer will not be able to determine which virtual authenticator it is communicating with. For example, multiple virtual authenticators can share a MAC address, but utilize different NAS-Identifiers.

(d) It is possible that the authenticator will not be able to determine which virtual peer it is communicating with. Multiple virtual peers can share a MAC address, but utilize different Peer-Ids.

(e) It is possible that the EAP peer and server will not be able to verify the authenticator identity via Channel Binding.

For example, problems (a), (c) and (e) occur in [IEEE-802.11], which utilizes peer and authenticator MAC addresses within the 4-way handshake. Problems (b) and (d) do not occur since [IEEE-802.11] only allows a virtual peer to utilize a single port.

The following steps enable lower layer identities to be securely verified by all parties:

(f) Specify the lower layer parameters used to identify the authenticator and peer. As noted earlier, endpoint or port identifiers are not recommended for identification of the authenticator or peer when it is possible for them to have multiple

ports.

(g) Communicate the lower layer identities between the peer and authenticator within phase 0. This allows the peer and authenticator to determine the key scope if a key cache is utilized.

(h) Communicate the lower layer authenticator identity between the authenticator and backend server within the NAS-Identifier attribute.

(i) Include the lower layer identities within Channel Bindings (if supported) in phase 1a, ensuring that they are communicated between the EAP peer and server.

(j) Support the integrity-protected exchange of identities within phase 2a.

(k) Utilize the advertised lower layer identities to enable the peer and authenticator to verify that keys are maintained within the advertised scope.

### 2.3.1. Virtual Authenticators

When a single physical authenticator advertises itself as multiple virtual authenticators, if the virtual authenticators do not maintain logically separate key caches, then by authenticating to one virtual authenticator, the peer can gain access to the other virtual authenticators sharing a key cache.

For example, where a physical authenticator implements "Guest" and "Corporate Intranet" virtual authenticators, an attacker acting as a peer could authenticate with the "Guest" virtual authenticator and derive EAP keying material. If the "Guest" and "Corporate Intranet" 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" network.

The following steps can be taken to mitigate this vulnerability:

(a) Authenticators are REQUIRED to cache associated authorizations along with EAP keying material and parameters and to apply authorizations to the peer on each network access, regardless of which virtual authenticator is being accessed. This ensures that an attacker cannot obtain elevated privileges even where the key cache is shared between virtual authenticators, and a peer obtains access to one virtual authenticator utilizing a key cache entry created for use with another virtual authenticator.

(b) It is RECOMMENDED that physical authenticators maintain separate key caches for each virtual authenticator. This ensures that a cache entry created for use with one virtual authenticator cannot be used for access to another virtual authenticator. Since a key cache entry can no longer be shared between virtual authentications, this step provides protection beyond that offered in (a). This is valuable in situations where authorizations are not used to enforce access limitations. For example, where access is limited using a filter installed on a router rather than using authorizations provided to the authenticator, a peer can gain unauthorized access to resources by exploiting a shared key cache entry.

(c) It is RECOMMENDED that each virtual authenticator identify itself consistently to the peer and to the backend authentication server, so as to enable the peer to verify the authenticator identity via Channel Binding (see Section 5.3.3).

(d) It is RECOMMENDED that each virtual authenticator identify itself distinctly, in order to enable the peer and backend server to tell them apart. For example, this can be accomplished by utilizing a distinct value of the NAS-Identifier attribute.

## 2.4. Peer Identification

As described in [RFC3748] Section 7.3, the peer identity provided in the EAP-Response/Identity can be different from the peer identities authenticated by the EAP method. For example, the identity provided in the EAP-Response/Identity can be a privacy identifier as described in "The Network Access Identifier" [RFC4282] Section 2. As noted in [RFC4284], it is also possible to utilize a Network Access Identifier (NAI) for the purposes of source routing; an NAI utilized for source routing is said to be "decorated" as described in [RFC4282] Section 2.7.

When EAP peer provides the Network Access Identity (NAI) within the EAP-Response/Identity, as described in [RFC3579], the authenticator copies the NAI included in the EAP-Response/Identity into the User-Name attribute included within the Access-Request. As the Access-Request is forwarded toward the backend authentication server, AAA proxies remove decoration from the NAI included in the User-Name Attribute; the NAI included within the EAP-Response/Identity encapsulated in the Access-Request remains unchanged. As a result, when the Access-Request arrives at the backend authentication server, the EAP-Response/Identity can differ from the User-Name Attribute (which can have some or all of the decoration removed). In the absence of a Peer-Id, the backend authentication server SHOULD use the contents of the User-Name Attribute, rather than the EAP-

Response/Identity as the peer identity.

It is possible for more than one Peer-Id to be exported by an EAP method. For example, a peer certificate can contain more than one peer identity; in a tunnel method peer identities can be authenticated both within an outer and inner exchange and these identities could be different in type and contents. For example, an outer exchange could provide a Peer-Id in the form of an RDN, whereas an inner exchange could identify the peer via its NAI or MAC address. Where EAP keying material is determined solely from the outer exchange, only the outer Peer-Id(s) are exported; where the EAP keying material is determined from both the inner and outer exchanges, then both the inner and outer Peer-Id(s) are exported by the tunnel method.

## 2.5. Server Identification

It is possible for more than one Server-Id to be exported by an EAP method. For example, a server certificate can contain more than one server identity; in a tunnel method server identities could be authenticated both within an outer and inner exchange and these identities could be different in type and contents. For example, an outer exchange could provide a Server-Id in the form of an IP address, whereas an inner exchange could identify the server via its FQDN or hostname. Where EAP keying material is determined solely from the outer exchange, only the outer Server-Id(s) are exported by the EAP method; where the EAP keying material is determined from both the inner and outer exchanges, then both the inner and outer Server-Id(s) are exported by the EAP method.

As shown in Figure 3, an authenticator can be configured to communicate with multiple EAP servers; the EAP server that an authenticator communicates with can vary according to configuration and network and server availability. While the EAP peer can assume that all EAP servers within a realm have access to the credentials necessary to validate an authentication attempt, it cannot assume that all EAP servers share persistent state.

Authenticators can be configured with different primary or secondary EAP servers, in order to balance the load. Also, the authenticator can dynamically determine the EAP server to which requests will be sent; in event of a communication failure, the authenticator can fail over to another EAP server. For example, in Figure 3, Authenticator2 can be initially configured with EAP server1 as its primary backend authentication server, and EAP server2 as the backup, but if EAP server1 becomes unavailable, EAP server2 can become the primary server.

In general, the EAP peer cannot direct an authentication attempt to a particular EAP server within a realm; this decision is made by AAA clients. Nor can the peer determine which EAP server it will be communicating with, prior to the start of the EAP method conversation. The Server-Id is not included in the EAP-Request/Identity, and since the EAP server may be determined dynamically, it typically is not possible for the authenticator to advertise the Server-Id during the discovery phase. Some EAP methods do not export the Server-Id so that is is possible that the EAP peer will not learn which server it was conversing with after the EAP conversation completes successfully.

As a result, an EAP peer, on connecting to a new authenticator or reconnecting to the same authenticator, can find itself communicating with a different EAP server. Fast reconnect, defined in [RFC3748] Section 7.2, can fail if the EAP server that the peer communicates with is not the same one with which it initially established a security association. For example, an EAP peer attempting an EAP-TLS session resume can find that the new EAP-TLS server will not have access to the TLS Master Key identified by the TLS Session-Id, and therefore the session resumption attempt will fail, requiring completion of a full EAP-TLS exchange.

EAP methods that export the Server-Id MUST authenticate the server. However, not all EAP methods supporting mutual authentication provide a non-null Server-Id; some methods only enable the EAP peer to verify that the EAP server possesses a long-term secret, but do not provide the identity of the EAP server. In this case the EAP peer will know that an authenticator has been authorized by an EAP server, but will not confirm the identity of the EAP server. Where the EAP method does not provide a Server-Id, the peer cannot identify the EAP server with which it generated keying material. This can make it difficult for the EAP peer to identify the location of a key possessed by that EAP server.

As noted in [I-D.simon-emu-rfc2716bis] Section 5.2, EAP methods supporting authentication using server certificates can determine the Server-Id from the subject or subjectAltName fields in the server certificate. Validating the EAP server identity can help the EAP peer to decide whether a specific EAP server is authorized. In some cases, such as where the certificate extensions defined in [RFC4334] are included in the server certificate, it can even be possible for the peer to verify some Channel Binding parameters from the server certificate.

It is possible for problems to arise in situations where the EAP server identifies itself differently to the EAP peer and authenticator. For example, it is possible that the Server-Id

exported by EAP methods will not be identical to the Fully Qualified Domain Name (FQDN) of the backend authentication server. Where certificate-based authentication is used within RADIUS or Diameter, it is possible that the subjectAltName used in the backend authentication server certificate will not be identical to the Server-Id or backend authentication server FQDN. This is not normally an issue in EAP, as the authenticator will be unaware of the identities used between the EAP peer and server. However, this can be an issue for key caching, if the authenticator is expected to locate a backend authentication server corresponding to a Server-Id provided by an EAP peer.

Where the backend authentication server FQDN differs from the subjectAltName in the backend authentication server certificate, it is possible that the AAA client will not be able to determine whether it is talking to the correct backend authentication server. Where the Server-Id and backend server FQDN differ, it is possible that the combination of the key scope (Peer-Id(s), Server- Id(s)) and EAP conversation identifier (Session-Id) will not be sufficient to determine where the key resides. For example, the authenticator can identify backend servers by their IP address (as occurs in RADIUS), or using a Fully Qualified Domain Name (as in Diameter). If the Server-Id does not correspond to the IP address or FQDN of a known backend authentication server, then it may not be possible to locate which backend authentication server possesses the key.

## 3. Security Association Management

EAP as defined in [RFC3748] supports key derivation, but does not provide for the management of lower layer security associations. Missing functionality includes:

(a) Security Association negotiation. EAP does not negotiate lower layer unicast or multicast security associations, including cryptographic algorithms or traffic profiles. EAP methods only negotiate cryptographic algorithms for their own use, not for the underlying lower layers. EAP also does not negotiate the traffic profiles to be protected with the negotiated ciphersuites; in some cases the traffic to be protected can have lower layer source and destination addresses different from the lower layer peer or authenticator addresses.

(b) Re-key. EAP does not support re-key of exported EAP keying material without EAP re-authentication, although EAP methods can support "fast reconnect" as defined in [RFC3748] Section 7.2.1.

(c) Key delete/install semantics. EAP does not synchronize installation or deletion of keying material on the EAP peer and

authenticator.

(d) Lifetime negotiation. EAP does not support lifetime negotiation for exported EAP keying material, and existing EAP methods also do not support key lifetime negotiation.

(e) Guaranteed TSK freshness. Without a post-EAP handshake, TSKs can be reused if EAP keying material is cached.

These deficiencies are typically addressed via a post-EAP handshake known as the Secure Association Protocol.

### 3.1. Secure Association Protocol

Since neither EAP nor EAP methods provide for establishment of lower layer security associations, it is RECOMMENDED that these facilities be provided within the Secure Association Protocol, including:

(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.

(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. Authenticator identity verification is discussed in Section 2.3.

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

(d) Key naming and selection. Where key caching is supported, it is 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 Section 1.4.1 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 keying material, the EAP peer lower layer can initiate a new session using keying material that was derived in a previous session. Were the TSKs to be derived solely 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 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 transported 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 keying material in order to generate fresh unicast (phase 2a) and possibly multicast (phase 2b) session keys. By not using exported 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 re-key of EAP keying material without re-authentication and existing EAP methods do not support key lifetime negotiation. 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. For example, a difference between IKEv1 [RFC2409] and IKEv2 [RFC4306] 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 state 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 it may not be possible for the peer to determine before attempting to use a key whether it exists within the authenticator cache. It is therefore RECOMMENDED for the EAP lower layer to provide a mechanism for key state resynchronization, either via the SAP or using a lower layer indication (see [RFC3748] Section 3.4). Where the peer and authenticator do not jointly possess a key with which to protect the resynchronization exchange, secure resynchronization is not possible and alternatives (such as a initiation of EAP re-authentication after expiration of a timer) is needed to ensure timely resynchronization.

(h) Key scope synchronization. To support 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) Traffic profile negotiation. The traffic to be protected by a lower layer security association will not necessarily have the same lower layer source or destination address as the EAP peer and authenticator, and it is possible for the peer and authenticator to negotiate multiple security associations, each with a different traffic profile. Where this is the case, the profile of protected traffic SHOULD be explicitly negotiated. For example, in IKEv2 it is possible for an Initiator and Responder to utilize EAP for authentication, then negotiate a Tunnel Mode Security Association (SA) which permits passing of traffic originating from hosts other than the Initiator and Responder. Similarly, in IEEE 802.16e a Subscriber Station (SS) can forward traffic to the Base Station (BS) which originates from the Local Area Network (LAN) to which it is attached. To enable this, Security Associations within IEEE 802.16e are identified by the Connection Identifier (CID), not by the EAP peer and authenticator MAC addresses. In both IKEv2 and IEEE 802.16e, multiple security associations can exist between the EAP peer and authenticator, each with their own traffic profile and quality of service parameters.

(j) 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.

(k) 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. See [RFC3748] Section 2.4 for more discussion.

## 3.2. Key Scope

Absent explicit specification within the lower layer, after the completion of phase 1b, transported keying material and parameters are bound to the EAP peer and authenticator, but are not bound to a specific peer or authenticator port.

While EAP keying material passed down to the lower layer is not intrinsically bound to particular authenticator and peer ports, TSKs 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).

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 keying material. For example, in IEEE 802.11, the backend authentication server can provide the authenticator with a list of authorized Called or Calling-Station-Ids and/or SSIDs for which 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.

## 3.3. Parent-Child Relationships

When an EAP re-authentication takes place, new EAP keying material is exported by the EAP method. In EAP lower layers where EAP re-authentication eventually results in TSK replacement, the maximum lifetime of derived keying material (including TSKs) can be less than or equal to that of EAP keying material (MSK/EMSK), but it cannot be greater.

Where TSKs are derived from or are wrapped by exported EAP keying material, compromise of that exported EAP keying material implies compromise of TSKs. Therefore if EAP keying material is considered stale, not only SHOULD EAP re-authentication be initiated, but also replacement of child keys, including TSKs.

Where EAP keying material is used only for entity authentication but not for TSK derivation (as in IKEv2), compromise of exported EAP keying material does not imply compromise of the TSKs. Nevertheless, the compromise of EAP keying material could enable an attacker to impersonate an authenticator, so that EAP re-authentication and TSK re-key are RECOMMENDED.

With respect to IKEv2, "IKEv2 Clarifications and Implementation Guidelines" [RFC4718] Section 5.2 states:

> Rekeying the IKE-SA and reuathentication are different concepts in IKEv2. Rekeying the IKE_SA establishes new keys for the IKE_SA and resets the Message ID counters, but it does not authenticate the parties again (no AUTH or EAP payloads are involved)... This means that reauthentication also establishes new keys for the IKE_SA and CHILD_SAs. Therefore while rekeying can be performed more often than reauthentication, the situation where "authentication lifetime" is shorter than "key lifetime" does not make sense.

Child keys that are used frequently (such as TSKs which are used for traffic protection) can expire sooner than the exported EAP keying material they are dependent on, so that it is advantageous to support re-key of child keys prior to EAP re-authentication. Note that deletion of the MSK/EMSK does not necessarily imply deletion of TSKs or child keys.

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

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

## 3.4. 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 can re-key TEKs during an EAP 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 EAP keying material (TEKs) which can persist for multiple EAP conversations when fast reconnect is used [RFC3748]. For example, EAP methods based on TLS (such as EAP-TLS [I-D.simon-emu-rfc2716bis]) derive and cache the TLS Master Secret, typically for substantial time periods. The lifetime of other local EAP 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 smart-card holding the private key for EAP-TLS can 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.

## 3.5. Exported and Calculated Key Lifetimes

The following mechanisms are available for communicating the lifetime of keying material between the EAP peer, server and authenticator:

AAA protocols (backend server and authenticator)
Lower layer mechanisms (authenticator and peer)
EAP method-specific negotiation (peer and server)

Where the EAP method does not support the negotiation of the lifetime of exported EAP keying material, and a key lifetime negotiation mechanism is not provided by the lower layer, it is possible that there will not be a way for the peer to learn the lifetime of keying material. This can leave the peer uncertain how long the authenticator will maintain keying material within the key cache. In

this case the lifetime of keying material can be managed as a system parameter on the peer and authenticator; a default lifetime of 8 hours is RECOMMENDED.

### 3.5.1. AAA Protocols

AAA protocols such as RADIUS [RFC2865] and Diameter [RFC4072] can be used to communicate the maximum key lifetime from the backend authentication server to the authenticator.

The Session-Timeout attribute is defined for RADIUS in [RFC2865] and for Diameter in [RFC4005]. Where EAP is used for authentication, [RFC3580] Section 3.17 indicates that a Session-Timeout attribute sent in an Access-Accept along with a Termination-Action value of RADIUS-Request specifies the maximum number of seconds of service provided prior to EAP re-authentication.

However, there is also a need to be able to specify the maximum lifetime of cached keying material. Where EAP pre-authentication is supported, cached keying material can be pre-established on the authenticator prior to session start, and will remain there until expiration. EAP lower layers supporting caching of keying material MAY also persist that material after the end of a session, enabling the peer to subsequently resume communication utilizing the cached keying material. In these situations it can be desirable for the backend authentication server to specify the maximum lifetime of cached keying material.

To accomplish this, [IEEE-802.11] overloads the Session-Timeout attribute, assuming that it represents the maximum time after which transported keying material will expire on the authenticator, regardless of whether transported keying material is cached.

An IEEE 802.11 authenticator receiving transported keying material is expected to initialize a timer to the Session-Timeout value, and once the timer expires, the transported keying material expires. Whether this results in session termination or EAP re-authentication is controlled by the value of the Termination-Action attribute. Where EAP re-authentication occurs the transported keying material is replaced, and with it, new calculated keys are put in place. Where session termination occurs, transported and derived keying material is deleted.

Overloading the Session-Timeout attribute is problematic in situations where it is necessary to control the maximum session time and key lifetime independently. For example, it might be desirable to limit the lifetime of cached keying material to 5 minutes while permitting a user once authenticated to remain connected for up to an

hour without re-authenticating. As a result, in the future additional attributes can be specified to control the lifetime of cached keys; these attributes MAY modify the meaning of the Session-Timeout attribute in specific circumstances.

Since the TSK lifetime is often determined by authenticator resources, and the backend authentication server has no insight into the TSK derivation process, 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.

### 3.5.2. Lower Layer Mechanisms

Lower layer mechanisms can be used to enable the lifetime of keying material to be negotiated between the peer and authenticator. This can be accomplished either using the Secure Association Protocol or within the lower layer transport.

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 re-key. Where the TSK is taken directly 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.

### 3.5.3. EAP Method-Specific Negotiation

As noted in [RFC3748] Section 7.10:

> In order to provide keying material for use in a subsequently negotiated ciphersuite, an EAP method supporting key derivation MUST export a Master Session Key (MSK) of at least 64 octets, and an Extended Master Session Key (EMSK) of at least 64 octets. EAP Methods deriving keys MUST provide for mutual authentication between the EAP peer and the EAP Server.

However, EAP does not itself support the negotiation of lifetimes for exported EAP keying material such as the MSK, EMSK and IV.

While EAP itself does not support lifetime negotiation, it would be possible to specify methods that do. However, systems that rely on key lifetime negotiation within EAP methods would only function with these methods. Also, there is no guarantee that the key lifetime negotiated within the EAP method would be compatible with backend authentication server policy. In the interest of method independence and compatibility with backend server implementations, management of the lifetime of keying material SHOULD NOT be provided within EAP

methods.

## 3.6. Key Cache Synchronization

Key lifetime negotiation alone cannot guarantee key cache synchronization. Even where a lower layer exchange is run immediately after EAP in order to determine the lifetime of keying material, it is still possible for the authenticator to purge all or part of the key cache prematurely (e.g. due to reboot or need to reclaim memory).

The lower layer can utilize the Discovery phase 0 to improve key cache synchronization. For example, if the authenticator manages the key cache by deleting the oldest key first, the relative creation time of the last key to be deleted could be advertised within the Discovery phase, enabling the peer to determine whether keying material had been prematurely expired from the authenticator key cache.

## 3.7. Key Strength

As noted in Section 2.1, EAP lower layers determine TSKs in different ways. Where exported EAP keying material is utilized in the derivation, encryption or authentication of TSKs, it is possible for EAP key generation to represent the weakest link.

In order to ensure that methods produce EAP keying material of an appropriate symmetric key strength, it is RECOMMENDED that EAP methods utilizing public key cryptography choose a public key that has a cryptographic strength providing the required level of attack resistance. This is typically provided by configuring EAP methods, since there is no coordination between the lower layer and EAP method with respect to minimum required symmetric key strength.

BCP 86 [RFC3766] Section 5 offers advice on the required RSA or DH module and DSA subgroup size in bits, for a given level of attack resistance in bits. The National Institute for Standards and Technology (NIST) also offers advice on appropriate key sizes in [SP800-57].

## 3.8. Key Wrap

The key wrap specified in [RFC2548], which is based on an MD5-based stream cipher, has known problems, as described in [RFC3579] Section 4.3. RADIUS uses the shared secret for multiple purposes, including per-packet authentication and 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 concerns exist relating to the security of this hash [MD5Collision].

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, [RFC3579] Section 4.2 recommends running RADIUS over IPsec. The same approach is taken in Diameter EAP [RFC4072], which defines clear-text key attributes, to be protected by IPsec or TLS.

## 4. Handoff Vulnerabilities

A handoff occurs when an EAP peer moves to a new authenticator. Several mechanisms have been proposed for reducing handoff latency within networks utilizing EAP. These include:

EAP pre-authentication
: In EAP pre-authentication, an EAP peer pre-establishes EAP keying material with an authenticator prior to arrival. EAP pre-authentication only affects the timing of EAP authentication, but does not shorten or eliminate EAP (phase 1a) or AAA (phase 1b) exchanges; Discovery (phase 0) and Secure Association Protocol (phase 2) exchanges occur as described in Section 1.3. As a result, the primary benefit is to enable EAP authentication to be removed from the handoff critical path, thereby reducing latency. Use of EAP pre-authentication within IEEE 802.11 is described in [IEEE-802.11] and [8021XPreAuth].

Proactive key distribution
: In proactive key distribution, keying material and authorizations are transported from the backend authentication server to a candidate authenticator in advance of a handoff. As a result, EAP (phase 1a) is not needed, but the Discovery (phase 0), and Secure Association Protocol exchanges (phase 2) are still necessary. Within the AAA exchange (phase 1b), authorization and key distribution functions are typically supported, but not authentication. Proactive key distribution is described in [MishraPro], [IEEE-03-084] and [I-D.irtf-aaaarch-handoff].

Key caching
: Caching of EAP keying material enables an EAP peer to re-attach to an authenticator without requiring EAP (phase 1a) or AAA (phase 1b) exchanges. However, Discovery (phase 0) and Secure Association Protocol (phase 2) exchanges are still needed. Use of key caching within IEEE 802.11 is described in [IEEE-802.11].

Context transfer

In context transfer schemes, keying material and authorizations are transferred between a previous authenticator and a new authenticator. This can occur in response to a handoff request by the EAP peer, or in advance, as in proactive key distribution. As a result, EAP (phase 1a) is eliminated, but not the Discovery (phase 0) or Secure Association Protocol exchanges (phase 2). If a secure channel can be established between the new and previous authenticator without assistance from the backend authentication server, then the AAA exchange (phase 1b) can be eliminated; otherwise, it is still needed, although it can be shortened. Context transfer protocols are described in [IEEE-802.11F] (now deprecated) and "Context Transfer Protocol (CXTP)" [RFC4067]. "Fast Authentication Methods for Handovers between IEEE 802.11 Wireless LANs" [Bargh] analyzes fast handoff techniques, including context transfer mechanisms.

Token distribution

In token distribution schemes the EAP peer is provided with a credential, subsequently enabling it to authenticate with one or more additional authenticators. During the subsequent authentications, EAP (phase 1a) is eliminated or shortened; the Discovery (phase 0) and Secure Association Protocol exchanges (phase 2) still occur, although the latter can be shortened. If the token includes authorizations and can be validated by an authenticator without assistance from the backend authentication server, then the AAA exchange (phase 1b) can be eliminated; otherwise it is still needed, although it can be shortened. Token-based schemes, initially proposed in early drafts of IEEE 802.11i [IEEE-802.11i], are described in [Token], [Tokenk] and [I-D.friedman-ike-short-term-certs].

The sections that follow discuss the security vulnerabilities introduced by the above schemes.

## 4.1. EAP Pre-authentication

EAP pre-authentication differs from a normal EAP conversation primarily with respect to the lower layer encapsulation. For example, in [IEEE-802.11], EAP pre-authentication frames utilize a distinct Ethertype, but otherwise conforms to the encapsulation described in [IEEE-802.1X]. As a result, an EAP pre-authentication conversation differs little from the model described in Section 1.3, other than the introduction of a delay between phase 1 and phase 2.

EAP pre-authentication relies on lower layer mechanisms for discovery of candidate authenticators. Where discovery can provide information on candidate authenticators outside the immediate listening range,

and the peer can determine whether it already possesses valid EAP keying material with candidate authenticators, the peer can avoid unnecessary EAP pre-authentications and can establish EAP keying material well in advance, regardless of the coverage overlap between authenticators. However, if the peer can only discover candidate authenticators within listening range and cannot determine whether it can reuse existing EAP keying material, then it is possible that the peer will not be able to complete EAP pre-authentication prior to connectivity loss or that it can pre-authenticate multiple times with the same authenticator, increasing backend authentication server load.

Since a peer can complete EAP pre-authentication with an authenticator without eventually attaching to it, it is possible that phase 2 will not occur. In this case an Accounting-Request signifying the start of service will not be sent, or will only be sent with a substantial delay after the completion of authentication.

As noted in "IEEE 802.1X RADIUS Usage Guidelines" [RFC3580], the AAA exchange resulting from EAP pre-authentication differs little from an ordinary exchange described in "RADIUS Support for EAP" [RFC3579]. For example, since in IEEE 802.11 [IEEE-802.11] an Association exchange does not occur prior to EAP pre-authentication, the SSID is not known by the authenticator at authentication time, so that an Access-Request cannot include the SSID within the Called-Station-Id attribute as described in [RFC3580] Section 3.20.

Since only the absence of an SSID in the Called-Station-Id attribute distinguishes an EAP pre-authentication attempt, if the authenticator does not always include the SSID for a normal EAP authentication attempt, it is possible that the backend authentication server will not be able to determine whether a session constitutes an EAP pre-authentication attempt, potentially resulting in authorization or accounting problems. Where the number of simultaneous sessions is limited, the backend authentication server can refuse to authorize a valid EAP pre-authentication attempt or can enable the peer to engage in more simultaneous sessions than they are authorized for. Where EAP pre-authentication occurs with an authenticator which the peer never attaches to, it is possible that the backend accounting server will not be able to determine whether the absence of an Accounting-Request was due to packet loss or a session that never started.

In order to enable pre-authentication requests to be handled more reliably, it is RECOMMENDED that AAA protocols explicitly identify EAP pre-authentication. In order to suppress unnecessary EAP pre-authentication exchanges, it is RECOMMENDED that authenticators unambiguously identify themselves as described in Section 2.3.

### 4.2. Proactive Key Distribution

In proactive key distribution schemes, the backend authentication server transports keying material and authorizations to an authenticator in advance of the arrival of the peer. The authenticators selected to receive the transported key material are selected based on past patterns of peer movement between authenticators known as the "neighbor graph". In order to reduce handoff latency, proactive key distribution schemes typically only demonstrate proof of possession of transported keying material between the EAP peer and authenticator. During a handoff, the backend authentication server is not provided with proof that the peer successfully authenticated to an authenticator; instead, the authenticator generates a stream of accounting messages without a corresponding set of authentication exchanges. As described in [MishraPro], knowledge of the neighbor graph can be established via static configuration or analysis of authentication exchanges. In order to prevent corruption of the neighbor graph, new neighbor graph entries can only be created as the result of a successful EAP exchange, and accounting packets with no corresponding authentication exchange need to be verified to correspond to neighbor graph entries (e.g. corresponding to handoffs between neighbors).

In order to prevent compromise of one authenticator from resulting in compromise of other authenticators, cryptographic separation needs to be maintained between the keying material transported to each authenticator. However, even where cryptographic separation is maintained, an attacker compromising an authenticator can still disrupt the operation of other authenticators. As noted in [RFC3579] Section 4.3.7, in the absence of spoofing detection within the AAA infrastructure, it is possible for EAP authenticators to impersonate each other. By forging NAS identification attributes within authentication messages, an attacker compromising one authenticator could corrupt the neighbor graph, tricking the backend authentication server into transporting keying material to arbitrary authenticators. While this would not enable recovery of EAP keying material without breaking fundamental cryptographic assumptions, it could enable subsequent fraudulent accounting messages, or allow an attacker to disrupt service by increasing load on the backend authentication server or thrashing the authenticator key cache.

Since proactive key distribution requires the distribution of derived keying material to candidate authenticators, the effectiveness of this scheme depends on the ability of backend authentication server to anticipate the movement of the EAP peer. Since proactive key distribution relies on backend authentication server knowledge of the neighbor graph it is most applicable to intra-domain handoff scenarios. However, in inter-domain handoff where there can be many

authenticators, peers can frequently connect to authenticators that have not been previously encountered, making it difficult for the backend authentication server to derive a complete neighbor graph.

Since proactive key distribution schemes typically require introduction of server-initiated messages as described in [RFC3576bis] and [I-D.irtf-aaaarch-handoff], security issues described in [RFC3576bis] Section 6 are applicable, including authorization (Section 6.1) and replay detection (Section 6.3) problems.

### 4.3. AAA Bypass

Fast handoff techniques which enable elimination of the AAA exchange (phase 1b) differ fundamentally from typical network access scenarios (dial-up, wired LAN, etc.) which include user authentication as well as authorization for the offered service. Where the AAA exchange (phase 1b) is omitted, authorizations and keying material are not provided by the backend authentication server, and as a result they need to be supplied by other means. This section describes some of the implications.

#### 4.3.1. Key Transport

Where transported keying material is not supplied by the backend authentication server, it needs to be provided by another party authorized to access that keying material. As noted in Section 1.5, only the EAP peer, authenticator and server are authorized to possess transported keying material. Since EAP peers do not trust each other, if the backend authentication server does not supply transported keying material to a new authenticator, it can only be provided by a previous authenticator.

As noted in Section 1.5, the goal of the EAP conversation is to derive session keys known only to the peer and the authenticator. If keying material is replicated between a previous authenticator and a new authenticator, then the previous authenticator can possess session keys used between the peer and new authenticator. Also, the new authenticator can possess session keys used between the peer and the previous authenticator.

If a one-way function is used to derive the keying material to be transported to the new authenticator, then the new authenticator cannot possess previous session keys without breaking a fundamental cryptographic assumption.

### 4.3.2. Authorization

As a part of the authentication process, the backend authentication server determines the user's authorization profile and transmits the authorizations to the authenticator along with the transported keying material. Typically, the profile is determined based on the user identity, but a certificate presented by the user can also provide authorization information.

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

(a) Is this a legitimate user of this network?

(b) Is the user allowed to access this network?

(c) Is the user permitted to access this network on this day and at this time?

(d) Is the user within the concurrent session limit?

(e) Are there any fraud, credit limit, or other concerns that could lead to access denial?

(f) If access is to be granted, what are the service parameters (mandatory tunneling, bandwidth, filters, and so on) to be provisioned for the user?

While the authorization decision is in principle simple, the distributed decision making process can add complexity. Where brokers 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 example, a broker can deny 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 concurrent session limits). In addition to the Accept/Reject decisions made by AAA entities, service 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 of day and the capabilities of the authenticator?

### 4.3.3. Correctness

When the AAA exchange (phase 1b) is bypassed, several challenges arise in ensuring correct authorization:

Theft of service
: Bypassing the AAA exchange (phase 1b) SHOULD NOT enable a user to extend their network access or gain access to services they are not entitled to.

Consideration of network-wide state
: Handoff techniques SHOULD NOT render the backend authentication server incapable of keeping track of network-wide state. For example, a backend authentication server can need to keep track of simultaneous user sessions.

Elevation of privilege
: 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. In this situation, bypassing the AAA exchange could enable unauthorized access unless the restrictions are explicitly encoded within the authorizations provided by the backend authentication server.

A handoff mechanism that provides proper authorization is said to be "correct". One condition for correctness is as follows:

> For a handoff to be "correct" it MUST establish on the new authenticator the same authorizations as would have been created had the new authenticator 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" authorizations, it is preferable for it to fail. Where the supported services differ between authenticators, a handoff that bypasses the backend authentication server is likely to fail. [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.

This behavior 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 authenticator is provided with RADIUS authorizations for a known but unavailable service, then it MUST process these authorizations 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 authenticator is provided with authorizations including unknown attributes, then these attributes MAY be ignored. 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]. Although it can 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 an authenticator provide a service that it does not implement. This implies that if the new authenticator were to complete a AAA conversation, it would be likely to receive different service instructions. Failure of the handoff is the desired result since it will cause the new authenticator to go back to the backend server in order to receive the appropriate service definition.

Handoff mechanisms which bypass the backend authentication server are most likely to be successful when employed in a homogeneous deployment within a single administrative domain. In a heterogeneous deployment, the backend authentication server can 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. Where a backend authentication server would send different authorizations to the new authenticator than were sent to a previous authenticator, transferring authorizations between the previous authenticator and the new authenticator will result in incorrect authorization.

Virtual LAN (VLAN) support is defined in [IEEE-802.1Q]; RADIUS support for dynamic VLANs is described in [RFC3580] and [RFC4675]. If some authenticators support dynamic VLANs while others do not, then attributes present in the Access-Request (such as the NAS-Port-Type, NAS-IP-Address, NAS-IPv6-Address and NAS-Identifier) could be

examined by the backend authentication server to determine when VLAN attributes will be returned, and if so, which ones. However, if the backend authenticator is bypassed, then a handoff occurring between authenticators supporting different VLAN capabilities could result in a user obtaining access to an unauthorized VLAN (e.g. a user with access to a guest VLAN being given unrestricted access to the network).

Similarly, it is preferable for a handoff between an authenticator providing confidentiality and another that does not to fail, since if the handoff were successful, the user would be moved from a secure to an insecure channel without permission from the backend authentication server.

## 5. Security Considerations

The EAP threat model is described in [RFC3748] Section 7.1. The security properties of EAP methods (known as "security claims") are described in [RFC3748] Section 7.2.1. 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. In developing the threat model, it is assumed that:

All traffic is visible to the attacker.
The attacker can alter, forge or replay messages.
The attacker can reroute messages to another principal.
The attacker can be a principal or an outsider.
The attacker can compromise any key that is sufficiently old.

Threats arising from these assumptions include:

(a) An attacker can compromise or steal an EAP peer or authenticator, in an attempt to gain access to other EAP peers or authenticators or to obtain long-term secrets.

(b) An attacker can attempt a downgrade attack in order to exploit known weaknesses in an authentication method or cryptographic algorithm.

(c) An attacker can try to modify or spoof packets, including Discovery or Secure Association Protocol frames, EAP or AAA packets.

(d) An attacker can 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.

(e) An attacker can alter, forge or replay packets.

(f) An attacker can cause an EAP peer, authenticator or server to reuse a stale key. Use of stale keys can also occur unintentionally. For example, a poorly implemented backend authentication server can provide stale keying material to an authenticator, or a poorly implemented authenticator can reuse nonces.

(g) An authenticated attacker can attempt to obtain elevated privilege in order to access information that it does not have rights to.

(h) An attacker can attempt a man-in-the-middle attack in order to gain access to the network.

(i) An attacker can compromise an EAP authenticator in an effort to commit fraud. For example, a compromised authenticator can 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.

(j) An attacker can launch a denial of service attack against the EAP peer, authenticator or backend authentication server.

In order to address these threats, [RFC4962] Section 3 describes required and recommended security properties. The sections that follow analyze the compliance of EAP methods, AAA protocols and Secure Association Protocols with those guidelines.

## 5.1. Peer and Authenticator Compromise

Requirement: In the event that an authenticator is compromised or stolen, an attacker can gain access to the network through that authenticator, or can obtain the credentials needed for the authenticator/AAA client to communicate with one or more backend authentication servers. Similarly, if a peer is compromised or stolen, an attacker can obtain credentials needed to communicate with one or more authenticators. Mandatory requirement from [RFC4962] Section 3:

> Prevent the Domino effect
>
> Compromise of a single peer MUST NOT compromise keying material

held by any other peer within the system, including session keys and long-term keys. Likewise, compromise of a single authenticator MUST NOT compromise keying material held by any other authenticator within the system. In the context of a key hierarchy, this means that the compromise of one node in the key hierarchy must not disclose the information necessary to compromise other branches in the key hierarchy. Obviously, the compromise of the root of the key hierarchy will compromise all of the keys; however, a compromise in one branch MUST NOT result in the compromise of other branches. There are many implications of this requirement; however, two implications deserve highlighting. First, the scope of the keying material must be defined and understood by all parties that communicate with a party that holds that keying material. Second, a party that holds keying material in a key hierarchy must not share that keying material with parties that are associated with other branches in the key hierarchy.

Group keys are an obvious exception. Since all members of the group have a copy of the same key, compromise of any one of the group members will result in the disclosure of the group key.

Some of the implications of the requirement are as follows:

Key Sharing
: In order to be able to determine whether keying material has been shared, it is necessary for the identity of the EAP authenticator (NAS-Identifier) to be defined and understood by all parties that communicate with it. EAP lower layer specifications such as [IEEE-802.11], [IEEE-802.16e], [IEEE-802.1X], IKEv2 [RFC4306] and PPP [RFC1661] do not involve key sharing.

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.

Compromise of Long-Term Credentials
: An attacker obtaining keying material (such as TSKs, TEKs or 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. The mandatory requirements of [RFC4017] Section 2.2 include generation of EAP keying material, capability to generate EAP keying material with 128-bits of effective strength, resistance to dictionary attacks, shared state

equivalence and protection against man-in-the-middle attacks.

## 5.2. Cryptographic Negotiation

Mandatory requirements from [RFC4962] Section 3:

> Cryptographic algorithm independent
>
> The AAA key management protocol MUST be cryptographic algorithm independent. However, an EAP method MAY depend on a specific cryptographic algorithm. The ability to negotiate the use of a particular cryptographic algorithm provides resilience against compromise of a particular cryptographic algorithm. Algorithm independence is also REQUIRED with a Secure Association Protocol if one is defined. This is usually accomplished by including an algorithm identifier and parameters in the protocol, and by specifying the algorithm requirements in the protocol specification. While highly desirable, the ability to negotiate key derivation functions (KDFs) is not required. For interoperability, at least one suite of mandatory-to-implement algorithms MUST be selected. Note that without protection by IPsec as described in [RFC3579] Section 4.2, RADIUS [RFC2865] does not meet this requirement, since the integrity protection algorithm cannot be negotiated.
>
> This requirement does not mean that a protocol must support both public-key and symmetric-key cryptographic algorithms. It means that the protocol needs to be structured in such a way that multiple public-key algorithms can be used whenever a public-key algorithm is employed. Likewise, it means that the protocol needs to be structured in such a way that multiple symmetric-key algorithms can be used whenever a symmetric-key algorithm is employed.
>
> Confirm ciphersuite selection
>
> The selection of the "best" ciphersuite SHOULD be securely confirmed. The mechanism SHOULD detect attempted roll-back attacks.

EAP methods satisfying [RFC4017] Section 2.2 mandatory requirements and AAA protocols utilizing transmission layer security are capable of addressing downgrade attacks. [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 method conversation, as well as to integrity protect the ciphersuite negotiation. [RFC4017] Section 2.2 requires EAP methods satisfying this security claim. Since TLS v1.2

[I-D.ietf-tls-rfc4346-bis] supports negotiation of Key Distribution Functions (KDFs), EAP methods based on TLS will, if properly designed, inherit this capability. However, negotiation of KDFs is not required by RFC 4962 [RFC4962], and EAP methods not based on TLS typically do not support KDF negotiation.

Diameter [RFC3588] provides support for cryptographic algorithm negotiation via use of IPsec [RFC4301] or TLS [RFC4346]. Since IKEv2 [RFC4306] does not support KDF negotiation, support for KDF negotiation is only available when Diameter runs over TLS v1.2. RADIUS [RFC3579] does not support cryptographic algorithm negotiation and relies on MD5 for integrity protection, authentication and confidentiality. Given the known weaknesses in MD5 [MD5Collision] this is undesirable, and can be addressed via use of RADIUS over IPsec, as described in [RFC3579] Section 4.2.

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 well as KDF negotiation.

As described in [RFC1968], PPP ECP does not support secure ciphersuite negotiation. While [IEEE 802.16e] and [IEEE-802.11] support ciphersuite negotiation for protection of data, they do not support negotiation of the cryptographic primitives used within the Secure Association Protocol, such as message integrity checks (MICs) and KDFs.

### 5.3. Confidentiality and Authentication

Mandatory requirements from [RFC4962] Section 3:

> Authenticate all parties
>
> Each party in the AAA key management protocol MUST be authenticated to the other parties with whom they communicate. Authentication mechanisms MUST maintain the confidentiality of any secret values used in the authentication process. When a secure association protocol is used to establish session keys, the parties involved in the secure association protocol MUST identify themselves using identities that are meaningful in the lower-layer protocol environment that will employ the session keys. In this situation, the authenticator and peer may be known by different identifiers in the AAA protocol environment and the lower-layer protocol environment, making authorization decisions difficult without a clear key scope. If the lower-layer identifier of the

peer will be used to make authorization decisions, then the pair of identifiers associated with the peer MUST be authorized by the authenticator and/or the AAA server.

AAA protocols, such as RADIUS [RFC2865] and Diameter [RFC3588], 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.

When multiple base stations and a "controller" (such as a WLAN switch) comprise a single EAP authenticator, the "base station identity" is not relevant; the EAP method conversation takes place between the EAP peer and the EAP server. Also, many base stations can share the same authenticator identity. The authenticator identity is important in the AAA protocol exchange and the secure association protocol conversation.

Authentication mechanisms MUST NOT employ plaintext passwords. Passwords may be used provided that they are not sent to another party without confidentiality protection.

Keying material confidentiality and integrity

While preserving algorithm independence, confidentiality and integrity of all keying material MUST be maintained.

Conformance to these requirements are analyzed in the sections that follow.

### 5.3.1. Spoofing

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] Section 2.2 requires 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. IKEv2 [RFC4306] supports per-frame integrity protection and authentication, as does the Secure Association

Protocol defined in [IEEE-802.16e]. [IEEE-802.11] supports per-frame integrity protection and authentication on all messages within the 4-way handshake except the first message. An attack leveraging this omission is described in [Analysis].

### 5.3.2. Impersonation

Both RADIUS [RFC2865] and Diameter [RFC3588] implementations are potentially vulnerable to a rogue authenticator impersonating another authenticator. While both protocols support mutual authentication between the AAA client/authenticator and the backend authentication server, the security mechanisms vary.

In RADIUS, the shared secret used for authentication is determined by the source address of the RADIUS packet. However, when RADIUS Access-Requests are forwarded by a proxy, the NAS-IP-Address, NAS-Identifier or NAS-IPv6-Address attributes received by the RADIUS server will not correspond to the source address. As noted in [RFC3579] Section 4.3.7, if the first-hop proxy does not check the NAS identification attributes against the source address in the Access-Request packet, it is possible for a rogue authenticator to forge NAS-IP-Address [RFC2865], NAS-IPv6-Address [RFC3162] or NAS-Identifier [RFC2865] attributes in order to impersonate another authenticator; attributes such as the Called-Station-Id [RFC2865] and Calling-Station-Id [RFC2865] can be forged as well. Among other things, this can result in messages (and transported keying material) being sent to the wrong authenticator.

While [RFC3588] requires use of the Route-Record AVP, this utilizes Fully Qualified Domain Names (FQDNs), so that impersonation detection requires DNS A, AAAA and PTR Resource Records (RRs) to be properly configured. As a result, Diameter is as vulnerable to this attack as RADIUS, if not more so. [RFC3579] Section 4.3.7 recommends mechanisms for impersonation detection; to prevent access to keying material by proxies without a "need to know", it is necessary to allow the backend authentication server to communicate with the authenticator directly, such as via the redirect functionality supported in [RFC3588].

### 5.3.3. 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 can 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 EAP conversation. Within the Secure Association Protocol, the EAP peer and authenticator only demonstrate mutual possession of the transported keying material; they do not mutually authenticate. This creates a potential security vulnerability, described in [RFC3748] Section 7.15.

As described in [RFC3579] Section 4.3.7, it is possible for a first-hop AAA proxy to detect a AAA client attempting to impersonate another authenticator. 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. Also, a pass-through authenticator acting as a AAA client can provide an incorrect peer Calling-Station-Id [RFC2865][RFC3580] to the backend authentication 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. Typically the EAP method imports channel binding parameters from the lower layer on the peer, and transmits them securely to the EAP server, which exports them to the lower layer or AAA layer. However, transport can occur from EAP server to peer, or can be bi-directional. On the side of the exchange (peer or server) where Channel Binding is verified, the lower layer or AAA layer passes the result of the verification (TRUE or FALSE) up to the EAP method. While the verification can be done either by the peer or the server, typically only the server has the knowledge to determine the correctness of the values, as opposed to merely verifying their equality. For further discussion, see [I-D.arkko-eap-service-identity-auth].

It is also possible to perform Channel Binding without transporting data over EAP, as described in [I-D.ohba-eap-channel-binding]. In this approach the EAP method includes channel binding parameters in the calculation of exported EAP keying material, making it impossible for the peer and authenticator to complete the Secure Association

Protocol if there is a mismatch in the channel binding parameters. However, this approach can only be applied where methods generating EAP keying material are used along with lower layers that utilize EAP keying material. For example, this mechanism would not enable verification of Channel Binding on wired IEEE 802 networks using [IEEE-802.1X].

### 5.3.4. Mutual Authentication

[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.

[RFC3748] Section 7.2.1 also describes the "Cryptographic binding" security claim, and [RFC4017] Section 2.2 requires support for this claim. As described in [I-D.puthenkulam-eap-binding], EAP method sequences and compound authentication mechanisms can 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 EAP keying material, it MAY be desirable to derive the compound key from a portion of the EMSK. Where this is done, 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.

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 authenticator/AAA client and backend authentication server communicate directly and credible key wrap is used (see Section 3.8), this ensures that the AAA Key Transport (phase 1b) achieves its security objectives: mutually authenticating the AAA client/authenticator and backend authentication server and providing transported keying material to the EAP authenticator and to no other party.

[RFC2607] Section 7 describes the security issues occurring when the

authenticator/AAA client and backend authentication server do not communicate directly. Where a AAA intermediary is present (such as a RADIUS proxy or a Diameter agent), and data object security is not used, transported keying material can be recovered by an attacker in control of the intermediary. As discussed in Section 2.1, unless the TSKs are derived independently from EAP keying material (as in IKEv2), possession of transported keying material enables decryption of data traffic sent between the peer and the authenticator to whom the keying material was transported. It also allows the AAA intermediary to impersonate the authenticator or the peer. Since the peer does not authenticate to a AAA intermediary it has no ability to determine whether it is authentic or authorized to obtain keying material.

However, as long as transported keying material or keys derived from it are 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 compromise of a AAA intermediary can be mitigated by implementation of redirect functionality, as described in [RFC3588] and [RFC4072].

The Secure Association Protocol does not provide for mutual authentication between the EAP peer and authenticator, only mutual proof of possession of transported keying material. In order for the peer to verify the identity of the authenticator, mutual proof of possession needs to be combined with impersonation prevention and Channel Binding. Impersonation prevention (described in Section 5.3.2) enables the backend authentication server to determine that the transported keying material has been provided to the correct authenticator. When utilized along with impersonation prevention, Channel Binding (described in Section 5.3.3) enables the EAP peer to verify that the EAP server has authorized the authenticator to possess the transported keying material. Completion of the Secure Association Protocol exchange demonstrates that the EAP peer and the authenticator possess the transported keying material.

## 5.4. Key Binding

Mandatory requirement from [RFC4962] Section 3:

> Bind key to its context
>
> Keying material MUST be bound to the appropriate context. The context includes the following:
>
> o The manner in which the keying material is expected to be used.

- The other parties that are expected to have access to the keying material.

- The expected lifetime of the keying material. Lifetime of a child key SHOULD NOT be greater than the lifetime of its parent in the key hierarchy.

Any party with legitimate access to keying material can determine its context. In addition, the protocol MUST ensure that all parties with legitimate access to keying material have the same context for the keying material. This requires that the parties are properly identified and authenticated, so that all of the parties that have access to the keying material can be determined.

The context will include the peer and NAS identities in more than one form. One (or more) name form is needed to identify these parties in the authentication exchange and the AAA protocol. Another name form may be needed to identify these parties within the lower layer that will employ the session key.

Within EAP, exported keying material (MSK, EMSK,IV) is bound to the Peer-Id(s) and Server-Id(s) which are exported along with the keying material. However, not all EAP methods support authenticated server identities (see Appendix A).

Within the AAA protocol, transported keying material is destined for the EAP authenticator identified by the NAS-Identifier Attribute within the request, and is for use by the EAP peer identified by the Peer-Id(s), User-Name [RFC2865] or Chargeable User Identity (CUI) [RFC4372] attributes. The maximum lifetime of the transported keying material can be provided, as discussed in Section 3.5.1. Key usage restrictions can also be included as described in Section 3.2. Key lifetime issues are discussed in Sections 3.3, 3.4 and 3.5.

## 5.5. Authorization

Requirement: The Secure Association Protocol (phase 2) conversation may utilize different identifiers from the EAP conversation (phase 1a), so that binding between the EAP and Secure Association Protocol identities is REQUIRED.

Mandatory requirement from [RFC4962] Section 3:

> Peer and authenticator authorization
>
> Peer and authenticator authorization MUST be performed. These entities MUST demonstrate possession of the appropriate keying material, without disclosing it. Authorization is REQUIRED

whenever a peer associates with a new authenticator. The authorization checking prevents an elevation of privilege attack, and it ensures that an unauthorized authenticator is detected.

Authorizations SHOULD be synchronized between the peer, NAS, and backend authentication server. Once the AAA key management protocol exchanges are complete, all of these parties should hold a common view of the authorizations associated with the other parties.

In addition to authenticating all parties, key management protocols need to demonstrate that the parties are authorized to possess keying material. Note that proof of possession of keying material does not necessarily prove authorization to hold that keying material. For example, within an IEEE 802.11, the 4-way handshake demonstrates that both the peer and authenticator possess the same EAP keying material. However, by itself, this possession proof does not demonstrate that the authenticator was authorized by the backend authentication server to possess that keying material. As noted in [RFC3579] in Section 4.3.7, where AAA proxies are present, it is possible for one authenticator to impersonate another, unless each link in the AAA chain implements checks against impersonation. Even with these checks in place, an authenticator may still claim different identities to the peer and the backend authentication server. As described in [RFC3748] Section 7.15, channel binding is required to enable the peer to verify that the authenticator claim of identity is both consistent and correct.

Recommendation from [RFC4962] Section 3:

Authorization restriction

If peer authorization is restricted, then the peer SHOULD be made aware of the restriction. Otherwise, the peer may inadvertently attempt to circumvent the restriction. For example, authorization restrictions in an IEEE 802.11 environment include:

- Key lifetimes, where the keying material can only be used for a certain period of time;
- SSID restrictions, where the keying material can only be used with a specific IEEE 802.11 SSID;
- Called-Station-ID restrictions, where the keying material can only be used with a single IEEE 802.11 BSSID; and
- Calling-Station-ID restrictions, where the keying

material can only be used with a single peer IEEE 802 MAC address.

As described in Section 2.3, consistent identification of the EAP authenticator enables the EAP peer to determine the scope of keying material provided to an authenticator, 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.

Within the AAA protocol, the authorization attributes are bound to the transported keying material. While the AAA exchange provides the AAA client/authenticator with authorizations relating to the EAP peer, neither the EAP nor AAA exchanges provide 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 lower layers where the authenticator consistently identifies itself to the peer and backend authentication server and the EAP peer completes the Secure Association Protocol exchange with the same authenticator through which it completed the EAP conversation, authorization of the authenticator is demonstrated to the peer by mutual authentication between the peer and authenticator as discussed in the previous section. Identification issues are discussed in Sections 2.3, 2.4 and 2.5 and key scope issues are discussed in Section 3.2.

Where the EAP peer utilizes different identifiers within the EAP method and Secure Association Protocol conversations, peer authorization can be difficult to demonstrate to the authenticator without additional restrictions. This problem does not exist in IKEv2 where the Identity Payload is used for peer identification both within IKEv2 and EAP, and where the EAP conversation is cryptographically protected within IKEv2 binding the EAP and IKEv2 exchanges. However within [IEEE-802.11] the EAP peer identity is not used within the 4-way handshake, so that it is necessary for the authenticator to require that the EAP peer utilize the same MAC address for EAP authentication as for the 4-way handshake.

## 5.6. Replay Protection

Mandatory requirement from [RFC4962] Section 3:

> Replay detection mechanism
>
> The AAA key management protocol exchanges MUST be replay

> protected, including AAA, EAP and Secure Association Protocol exchanges. 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] Section 2.2 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, it is possible that the recipient will 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-802.11] supports replay protection on all messages within the 4-way handshake; IKEv2 [RFC4306] also supports this.

## 5.7. Key Freshness

Requirement: A session key SHOULD be considered compromised if it remains in use beyond its authorized lifetime. Mandatory requirement from [RFC4962] Section 3:

> Strong, fresh session keys
>
> While preserving algorithm independence, session keys MUST be strong and fresh. Each session deserves an independent session key. 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.
>
> Some EAP methods are capable of deriving keys of varying strength, and these EAP methods MUST permit the generation of keys meeting a minimum equivalent key strength. BCP 86 [RFC3766] offers advice on appropriate key sizes. The National Institute for Standards and Technology (NIST) also offers advice on appropriate key sizes in [SP800-57].

> A fresh cryptographic key is one that is generated specifically for the intended use. In this situation, a secure association protocol is used to establish session keys. The AAA protocol and EAP method MUST ensure that the keying material supplied as an input to session key derivation is fresh, and the secure association protocol MUST generate a separate session key for each session, even if the keying material provided by EAP is cached. A cached key persists after the authentication exchange has completed. For the AAA/EAP server, key caching can happen when state is kept on the server. For the NAS or client, key caching can happen when the NAS or client does not destroy keying material immediately following the derivation of session keys.
>
> Session keys MUST NOT be dependent on one another. Multiple session keys may be derived from a higher-level shared secret as long as a one-time value, usually called a nonce, is used to ensure that each session key is fresh. The mechanism used to generate session keys MUST ensure that the disclosure of one session key does not aid the attacker in discovering any other session keys.

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

> 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 [RFC4017] requires EAP methods supporting these claims as well as methods capable of providing equivalent key strength of 128 bits or greater. See Section 3.7 for more information on key strength.

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. As discussed in Section 3.5, the Session-Timeout attribute enables the backend authentication server to limit the exposure of transported

keying material.

The EAP Session-Id, described in Section 1.4, 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 keying material.

The Secure Association Protocol, described in Section 3.1, MUST generate a fresh session key for each session, even if the EAP keying material and parameters provided by methods are cached, or either 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. While [IEEE-802.11] and IKEv2 [RFC4306] satisfy this requirement, [IEEE-802.16e] does not, since randomness is only contributed from the Base Station.

## 5.8. Key Scope Limitation

Mandatory requirement from [RFC4962] Section 3:

> Limit key scope
>
> Following the principle of least privilege, parties MUST NOT have access to keying material that is not needed to perform their role. A party has access to a particular key if it has access to all of the secret information needed to derive it.
>
> Any protocol that is used to establish session keys MUST specify the scope for session keys, clearly identifying the parties to whom the session key is available.

Transported keying material is permitted to be accessed by the EAP peer, authenticator and server. The EAP peer and server derive EAP keying material during the process of mutually authenticating each other using the selected EAP method. During the Secure Association Protocol exchange, the EAP peer utilizes 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 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 (see Section 1.5); TSK derivation is discussed in Section 2.1. Since demonstration of authorization within the Secure Association Protocol exchange depends on possession of transported keying material, the backend authentication server can obtain TSKs unless it deletes the transported keying material after sending it.

## 5.9. Key Naming

Mandatory requirement from [RFC4962] Section 3:

> Uniquely named keys
>
> AAA key management proposals require a robust key naming scheme, particularly where key caching is supported. The key name provides a way to refer to a key in a protocol so that it is clear to all parties which key is being referenced. Objects that cannot be named cannot be managed. All keys MUST be uniquely named, and the key name MUST NOT directly or indirectly disclose the keying material. If the key name is not based on the keying material, then one can be sure that it cannot be used to assist in a search for the key value.

EAP key names (defined in Section 1.4.1), along with the Peer-Id(s) and Server-Id(s), uniquely identify EAP keying material, and do not directly or indirectly expose EAP keying material.

Existing AAA server implementations do not distribute key names along with the transported keying material, although Diameter EAP [RFC4072], provides the EAP-Key-Name AVP for this purpose. Since the EAP-Key-Name AVP is defined within the RADIUS attribute space, it can be used either with RADIUS or Diameter.

Since the authenticator is not provided with the name of the transported keying material by existing backend authentication server implementations, existing Secure Association Protocols do not utilize EAP key names. For example, [IEEE-802.11] supports PMK caching; to enable the peer and authenticator to determine the cached PMK to utilize within the 4-way handshake the PMK needs to be named. For this purpose [IEEE-802.11] utilizes a PMK naming scheme which is based on the key. Since IKEv2 [RFC4306] does not cache transported keying material, it does not need to refer to transported keying material.

## 5.10. Denial of Service Attacks

Key caching can result in vulnerability to denial of service attacks. For example, EAP methods that create persistent state can 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 can limit the persistent state created by an EAP peer. For example, for each peer an EAP server can 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 can choose to limit the persistent state corresponding to each peer. This can be accomplished by limiting each peer to persistent state corresponding to a few EAP conversations, distinguished by the EAP Session-Id.

Whether creation of new TSKs implies deletion of previously derived TSKs depends on the EAP lower layer. Where there is no implied deletion, the authenticator can choose to limit the number of TSKs and associated state that can be stored for each peer.

## 6. IANA Considerations

This specification does not request the creation of any new parameter registries, nor does it require any other IANA assignments.

## 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.

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

[RFC4962] Housley, R. and B. Aboba, "Guidance for AAA Key Management", RFC 4962, July 2007.

### 7.2. Informative References

[8021XPreAuth] Pack, S. and Y. Choi, "Pre-Authenticated Fast Handoff in a Public Wireless LAN Based on IEEE 802.1x Model", Proceedings of the IFIP TC6/WG6.8 Working Conference on Personal Wireless Communications, p.175-182, October 23-25, 2002.

[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.

[Bargh] Bargh, M., Hulsebosch, R., Eertink, E., Prasad, A., Wang, H. and P. Schoo, "Fast Authentication Methods for Handovers between IEEE 802.11 Wireless LANs", Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots, October, 2004.

[GKDP] Dondeti, L., Xiang, J. and S. Rowles, "GKDP: Group Key Distribution Protocol", Internet draft (work in progress), draft-ietf-msec-gkdp-01, March 2006.

[He] He, C., Sundararajan, M., Datta, A. Derek, A. and J. C. Mitchell, "A Modular Correctness Proof of TLS and IEEE 802.11i", ACM Conference on Computer and Communications Security (CCS '05), November, 2005.

[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 802.11-2007, 2007.

[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] IEEE Standards for Local and Metropolitan Area Networks: Draft Standard for Virtual Bridged Local Area Networks, P802.1Q-2003, January 2003.

[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 LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security", IEEE 802.11i/D1, 2001.

[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-802.16e] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks: Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems: Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operations in Licensed Bands" IEEE 802.16e, August 2005.

[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.

[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-04.txt Internet draft (work in progress), October 2005.

[I-D.friedman-ike-short-term-certs] Friedman, A., Sheffer, Y. and A. Shaqed, "Short-Term Certificates", draft-friedman-ike-short-term-certs-02, Internet draft (work in progress), June 2007.

[I-D.irtf-aaaarch-handoff] Arbaugh, W. and B. Aboba, "Handoff Extension to RADIUS", draft-irtf-aaaarch-handoff-04.txt, Internet Draft (work in progress), October 2003.

[I-D.ohba-eap-channel-binding] Ohba, Y., Parthasrathy, M. and M. Yanagiya, "Channel Binding Mechanism Based on Parameter Binding in Key Derivation", draft-ohba-eap-channel-binding-02.txt, Internet draft (work in progress), December 2006.

[I-D.puthenkulam-eap-binding] Puthenkulam, J., Lortz, V., Palekar, A. and D. Simon, "The Compound Authentication Binding Problem", draft-puthenkulam-eap-binding-04, Internet draft (work in progress), October 2003.

[I-D.simon-emu-rfc2716bis] Simon, D., Aboba, B. and R. Hurst, "The EAP TLS Authentication Protocol", draft-simon-emu-rfc2716bis-11.txt, Internet Draft (work in progress), July 2007.

[I-D.ietf-tls-rfc4346-bis]
Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", draft-ietf-tls-rfc4346-bis-05.txt, Internet draft (work in progress), September 2007.

[MD5Collision] Klima, V., "Tunnels in Hash Functions: MD5 Collisions Within a Minute", Cryptology ePrint Archive, March 2006, http://eprint.iacr.org/2006/105.pdf

[MishraPro] Mishra, A., Shin, M. and W. Arbaugh, "Pro-active Key Distribution using Neighbor Graphs", IEEE Wireless Communications, vol. 11, February 2004.

[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.

[RFC2230] Atkinson, R., "Key Exchange Delegation Record for the DNS", RFC 2230, November 1997.

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

[RFC2782] Gulbrandsen, A., Vixie, P. and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D. and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC2865] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3007] Wellington, B., "Simple Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[RFC3162] Aboba, B., Zorn, G. and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.

[RFC3547] Baugher, M., Weis, B., Hardjono, T. and H. Harney, "The Group Domain of Interpretation", RFC 3547, July 2003.

[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.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M. and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 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.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4067] Loughney, J., Nakhjiri, M., Perkins, C. and R. Koodli, "Context Transfer Protocol (CXTP)", RFC 4067, July 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.

[RFC4186] Haverinen, H. and J. Salowey, "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, January 2006.

[RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, January 2006.

[RFC4282] Aboba, B., Beadles, M., Arkko, J. and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC4284] Adrangi, F., Lortz, V., Bari, F. and P. Eronen, "Identity Selection Hints for the Extensible Authentication Protocol", RFC 4284, January 2006.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4372] Adrangi, F., Lior, A., Korhonen, J. and J. Loughney, "Chargeable User Identity", RFC 4372, January 2006.

[RFC4334] Housley, R. and T. Moore, "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)", RFC 4334, February 2006.

[RFC4535] Harney, H., Meth, U., Colegrove, A. and G. Gross, "GSAKMP: Group Secure Association Group Management Protocol", RFC 4535, June 2006.

[RFC4763] Vanderveen, M. and H. Soliman, "Extensible Authentication Protocol Method for Shared-secret Authentication and Key Establishment (EAP-SAKE)", RFC 4763, November 2006.

[RFC4675] Congdon, P., Sanchez, M. and B. Aboba, "RADIUS Attributes for Virtual LAN and Priority Support", RFC 4675, September 2006.

[RFC4718] Eronen, P. and P. Hoffman, "IKEv2 Clarifications and Implementation Guidelines", RFC 4718, October 2006.

[RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: a Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, January 2007.

[RFC3576bis] Chiba, M., Dommety, G., Eklund, M., Mitton, D. and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", draft-ietf-radext-rfc3576bis-13.txt, Internet draft (work in progress), October 2007.

[SP800-57] National Institute of Standards and Technology, "Recommendation for Key Management", Special Publication 800-57, May 2006.

[Token] Fantacci, R., Maccari, L., Pecorella, T. and F. Frosali, "A secure and performant token-based authentication for infrastructure and mesh 802.1X networks", IEEE Conference on Computer Communications, June 2006.

[Tokenk] Ohba, Y., Das, S. and A. Duttak, "Kerberized Handover Keying: A Media-Independent Handover Key Management Architecture", Mobiarch 2007.

## Acknowledgments

Thanks to Ashwin Palekar, Charlie Kaufman and Tim Moore of Microsoft, Jari Arkko of Ericsson, Dorothy Stanley of Aruba Networks, Bob Moskowitz of TruSecure, Jesse Walker of Intel, Joe Salowey of Cisco and Russ Housley of Vigil Security for useful feedback.

Authors' 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

Pasi Eronen
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

EMail: pasi.eronen@nokia.com

## Appendix A - Exported Parameters in Existing Methods

This Appendix specifies Session-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 Session-Id, Peer-Id and Server-Id (could be the null string).

### EAP-Identity

The EAP-Identity method is defined in [RFC3748]. It does not derive keys, and therefore does not define the Session-Id. The Peer-Id and Server-Id are the null string (zero length).

### EAP-Notification

The EAP-Notification method is defined in [RFC3748]. It does not derive keys and therefore does not define the Session-Id. The Peer-Id and Server-Id are the null string (zero length).

### EAP-MD5-Challenge

The EAP-MD5-Challenge method is defined in [RFC3748]. It does not derive keys and therefore does not define the Session-Id. The Peer-Id and Server-Id are the null string (zero length).

### EAP-GTC

The EAP-GTC method is defined in [RFC3748]. It does not derive keys and therefore does not define the Session-Id. The Peer-Id and Server-Id are the null string (zero length).

### EAP-OTP

The EAP-OTP method is defined in [RFC3748]. It does not derive keys and therefore does not define the Session-Id. The Peer-Id and Server-Id are the null string (zero length).

### EAP-AKA

EAP-AKA is defined in [RFC4187]. The EAP-AKA Session-Id is the concatenation of the EAP Type Code (0x17) with 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 EAP re-authentication identity. The Server-Id is the null string (zero length).

EAP-SIM

EAP-SIM is defined in [RFC4186]. The EAP-SIM Session-Id is the concatenation of the EAP Type Code (0x12) with 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 EAP re-authentication identity. The Server-Id is the null string (zero length).

EAP-PSK

EAP-PSK is defined in [RFC4764]. The EAP-PSK Session-Id is the concatenation of the EAP Type Code (0x2F) with the peer (RAND_P) and server (RAND_S) nonces. The Peer-Id is the contents of the ID_P field and the Server-Id is the contents of the ID_S field.

EAP-SAKE

EAP-SAKE is defined in [RFC4763]. The EAP-SAKE Session-Id is the concatenation of the EAP Type Code (0x30) with the contents of the RAND_S field from the AT_RAND_S attribute, followed by the contents of the RAND_P field in the AT_RAND_P attribute. Note that the EAP-SAKE Session-Id is not the same as the "Session ID" parameter chosen by the Server, which is sent in the first message, and replicated in subsequent messages. The Peer-Id is contained within the value field of the AT_PEERID attribute and the Server-Id, if available, is contained in the value field of the AT_SERVERID attribute.

EAP-TLS

For EAP-TLS, the Peer-Id, Server-Id and Session-Id are defined in [I-D.simon-emu-rfc2716bis].

## Full Copyright Statement



## Intellectual Property



## 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/