Archived NIST Technical Series Publication

The attached publication has been archived (withdrawn), and is provided solely for historical purposes. It may have been superseded by another publication (indicated below).

Archived Publication

Series/Number:	NIST Special Publication 800-120
Title:	Recommendation for EAP Methods Used in Wireless Network Access Authentication
Publication Date(s):	September 2009
Withdrawal Date:	October 19, 2018
Withdrawal Note:	This publication is out of date. Refer to relevant standards from the Internet Engineering Task Force (IETF), http://www.ietf.org/ .

Superseding Publication(s)

The attached publication has been **superseded by** the following publication(s):

Series/Number:	
Title:	
Author(s):	
Publication Date(s):	
URL/DOI:	

Additional Information (if applicable)

Contact:	Computer Security Division (Information Technology Laboratory)
Latest revision of the attached publication:	
Related information:	https://csrc.nist.gov https://csrc.nist.gov/publications/detail/sp/800-120/archive/2009-09-17
Withdrawal announcement (link):	N/A

National Institute of Standards and Technology
U.S. Department of Commerce

Date updated: October 19, 2018

NIST Special Publication 800-120

Recommendation for EAP Methods Used in Wireless Network Access Authentication

Katrin Hoeper and Lily Chen

Computer Security Division

Information Technology Laboratory

C O M P U T E R S E C U R I T Y

September 2009

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Patrick Gallagher, Deputy Director

Abstract

This Recommendation specifies security requirements for authentication methods with key establishment supported by the Extensible Authentication Protocol (EAP) defined in IETF RFC 3748 for wireless access authentications to federal networks.

KEY WORDS: EAP methods, authentication, key establishment.

Acknowledgments

The authors, Katrin Hoeper and Lily Chen, wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors gratefully acknowledge and appreciate contributions by Elaine Barker, William Burr, Sheila Frankel, Antonio Izquierdo, Ray Perlner, and Tim Polk of NIST. The authors also thank the many contributions by the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This Recommendation has been prepared for use by Federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Table of Contents

Figures

Tables

1. Introduction

As different wireless technologies are launched to enable user mobility and provide pervasive network and service accessibility, security has been a prominent requirement for the U.S. Federal Government in such access environments. Access authentication and the establishment of keys that protect wireless traffic are both core security components in wireless applications.

The Extensible Authentication Protocol (EAP), specified in IETF RFC 3748 [18], is a framework for access authentication, which supports different authentication methods that are specified as EAP methods. Numerous EAP methods have been published as IETF RFCs and implemented by various vendors. Currently, EAP has been adopted by various wireless standards as an access authentication and key establishment protocol. As described in RFC 4017[19], it is desirable for EAP methods used for wireless LAN to support mutual authentication and key derivation.

Informally, in an EAP execution for wireless access, there are usually three players, a wireless station such as a laptop computer (called a "peer" in EAP), a wireless point of attachment (PoA) (may be collocated with an "authenticator" in EAP) and a backend EAP server, which determines whether the access authentication of a peer succeeds or fails using EAP. Based on the authentication results, the PoA authorizes or denies the wireless access of the peer. For example, IEEE 802.11 [14] for Wireless Local Area Networks (WLANs) uses IEEE 802.1X [15] as a way to encapsulate EAP messages over LAN (EAPOL) for providing access authentication and key establishment. Even though EAP is also used as an authentication framework for wired access, this Recommendation will focus on requirements for wireless access authentication and key establishment.

EAP was originally designed and used to support user password authentication to Internet service providers for dial-up services using the Point to Point Protocol (PPP). At that time, the point-to-point nature of the connection-oriented wireline phone network and the consequently relatively limited, applicable attack models did not demand extensive security provisions for the use of EAP. As a result, the EAP methods defined in [18] support neither mutual authentication nor key derivation. Today dial-up Internet service is comparatively rare, but shared media Ethernets, as well as wireless networks supporting various degrees of mobility, are quite common, while authentication of both users and machines is increasingly considered a basic prerequisite for network access. In such environments and with much more sophisticated modern Internet attack models, naive implementations of early EAP methods as defined in [18] are insecure.

In addition to these older and less secure methods, there are now a number of stronger EAP methods intended for integrated use with standard wireless access protocols, such that the keys generated in an EAP execution are used to protect against wireless eavesdroppers and more sophisticated active man-in-the middle attacks. Many EAP methods define a set of supported cryptographic schemes and algorithms—for example, for authentication, key establishment, and/or message protection—called a ciphersuite. Other EAP methods do not offer such a choice, and only support one cryptographic algorithm for each security functionality. The diversity of EAP methods enables the implementation and use of a variety of authentication and key establishment methods to protect wireless network access.

Security assessments of each EAP method with a specific set of cryptographic algorithms and schemes are crucial for securely launching wireless applications. These assessments must be based on a well-defined set of common security requirements for EAP methods used for wireless access authentication and key establishment for link protection.

This Recommendation is applicable to EAP servers operated by or for federal agencies and federal mobile terminals when they are used with the federal servers. Federal users may encounter the use of EAP methods in other contexts, such as travelers desiring wireless access in hotels and airports, in meetings, in public "hotspots" and the like, as well as by commercial public wireless Internet service providers. This Recommendation is not intended to constrain mobile federal users from the use of non-federal wireless services that do not implement EAP authentication as specified here; indeed it may be difficult for users to tell which precise methods are used.

The requirements presented in this Recommendation target a security level such that the users of agency intranets that employ complying EAP methods with appropriate wireless interface encryption and authentication, with the keys established through EAP, may consider that their connection is as secure as a wired connection to the intranet, in the sense that traffic eavesdropping and injection requires a physical compromise of the communication equipment. Similarly, agencies may treat such wireless terminals as they do other stations on the intranet. However, when wireless users log onto non-federal points of attachment, they should assume that the wireless interface may not be protected, and they are subject to attack. In these cases, wireless users may either restrict their use of the network to avoid exposing sensitive information, or establish end-to-end protection by using such methods as virtual private networks and protocols such as the Transport Layer Security (TLS)[26].

2. Scope and Purpose

This Recommendation formalizes a set of core security requirements for EAP methods when employed by the U.S. Federal Government for wireless access authentication and key establishment. The requirements **should** be considered as generic, in the sense that they are independent of specific wireless technologies. When there are differences between this Recommendation and the referenced IEEE and IETF standards, this Recommendation **shall** have precedence for U.S. Government applications. This Recommendation addresses the validation of a few selected EAP methods, in order to explain the requirements.

3. Definitions, Symbols and Abbreviations

3.1 Definitions

Approved	FIPS-approved or NIST Recommended. An algorithm or technique that meets at least one of the following: 1) is specified in a FIPS or NIST Recommendation, 2) is adopted in a FIPS or NIST Recommendation or 3) is specified in a list of NIST-approved security functions (e.g., specified as approved in the annexes of FIPS 140-2).
Access authentication	A procedure to obtain assurance of the accuracy of the claimed identity of an entity for the purpose of authorizing network access. This is sometimes referred to as network access authentication. Access authentication is an entity authentication for access control purposes (see entity authentication).

Authentication credentials	A piece of information, the possession of which can be used to obtain assurance of the accuracy of the claimed identity of an entity.
Authenticator	A network entity that executes access authentication protocols with the entity that requests access to a network. An authenticator may use an EAP server to conduct authentication operations. In wireless access networks, an authenticator may be collocated with a Point of Attachment (PoA). (See the definition of PoA).
Authentication framework	Method-independent specification of the authentication message format, message sequences, and protocol state machine.
Authentication, Authorization, and Accounting (AAA)	The framework for access control in which a server verifies the authentication and authorization of entities that request network access and manages their billing accounts. AAA protocols with EAP support are RADIUS [17] and Diameter[20]. (See the definition of Authorization.)
Authentication method	A cryptographic scheme used by an entity to provide assurance of its identity to another entity. For example, by proving the knowledge of some secret information, called authentication credentials, such as a secret or private key or by demonstrating possession of some token, such as a smartcard.
Authorization	A procedure to verify whether an entity is eligible to access a requested network or service.
Ciphersuite	A set of cryptographic algorithms and parameter specifications. For example, EAP method ciphersuites typically contain authentication and key establishment algorithms, as well as algorithms used for encryption and integrity protection, including corresponding key sizes and other parameters.
Ciphersuite negotiation	A procedure executed between two entities to agree on a ciphersuite that will be used in subsequent communications. In this Recommendation, a peer and EAP server negotiate the ciphersuite that they will use in the remainder of the current EAP execution. (See the definition of peer).
Compound key	In a tunnel-based EAP method, a key derived from the tunnel key and the inner keys established by the tunneled authentication (and key establishment) methods.
Cryptographic binding	A specific procedure in a tunnel-based EAP method that binds together the tunnel protocol and the tunneled authentication method(s) that is executed within the protective tunnel. In this Recommendation, cryptographic binding is a procedure to assure the server that it executed tunnel and inner method with the same peer.

Extensible Authentication Protocol (EAP)	An authentication framework defined in IETF RFC 3748 [18]. The Extensible Authentication Protocol can support different authentication methods.
EAP execution	An EAP execution conducts authentication through a single authentication method, if it is not tunneled and usually starts with an EAP-Request/Identity message, and terminates with an EAP-Success/Failure message. An EAP execution may consist of multiple authentication methods, if they are tunneled. (See tunneled authentication methods.)
EAP method	An authentication method carried out by EAP. In this Recommendation, it implies a specific way to use the EAP message format to carry the data for authentication, as well as the data for other cryptographic purposes.
EAP layer	A virtual network layer to carry EAP data frames. It is defined relative to the lower layers. (See the definition of a lower layer.)
EAP server	A network server that executes EAP with a peer. The EAP server is generally located in the protected (wired) network. When EAP is used as an access authentication protocol, the EAP server may be collocated with an AAA server.
Extended Master Session Key (EMSK)	A key derived by the communication endpoints of a successful EAP method execution, i.e., typically by a peer and the EAP server. The key is not shared with any other entities. The EMSK is derived from the master key. (See the definition of master key).
Entity	An individual (person), organization, device or a combination of them. "Party" is a synonym. In this Recommendation, an entity can be a physical unit or a functional unit. When an entity is a functional unit, it may be located in a physical unit. For example, an authenticator can be a functional unit and located in a PoA, which is considered as a physical unit.
Entity authentication	A procedure to obtain assurance of the claimed identity of an entity. In a two party protocol, entity authentications may be unidirectional or mutual. (See the definition of mutual authentication).

Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions are designed to satisfy the following properties: 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and 2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output. Approved hash functions are specified in FIPS 180-3 [1]. In some EAP methods, hash functions are used in digital signatures and to build key derivation functions and message authentication codes.
Inner key	A key established in a tunneled authentication (and key establishment) method.
Implicit key authentication	A property of key establishment protocols that provides assurance to one protocol participant that the other protocol participant is the only other party that could possibly be in possession of the correct established key.
Key confirmation	A procedure to provide assurance to one party (the key confirmation recipient) that another party (the key confirmation provider) actually derived the correct secret keying material as a result of a key establishment.
Key confirmation key	A cryptographic key derived from some keying material and used for key confirmation of this keying material.
Key derivation	The process that derives keys from another key or from a secret output value, called shared secret in [8] and [9], obtained through a key establishment procedure.
Key derivation function	A function that is used to derive keys.
Key establishment	A procedure, conducted by two or more participants, which culminates in the derivation of keying material by all participants (see keying material). Key establishment can be based on pre-shared keys or on public key-based schemes. For example, EAP key establishment is executed between a peer and the EAP server to derive EAP keying material.
Key export	A mechanism by which a key is delivered from an EAP layer to a lower layer [18].

Key hierarchy	A tree structure that represents the relationship of different keys. In a key hierarchy, a node represents a key used to derive the keys represented by the descendent nodes. A key can only have one precedent, but may have multiple descendent nodes.
Key holder	An entity that is entitled to hold a specific key and use it for cryptographic operations, including deriving other keys from that key.
Keying material	The output of a key derivation function, a segment of which, with the required length, can be used as a cryptographic key.
Key transport	A procedure to deliver a key from one entity to another entity in a protected manner. In this Recommendation, key transport refers to key delivery from the EAP server to another entity, for example, to an authenticator.
Long-term credentials	Authentication credentials used for access authentication to provide assurance of the correctness of the claimed identity. In this Recommendation, long-term credentials could be a pair of public/private keys, where the public key is certified by a trusted third party, or a symmetric key shared between the entity to be authenticated and an EAP server. They are called long-term credentials, because, different from session keys, the same credentials, once certified or distributed, will be used in different executions of an authentication protocol.
Lower layer	A virtual network layer that is defined relative to the EAP layer. The lower layers may include the layers in which the actual transport protocols are defined to carry EAP data. (See definition of EAP layer).
Master Key (MK)	In this Recommendation, the master key refers to the keying material derived by participating parties upon successfully completing a key establishment protocol. The derived keying material may be used to derive further keys. (See definition of Key Establishment)
Message Authentication Code (MAC) algorithm	A family of one-way cryptographic functions that is parameterized by a symmetric key and produces a *MAC* on arbitrary data. A MAC algorithm can be used to provide data origin authentication, as well as data integrity.
Master Session Key (MSK)	A key shared by all parties participating in an EAP method upon a successful protocol completion. The MSK may be exported to a lower layer and transported to an authenticator. The master session key may be derived from the master key or may be the master key.

Mutual authentication	A procedure in which both participating entities obtain assurance of the claimed identity of the other entity. Therefore, an authentication procedure is executed in each direction, where the messages of the two procedures are interleaved to ensure that both entities participate in the same protocol execution. In this Recommendation, mutual authentication refers to an access authentication during which 1) the peer that requests network access authenticates to the EAP server, and 2) the EAP server also authenticates to the peer.
Nonce	A time-varying value that has at most a negligible chance of repeating, for example, a random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these.
Peer	In this Recommendation, a peer is the entity attempting to access the network.
Point of Attachment (PoA)	A device that connects wireless stations to (usually wired) networks, and to each other, through wireless links. In this Recommendation, a Point of Attachment (PoA) is used as a media-independent generic term, e.g. it could describe an access point in IEEE 802.11 networks.
Protected communication	In this Recommendation, it refers to applying encryption and/or message authentication to the communicated information to provide confidentiality and authenticity, where authenticity includes information integrity and source authenticity.
Session key	A cryptographic key established for use for a relatively short period of time. In this Recommendation, an EAP execution may establish session keys, such as MSK, EMSK, and TEK, from each of which further session keys can be derived and used in data protection algorithms, such as an encryption or a message authentication.
Shall	This term is used to indicate a requirement of a Federal Information Processing Standard (FIPS) or a requirement that needs to be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.
Should	This term is used to indicate an important recommendation. Ignoring the recommendation could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.
Transient EAP Key (TEK)	A session key used to protect messages or to derive further session keys used at the EAP layer

Trusted third party	A party trusted by all other parties, e.g. a peer and the EAP server, involved in an EAP execution, such as a certificate authority (CA) that issues certificates when public/private key pairs are used as long-term credentials.
Tunnel-based EAP method	An EAP method in which a protective tunnel is established by executing a tunnel protocol between a peer and the EAP server, followed by the execution of one or more authentication methods within the established protective tunnel. Examples of tunnel-based EAP methods are PEAP [33], EAP-TTLS [28], and EAP-FAST [23]. (See the definition of tunnel protocol).
Tunneled authentication method	An authentication method that is executed inside a protective tunnel that has been established by a tunnel protocol (See the definition of tunnel protocol).
Tunnel Key (TK)	In this Recommendation, a key derived by a peer and the EAP server as a result of a successful tunnel protocol execution. (See the definition of tunnel protocol).
Tunnel protocol	A protocol, e.g. TLS [26], used to establish a tunnel key (TK) between two parties. The key is then used to establish a protective communication link between these parties; the protected link is also referred to as a protective tunnel.
Wireless station	A wireless terminal device. In this Recommendation, when EAP is executed as a wireless access authentication protocol, the peer is accommodated in a wireless station.

3.2 Symbols and Abbreviations

$[m]_K$	The output of a Message Authentication Code over a message m using a secret key K
AAA	Authentication, Authorization and Accounting
AP	Access Point
AVP	Attribute Value Pair
CTK	Compound Key
DH	Diffie-Hellman (a key establishment algorithm)
EAP	Extensible Authentication Protocol
EMSK	Extended Master Session Key

GPSK	Generalized Pre-Shared Key (see [32])
IETF	The Internet Engineering Task Force
INK	Inner Key
KCK	Key Confirmation Key
KDF	Key Derivation Function
MAC	Message Authentication Code
MK	Master Key
MSK	Master Session Key
NAK	Not acknowledged. A message used in EAP to indicate the requested information or function is not available.
PoA	Point of Attachment
PRF	PseudoRandom Function
RADIUS	Remote Authentication Dial In User Service
RSA	Rivest, Shamir, Adleman (an algorithm)
SR-XX-i	Security Requirement number i for subroutine XX, where XX is one of the following: AUTH authentication, CB channel binding. CN ciphersuite negotiation, KD key derivation, KE key establishment, MP message protection, TBEAP tunnel-based EAP method, TP tunnel protocol, TEAP tunneled authentication method. These requirements are mandatory, and are written using **shall** statements.

[SR-XX-i]	Security Requirement number i for subroutine XX. The square brackets are used to indicate that these requirements are strongly recommended, but not mandatory; they are written using **should** statements. (See the definition of SR-XX-i for a list of subroutines for XX).
TCK	Transient Cipher (encryption) Key
TEK	Transient EAP Key
TIK	Transient Integrity (protection) Key
TK	Tunnel Key
TLS	Transport Layer Security (see [26])
TTLS	Tunneled Transport Layer Security (see [28])

4. EAP Overview

This section provides an overview of the Extensible Authentication Protocol (EAP).

EAP is an access authentication framework that was originally developed to support peer authentication before granting the peer access to the network. The actual cryptographic schemes used for achieving the desirable security objectives are defined in the EAP methods. With the growing complexity of applications and security demands, the scope of the security objectives and features has been extended to include server authentication, key establishment, privacy and many other features. This section provides an introduction to EAP and the EAP methods before discussing their security vulnerabilities in Section 5, and the security requirements in Section 8 and Section 9.

4.1 EAP Communication Links and Involved Parties

As specified in [18], EAP is a framework for two party authentication protocols that are executed between a peer and an authenticator. Typically, a backend server, called an EAP server, executes the cryptographic operations to authenticate peers and has access to the necessary authentication credentials. Therefore, EAP is actually executed between a peer and the EAP server, while an authenticator is employed as a pass-through device to pass messages from the peer to the server and vice versa. As a result, an EAP execution typically involves three parties, with communications across two links (see Figure 1). On the other hand, in some implementations, the authenticator takes over the role of the EAP server, i.e. only two parties are involved in the EAP execution. However, the remainder of this document focuses on the more common pass-through mode. The first communication link between the peer and the authenticator is referred to as CL1 in the remainder of this document. Since this Recommendation discusses security requirements for wireless applications, CL1 is assumed to have a wireless portion. When an authenticator is collocated with a PoA, which is often the case, CL1 is a wireless link. The second communication link (referred to as

CL2) is a wired link in the network between the authenticator and the EAP server[1]. Note that over CL2, an AAA protocol, such as RADIUS or Diameter, can be used to carry EAP packets. That is, the EAP Server is accessed through AAA protocols. The EAP server is sometimes located within an AAA server, in which case the authenticator acts as an AAA client. When the authenticator and the EAP server are collocated in a single device, CL2 is considered as physically protected. The EAP parties and communication links are illustrated in Figure 1, where the authenticator is shown as collocated with a PoA and interfaces with the peer through a wireless link CL1.

Figure 1: EAP Communication Model

The security requirements for the EAP methods presented in this Recommendation must be observed by federal peers and federal EAP servers employing EAP for secure access to federal intranets. However, the requirements do not apply to authenticators, because authenticators using the pass-through mode do not need to implement EAP methods. A brief description of the three participating parties is given as follows:

- **Peer:** A wireless (mobile) client who wishes to access a network through a wireless connection in order to use a provided service or access data in this network.
- **Authenticator:** A network entity that interfaces with the peer in an EAP execution. The remainder of this document focuses on EAP executions in which an authenticator passes the EAP packets between the peer and the EAP server, i.e. the EAP is using the pass-through mode, and the EAP server informs the authenticator of the authentication outcome. Based on this result, the authenticator grants or denies peer access to the network. If the used EAP method derives keying material, the server delivers some of the freshly derived keying material to the authenticator.
- **EAP server:** A backend server that performs peer authentications through an EAP execution. If the EAP server does not store authentication credentials, the EAP server might need to access an external database that stores the credentials and policies defining when a peer is authorized to access the network. The EAP server communicates with peers through authenticators that are acting as pass-through devices, and informs the participating authenticators about the authentication (and authorization) results. If keying material was derived as part of the EAP execution, the server transports freshly established EAP keys to participating authenticators.

[1] Note that in some networks, and especially in roaming scenarios, additional message forwarding entities—such as proxies and routers— may be located between the authenticator and the EAP server. In the remainder of this document, such entities are not explicitly mentioned again, but the security considerations and solutions discussed for authenticators can be extended to these entities.

4.2 EAP Message Flows

EAP, as defined in RFC 3748 [18], consists of four different message types: *request*, *response*, *success*, and *failure*. Some new EAP message types are introduced in EAP re-authentication extensions [31]. For illustration purposes, this Recommendation will focus on the original messages defined in [18], where all presented security requirements are actually message type-independent. Request messages are sent by an authenticator or EAP server, while response messages are sent by a peer to the authenticator, and may be forwarded to the EAP server. Success/failure messages are sent either by an authenticator or EAP server. Request and response messages are typically paired through a type field *EAP-TYPE*, where each pair must be of the same EAP type to define the information and format of the request and response. This type of pairing is not true for NAKs in the response message, which indicates that the requested information or function is not available, and another request message with different options must be sent.

An *EAP execution* usually starts with an EAP-Request/EAP-Type=Identity message, and terminates with an EAP-Success/Failure message. A typical message flow for an EAP execution in pass-through mode is illustrated in Figure 2. The first pair of request and response messages is of type *identity*, i.e., the authenticator requests the peer's identity, and the peer returns its claimed identity in the response message. In this Recommendation, the pass-through mode is assumed, i.e. all request messages—except the initial identity request—as well as the success/failure message are sent by the EAP server, and all response messages by the peer are returned to the server. Upon receiving the identity response message from the peer, the server selects an EAP method and sends the first EAP-message of Type *T_d*. If the peer supports and accepts the selected EAP method, it replies with the corresponding response message of the same type. Otherwise, the peer sends a NAK, and the EAP server either selects another EAP method or aborts the EAP execution with a failure message. The selected EAP method determines the number of request/response pairs. While the success or failure of an entire EAP execution is indicated by the exchanged EAP-Success/Failure messages, intermediate steps, such an authentication or key derivations, may have their own result indications. For example, a failed peer authentication may lead the EAP server to silently drop the session, whereas in other cases, a message requesting another attempt to execute the failed procedure is sent. This type of behavior is defined in the state machine of an EAP method, and in the remainder of this Recommendation, it is assumed that the EAP methods under consideration provide such result indications.

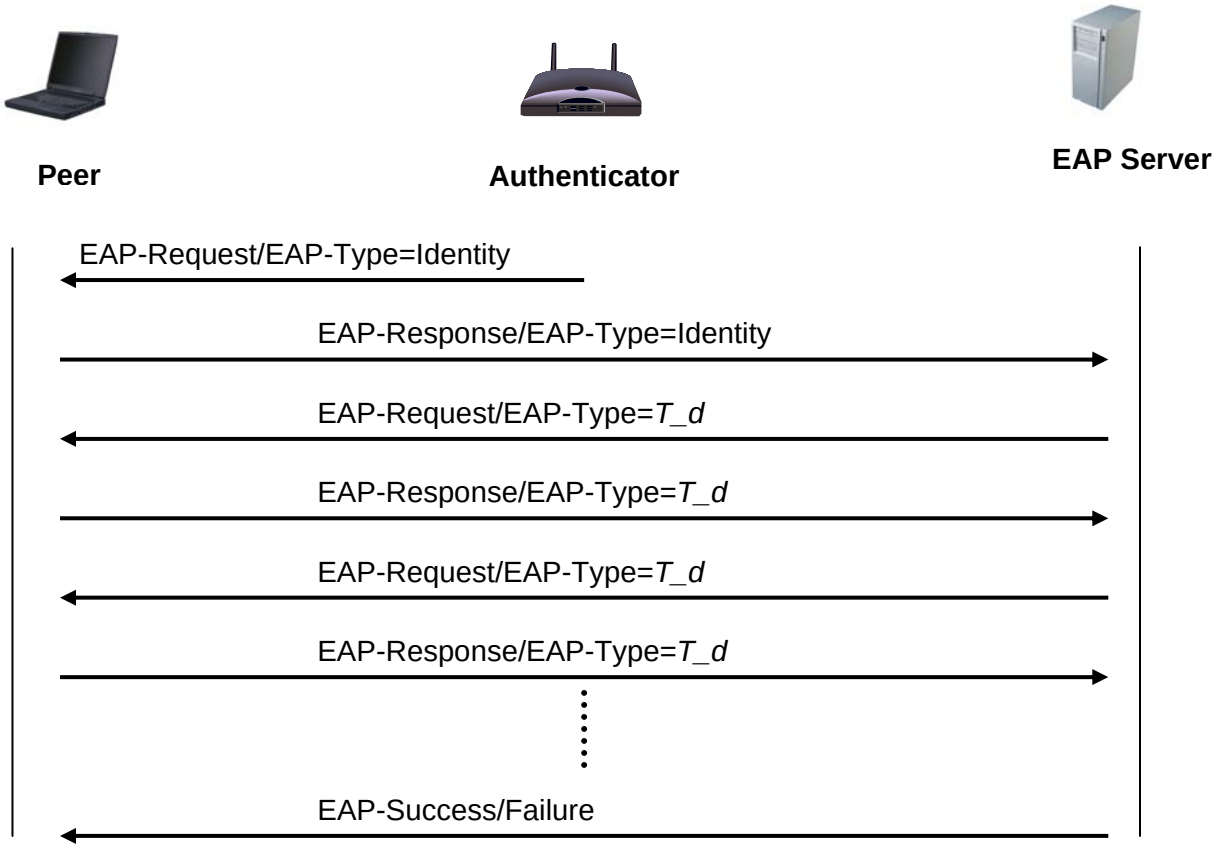

Figure 2: Example Message Flow of an EAP Execution in Pass-Through Mode

4.3 EAP Protocol Stacks

As mentioned in Section 4.1, EAP operates over a wireless communication link (CL1) and a wired link (CL2). For example, CL1 might be employing IEEE 802.11, whereas CL2 could employ an AAA protocol for communications. As a consequence of these different communication media and employed communication protocols, EAP is executed across different communication protocol stacks. A peer and an authenticator typically communicate over a lower layer protocol specified by the employed wireless technology. On the other hand, an authenticator and the EAP server commonly communicate over layers that are higher in the protocol stack, such as the AAA and IP layers. A typical EAP protocol stack for the three involved parties is depicted in Figure 3. Note that authenticators in pass-through mode do not need to support any EAP method and, thus, do not have an EAP method layer.

In this Recommendation, an authenticator is shown as a point of attachment (PoA). For the purpose of an EAP execution, an authenticator is a functional entity that may reside in a PoA, while the CL1 communication link is implemented by a PoA through lower layer protocols, as shown in Figure 3.

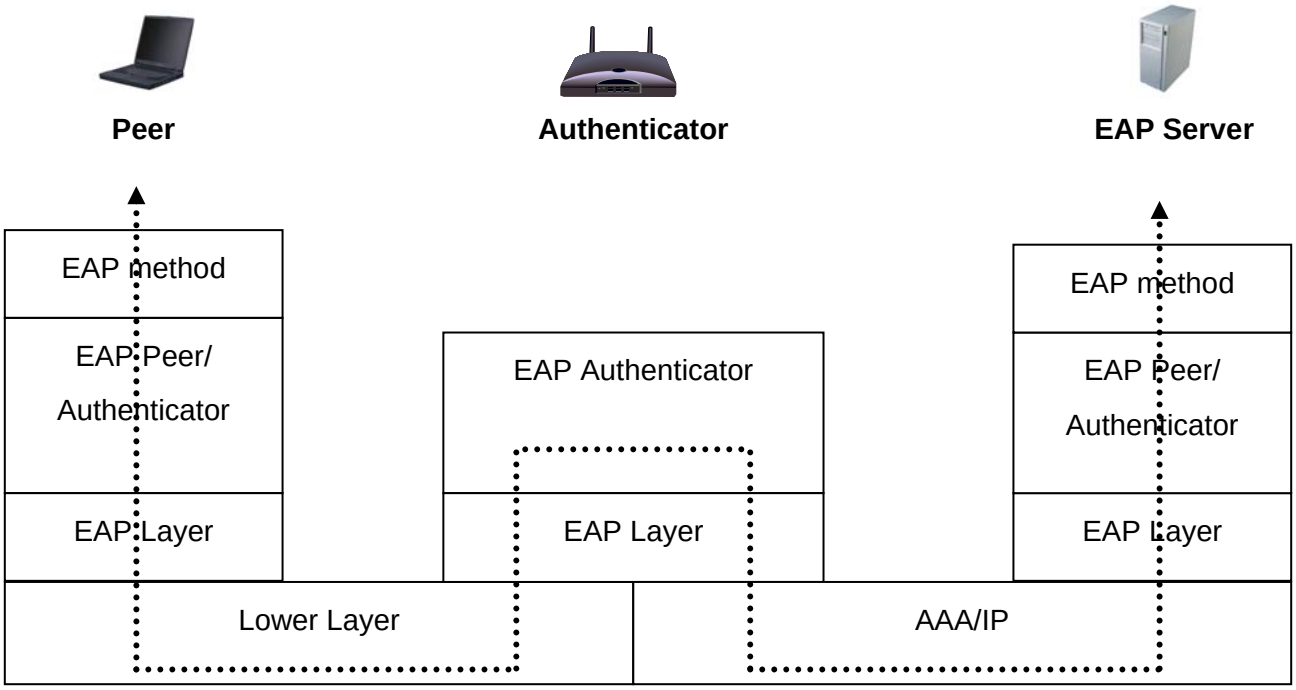

Figure 3: EAP Protocol Stacks

4.4 Tunnel-based EAP Methods

In some EAP methods, one or more authentication methods are executed in a protective tunnel, which is referred to as a *tunnel-based EAP method*. Tunnel-based EAP methods consist of two phases; in the first phase, the peer and EAP server execute a *tunnel protocol* to establish a protected connection. Then, in the second phase, both parties execute authentication methods within the protective tunnel. An authentication method executed within the tunnel is referred to as a *tunneled EAP method* in the remainder of this Recommendation. Commonly, the TLS protocol [26] is used to establish the tunnel. The established tunnel key (referred to as *TK* in the remainder of this Recommendation) is used to protect the authentication(s) executed in the second phase. The authentication conducted inside the tunnel is sometimes called an inner authentication method and is typically used for peer authentication and, optionally, to derive keying material. Inner authentication methods can be EAP methods or other authentication methods. Examples of tunnel-based EAP methods are: EAP-TTLSv0 [28], PEAP [33], and EAP-FAST [23], which all use the TLS protocol to establish a tunnel. An overview of a typical tunneled EAP method with its two phases and derived keys is illustrated in Figure 4.

Tunnel-based EAP methods were introduced for two reasons: first, and most importantly, tunnel-based EAP methods enable the use of password-based authentication methods for peers. Without tunneling, widely deployed password-based authentication methods are insecure. Secondly, tunnel-based EAP methods can enable privacy protection, because the peer and, optionally, the server identifiers can be exclusively exchanged in the tunnel and, thus, prevent an eavesdropper from identifying these entities. However, in this Recommendation, the server is required to be authenticated during tunnel establishment (see Section 9.2). Therefore, only peer privacy can be protected by the tunneled methods.

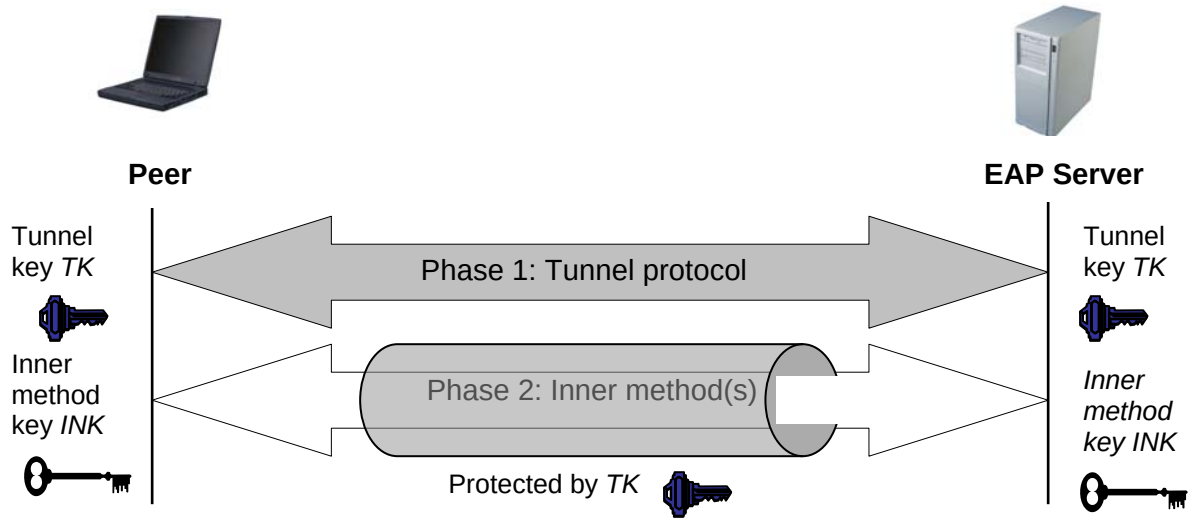

Figure 4: Overview Tunnel-based EAP Method

While RFC 3748 [18] prohibits the use of multiple authentication methods within a single EAP execution, due to its vulnerability to man-in-the-middle attacks and incompatibility with existing implementations, the prohibition does not apply to tunnel-based EAP methods. A tunnel-based EAP method is considered as one authentication method and, thus, multiple authentication methods may be executed within the protective tunnel. The tunnel-based EAP method is intended to protect all inner methods from attacks. Note that the feature of executing several authentication methods within a protective tunnel can be useful if several layers of peer authentication are necessary, e.g. first authenticating the peer device, then authenticating the user operating the device.

4.5 EAP Key Derivation and Key Hierarchy

An EAP method that provides key establishment either establishes a master key (MK) between a peer and the EAP server, or assumes the existence of such a key. In the latter case—i.e. an MK is pre-shared—the key establishment protocol is used to exchange fresh input that is used in combination with MK to derive further keys. If a pre-shared key is not used, the key establishment protocol outputs a fresh MK that is then directly used to derive further keys. All EAP methods that provide key establishment derive a Master Session Key (MSK) and an Extended Master Session Key (EMSK). In addition, the methods may derive Transient EAP Keys (TEKs), which may be used to further derive session keys, e.g. Transient Cipher (encryption) Keys (TCKs) and Transient Integrity protection Keys (TIKs). A typical EAP key hierarchy is shown in Figure 5, where dashed lines indicate optional keys.

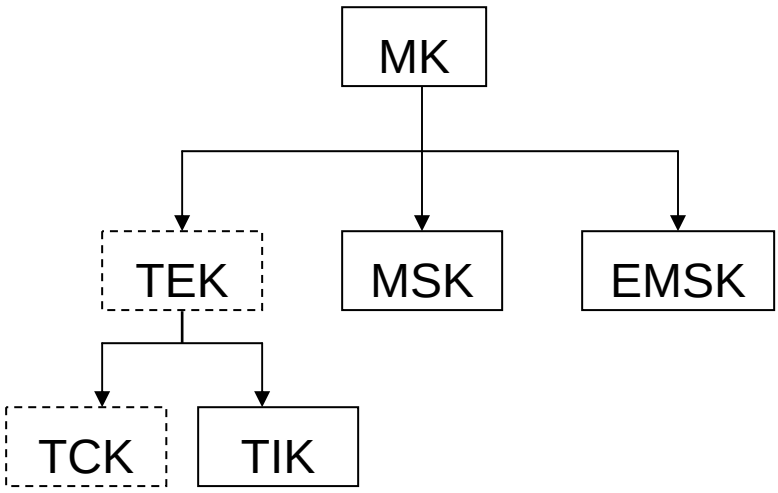

Figure 5: EAP Key Hierarchy

The use of the keys is assigned as follows:

1. MSK is exported to the lower layers and may be transported to the authenticator to derive keys to protect the CL1 wireless link upon the completion of an EAP execution.
2. EMSK remains on the server and is reserved for future use. Recently, EMSK has been considered for deriving handover keys (see [30]).
3. TEKs are used to derive session keys, such as transient encryption keys (TCK) and transient integrity protection keys (TIK), to protect the messages of the ongoing EAP execution. In other words, TEKs are used for message protection at the EAP layer[2].

4.6 EAP Ciphersuite Negotiation

In many EAP methods, the peer and EAP server agree on a cryptographic ciphersuite defining all cryptographic algorithms, including parameters that will be used to protect the remainder of the EAP execution. While some EAP methods support a large number of ciphersuites, others only support a few (e.g. EAP-GPSK supports two) or a single ciphersuite (e.g. EAP-AKA [21]). In the latter case, all cryptographic algorithms and parameters are pre-defined in the EAP method and are not negotiable. Ciphersuite negotiations provide crypto-agility and backward compatibility and are part of the EAP execution. For example, a suite may include some of the following algorithm types: authentication (*AUTH*), key establishment (*KE*), key derivation function (*KDF*), message encryption (*ENC*), and message integrity protection (*INT*). A ciphersuite could be denoted as $CS_i=\{AUTH_i\ KE_i,\ KDF_i,\ ENC_i,\ INT_i,\}$. Here, it is assumed that each of the algorithms includes a selection of related parameters. In addition to the ciphersuites, the version numbers of protocols and other features, such as channel binding and cryptographic bindings, which will be introduced in Section 8.4 and Section 9.1 respectively, might be negotiated at the beginning of an EAP execution.

[2] In some of the EAP methods, TEK is not derived explicitly. In those cases, TIK (and TCK, if encryption is applied to EAP messages) can be derived directly from MK.

During a typical ciphersuite negotiation, an offering party (the peer or EAP server) offers a choice of *n* supported and acceptable ciphersuites $CS\_offer = \{CS_1, CS_2, ..., CS_n\}$ to the selecting party (the EAP server or peer). The selecting party selects a supported and acceptable ciphersuite out of the offer *CS_offer*, with $CS\_select = \{CS_i\}$ where $i \in \{1,..n\}$. Here, *acceptable* means that the offered (selected) ciphersuite is in compliance with the offering (selecting) party's security policy. This type of negotiation is referred to as a two-flow negotiation in this Recommendation. An example of the message flow for a two-flow negotiation is illustrated in Figure 6, where the EAP server acts as the offering party. There are other types of ciphersuite negotiations, e.g. so-called bidding negotiations in which the offering party repeatedly offers a ciphersuite until the selecting party accepts the offer.

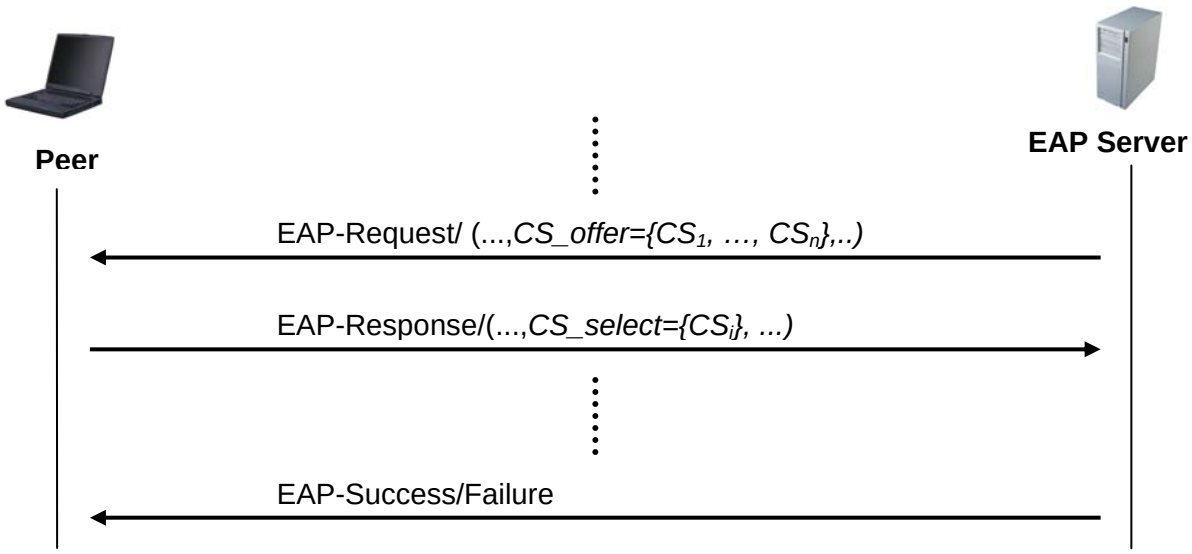

Figure 6: Example Two-way Ciphersuite Negotiation

5. Vulnerabilities of EAP in Wireless Applications

This section discusses vulnerabilities of EAP in wireless applications under attacks by outsiders and insiders, where insiders include rogue peers, authenticators, intermediary entities in the backend (such as proxies) and the EAP server. Existing EAP methods used in wireless applications may suffer from several security vulnerabilities if they are not properly protected or configured. These vulnerabilities are due to certain properties of the EAP framework, and weaknesses of particular EAP methods or ciphersuites, as well as the wireless application environment under consideration. The application environment also has an impact on the severity of potential risks, e.g. attacks could be more lucrative in some applications.

5.1 Wireless Links

The wireless communication link between mobile peers and authenticators makes EAP methods susceptible to a variety of passive attacks by outside attackers in communication range. Unlike wired systems, an adversary does not need to connect physically to the system, but simply needs to be in communication range. Passive attacks include:

- Eavesdropping

- Traffic analysis

Through eavesdropping, an adversary can intercept a communication and, thus, access all unprotected information that is exchanged over this link. Traffic analysis can be used to track users, e.g. by correlating peer identifiers used in multiple EAP executions. As discussed in Section 6.1, EAP can provide privacy for higher layer identifiers, thus preventing the use of such identifiers in traffic analysis. However, privacy protection for lower-layer identifiers, such as Media Access Control (MAC) addresses, cannot be provided by an EAP method. Privacy protection of such identifiers is outside the scope of this Recommendation.

For the same reason as for passive attacks, active attacks on a wireless link are far more likely than on a wired system. Such active attacks include:

- Impersonation attacks, in which an attacker assumes the identity of a legitimate party and attempts to convince a verifier that he is that party. Impersonation attacks are conducted through, but are not limited to, the following methods:

  a. *masquerading attacks*, in which a party directly claims to be somebody else;

  b. *man-in-the-middle attacks*, in which an adversary may replay, relay, reflect, interleave and/or modify messages in one or more protocol executions between two parties to fool at least one of those parties about the identity of the other party;

  c. *replay attacks*, in which an adversary replays messages from a previously-observed protocol execution;

  d. *extraction of authentication credentials*, in which an adversary tries to get information about the long-term authentication credentials. This can be done through

    i. *dictionary attacks*, in which an adversary breaks a weak password and uses it in subsequent sessions;

    ii. *chosen-text attacks*, in which an adversary strategically chooses challenges in an attempt to extract information about the claimant's long-term credentials.

  It can be observed that impersonation attacks can be conducted at the protocol level (e.g. man-in-the-middle attacks) and/or at the cryptographic level (e.g. extraction of authentication credentials).

- Key extraction attacks, in which an adversary obtains secret keying material by manipulating or breaking the employed key establishment scheme.

5.2 Negotiable Cryptographic Algorithms

As mentioned in Section 4.6, at the beginning of many EAP method executions, the ciphersuite used to protect the remainder of the execution is negotiated by the peer and the EAP server. Such negotiations are vulnerable to downgrading attacks, in which an inside or outside attacker tries to force the peer and EAP server to agree on a weak ciphersuite that contains cryptographic algorithms that are susceptible to attacks. For example, a rogue PoA or a man-in-the-middle could reduce the offered set of ciphersuites that is sent over the wireless link to a subset consisting of only weak ciphersuites. As a result, the selecting party can only choose from weak ciphersuites.

5.3 Sensitive Information and Data Confidentiality

The wireless applications under consideration in this Recommendation make impersonating federal peers and/or federal networks lucrative and, thus, increase the likelihood of attacks. For example, federal networks generally store some confidential information, which makes impersonating federal peers in order to access these data attractive. Furthermore, many applications require the peer to send security-sensitive information, such as personal identification information, passwords, and other sensitive information over the wireless link. This makes accessing the user traffic by impersonating a legitimate network attractive.

Other applications that are out of scope of this document may also increase the likelihood of attacks. For example, commercial mobile wireless applications that provide wireless Internet access or International roaming make impersonating a legitimate subscriber or stealing a service by attacking EAP more attractive to outside attackers. For the same reason, rogue networks, authenticators or intermediary entities may masquerade as a peer's home network to lure the peer to connect to their network and collect roaming fees in higher service rates.

5.4 Tunnel-based EAP Methods

When the tunnel protocols and inner authentication methods are not securely tied together, the tunnel based EAP methods are vulnerable to man-in the-middle attack as identified in [37]. These attacks exploit the fact that tunnel protocols and inner authentication methods are not tied together, and that executions of authentication methods outside or within a tunnel are indistinguishable. In an attack, the adversary initiates a tunnel-based EAP method with an EAP server in which only server authentication is provided. Once the tunnel is established, the adversary initiates an EAP execution with a peer by pretending to be an authenticator connected to a legitimate EAP server. The adversary replays the peer's responses as its own responses to the EAP server through the tunnel. Hence, the adversary can successfully impersonate the peer to the EAP server. Such a man-in-the-middle attack will be further elaborated in Section 9.1.

Similar man-in-the-middle attacks are feasible on sequences of EAP methods that are executed within a protective tunnel.

5.5 Vulnerability of the Points of Attachment

Unlike the EAP server and other network entities in the backend network, points of attachments are usually more accessible and, thus, more prone to compromises. For example, in non-federal access networks, PoAs are typically distributed in a large geographic area and are often located in public places, such as airports or libraries. Even in the considered federal intranets, PoAs are typically more exposed than the EAP server that can be placed in a locked server room, while a PoA may be installed in a hallway.

Therefore, attacks to an EAP execution can be launched through PoAs. We consider two kinds of PoAs: *compromised PoAs* or *rogue PoAs*. Compromised PoAs are authorized parts of a legitimate network, but under the control of attackers. A compromised PoA can be detected if the PoA behaves in an unexpected way, e.g. broadcasting false information to peers, modifying messages from peers and the EAP server, etc. On the other hand, compromised PoAs that faithfully execute all protocol steps cannot be detected. In this case, freshly derived keying material is still delivered

to the compromised PoA. Hence, physically breaking into the device enables the attacker to eavesdrop.

Rogue PoAs are placed in a network by an attacker. A rogue PoA may launch more active attacks, such as advertising false information and inducing the users to be connected to a network with which the users would not otherwise connect, as discussed in [36]. Such a rogue PoA is referred as a *lying PoA* and can be detected and the attack prevented by an EAP property called channel binding, which will be discussed in Section 8.4. Through channel binding, attacks by rogue PoAs placed in a federal intranet can be prevented by EAP.

The above discussed attacks using rogue PoAs or compromised PoAs can happen, no matter whether the PoA is collocated with an authenticator or not. Therefore, in the remainder of this Recommendation, no difference is made between rogue (compromised) PoAs and rogue (compromised) authenticators.

6. EAP Objectives for Wireless Network Access Authentications

6.1 Objectives and Features

The general objective of EAP is to verify the identity and authorization of a peer before granting access to the network. Depending on the EAP method and the application environment, more complex objectives may apply. Since this Recommendation considers EAP for wireless network access authentications, EAP methods need to thwart the attacks outlined in Section 5. For example, wireless applications demand mutual authentication to protect federal peers from fraudulent networks and federal networks from unauthorized access. In addition, key establishment is necessary to enable the protection of subsequent communications over the wireless link.

Hence, an EAP method employed in the federal wireless scenarios under consideration **shall** have the following two security objectives:

O-1. Secure mutual authentication and authorization between a peer and the wireless access network;
O-2. Secure key establishment between a peer and the EAP server.

The first security objective includes an authentication and authorization check of the EAP peer by the EAP server and the authentication of the EAP server to the peer. In the remainder of this Recommendation, peer authorization is considered as a part of the peer authentication and not explicitly mentioned again, because the authorization check of a peer does not occur over any EAP-protected communication links. The second security objective is necessary to protect the remainder of on-going EAP executions, as well as to make keying material available to protect the CL1 wireless link and, potentially, other applications.

In addition to the above security objectives, many EAP methods provide additional features. While numerous features exist, an EAP method may provide the following feature to thwart traffic analysis, e.g. to prevent tracking mobile users:

F-1. Privacy protection of the peers.

This feature refers to the property that the peer's identity is not revealed during protocol execution. It is important to note that supporting privacy and/or any other feature **shall not** violate the two aforementioned security objectives.

6.2. Procedures

Every EAP method consists of several procedures that are necessary to ensure the security goals, enable crypto-agility and backward compatibility and prevent attacks. The EAP methods under consideration may include the following procedures:

1. Ciphersuite negotiation to enable crypto-agility and backward-compatibility;
2. Mutual authentication of a peer and the EAP server to ensure that federal peers and federal EAP servers provide assurance of their acclaimed identities to each other;
3. Key establishment between a peer and the EAP server to provide keying material to protect the remainder of the EAP execution and the wireless link;
4. Service information exchange to ensure the detection of malicious information sent by rogue authenticators or other rogue intermediary entities;
5. Message protection to utilize the established keying material to protect the remainder of the EAP execution.

These procedures can be executed sequentially, in parallel or in an interleaved fashion. For example, authentication and key establishment could be combined into an authenticated key establishment process. If the authentication or key establishment algorithm is not negotiated as part of the ciphersuite negotiation, both the ciphersuite negotiation and the algorithm can be started at the same time. For the sake of an easier discussion, all procedures as listed above are treated separately in the remainder of this document.

In order to achieve the two security objectives in Section 6.1, certain security requirements are necessary for the five procedures listed above. These requirements are identified and described in detail in Section 8 for non-tunneled EAP methods, and in Section 9 for tunnel-based EAP methods. Note that some of the procedures may not be included in certain EAP methods and, as discussed in Sections 8 and 9, only procedures two, three and five are mandatory to comply with the requirements in this Recommendation, while support of procedure four is encouraged.

7. Pre-conditions for EAP

The pre-conditions for EAP are a set of system prerequisites that are necessary to enable the secure execution of any EAP method in a particular environment. The security discussion of an EAP method is only meaningful when all pre-conditions are met. However, the methods for achieving these pre-conditions are outside the scope of this Recommendation.

The following pre-conditions apply to any EAP method executed in the wireless applications under consideration.

7.1 Secure Set Up of Long-Term Credentials

EAP methods based on shared secret keys or passwords for peer authentication or mutual authentication require securely provisioning the secrets prior to EAP executions. In addition, all long-term secret keying material must be securely stored. In order to prevent domino effects, each peer needs to set up keys with the EAP server pairwisely. This ensures that if one peer's long-term secret key is compromised, another peer's long-term secret key is not affected.

Note that a symmetric key, if used as a long-term credential, can be generated by the peer, the server, or a trusted third party. In any case, the key must be kept secret and be distributed in a protected manner. Such secret keys are used as long-term credentials in purely symmetric EAP methods, such as EAP-GPSK [32], as well as hybrid EAP methods in which the server authenticates to the peer using a public key certificate, while the peer uses a password to authenticate to the server, as is possible in EAP-FAST [23] for example.

Whenever public keys are used as long-term authentication credentials, the respective certificates must be issued prior to EAP executions. A certificate must be accessible during an EAP execution, and the receiver of a public key certificate (i.e., a peer and/or EAP server) must be able to verify the certificate and trust the party that issued the certificate. This includes the capability of a receiver to check whether a certificate has been revoked.

7.2 Secure Connections in Accessed Backend Network

The CL2 communication link between an authenticator and an EAP server cannot be secured by EAP methods. For this reason, the EAP framework is based on the assumption that the authenticator, the EAP server and other network entities in the wired backend network that are involved in the EAP execution are able to securely communicate with each other. This may include, but is not limited to, message authentication, integrity protection, and confidentiality protection. Typically, the entities in the backend network, such as the authenticator and the EAP server, communicate using AAA protocols (such as RADIUS [17] or Diameter [20]). In that case, the AAA protocol needs to provide all necessary security properties for protecting the CL2 link.

7.3 Authorization and Authentication Information of Authenticators and other Entities in the Backend Network

In order to address threats by compromised or rogue authenticators and other intermediary entities (as described in Section 5.5), the EAP server needs to have access to the information that each legitimate authenticator is supposed to broadcast to peers, as well as the information that the EAP server is supposed to receive from the authenticators and other intermediary entities via the CL2 link during an EAP execution. Basically, the EAP server must be able to verify the correctness, authenticity and authorization of information sent by all entities in the backend network that participate in an EAP execution. For example, such information could be securely stored in a protected database and only be accessible by authorized parties. Please refer to [36] for the information that such a database should contain and how it could be set up.

8. Security Requirements for Non-tunneled EAP Methods

This section specifies the security requirements for non-tunneled EAP methods supporting key establishment that are used in wireless applications. The derived requirements are independent of any specific wireless technology and **shall** be applied whenever EAP is employed by a federal peer for access authentication to a federal wireless network.

As previously mentioned, EAP methods may support ciphersuite negotiation or only provide one set of non-negotiable cryptographic algorithms. Both strategies comply with this Recommendation, as long as the algorithms in the negotiated or provided ciphersuite meet all the security requirements. The security requirements for ciphersuite negotiation are provided in Section 8.1, while the security requirements for the authentication, key establishment and message protection algorithms are

specified in Sections 8.2, 8.3 and 8.5, respectively. The security requirements for channel bindings are in provided in Section 8.4.

8.1 Protected Ciphersuite Negotiation

This Recommendation classifies ciphersuites containing only the algorithms satisfying the security requirements specified in each corresponding FIPS publications and NIST Special Publications as approved in the wireless scenarios under consideration. The peer may support ciphersuites that are not approved for (backward) compatibility reasons in access authentications in non-federal settings. However, a ciphersuite negotiation under consideration must enable the federal EAP server and federal peer to select an approved ciphersuite. In fact, only approved cipersuites are considered acceptable, as defined in Section 4.6, by federal EAP servers, whereas non-approved ciphersuites may be acceptable in non-federal settings. Such a requirement is summarized as follows.

SR-CN-1 Each supported EAP method **shall** at least offer one approved ciphersuite.

In general, ciphersuites are negotiated before transient EAP keys (*TEK*) are available and the algorithms are agreed upon to protect the messages of the negotiation. Therefore, it is always possible for a man-in-the-middle to reduce the set of offered ciphersuites $CS\_offer = \{CS_1, CS_2, \ldots, CS_n\}$ so that only the weakest of the ciphersuite(s) is offered (e.g., $CS\_offer' = \{CS_w\}$). As a result, the selecting party has no choice but to select the weakest ciphersuite(s). The described attack assumes that the weakest ciphersuite(s) is (are) available and acceptable by the selecting party. The only way to detect such an attack is by providing post-verification of the negotiation. That is, once the *TEK*s are available, both parties derive a transient integrity protection key *TIK* and send integrity protected verification messages, which include the sent and received messages prior to *TEK* establishment, to each other. An example message flow of a two-flow ciphersuite negotiation with post-verification is illustrated in Figure 7, where $AUTH_i\text{-}T$ and $KE_i\text{-}T$, $T = 1, 2$, represent the authentication date and key exchange data respectively.

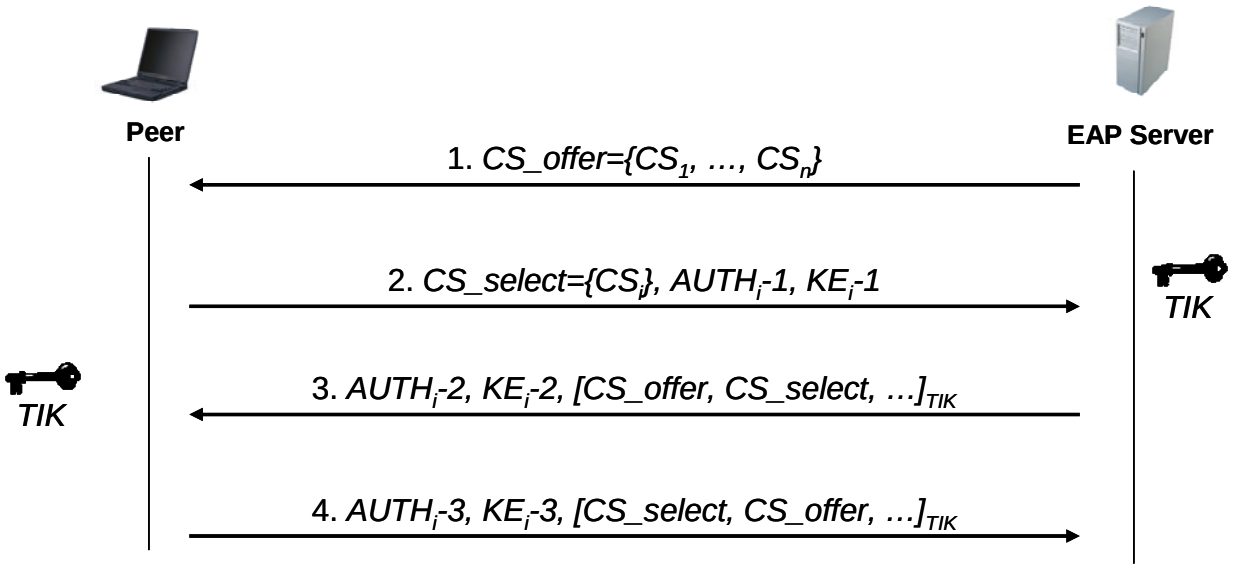

Figure 7: Ciphersuite Negotiation with Post-Verification

Besides ciphersuites, in some EAP methods, the protocol versions and features are also negotiable. If the negotiations must be done before TEKs are available, then the following requirement applies.

[SR-CN-2] Each supported EAP method supporting negotiations of ciphersuites, protocol versions, and features **should** include post-verification.

Note that the described downgrading attack could not be detected in EAP methods without post-verification. However, the attacker could not subsequently successfully attack any of the negotiated algorithms, because—according to the requirements in this Recommendation—any acceptable ciphersuite only contains cryptographically strong algorithms. The consequences of EAP methods not supporting approved ciphersuites or implementations in which acceptable ciphersuites do not meet the requirement of this Recommendation could have severe consequences, as described in [35].

8.2 Mutual Authentication

The wireless applications considered here require mutual authentication between the peer and EAP server for reasons motivated in Section 5 and discussed in Section 6.

SR-AUTH-1 Each EAP method **shall** provide mutual authentication between a peer and the EAP server.

This requirement applies to the authentication algorithm that is part of the negotiated ciphersuite or is specified as the only choice in the used EAP method.

Entity authentication employs a cryptographic algorithm that demonstrates knowledge of certain secret information, for example, a cryptographic key. Usually, the claimant generates a digital signature or a message authentication code (MAC) over some data, depending on whether a public key-based or symmetric key-based method is used. In order to make sure that the claimant has to use its secret information for each authentication, the data may include a nonce.

In order to prevent attacks on the cryptographic algorithms employed by the mutual authentication procedure, the following requirements apply.

SR-AUTH-2 Approved cryptographic schemes **shall** be employed for authentication that each satisfies the security strength requirements for algorithms and key sizes in NIST SP 800-57 [10].

SR-AUTH-3 When symmetric key-based MACs are employed for entity authentication, approved algorithms **shall** be used as specified in FIPS 198 [4] for HMAC, and in NIST SP800-38B [5] for CMAC; the selected key size for the MAC **shall** satisfy the security strength requirements specified in SP 800-57 [10].

SR-AUTH-4 When a digital signature algorithm is employed for entity authentication, an approved algorithm and key size **shall** be used that satisfies the security strength requirements specified by FIPS 186-3 [2] and NIST SP 800-57 [10].

In an authentication procedure, re-using (i.e., replaying) the digital signature or MAC generated in a previous procedure may allow an impersonation attack (as listed in Section 5.1). In order to prevent such replay attacks, each digital signature or MAC used for entity authentication may be generated by the entity to be authenticated using a nonce. The nonce can be a random number generated by

another entity and sent as a challenge, a sequence number or a time stamp; note that the sequence number or time stamp may not be explicitly sent to the entity to be authenticated, but may be implicitly known. The signature or MAC is sent as an *authentication response*, whether or not the nonce is explicitly sent. In order to prevent impersonation through replay attacks on the authentication protocol, the following requirement applies.

SR-AUTH-5 An authentication response **shall** resist replay attacks by using non-repeating nonces.

Using random nonces requires two flows for replay prevention, but is typically straight forward to implement. On the other hand, sequence numbers and timestamps both only require one communication flow; however, the use of sequence numbers requires careful tracking for each session and each verifier, while timestamps depend on a frequent synchronization of clocks.

Once authenticated, it must be ensured that all of the remaining messages continue to be exchanged between the authenticated parties throughout the remainder of the EAP session. This leads to the following requirement.

SR-AUTH-6 The EAP method to be used **shall** ensure that no entity but the authenticated parties can take over an EAP execution after the successful completion of the authentication subroutine.

Typically, this requirement can be satisfied by binding the authentication and key establishment procedures (e.g. by applying digital signatures or MACs to key establishment messages), and subsequently using the derived, authenticated keying material to protect the remainder of the protocol execution.

8.3 Key Establishment

In order to protect the data exchanged during an EAP execution, e.g. to thwart some of the attacks outlined in Section 5, only EAP methods with key establishment (a.k.a. key derivation as specified in [18]) **shall** be used. The key establishment algorithm specified either by the negotiated ciphersuite or by the used EAP method needs to meet all the requirements described in the remainder of this section.

In order to prevent attacks on the cryptographic algorithms employed by the key establishment procedure, the following requirements apply.

SR-KE-1 Approved cryptographic schemes for key establishment **shall** be employed that follow the security strength requirements for algorithms and key sizes in NIST SP 800-57 [10].

SR-KE-2 Key establishment schemes using public key cryptography **shall** conform to the requirements in NIST SP 800-56A [8] for discrete logarithm-based key establishment and SP 800-56B [9] for integer factorization-based key establishment.

SR-KE-3 Whenever authentication and key establishment subroutines are combined as a mutually authenticated key establishment subroutine, it **shall** comply with all the requirements for the authentication subroutine stated in Section 8.2.

In order to prevent attacks at the protocol level, the following requirements apply to a key establishment protocol employed by an EAP method.

SR-KE-4 A key establishment procedure **shall** provide mutual implicit *key authentication,* i.e., the established keying material is only known to the peer and the EAP server.

SR-KE-5 A key establishment procedure **shall** provide *key freshness*, i.e. the established key is (pseudo-) random and the probability to repeat a previously established key is small.

[SR-KE-6] A key establishment procedure **should** provide *key control*, i.e., the peer and the EAP server should both contribute data for the key computation.

This property prevents a single protocol participant from controlling the value of an established key. In this way, protocol participants can ensure that generated keys are fresh and have good random properties.

[SR-KE-7] A key establishment procedure **should** provide *key confirmation,* i.e., the peer and the EAP server should both obtain assurance that they computed *MK* correctly. Key confirmation is commonly achieved by using one of the derived keys to generate a message authentication code. Mutual key confirmation, combined with mutual implicit key authentication, provides mutual explicit key authentication.

[SR-KE-8] A key establishment procedure (for public-key based key establishment schemes) **should** provide *forward secrecy (FS)*, i.e., a compromise of long-term private or pre-shared secret keys does not enable an adversary to compute the *MK* generated in previous EAP executions.

This property is typically achieved by executing an ephemeral Diffie-Hellman key establishment scheme.

8.3.1 Key Hierarchies and Key Derivation Functions

In order to prevent attacks on the derived keying material and to limit the impact of key disclosure, the key derivation functions and derived key hierarchies need to meet the following requirements.

SR-KD-1 The key derivation functions **shall** comply with NIST SP 800-108 [13].

SR-KD-2 The key hierarchy **shall** conform to the requirements in NIST SP 800-108 [13].

8.4 Service Information Exchange

An EAP peer is neither able to authenticate an authenticator nor verify the information received from it. As a result, EAP methods that do not support the exchange of additional service information are susceptible to the lying PoA problem and other attacks by rogue authenticators (see Section 5.5). This demands that keys only be transported from the EAP server to an appropriate authenticator that the peer intended to connect to (not necessarily a particular authenticator, but rather an authenticator of a particular network) and that is authorized and authenticated by the EAP server. This can be achieved by so-called *channel bindings* in which all participating entities (i.e. the EAP peer, the authenticator, any other intermediary entity and the EAP server) are securely bound to an EAP method execution [18] [36]. This ensures the consistency of the information provided to the EAP peer and the EAP server by any intermediary entity. EAP channel binding may require the following steps as described in [36]:

1. The peer sends the information received from the authenticator to the server with integrity protection;

2. The EAP server checks the consistency of the received information from the EAP peer, as well as the information received from the authenticator (or the last entity in the communication chain in the CL2 link) with the information stored in its protected database.
3. The EAP server sends the verification result to the EAP peer in an integrity protected message.

Please note that steps 1 and 3 require dedicated data fields which are integrity-protected for a peer and EAP server to exchange the service information and verification results. Considering that EAP methods complying with this Recommendation derive keys (see Section 8.3) and provide EAP message protections (see Section 8.5), the following requirement allows introducing channel binding as a future extension.

[SR-CB-1] Each EAP method **should** be capable of providing integrity protection for additional payloads to securely exchange service information necessary for providing channel bindings.

The payload can be used to carry service information to provide channel binding. EAP channel bindings can be achieved by using encapsulated AVPs described in [36]. For the EAP methods which are not explicitly providing channel binding, satisfying [SR-CB-1] allows the addition of an integrity protected data field as a container for the service information as a future extension that will provide channel binding.

The second step in the above requires the EAP server to be capable of checking whether the received information from the peer and authenticator is consistent with the server's stored information, which is described as a system pre-requisite in Section 7.3.

8.5 EAP Message Protections

After fresh EAP keys are established, and the protection algorithms are agreed upon, all subsequent EAP messages can be protected, thus, preventing many of the attacks outlined in Section 5. Typically, MACs are used for message authentication and integrity protection, whereas symmetric key encryption algorithms are used for message confidentiality. Before a ciphersuite is negotiated and protection keys are available, no EAP messages requiring confidentiality can be exchanged. On the other hand, the authenticity and integrity of information exchanged before the ciphersuite negotiation and key establishment can be ensured by post-verification (see Section 8.1). The requirements are summarized as follows.

SR-MP-1 Post-verification **shall** be provided for all integrity-vulnerable information that has been exchanged before a transient integrity key is available.

SR-MP-2 Confidential information **shall not** be exchanged unless encryption becomes available and is applied.

SR-MP-3 After a transient integrity key is available, all messages **shall** be integrity protected.

To comply with this Recommendation, the following requirements apply to the cryptographic algorithms used for message-protection (i.e. integrity- and confidentiality-protection, as well as message authentication).

SR-MP-4 Algorithms used for integrity-protection and confidentiality **shall** follow the requirements on cryptographic strength for algorithms and key sizes in NIST SP 800-57 [10].

SR-MP-5 Algorithms used for integrity-protection **shall** comply with FIPS 198 [4], when HMAC is employed, and NIST SP 800-38B [5], when CMAC is employed.

SR-MP-6 Algorithms used for confidentiality-protection **shall** comply with FIPS 197 [3], when AES is used, and NIST SP 800-67 [12], when TDES is used.

SR-MP-7 Algorithms used for authenticated encryption **shall** comply with NIST SP 800-38C [6] and 800-38D [7].

9. Requirements for Tunnel-based EAP Methods

A tunnel-based EAP method describes a framework for executing authentication methods inside a protective tunnel that has been established by a tunnel protocol (see Section 4.4). Tunnel-based EAP methods (such as EAP-TTLSv0, PEAP and EAP-FAST) specify how to encapsulate a tunnel protocol (typically TLS) into EAP messages, and then execute EAP method(s) or other authentication method(s) inside the tunnel. Generally, the tunnel-based EAP methods specify which tunnel protocol is used, but do not restrict which authentication methods can be used as inner methods. This section describes the security requirements for all components of tunnel-based EAP methods, namely, the tunnel-based method itself, the employed tunnel protocol and the authentication method(s) executed within the tunnel.

9.1 Tunnel-based EAP Methods

Under some conditions, tunnel-based EAP methods are vulnerable to a particular man-in-the-middle attack described in [37]. In this attack, an adversary—masquerading as a peer—initiates a tunnel-based EAP method with the EAP server. As part of this EAP method, the adversary executes a tunnel protocol with the EAP server in which the EAP server authenticates to the adversary (thinking it is the peer). Upon a successful tunnel protocol execution, both the adversary and the EAP server are in possession of the established tunnel key *TK*. The server then initiates an inner authentication method inside the protective tunnel. The adversary—acting as an EAP server—initiates a parallel session with a peer using the same authentication method outside a tunnel. The adversary then replays the peer's response into the tunnel, making the EAP server believe that the messages are coming from the other end of the tunnel. Hence, the inner authentication method, and the tunnel-based EAP method are executed successfully, and both the adversary and the EAP server subsequently share the established *MSK* if it is derived from the tunnel key *TK*. The attack is illustrated in Figure 8.

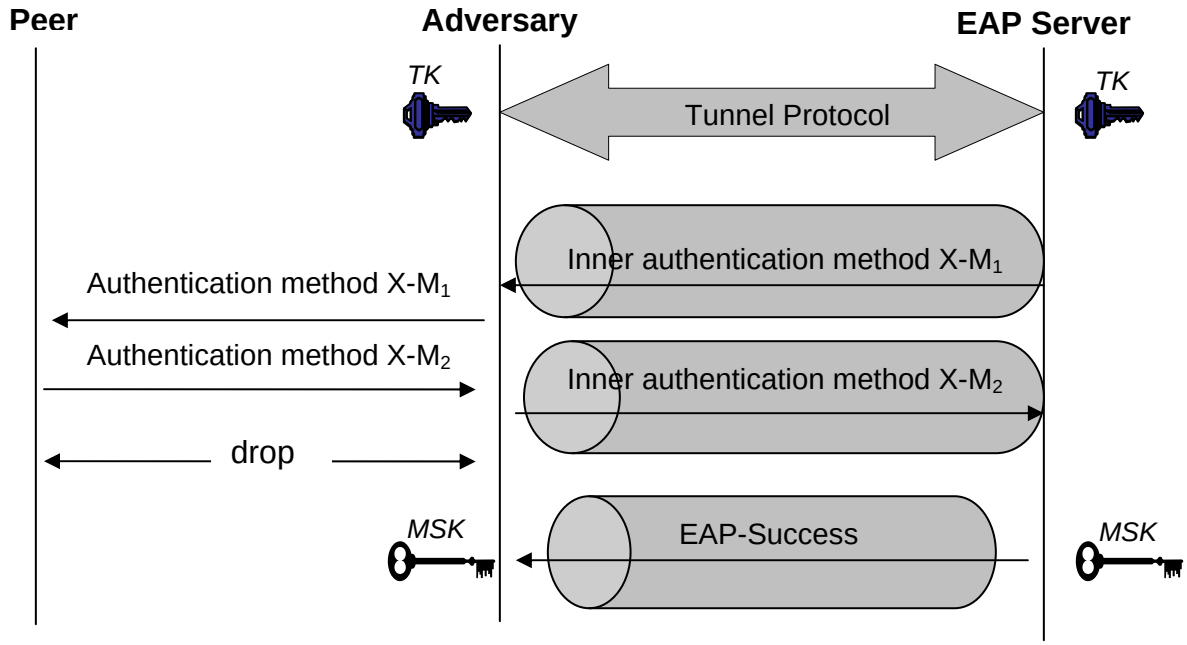

Figure 8: Man-in-the-middle Attack on Tunnel-based EAP methods

The described man-in-the-middle attack can be mitigated if a tunneled EAP method includes a procedure to assure the EAP server that the inner authentication and the tunnel are executed with the same entity. For example, EAP methods could provide such assurance through a combination of *cryptographic bindings* provided by the tunnel-based method and server- and/or peer-enforced system requirements. Cryptographic bindings bind the inner authentication method(s) to the tunnel protocol by computing a compound key *CTK* using the tunnel key *TK* and the derived key INK_i from each inner authentication method *i* as inputs. The compound key is then used to derive further keying material and applied in some subsequent EAP messages to provide assurance to the server and the peer that the tunnel protocol and all inner authentication methods are executed with the same entity. Compound key derivation in tunnel-based EAP methods supporting cryptographic bindings is depicted in Figure 9, where KCK is used to confirm the derivation of CTK.

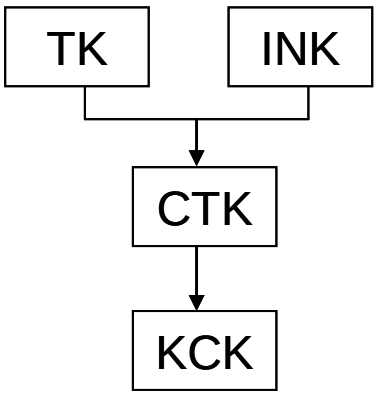

Figure 9: Compound Key Derivation in Cryptographic Bindings with Key Confirmation

It needs to be emphasized that only inner authentication methods with key establishment contribute a non-zero input to the compound key computations, resulting in a non-trivial cryptographic

binding. Conversely, inner methods that do not derive keys do not contribute to the compound key computation (typically a zero string will serve as input here), and the resulting trivial cryptographic bindings do not mitigate the described man-in-the-middle attacks. Furthermore, inner authentication methods with key establishment processes that are vulnerable to attacks when executed outside a tunnel might also lead to insecure cryptographic bindings that can be broken by the adversary, as discussed in [35]. Here, the adversary breaks the key establishment scheme of the authentication method and is then able to compute the compound key *CTK*.

Possible ways for mitigating the man-in-the-middle attack on tunnel-based EAP methods using cryptographic bindings can be summarized as follows:

1 Only permit tunnel-based EAP methods supporting cryptographic bindings where inner authentication methods have key establishment and are not vulnerable to attacks.

2 Only permit tunnel-based EAP methods supporting cryptographic bindings. If inner authentication methods that do not establish keys or are vulnerable to attacks are used, their execution is only permitted within a protective tunnel.

The first mitigation option is the most preferable from a security point of view, because supporting cryptographic bindings plus allowing only inner methods with non-vulnerable key establishment can be enforced by the EAP server. The exclusive use of inner methods with key establishment guarantees non-trivial cryptographic bindings. The second option is less favorable, because the peers must enforce the server policy that certain authentication methods can only be executed within a protective tunnel. Unlike EAP servers, peers might not be aware of their policy configurations and whether they are executing an EAP method inside or outside a protective tunnel. In addition, peers are more vulnerable to attacks that could change their configurations.

For practicability and backward compatibility reasons, the second mitigation option is recommended, because—unlike the first option—this option enables the use of password-based authentication methods within a protective tunnel − one of the original motivations for introducing tunnel-based EAP methods. At the same time, the second option supports cryptographic bindings for inner authentication methods with key establishment, facilitating more secure implementations. The same mitigation option is chosen in the current IETF draft "Requirements for a Tunnel Based EAP Method" [34].

Besides cryptographic binding, there are other ways to assure that the two endpoints of all inner authentications and the tunnel protocol are the same. However, those methods may require changes to the inner authentication schemes and/or the tunnel protocols. Such changes contradict one of the original purposes of implementing tunnel-based EAP methods, namely, enabling the secure use of already deployed authentication methods. From the above discussions about the mitigation of man-in-the-middle attacks, the following requirements for the federal wireless network access authentication apply.

SR-TBEAP-1 The tunnel-based EAP method to be used **shall** provide assurance to the peer and EAP server that all inner authentications and the tunnel protocol are executed with the same entity.

SR-TBEAP-2 Every tunnel-based EAP method supporting cryptographic bindings **shall** provide key confirmation for the derived compound key *CTK*.

Any tunnel-based EAP method **shall** derive the same EAP session keys—such as MSK and EMSK— as a non-tunneled EAP method (see Figure 5). This key hierarchy may be derived from either CTK or TK acting as master key MK. The key derivations and key hierarchy in tunnel-based EAP methods (including the derivations of CTK and KCK in methods with cryptographic bindings) must satisfy the same security requirements as the ones for non-tunneled methods.

SR-TBEAP-3 The key hierarchy and key derivations of a tunnel-based EAP method **shall** satisfy the same requirements as for non-tunneled EAP methods, i.e. all requirements in Section 8.3.1.

A Tunnel-based EAP method may involve negotiations on its tunnel protocol and tunneled authentication methods. The following requirement applies.

SR-TBEAP-4 Each supported tunnel-based EAP method **shall** at least offer one approved tunnel protocol as specified in Section 9.2 and an approved tunneled authentication method as specified in Section 9.3.

For a tunnel-based EAP method, if any of the negotiations must be done before protections are available, then the following requirement applies.

[SR-TBEAP-5] Each supported tunnel-based EAP method supporting negotiations on its tunnel protocol, tunneled authentication methods, and features **should** include post-verification;

A tunnel-based EAP method in pass-through mode is also subject to rogue PoA attacks, as discussed in Section 5.5. As a consequence, tunnel-based EAP methods also rely on channel binding to assure that the information of a PoA received by the peer and the EAP server are consistent. The following requirement applies:

[SR-TBEAP-6] Each tunnel-based EAP method **should** satisfy [SR-CB-1], as specified in Section 8.4.

As mentioned in Section 4.4, it is permitted per [18] that multiple authentication methods are executed within a protective tunnel during an EAP execution. This Recommendation distinguishes between the *concurrent* and *sequential execution* of authentication methods within the tunnel. Concurrent execution means that the execution of an authentication method within a protective tunnel can be initiated at any time, and is independent of all other instances of authentication methods that may be executed in the same tunnel during an EAP execution. Sequential execution means that authentication methods are executed sequentially within a protective tunnel. In that case, the execution of an inner method can only start upon the completion of the previous authentication method. In other words, a new authentication method may be initiated within the protective tunnel upon receiving a Success or Failure message from the previous method. Both, concurrent as well as sequential execution of inner methods, comply with this Recommendation. However, the requirements for compound key computations differ for both scenarios.

[SR-TBEAP-7] When n inner authentication methods are concurrently executed inside one single tunnel, each individual compound key CTK_i **should** be computed upon the completion of each inner method i, i.e. $CTK_i = f(INK_i, TK)$ for $0 < i \leq n$.

[SR-TBEAP-8] When the sequential execution of *n* inner methods is used, a *chained compound key* CTK_i **should** be computed upon the completion of each inner method *i*, such that it contains the compound key of all previous inner methods, i.e. $CTK_i = f(CTK_{i-1}, INK_i)$ with $0 < i \leq n$ and $CTK_0 = TK$.

9.2 Tunnel Protocol

To comply with this Recommendation, any tunnel protocol needs to provide server authentication and establish fresh keying material between the peer and EAP server. The established keying material is used to derive a tunnel key *TK*. The keys, used for encryption and authentication, are also derived from the established keying material or from *TK*.

SR-TP-1 The key establishment process of a tunnel protocol **shall** satisfy all requirements specified in Section 8.3[3].

SR-TP-2 Tunnel protocols **shall** provide unidirectional authentication from the server to the peer. Furthermore, all other requirements specified in Section 8.2 **shall** be satisfied by a tunnel protocol.

SR-TP-3 The integrity and confidentiality message-protection established through the tunnel protocols **shall** satisfy all requirements specified in Section 8.5[4].

Please observe that security requirement SR-TP-2 implies that bidirectional anonymous key establishments—as sometimes used for backward compatibility reasons or in an attempt to provide peer as well as server privacy—do not comply with this Recommendation. For example, anonymous DH-key establishment, as defined in some TLS v1.0 [16] ciphersuites and supported by EAP-FAST [23], PEAP [33] and EAP-TTLSv0 [28], is non-compliant. Please refer to [35] for a detailed description of risks when bidirectional anonymous tunnels are used in tunnel-based EAP methods. Therefore, for tunnel protocols, certain ciphersuites, protocol versions, and features may not be acceptable. If they are negotiable, then the following requirement applies.

SR-TP-4 Each tunnel protocol used in a tunnel-based EAP method **shall** at least offer one approved ciphersuite, i.e. a ciphersuite that only contains cryptographic algorithms complying with this Recommendation.

Besides ciphersuites, in some tunnel protocols used in tunnel-based EAP methods, the protocol versions and features are also negotiable. If the negotiations are conducted before protections are available, then the following requirement applies.

[SR-TP-5] Each tunnel protocol used in a tunnel-based EAP method supporting negotiations on ciphersuites, protocol versions, and features **should** include post-verification.

---

[3] In scenarios in which the tunnel protocol only provides server authentication, the requirement of mutual implicit key authentication is not applicable.

[4] The protections established through the tunnel protocol may not be applied at the EAP layer as described in 8.5. However, the same security requirements apply.

9.2.1 TLS as a Tunnel Protocol

TLS is the de facto standard for establishing a protective tunnel between an EAP peer and the EAP server in tunnel-based EAP methods. For this reason, TLS is briefly reviewed in this section, and guidelines for TLS as a tunnel protocol are introduced. Currently, three TLS versions exist, TLS v1.0 [16], TLS v1.1 [22], and TLS v1.2 [26]. TLS is typically public key-based and employs public key certificates for authentication, but pre-shared key-based TLS implementations do exist. As for EAP methods, the cryptographic algorithms used during the protocol execution are defined in a ciphersuite that is negotiated in the beginning of the protocol execution. TLS ciphersuites have the form

TLS_KE_(AUTH)_WITH _ENC_HASH.

The notations used are explained as follows: KE is the algorithm for key establishment; AUTH is the authentication algorithm, if it is not defined as part of KE; ENC is the encryption algorithm; and HASH is the hash function used to form a message authentication code (MAC) algorithm. The IETF identifies one mandatory-to-implement TLS ciphersuite for each TLS version (i.e. v1.0, v1.1, and v1.2). However, a large number of TLS ciphersuites exist, supporting a variety of key establishment, encryption and message authentication algorithms. Please notice that the pseudo-random function (PRF), used as a building block for key derivation functions, is not defined as a part of TLS ciphersuite. In TLS v1.0 and v1.1, the XOR of the outputs of HMAC-MD5 and HMAC-SHA1 is used as a PRF, while TLS v1.2 specifies HMAC-SHA-256 as the PRF for the existing ciphersuites.

Not all TLS ciphersuites are suitable to meet the requirements for tunnel protocols defined in the previous section, i.e. the requirements from SR-TP-1 to [SR-TP-5]. As a general rule to comply with this Recommendation, approved TLS ciphersuites only consist of approved cryptographic algorithms. However, not all such combinations are necessarily secure. NIST SP 800-57, Part 3 [11] specifies ciphersuites for TLS v1.0, v1.1, and v1.2, approved for federal use. Therefore, in order to comply with this Recommendation, the following requirement applies to TLS ciphersuites.

SR-TLS-1 If TLS v1.0, v1.1. or v1.2 is used as the tunnel protocol in a tunnel-based EAP method, at least one approved TLS ciphersuite, i.e. a TLS ciphersuite that is listed in NIST SP 800-57, Part 3 [11], **shall** be supported.

Note that requirement SR-TLS-1 replaces requirements on tunnel protocols from SR-TP-1 to [SR-TP-5] whenever TLS is used as the tunnel protocol in a tunnel-based EAP method.

9.3 Tunneled Authentication Method

If the tunnel protocol is secure (i.e. SR-TP-I, I = 1, 2, 3, 4 and [SR-TP-5] are satisfied)[5], the security requirements for tunneled authentication methods can be relaxed to the following.

[5] If the tunnel protocol is not secure (i.e. one of the requirements among SR-TP-I. I =1, 2, 3, 4 and [SR-TP-5] is not satisfied), man-in-the-middle attacks and other attacks on the inner authentication method(s) may be feasible. In that case, all inner authentication methods **shall** be in compliance with the same security requirements as non-tunneled EAP methods (see Section 8). Note that in this scenario, the tunnel protocol does not add any security to the EAP method and is, in fact, redundant.

SR-TEAP-1 Any inner authentication method **shall** provide unidirectional authentication from the peer to the EAP server.

It can be observed that the security requirement for tunneled authentication methods is significantly relaxed, compared to the requirements for non-tunneled EAP methods.

To mitigate the man-in-the-middle-attack illustrated in Figure 8 when supporting the use of legacy password-based authentication methods that do not derive keys, the following peer-enforced server policy is required.

SR-TEAP-2 Every authentication method that does not establish keys or is vulnerable to attacks **shall** only be executed as an inner authentication method within tunnel-based EAP methods. In other words, such authentication methods **shall not** be executed as an autonomous authentication method outside a protective tunnel.

Under certain conditions, a sequential execution of multiple authentication methods enables the secure use of methods that are vulnerable to attacks without system requirement SR-TEAP-2. This can be done using the chained compound keys described in [SR-TBEAP-8], as long as at least one of the inner authentication methods provides key establishment and resists attacks when executed outside a protective tunnel. To ensure that at least one of the authentication methods provides key establishment, the EAP server and peer could first negotiate the methods, which will be executed sequentially inside the tunnel, the tunnel-based EAP method could abort if none of the inner methods derived keying material.

10. Summary

EAP is widely deployed to secure a growing number of wireless mobile applications. In such applications, a mobile station attempts to access a network over a wireless link. Hence, the security of EAP methods used to secure wireless mobile applications is of paramount importance. This Recommendation summarizes the security requirements that **shall** or **should** be met by implemented EAP methods, as well as by systems implementing such protocols.

It is strongly encouraged that all cryptographic algorithms employed in an EAP method (including authentication algorithms, key establishment algorithms, key derivation functions, MACs, public and symmetric encryption schemes, as well as digital signature schemes) are in compliance with existing FIPS publications and NIST Special Publications (e.g. SP 800-38B, SP 800-56A, SP 800-57, SP 800-108, FIPS 186-3, FIPS 196, and FIPS 198). In addition to the recommendations for the security strength of the cryptographic algorithms and associated keys, this Recommendation includes requirements for authentication and key exchange protocols that are aimed at preventing common attacks on such protocols. Furthermore, requirements for derived keying material and relations among keys are discussed in detail.

Only ciphersuites that meet all requirements in this Recommendation are acceptable in the federal applications under consideration. This prevents downgrading attacks, as well as cryptographic attacks on the authentication, key establishment and message protections.

This Recommendation distinguishes between non-tunneled and tunnel-based EAP methods and specifies the necessary requirements to allow backward compatibility. Attacks, which are feasible if such requirements are not met, are outlined.

In addition to the requirements that should be met by any supported EAP method, this Recommendation specifies necessary system pre-requisites that **should** be met by all systems supporting EAP for wireless mobile access control.

Annex A discusses how this Recommendation may be used to check the compliance of EAP methods with the security requirements listed in this document.

Annex A: Discussion of Selected EAP Methods

This section discusses how this Recommendation may be used to check the security compliance of EAP methods when used in Federal wireless applications. Compliance checks are provided for a selection of well-known methods (namely EAP-GPSK [32], EAP-TLS [25], EAP-TTLSv0 [28], EAP-FAST [23], and PEAP [33], representing symmetric key-based, public key-based and tunnel-based EAP methods). However, this selection is purely for illustrative purpose and does not imply that the selected methods are approved for federal use.

In order to check compliance with this Recommendation, the security requirements are checked for the approved ciphersuites among the specified by the considered EAP method. The results of the provided compliance checks are summarized in Table 2 for EAP-GPSK, Table 3 for EAP-TLS, Table 4 for EAP-FAST, Table 5 for EAP-TTLS, and Table 6 for PEAP, respectively. All "**shall**" and "**should**" requirements derived in this document are listed in the rows, where some or all ciphersuites of the respective EAP method are represented in the columns. The notation used in these tables is summarized in Table 1.

Requirement	Satisfied	Not satisfied	Not applicable[6]
SHALL	✓	×	N/A
SHOULD	✓	○	N/A

Table 1: Notations for Compliance Checks

Only when a considered EAP method using a specific ciphersuite satisfies all "**shall**" requirements will the EAP method and the given ciphersuite be in compliance with this Recommendation, and considered safe to use in a Federal wireless application.

Please recall that pre-conditions are EAP method independent and must be checked separately for any system that supports EAP for access control.

A.1 EAP-GPSK

The EAP Generalized Pre-Shared Key (EAP-GPSK) method is specified in RFC 5433 [32] of the IETF EMU working group (EAP Method Update). EAP-GPSK specifies an EAP method based on pre-shared keys and employs secret key-based cryptographic algorithms. Hence, this method is efficient in terms of message flows and computational costs, but requires the existence of pre-shared keys between each peer and EAP server. The set up of these pairwise secret keys is part of the peer registration, and thus, must satisfy the system pre-conditions.

During an EAP-GPSK execution, a peer and server exchange nonces that are used together with the pre-shared key to derive the EAP key hierarchy. Hence, the security of the key establishment

[6] Not applicable (N/A) as check result indicates that a conditional requirement does not apply to a particular EAP method and/or ciphersuite. For example, requirements on digital signatures do not apply to purely symmetric key-based schemes.

depends on the key derivation function (KDF) used and the randomness of the exchanged nonces. The EAP-GPSK RFC specifies two ciphersuites (referred to as CS-GPSK1 and CS-GPSK2 in the remainder of this discussion) of the form CS={ENC, MAC, KDF}. CS-GPSK1 is mandatory-to-implement and defined as CS-GPSK1={AES-CBC-128, AES-CMAC-128, GKDF}; the second ciphersuite is defined as CS-GPSK2={NULL, HMAC-SHA256, GKDF}. "Null" indicates that ciphersuite 2 does not provide encryption and, thus, does not enable confidential communications. GKDF is defined in [32] and utilizes the MAC function specified in the ciphersuite, i.e. KDF=AES-CMAC-128 for CS-GPSK1 and HMAC-SHA256 for CS-GPSK2, respectively.

Table 2 summarizes the compliance check of the two ciphersuites supported by EAP-GPSK. It can be observed that the current version of EAP-GPSK meets all "**shall**" requirements and is, thus, in compliance with this Recommendation.

Security Requirement	**Compliance**	
SR-CN-1	✓	
	CS-GPSK1 **AES-CBC-128/AES-CMAC-128/AES-CMAC-128**	**CS-GPSK2** **-/HMAC-SHA-256/HMAC-SHA-256**
[SR-CN-2]	✓	✓
SR-AUTH-1	✓	✓
SR-AUTH-2	✓	✓
SR-AUTH-3	✓	✓
SR-AUTH-4	N/A	N/A
SR-AUTH-5	✓	✓
SR-AUTH-6	✓	✓
SR-KE-1	✓	✓
SR-KE-2	N/A	N/A
SR-KE-3	✓	✓
SR-KE-4	✓	✓
SR-KE-5	✓	✓
[SR-KE-6]	✓	✓

[SR-KE-7]	✓	✓
[SR-KE-8]	N/A	N/A
SR-KD-1	✓	✓
SR-KD-2	✓	✓
[SR-CB-1]	✓	✓
SR-MP-1	✓	✓
SR-MP-2	✓	✓[7]
SR-MP-3	✓	✓
SR-MP-4	✓	✓
SR-MP-5	✓	✓
SR-MP-6	✓	N/A
SR-MP-7	N/A	N/A

Table 2: EAP-GPSK Compliance Check

A.2 EAP-TLS

EAP-TLS [25] defines how the TLS protocol can be encapsulated in EAP messages. Per [25], every EAP-TLS implementation must support TLS v1.0 [16] and may support TLS v1.1 [22] and TLS v1.2 [26], as well as later versions that might be published in the future. Implementations may support several of the numerous existing TLS ciphersuites. Ciphersuites in EAP-TLS are of the format CS=TLS_KE_(AUTH)_WITH_ENC_HASH, i.e. each suite specifies key establishment, authentication, encryption and integrity-protection algorithms. Please note that HASH defines the MAC for integrity-protection (i.e. using HMAC). The MAC in EAP-TLS is HMAC with the hash function defined in each ciphersuite. The PRF is either an exclusive or of HMAC-MD5 and HMAC-SHA-1 as in version 1.0 and 1.1 or HMAC SHA-256 as in version 1.2 (see Section 9.2.1).

EAP-TLS supports options for mutual authentication and server authentication. Only options supporting mutual authentication comply with this Recommendation (see SR-AUTH-1). Note that mutual authentication in EAP-TLS requires peer certificates.

EAP-TLS may support peer privacy, which requires that the username is not transmitted in cleartext (instead, a privacy NAI is used), and the peer certificate is sent confidentially (i.e. in the tunnel).

[7] With this ciphersuite, EAP-GPSK does not allow exchanging confidential information.

This technique is in compliance with this Recommendation, as long as all security requirements are met.

EAP-TLS defines one ciphersuite that is mandatory-to-implement by both EAP servers and EAP peers:

CS-TLS-1. TLS_RSA_WITH_3DES_EDE_CBC_ SHA.

In addition, EAP-TLS recommends implementing the following ciphersuites on EAP servers and EAP peers:

CS-TLS-2. TLS_RSA_WITH_RC4-128_SHA
CS-TLS-3. TLS_RSA_WITH_AES-128-CBC_SHA.

EAP-TLS also recommends EAP servers to implement:

CS-TLS-4. TLS_RSA_WITH_RC4-128_MD5

This Recommendation follows the guidelines of NIST SP 800-57, Part 3 for ciphersuites defined in TLS v1.0, TLS v.1.1 and TLS v.1.2. Hence, only CS-TLS-1 and CS-TLS-3 comply with this Recommendation, while CS-TLS-2 and CS-TLS-4 do not comply with this Recommendation.

For the compliance checks in this section, the use of CS-TLS-1, CS-TLS-3 or another ciphersuite in compliance with this Recommendation is assumed. The results of the EAP-TLS compliance check (according to the security requirements specified in this Recommendation) are summarized in Table 3. It can be observed that EAP-TLS used with a compliant ciphersuite meets all "**shall**" requirements and is, thus, in compliance with this Recommendation.

Security Requirement	**Compliance**
SR-CN-1	✓
[SR-CN-2]	✓[8]
SR-AUTH-1	✓
SR-AUTH-2	✓
SR-AUTH-3	✓
SR-AUTH-4	✓
SR-AUTH-5	✓
SR-AUTH-6	✓

[8] The post-verification on ciphersuite negotiation is provided through TLS.

SR-KE-1	✓
SR-KE-2	✓
SR-KE-3	✓
SR-KE-4	✓
SR-KE-5	✓
[SR-KE-6]	✓
[SR-KE-7]	✓
[SR-KE-8]	✓
SR-KD-1	✓
SR-KD-2	✓
[SR-CB-1]	○[9]
SR-MP-1	✓
SR-MP-2	✓
SR-MP-3	✓
SR-MP-4	✓
SR-MP-5	✓
SR-MP-6	✓
SR-MP-7	N/A[10]

Table 3: EAP-TLS Compliance Check

A.3 EAP-FAST

EAP-FAST [23] is a tunnel-based EAP method (see Section 9) that extends EAP-TLS such that mutual authentication can be provided without requiring peer certificates. In particular, EAP-FAST employs TLS to establish a protective tunnel and legacy peer authentication protocols, and other authentication protocols can be executed within the tunnel.

[9] EAP-TLS does not support the exchange of additional payload in its EAP messages.

[10] Some newly specified ciphersuites for TLS (see [29] support authenticated encryption such as GCM [7].

EAP-FAST offers peer privacy as a special feature, in which case, peer identifiers are only submitted within the tunnel. This type of privacy does not violate the security requirements in this Recommendation, as long as the peer subsequently authenticates within the tunnel.

EAP-FAST offers cryptographic binding and method chaining.

EAP-FAST supports TLS v1.0, v1.1 and any later versions. The following TLS ciphersuites are mandatory-to-implement in any EAP-FAST implementation:

CS-FAST-1. TLS_RSA_WITH_RC4_128_SHA

CS-FAST-2. TLS_RSA_WITH_AES_128_CBC_SHA

CS-FAST-3. TLS_DHE_RSA_WITH_AES_128_CBC_SHA

To comply with this Recommendation, the TLS ciphersuites used to establish the tunnel **shall** be listed in NIST SP 800-57, Part 3. Hence, the mandatory-to-implement ciphersuites CS-FAST-2 and CS-FAST-3 are in compliance with this Recommendation.

If the requirements for the tunnel protocol are met, EAP-FAST can be used with any EAP or other authentication method that provides peer authentication under the system condition that this authentication method can only be executed within a protective tunnel (see SR-TEAP-2).

Table 4 summarizes results of the compliance checks of EAP-FAST. It can be observed that EAP-FAST meets all the security requirements as a tunnel-based EAP method when used with a recommended choice of TLS ciphersuites and compliant inner authentication methods. The compliance of the inner authentication methods cannot be generally checked for EAP-FAST, because EAP-FAST supports any type of authentication method as an inner method. Note that SR-TEAP-1 must be met by every inner authentication method, and SR-TEAP-2 is a system requirement.

Security Requirement	**Compliance**
SR-TBEAP-1	✓
SR-TBEAP-2	✓
SR-TBEAP-3	✓[11]
SR-TBEAP-4	✓
[SR-TBEAP-5]	✓
[SR-TBEAP-6]	✓
[SR-TBEAP-7]	N/A[12]

[11] This requirement consists of a set of "**shall**" and "**should**" requirements. EAP-Fast is compliant (as indicated here) because it meets all "**shall**" requirements.

[SR-TBEAP-8]	✓
SR-TLS-1	✓

Table 4: EAP-FAST Compliance Check

A.4 EAP-TTLSv0

EAP-TTLSv0 [28] is a tunnel-based EAP method (see Section 9) that extends EAP-TLS such that mutual authentication can be provided without requiring peer certificates. Therefore, EAP-TTLSv0 employs TLS to establish a protective tunnel, and an authentication method is executed within the tunnel. EAP-TTLSv0 supports TLS v1.0, v1.1, v1.2 and potential later versions. No mandatory-to-implement ciphersuites are defined in EAP-TTLSv0. However, the mandatory-to-implement ciphersuites in the negotiated TLS version (i.e. TLS v1.0, v1.1, or v1.2) apply to the EAP-TTLSv0 tunnel protocol.

EAP-TTLSv0 provides peer privacy, because peer identifiers are only submitted within the tunnel. However, server identifiers are exchanged as part of the tunnel protocol to be authenticated. Hence, EAP-TTLSv0's privacy feature is in compliance with this Recommendation.

EAP-TTLSv0 does not support cryptographic bindings or method chaining[13].

Table 5 summarizes the compliance check for EAP-TTLSv0. The tunnel protocol meets all requirements when used with a recommended choice of TLS ciphersuites, while the compliance of the inner authentication methods cannot be generally checked, because EAP-TTLSv0 supports any type of inner authentication method. In summary, this version of EAP-TTLS does not comply with this Recommendation, because it does not mitigate the man-in-the-middle attacks on the tunnel described in 9.1 (SR-TBEAP-1).

Security Requirement	**Compliance**
SR-TBEAP-1	✗
SR-TBEAP-2	N/A
SR-TBEAP-3	✓[14]
SR-TBEAP-4	✓

[12] EAP-FAST does not support the concurrent execution of multiple inner authentications inside the TLS tunnel.

[13] Currently, there are attempts within the IETF to add cryptographic bindings, method chaining and other additional features to EAP-TTLSv0. For example, these extensions can be found in the expired personal draft from S. Hanna and P. Funk, "Key Agility Extensions for EAP-TTLSv0", <draft-hanna-eap-ttls-agility-00.txt>, expired March 2008.

[14] This requirement consists of a set of "**shall**" and "**should**" requirements. EAP-TTLSv0 is compliant (as indicated here) because it meets all "**shall**" requirements.

[SR-TBEAP-5]	✓
[SR-TBEAP-6]	✓
[SR-TBEAP-7]	N/A[15]
[SR-TBEAP-8]	N/A[16]
SR-TLS-1	✓

Table 5: EAP-TTLSv0 Compliance Check

A.5 PEAP

Protected Extensible Authentication Protocol (PEAP)[33] is a tunnel-based EAP method (see Section 9) that extends EAP-TLS such that mutual authentication can be provided without requiring peer certificates. In particular, PEAP employs TLS to establish a protective tunnel, and any EAP or other authentication method can be executed within the tunnel.

PEAP offers peer privacy as a special feature, in which case, peer identifiers are only submitted within the tunnel. This type of privacy does not violate the security requirements in this Recommendation, as long as the peer subsequently authenticates within the tunnel.

PEAP offers cryptographic binding. However, it does not support the executions of multiple methods in the tunnel, neither sequentially nor concurrently.

PEAP supports TLS v1.0, in which the mandatory-to-implement ciphersuite is TLS_DHE_DSS_WITH_3DES_EDE_SHA, while PEAP specifies additional mandatory-to-implement ciphersuites TLS_RSA_WITH_RC4_128_MD5 and TLS_RSA_WITH_RC4_128_SHA. Combining these together, the following TLS ciphersuites are mandatory-to-implement in any PEAP implementation:

CS-PEAP-1. TLS_RSA_WITH_RC4_128_MD5

CS-PEAP-2. TLS_RSA_WITH_RC4_128_SHA

CS-PEAP-3. TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

To comply with this Recommendation, the TLS ciphersuites used to establish the tunnel **shall** be listed in NIST SP 800-57, Part 3. Hence, only the mandatory-to-implement ciphersuite CS-PEAP-3 is in compliance with this Recommendation.

[15] EAP-TTLSv0 does not support the concurrent execution of multiple inner authentication methods inside the TLS tunnel.

[16] EAP-TTLSv0 does not support the sequential execution of multiple inner authentications methods inside the TLS tunnel.

If and only if the requirements for the tunnel protocol are met, PEAP can be used with any EAP or other authentication method that provides peer authentication under the system condition that this authentication method can only be used in combination with a tunnel protocol (see SR-TEAP-2).

Table 6 summarizes results of the compliance checks of PEAP. It can be observed that PEAP meets all the security requirements as a tunnel-based EAP method when used with a recommended choice of TLS ciphersuite and a compliant inner authentication method. The compliance of the inner authentication methods cannot be generally checked for PEAP, because PEAP supports any type of authentication method as an inner method. Note that SR-TEAP-1 must be met by every inner authentication method, and SR-TEAP-2 is a system requirement.

Security Requirement	**Compliance**
SR-TBEAP-1	✓
SR-TBEAP-2	✓
SR-TBEAP-3	✓[17]
SR-TBEAP-4	✓
[SR-TBEAP-5]	✓
[SR-TBEAP-6]	✓
[SR-TBEAP-7]	N/A[18]
[SR-TBEAP-8]	N/A[19]
SR-TLS-1	✓

Table 6: PEAP Compliance Check

[17] This requirement consists of a set of "**shall**" and "**should**" requirements. PEAP is compliant (as indicated here) because it meets all "**shall**" requirements.

[18] PEAP does not support the concurrent execution of multiple inner authentications inside the TLS tunnel.

[19] PEAP does not support the sequential execution of multiple inner authentication methods inside the TLS tunnel.

Annex B: References (Informative)

[1] FIPS 180-3, “Secure Hash Standard”, Federal Information Processing Standards Publication, October 2008.

[2] FIPS 186-3, “Digital Signature Standard (DSS)”, Federal Information Processing Standards Publication, June 2009.

[3] FIPS 197, “Specification for the ADVANCED ENCRYPTION STANDARD (AES)”, Federal Information Processing Standards Publication, November 2001.

[4] FIPS 198-1, “The Keyed-Hash Message Authentication Code (HMAC)”, Federal Information Processing Standards Publication, July 2008.

[5] NIST SP 800-38B, “Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication”, NIST Special Publication, May 2005.

[6] NIST SP 800-38C, “Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality”, NIST Special Publication, May 2004.

[7] NIST SP 800-38D, “Recommendation for Block Cipher Modes of Operation: The Galois/Counter Mode (GCM) and GMAC ”, NIST Special Publication, November 2007.

[8] NIST SP 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”, NIST Special Publication, March 2007.

[9] NIST SP 800-56B, “DRAFT Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography”, NIST Special Publication, December 2008.

[10] NIST SP 800-57, “Recommendation for Key Management – Part 1: General (Revised)”, NIST Special Publication, March 2007

[11] NIST SP 800-57, Part 3 (DRAFT), “Recommendation for Key Management, Part 3 Application-Specific Key Management Guidance”, NIST Special Publication, October 2008.

[12] NIST SP 800-67, “Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher”, NIST Special Publication, revised March 2008.

[13] NIST SP 800-108, "Recommendation for Key Derivation Using Pseudorandom Functions", Recommendations of the National Institute of Standards and Technology, NIST Special Publication, November 2008.

[14] IEEE Standard 802.11-2007, Institute of Electrical and Electronics Engineers, "Standard for Local and metropolitan area networks - specific requirements - part 11: Wireless LAN Medium Access Control and Physical Layer specifications", 2007.

[15] IEEE Standard 801.1X-2004, Institute of Electrical and Electronics Engineers, “Standard for Local and metropolitan area networks, Port-Based Network Access Control”, 2004.

[16] RFC 2246, IETF Request for Comments, “The TLS Protocol Version 1.0”, T. Dierks, and C. Allen, January 1999, (obsoleted by RFC 4346 and RFC 5246).

[17] RFC 3579, IETF Request for Comments, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", B. Aboba and P. Calhoun, September 2003.

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

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

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

[21] RFC 4187, IETF Request for Comments, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", J. Arkko and H. Haverinen, January 2006.
[22] RFC 4346, IETF Request for Comments, "The Transport Layer Security (TLS) Protocol Version 1.1", T. Dierks and E. Rescorla, April 2006, (obsoletes RFC 2246 and obsoleted by RFC 5246).
[23] RFC 4851, IETF Request for Comments, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou, May 2007
[24] RFC 4962, IETF Request for Comments, "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management", R. Housley and B. Aboba, July 2007.
[25] RFC 5216, IETF Request for Comments, "The EAP TLS Authentication Protocol", D. Simon, B. Aboba, and Hurst R., March 2008. (obsoletes RFC 2716).
[26] RFC 5246, IETF Request for Comments, "The Transport Layer Security (TLS) Protocol Version 1.2", T. Dierks and E. Rescorla, August 2008. (obsoletes 3268, 4346, 4366, updates: 4492).
[27] RFC 5247, IETF Request for Comments, "Extensible Authentication Protocol (EAP) Key Management Framework". Bernard Aboba, Dan Simon, and P. Eronen, August 2008.
[28] RFC 5281, IETF Request for Comments, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", P. Funk and S. Blake-Wilson, August 2008.
[29] RFC 5288, IETF Request for Comments, "AES Galois Counter Mode (GCM) Cipher Suites for TLS" J. Salowey, A. Choudhury, and D. McGrew, August 2008.
[30] RFC 5295, IETF Request for Comments, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", J. Salowey, L. Dondeti, V. Narayanan, M. Nakhjiri, August 2008.
[31] RFC 5296, IETF Request for Comments, "EAP Extensions for EAP Re-authentication Protocol (ERP), V. Narayanan and L. Dondeti, August 2008.
[32] RFC 5433, IETF Request for Comments, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK)", T. Clancy and H. Tschofenig, February 2009.
[33] Microsoft White Paper "Protected Extensible Authentication Protocol (PEAP) Specification", http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS-PEAP%5D.pdf
[34] IETF Internet Draft, "Requirements for a Tunnel Based EAP Method", K. Hoeper, S. Hanna, H. Zhou and J. Salowey (Ed.), work in progress, <draft-ietf-emu-eaptunnel-req-03.txt>, July 2009
[35] K. Hoeper and L. Chen, "Where EAP Security Claims Fail", Qshine 2007, The 4th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, August 14-17, 2007, Vancouver, Canada.
[36] T. C. Clancy and K. Hoeper, "Making the case for EAP channel bindings", In proceedings of IEEE Sarnoff Symposium 2009, pp. 1-5, April 2009.
[37] Asokan, N., Niemi, V. and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", IACR ePrint Archive Report 2002/163, October 2002, <http://eprint.iacr.org/2002/163>.