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

December 2008

U.S. Department of Commerce

Carlos M. Gutierrez, 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.

Acknowledgements

The authors, Katrin Hoeper and Lily Chen of the National Institute of Standards and Technology (NIST), 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.

EAP access authentication and key establishment with an authentication server operated by a Federal agency over a wireless communication link must adhere to the requirements in this Recommendation in order to be validated under the CMVP. The requirements of this Recommendation are indicated by the word “shall.”

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 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) [1] is a framework for access authentication which supports different authentication methods that are specified in EAP methods. In addition to authentication, some new EAP methods derive keys that can be used to protect the wireless link. 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. For example, IEEE 802.11 [2] for Wireless Local Area Networks (WLANs) and IEEE 802.16 [3] for Wireless Metropolitan Area Networks (WMANs) both use IEEE 802.1X as a way to encapsulate EAP messages over LAN (EAPOL) for providing access authentication and key establishment. While the EAP methods defined in [1] did neither support mutual authentication nor key derivation, the use of EAP with wireless technologies has resulted in development of a new set of requirements. As described in RFC 4017 [19], it is desirable for EAP methods used for wireless LAN authentication to support mutual authentication and key derivation. Guidelines on security solutions of IEEE 802.11i are specified in SP 800-97 [20]. Other link layers can also make use of EAP to enable mutual authentication and key derivation.

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. Today dial-up Internet service is comparatively rare, but wired broadcast 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, naïve implementations of early EAP methods 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 all its supported cryptographic algorithms and schemes are crucial for securely launching wireless applications and providing mobility services. 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.

In a wireless or mobile application there are usually three players, a mobile terminal such as a laptop computer (called a “peer” by EAP), a wireless point of access (PoA) (called an

"authenticator" by EAP) and a back end EAP Authentication Server (AS). This is illustrated in Figure 1. This recommendation is applicable to EAP Authentication Servers operated by or for Federal agencies and Federal mobile terminals when they are used with the Federal servers. Federal users may well 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 nonfederal 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.

Users of internal agency intranets that employ these EAP methods with appropriate air interface encryption may consider that their connection is as secure as a wired connection to the Intranet and may reasonably make the same security assumptions that they do when using a wired terminal to access other services on the agency intranet. Similarly, agencies may treat such mobile terminals as they do other stations on the intranet. However, when mobile users log onto nonfederal points of access, they should assume that after a successful EAP execution, the air interface may be still unprotected and they are subject to attack. In these case mobile users may either restrict their use of the network to avoid exposing sensitive information, or establish end-to-end protection by using methods such as virtual private networks and protocols such as the Transport Layer Security (TLS) [10] protocol.

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).
Point of Attachment (PoA)	A device that connects wireless stations to (usually wired) networks, and to each other, through wireless links. In this Recommendation, 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 or a base station in IEEE 802.16 networks.

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.
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.
Authentication Server (AS)	A network server that executes access authentication operations. The AS is generally located in the protected (wired) network. When EAP is used as an access authentication protocol, then the authentication server is called an EAP server.
Authenticator	A network entity that executes access authentication protocols with the entity that requests access to a network. Unlike the authentication server, the authenticator is typically not located in the protected (wired) network. An authenticator may use an authentication server to conduct authentication operations. In this Recommendation, the authenticator is typically co-located with the 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 [6] and DIAMETER [7]. (See the definition of Authorization.)
Authentication Method	A cryptographic scheme used by an entity to prove its identity to another entity. For example, by proving the knowledge of some secret information, such as a secret or private key or by proving 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 authentication server negotiate the ciphersuite that they will use in the remainder of the current EAP execution. (See the definition of peer).

Cryptographic binding	An EAP tunnel-based specific procedure that binds the tunnel protocol and the tunneled authentication method(s) that is executed within the protective tunnel together. In this Recommendation, cryptographic binding is a procedure to use a key derived from the tunnel key and key(s) from the tunneled authentication method(s) for integrity protection.
Extensible Authentication Protocol (EAP)	An authentication framework defined in IETF RFC 3748. The Extensible Authentication Protocol can support different authentication methods.
EAP execution	A protocol executed between a peer, an authenticator and the authentication server that starts with an EAP-Request/Identity message, and terminates with an EAP-Success/Failure message. An EAP execution consists of a single or a sequence of EAP authentications mechanisms.
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 other cryptographic schemes.
EAP layer	A virtual network layer to carry EAP data frames. It is defined relatively to the lower layers. (See the definition of a lower layer.)
Ephemeral keys	Cryptographic keying material that is derived and used during an EAP method execution and erased upon the termination of the EAP method.
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 authentication server. The key is not shared with any other entities and per RFC 3748 reserved for future use. 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 may be a functional unit to execute certain processes, e.g. an authenticator relaying messages between peer and AS.
Entity authentication	A procedure to obtain assurance of the claimed identity of an entity by another entity. Entity authentications may be unidirectional or mutual. (See the definition of mutual authentication).
Full authentication	A synonym for access authentication. Full authentication implies an authentication with a backend and/or remote authentication server. Contrast with local authentication. (See the definition of local authentication.)

Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions 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 [8]. In EAP hash functions are sometimes used in digital signatures and to build key derivation functions and message authentication codes.
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 derivation	The process that derives keys from another key or from the shared secret of a key agreement scheme.
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, the EAP key establishment is executed between a peer and the authentication server to derive EAP keying material.
Key export	A mechanism by which a key is delivered from an EAP layer to a lower layer [1].
Key hierarchy	A tree structure to represent the relationship of different keys. In a key hierarchy, a node represents a key used to derive the keys used 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.

Key transport	A procedure to deliver a key, e.g. one that has been derived as the result of a key establishment procedure, to another entity. In this Recommendation, key transport refers to key delivery from the authentication server to another entity, for example, to an authenticator.
Local authentication	An access authentication that does not require contacting a remote authentication server, e.g. because it does not use long term credentials that are stored on a backend server. In this Recommendation, local authentication refers to authentications of local peers carried out by the authenticator without contacting the authentication server.
Long term credentials	Data used for access authentication to prove 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 authentication server. In this Recommendation, the long term credentials used for full authentications are distinguishable from the short term credentials used for local authentications.
Lower layer	A virtual network layer that is defined relative to the EAP layer. The lower layers may include the layers where 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. Unlike the EMSK, the MSK is not exclusively known to the communication endpoints and may be, e.g., exported to an authenticator. The master session key may be derived from the master key or may equal 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 both procedures need to be 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 entity that requests network access authenticates to the authentication server, and 2) the authentication server also authenticates to the entity.

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, peer refers to the device attempting to access the network.
Protected communication	In this Recommendation, the provision of confidentiality and/or authenticity for the communicated information. Authenticity includes information integrity and source authenticity.
Session key	A cryptographic key that is used in a specific session. In this Recommendation, a session key is used to provide either confidentiality via a specific encryption algorithm or integrity protection via a specific message authentication code during the period of one EAP execution.
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	A key derived from MK and used to derive further keying materials to protect EAP exchanges.
Trusted third party	A party trusted by all the other involved parties. In this Recommendation, the trusted third party is a certificate authority that issues certificates when public/private key pairs are used as long term credentials.
Tunnel-based EAP method	An EAP method in which first a protective tunnel is established by executing a tunnel protocol between a peer and the authentication server, and then executing one or more other EAP methods within the established protective tunnel. Examples of tunnel-based EAP methods are PEAP [14], EAP-TTLS [11], and EAP-FAST [15]. (See the definition of tunnel protocol).
Tunneled EAP method	An EAP 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 authentication server as a result of a successful tunnel protocol execution. (See the definition of tunnel protocol).

Tunnel protocol	A protocol, e.g. TLS [10], 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 protective tunnel.

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
AS	Authentication Server
TCK	Transient Cipher (encryption) Key
DH	Diffie-Hellman (an key establishment algorithm)
EAP	Extensible Authentication Protocol
EMSK	Extended Master Session Key
GPSK	Generalized Pre-Shared Key (see [9])
TIK	(Transient) Integrity (protection) Key
KDF	Key Derivation Function
MAC	Message Authentication Code
MK	Master Key
MSK	Master Session Key
PMK	Pre-Master Key
PoA	Point of Attachment
PRF	PseudoRandom Function
RADIUS	Remote Authentication Dial In User Service
RSA	Rivest, Shamir, Adleman (an algorithm)
SK	Session Key

SR-XX-i	Security Requirement number i for subroutine XX, with XX: 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 EAP 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 definition SR-XX-i for a list of subroutines XX).
TEK	Transient EAP Key
TK	Tunnel Key
TLS	Transport Layer Security (see [10])
TTLS	Tunneled Transport Layer Security (see [11])

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.

4.1 EAP Communication Links and Involved Parties

As specified in [1], EAP is a framework for two party authentication protocols that are executed between a peer and an authentication server (AS). The latter is sometimes referred to as "EAP server". The peer and AS typically cannot communicate directly with each other, in which case 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). 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 be a wireless link. The second communication link (referred to as CL2) is a wired link in the network between the authenticator and the authentication server[1]. In this Recommendation security requirements for federal peers and federal authentication servers are presented to ensure the secure use of EAP for wireless network access authentication and key establishment. Since the authenticator only acts as a pass-through device, no security requirements for authenticators are necessary and it is of no importance whether the authenticator is a federal or non-federal device. Please note that attacks by rogue authenticators as well as by rogue peers and authentication servers are considered in the list of potential threats in Section 5.

Note that the server has access to an AAA server as well as a local database (DB), sometimes co-located with the server. The AAA server stores all required credentials for the AAA connection between authenticator and AS, where the database stores the long-term authentication credentials of the peers, as well as additional information about authenticators, roaming agreements, network policies and other network information. The EAP parties and communication links are illustrated in Figure 1.

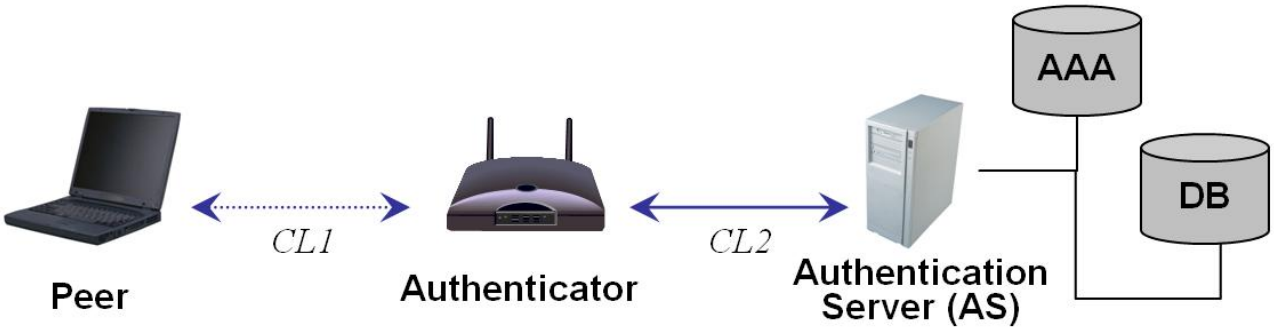

Figure 1: Three Party EAP Communication Model

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.

[1] Note that in some networks and especially in roaming scenarios additional message forwarding entities—such as proxies and routers— may be located between authenticator and AS. For the same reason as for authenticators, this Recommendation does not specify security requirements for these intermediary entities and it is irrelevant whether they are federal or non-federal entities. In the remainder of this document, such entities are not explicitly mentioned again but the security considerations and solutions discussed for rogue authenticators can be extended to these entities.

- **Authenticator:** A network entity that controls access to a network. The authenticator can be accessed by peers via a wireless interface, whereas the authenticator has a wired connection to the protected network. If the authenticator supports local authentication methods and recognizes the peer as a local client, the authenticator can conduct a local authentication. This local authentication does not require contacting the authentication server and might not utilize an EAP method. The remainder of this document focuses on EAP executions in which such local authentications are infeasible. In those cases an authenticator passes the EAP packets between the peer and the authentication server. This is called *pass-through mode*, and the authentication server informs the authenticator of the authentication outcome. Based on this result, the authenticator grants or denies 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.
- **Authentication Server (AS):** A backend server (sometimes referred to as an EAP server) that performs peer authentications and peer authorization checks. If the EAP server does not store authentication credentials, it needs to access an AAA server (e.g. RADIUS [6] or Diameter [7]) that stores the credentials. Furthermore, the authentication server might need to access an external database that stores the policies defining when a peer is authorized to access the network. The EAP server communicates with peers through authenticators acting as pass-through devices and informs the participating authenticators of 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.

4.2 EAP Message Flows

EAP consists of four different message types: *request*, *response*, *success*, and *failure*. Request messages are sent by an authenticator or authentication server, while response messages are sent by a peer to the authenticator and may be forwarded to the authentication server. Success/failure messages are sent either by an authenticator or authentication 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; where it is simply signaled that the requested information or function is not available and another request message with different options must be re-send.

An *EAP execution* 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 authentication 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_1*. 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 authentication 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. In the scenarios considered in this Recommendation, i.e. a federal peer using EAP for network authentication to a federal wireless network, a federal authentication server **shall** only select EAP

methods that comply with the security requirements in this Recommendation. In addition, a federal peer **shall** only accept EAP methods that comply with the security requirements in this Recommendation.

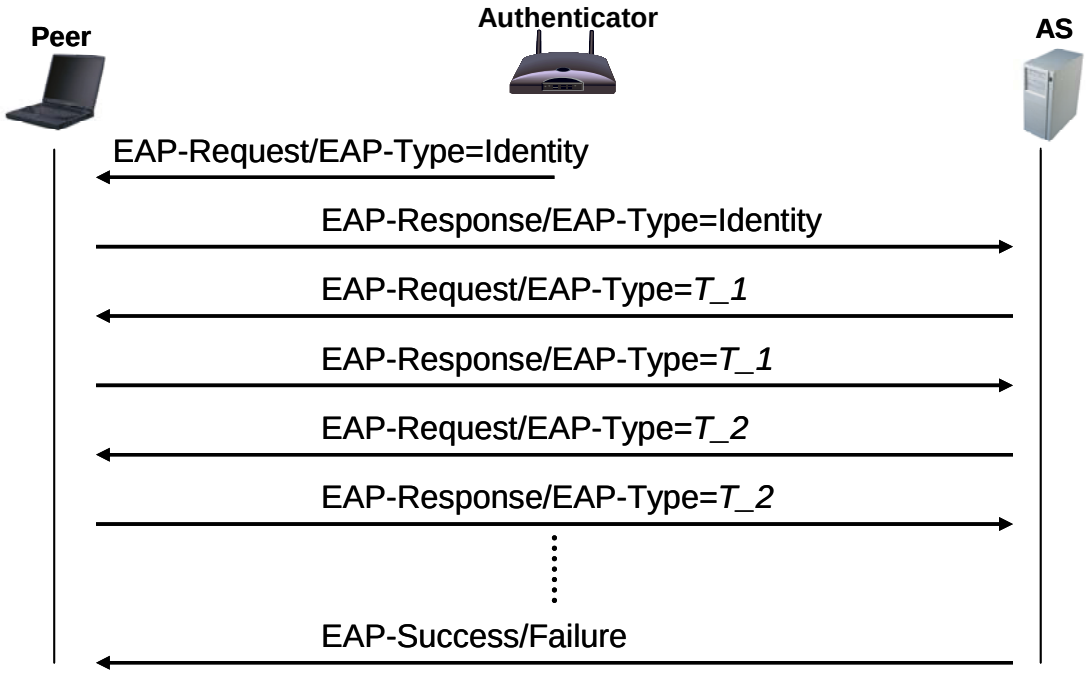

Figure 2: Example Message Flow of an EAP Execution in Pass-Through Mode

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 or IEEE 802.16, whereas CL2 employs an AAA protocol for communications. As a consequence of the 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 an authentication 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 access. For the purpose of an EAP execution, an authenticator is a functional entity that may reside in a point of access, while the communication link CL1 is implemented by a point of access through lower layer protocols as shown in Figure 3.

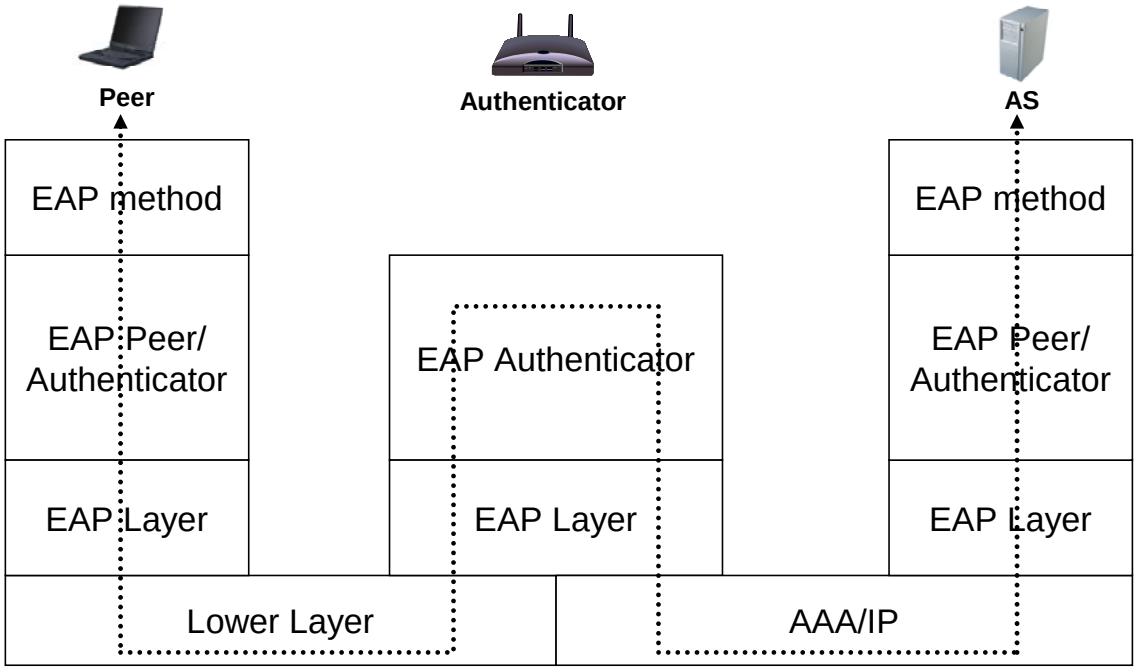

Figure 3: EAP Protocol Stacks

4.4 Tunnel-based EAP Methods

In some EAP methods, an authentication scheme is executed in a protective tunnel, which is referred to as *tunnel-based EAP method*. Tunnel-based EAP methods consist of two phases; in the first phase, the peer and authentication server execute a *tunnel protocol* to establish a protected connection. Then, in the second phase, both parties execute an authentication scheme within the protective tunnel. The authentication scheme is referred to as *tunneled EAP method* in the remainder of this document. Commonly, the TLS protocol [10] 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 schemes encapsulated in EAP methods. Examples of tunnel-based EAP methods are: EAP-TTLSv0 [11], PEAP [14], and EAP-FAST [15], 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 legacy authentication methods for peers. Without tunneling, widely deployed but weak authentication methods (such as password authentication) are insecure. Secondly, tunnel-based EAP methods can enable privacy protection, because peer and, optionally, server identifiers can be exclusively exchanged in the tunnel and, thus, prevent an eavesdropper from identifying these entities. Especially, tracking mobile peers is a threat that can be prevented by privacy protection.

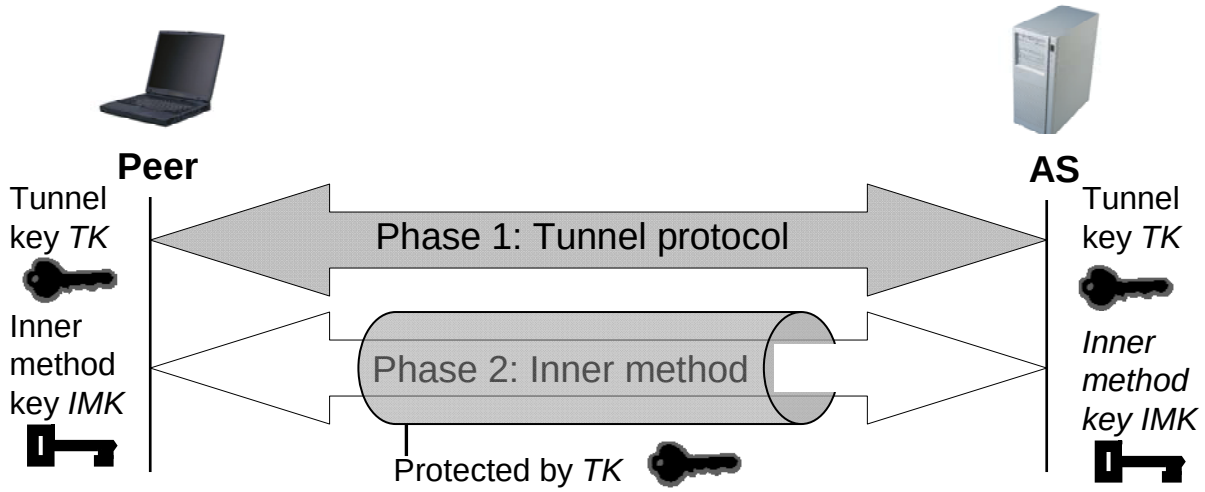

Figure 4: Overview Tunnel-based EAP Method

While RFC 3748 [1] prohibits the use of multiple authentication methods within a single EAP execution due to vulnerability to man-in-the-middle attacks and incompatibility with existing implementations, the prohibition does not apply to tunnel-based EAP methods. In that case, a tunnel-based EAP method is considered as one authentication method and, thus, multiple EAP methods may be executed within the protective tunnel. Here, 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 authentication 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 derive further dedicated 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.

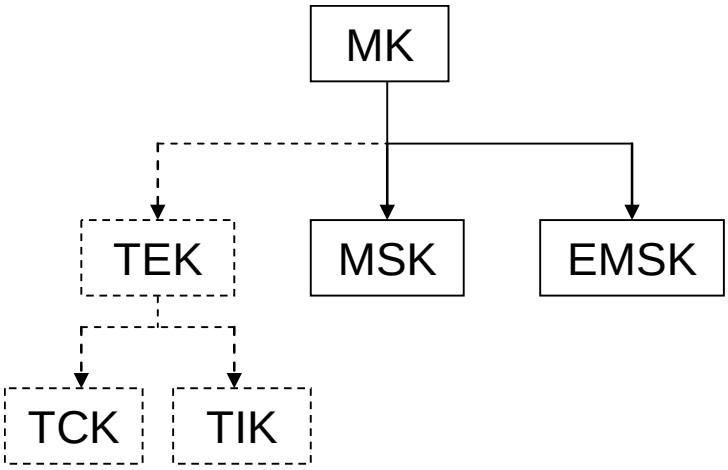

Figure 5: EAP Key Hierarchy

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 wireless link CL1.
2. EMSK remains on the server and is reserved for future use. Recently, EMSK has been considered for deriving handover keys (see [16]).
3. TEKs are used to derive session keys to protect the messages of the ongoing EAP execution. For example, transient encryption keys (TCK) and transient integrity protection keys (TIK). In other words, TEKs are used for message protection at the EAP layer.

4.6 EAP Ciphersuite Negotiation

In many EAP methods, the peer and authentication 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, some EAP methods only support a few (e.g. EAP-GPSK supports two) or none at all (e.g. EAP-AKA [17]). 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 (see Figure 2). 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*) algorithms and a ciphersuite could be denoted as $CS_i=\{AUTH_i\ KE_i,\ KDF_i,\ ENC_i,\ INT_{i,}\}$.

During a typical ciphersuite negotiation, an offering party (the peer or authentication server) offers a choice of n supported and acceptable ciphersuites $CS\_offer= \{CS_1,\ CS_2,\ldots,\ CS_n\}$ to the selecting party (the authentication 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 message flow of a two-way negotiation is illustrated in Figure 6, where the authentication 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.

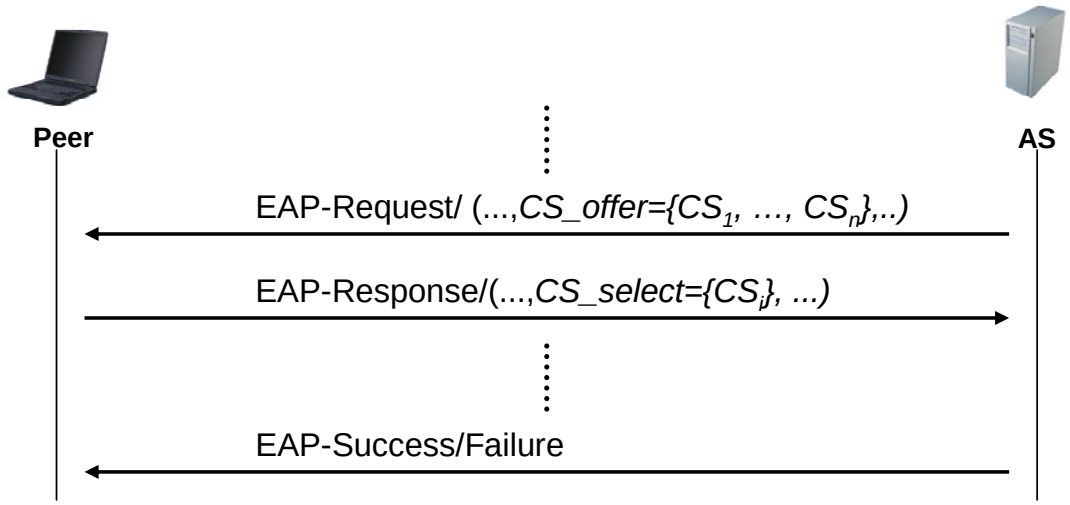

Figure 6: Example Two-way Ciphersuite Negotiation

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 authentication server. Existing EAP methods used in wireless mobile 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, 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 outsider 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 mobile users, e.g. by correlating all EAP executions carrying the same peer identifier.

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. It 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 key.

  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). For more details and additional impersonation attacks, please refer to Annex A.

- 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 authentication server. Such negotiations are vulnerable to downgrading attacks, in which an insider or outsider attacker tries to force the peer and authentication server to agree on a weak ciphersuite that contains cryptographic algorithms that are susceptible to attacks. For example, a rogue authenticator could send a reduced set of weak ciphersuites as offer or a man-in-the-middle could reduce the offered set that is sent over the wireless link.

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 this data attractive. Furthermore, many applications require the peer to send security-sensitive information, such as personal identification information, passwords, classified information etc, 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 service through attacking EAP more attractive to outsider 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 or higher service rates.

5.4 Tunnel-based EAP Methods

Man-in-the middle attacks on tunnel-based EAP methods have been identified in [33]. 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 authentication server in which only server authentication is provided. Once the tunnel is established, the adversary initiates an EAP execution with a peer pretending to be an authenticator connected to a legitimate authentication server. The adversary uses a weak proprietary authentication mechanism (e.g. a password-based mechanism) and replays the peer's responses as its own responses to the authentication server through the tunnel. Hence, the adversary can successfully impersonate the peer to the authentication server.

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 Access

Unlike the authentication server and other network entities in the backend network, points of access provide only weak physical protection. PoA are typically distributed in a large geographic area and often located in public places. Therefore, PoAs may be accessible by outsiders and are, thus, prone to compromises. An authenticator may be located in or as a function unit of a point of access and the freshly derived keys are delivered to it upon a successful EAP authentication. Hence, an adversary controlling an authenticator can access the protected communications over CL1. In addition, an adversary-controlled authenticator could attempt a *lying network access point* attack in which the authenticator advertises false information [32]. For example such an authenticator could provide false information about the network, offered security and offered services which can be exploited in many ways, e.g. to lure peers into networks they did not intend to connect to, charge higher fees, reduce the end-to-end security between peer and authentication server. Please refer to [32] for more attacks.

In addition, rogue points of access can be set up by an adversary and cannot be easily distinguished from legitimate points of access by a peer, e.g. in its list of points of access in range. Such a rogue authenticator can launch the same attacks as described for 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 protection of subsequent communications over the wireless link.

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

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

The first security objective includes the authentication and authorization check of the EAP peer by the authentication server, the authentication of the authentication server to the peer, as well as the authorization and authentication check of the authenticator by the authentication server. 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 wireless link CL1 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:

- 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;
2. Mutual authentication of a peer and the authentication server;
3. Key establishment between a peer and the authentication server;
4. Service information exchange;
5. Message protection.

Ciphersuite negotiation enables crypto-agility and backward-compatibility (see Section 4.6), mutual authentication ensures that federal peers and federal authentication servers prove their acclaimed identities to each other, key establishment provides keying material to protect the remainder of the EAP execution and the wireless link, the service information exchange ensures the detection of malicious information send by rogue authenticator or other rogue intermediary entities, and message protection utilizes the established keying material to protect the remainder of the EAP execution.

Procedures can be executed sequentially, in parallel or in an interleaved fashion. For example, authentication and key establishment could be combined to authenticated key establishment. Furthermore, 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 EAP tunnel-based 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 it is encouraged to support procedure four.

7. Pre-conditions for EAP

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, how these pre-conditions can be achieved is outside of 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, respectively, 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 pair of peer and authentication server need to share cryptographically separated keys. This ensures that once one peer is compromised the long-term secret key of no other peer is affected as well.

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 distributed in a protected manner. Such secret keys are used as long term credentials in purely symmetric EAP methods, such as EAP-GPSK [9], 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 possible in EAP-FAST [15] 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 (peer and/or authentication 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

Communication link CL2 between authenticators and the authentication server cannot be secured by EAP methods. For this reason, the EAP framework is based on the assumption that

authenticators, the authentication 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 authentication server, communicate using an AAA protocols (such as RADIUS [6] or Diameter [7]). In that case the AAA protocol needs to provide all necessary security properties for protecting CL2.

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 authentication server needs to have access to the information that each legitimate authenticator is supposed to broadcast to peers as well as the information the authentication server is supposed to receive from authenticators and other intermediary entities via CL2 during an EAP execution. Basically, the authentication server must be able to verify the correctness, authenticity and authorization of information send by all entities in the backend network that participate in an EAP execution. For example, such information could be securely stored in a local database server (as illustrated in Figure 1). The database needs to be protected and be only accessible by authorized parties. For information what information such a database should contain and how it could be set up please refer to [32].

8. Security Requirements for Non-tunneled EAP Methods in Wireless Applications

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 negotiated or hard-coded set of algorithms meets all the security requirements. The security requirements for ciphersuite negotiation are in Section 8.1, where 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 Section 8.4.

8.1 Protected Ciphersuite Negotiation

Peer and authentication server may support ciphersuites containing cryptographic algorithms that do not comply with the security requirements in this document for (backward) compatibility reasons and access authentications in non-federal settings. However, in the scenarios under consideration, the federal authentication server and federal peer **shall** only select a ciphersuite in which all algorithms comply with the security requirements in this Recommendation as the result of a ciphersuite negotiation. In other words—using the terminology from Section 4.6—federal peers and federal authentication server **shall** only classify ciphersuites complying with the security requirements in this Recommendation as acceptable in the wireless scenarios under consideration.

In general, ciphersuites are negotiated before transient EAP keys (*TEK*) are available 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, ..., CS_n}* to the weakest of the ciphersuite(s) (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, when the *TEK*s are available, both parties derive a transient integrity protection key *TIK* and send integrity protected verification messages to each other. The verification message of the offering party contains the original offer *CS_offer* and the received selection *CS_select'*, whereas the message of the selecting party contains the received offer *CS_offer'* and the original selection *CS_select*. The post-verification is only successful if both verifications pass (i.e. *CS_offer = CS_offer' and CS_select = CS_select'*). An example message flow of a two-flow ciphersuite negotiation with post-verification is illustrated in Figure 7.

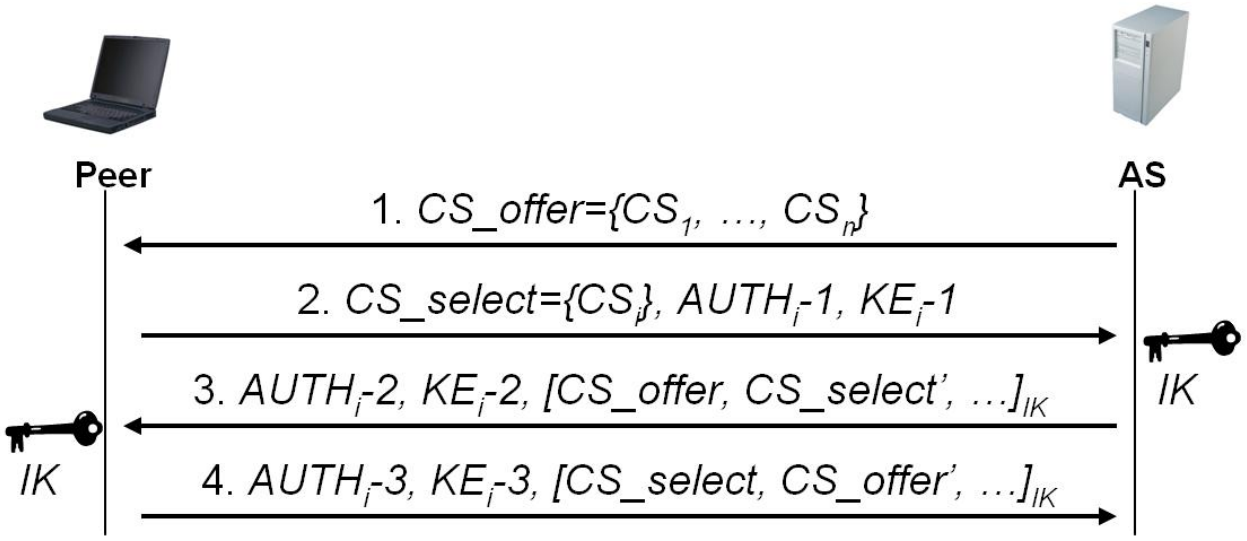

Figure 7: Ciphersuite Negotiation with Post-Verification

In conclusion:

SR-CN-1 Each supported EAP method **shall** at least offer one acceptable ciphersuite, i.e. a ciphersuite that only contains cryptographic algorithms complying with this Recommendation.

[SR-CN-2] Each supported EAP method providing ciphersuite negotiation **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 acceptable ciphersuites or implementations in which acceptable ciphersuites do not meet the requirement of this Recommendation could have severe consequences, as described in [5].

8.2 Mutual Authentication

The wireless mobile applications considered here, require mutual authentication between the peer and authentication 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 authentication server.

This requirement applies to the authentication algorithm that is part of the negotiated ciphersuite or 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 or password. 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 some nonce.

In order to prevent attacks on the cryptographic algorithms employed by the mutual authentication procedure:

SR-AUTH-2 Approved cryptographic schemes **shall** be employed for authentication that satisfy the security strength requirements for algorithms and key sizes in NIST SP 800-57 [22];

SR-AUTH-3 When symmetric key-based MACs are employed for entity authentication, approved algorithms **shall** be used as specified in FIPS 198 [22] when HMAC is used and in NIST SP800-38B [23] when CMAC is used that satisfy the security strength requirements specified in SP 800-57 [20];

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 [21] and NIST SP 800-57 [20].

A secure (mutual) authentication procedure prevents impersonation attacks (as listed in Section 5.1). In order to prevent impersonation through replay attacks on the authentication protocol:

SR-AUTH-5 An authentication response **shall** resist replay attacks by using previous generated authentication responses. Non-repeating nonces **shall** be used for generating an authentication response.

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, but sequence numbers require careful tracking for each session and each verifier and timestamp the synchronization of clocks.

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

SR-AUTH-6 Each EAP method **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 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 derivation (as specified in [1]) **shall** be used. The key establishment algorithm specified either by the negotiated ciphersuite or 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:

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 [22];

SR-KE-2 When symmetric key-based MACs are employed for key establishment[2], approved algorithms **shall** be used as specified in FIPS 198 [22] when HMAC is used and in NIST SP800-38B [23] when CMAC is used that satisfy the security strength requirements specified in SP 800-57 [20];

SR-KE-3 Key establishments using discrete logarithm cryptography **shall** conform to the requirements in NIST SP 800-56A [28] and SP 800-56B [29];

SR-KE-4 Whenever authentication and key establishment subroutines are combined, the mutual authenticated key establishment subroutine **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-5 A key establishment protocol **shall** provide mutual implicit *key authentication,* i.e., the master key *MK* is only known to the peer and the authentication server, and thus, *MK* is kept confidential.

SR-KE-6 A key establishment protocol **shall** provide *key freshness*, i.e. the peer and the authentication server are assured that the established MK is fresh, and re-using expired keying material is prevented. The freshness property is typically achieved by using nonces, sequence numbers, timestamps or a combination of these. (This property is related to SR-AUTH-5).

[SR-KE-7] A key establishment protocol **should** provide *key control*, i.e., the peer and the authentication server both contribute to the *MK* computation. This property prevents

[2] At the time of publication, no federal standard (e.g. FIPS and NIST Special publications) exists for symmetric key establishment. Once such standards become available, the specified approved algorithms **shall** be used for symmetric key establishments in EAP method.

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. Note that key control can only be ensured for *MK* in EAP, because the derivation of descendant keys by the respective key holders and the peer might be conducted in a non-interactive fashion.

[SR-KE-8] A key establishment protocol **should** provide *key confirmation,* i.e., the peer and AS both confirm that they computed *MK*. 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-9] A key establishment protocol (for public-key based key establishment schemes) **should** provide *forward secrecy (FS)*, i.e., the 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 derived key hierarchy **shall** prevent domino effects (see [30]). This has several implications:

a. Key derivation **shall** be one-way, i.e., a compromised key leads to the compromise of all descendant keys, but does not affect the security of any precedent key in the same key hierarchy;

b. Different keys derived from the same key **shall** be cryptographically separated, i.e., a compromised key **shall** not affect the security of other keys derived from the same root key.

SR-KD-2 The key hierarchies established through different EAP executions (see Figure 5) **shall** be cryptographically separated. In other words, the compromise of an ephemeral key established in one EAP execution **shall** not affect any key established in other EAP executions.

[SR-KD-3] The key derivation functions **should** comply with NIST SP 800-108 [18];

[SR-KD-4] Each derived key **should** be bound to all entities that derive, use, or hold this key by including the corresponding identifiers in the input of the key derivation function;

[SR-KD-5] Each derived key **should** be bound to its application or usage by including appropriate key labels in the input of the key derivation function;

[SR-KD-6] Each key **should** have a lifetime assigned that does not exceed the lifetime of any key higher in the key hierarchy.

8.4 Service Information Exchange

An EAP peer is neither able to authenticate an authenticator nor verify the information received by it. As a result, EAP methods that do not support the exchange of additional service information are susceptible to the lying access point problem and other attacks by compromised or rogues authenticators (see Section 5.5). This demands that keys are only transported to the 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 authentication 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 entities and the authentication server) are securely bound to an EAP method execution [1][31]. This ensures the consistency of the information provided to the EAP peer and the authentication server by any intermediary entities. EAP channel binding may require the following steps as described in [31]:

1. The peer sends the information received from the authenticator to the server over an integrity-protected channel;
2. The authentication 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 CL2) with the information stored in its protected database.
3. The authentication server sends the verification result to the EAP peer over an integrity-protected channel.

Please note that steps 1 and 3 require an integrity-protected channel between peer and authentication server. This does not pose an additional requirement since EAP methods complying with this Recommendation need to derive keying material (see Section 8.3) and provide message protection (see Section 8.5). The second step requires the authentication server to be capable of checking whether the received information from peer and authenticator is consistent with its stored information which is described as a system pre-requisite in Section 7.3.

In summary,

[SR-CB-1] Each EAP method **should** define messages to securely exchange service information necessary for providing channel bindings.

A method of how EAP channel bindings can be achieved by using encapsulated AVPs is described in [32].

8.5 EAP Message Protections

After fresh EAP keys are established and the protection algorithms are agreed on, 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 cipher suite 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).

In summary:

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;

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 for integrity-protection and message authentication shall follow the requirements on cryptography strength for algorithms and key sizes in NIST SP 800-57 [22].

SR-MP-5 Algorithms for integrity-protection and message authentication that employ HMACs shall follow the requirements on cryptographic strength specified by FIPS 198 [23];

SR-MP-6 Algorithms for integrity-protection and message authentication that employ CMACs shall follow the requirements on cryptographic strength specified by SP 800-38B [24];

SR-MP-7 Algorithm for confidentiality-protection **shall** either employ AES as specified by FIPS 197 [26] or TDEA as specified in NIST SP 800-67 [25] as encryption algorithm;

9. Requirements for Tunnel-based EAP Methods in Wireless Applications

A tunnel-based EAP method describes a framework for executing EAP 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 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 does not restrict which authentication methods can be used as inner method. 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 EAP method(s) executed within the tunnel.

9.1 Tunnel-based EAP Method

Under some conditions, tunnel-based EAP methods are vulnerable to a particular man-in-the-middle attack described in [32]. In this attack, an adversary—masquerading as a peer—initiates a tunnel-based EAP method with the authentication server. As part of this EAP method, the adversary executes a tunnel protocol with the authentication server, in which the authentication server authenticates to the adversary (thinking it is the peer). Upon a successful tunnel protocol execution, both the adversary and the authentication server are in possession of the established tunnel key *TK*. Now the server initiates an inner authentication method inside the protective tunnel. The

adversary—acting as an authentication 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 authentication server believe that the messages are coming from the other end of the tunnel. Hence, the inner authentication method, and thus the tunnel-based EAP method terminate successfully and both the adversary and the authentication server share the established *MSK*. The attack is illustrated in Figure 8. Concluding, the tunnel protocol does not provide any protection to the inner authentication method(s) if the described man-in-the-middle attack applies.

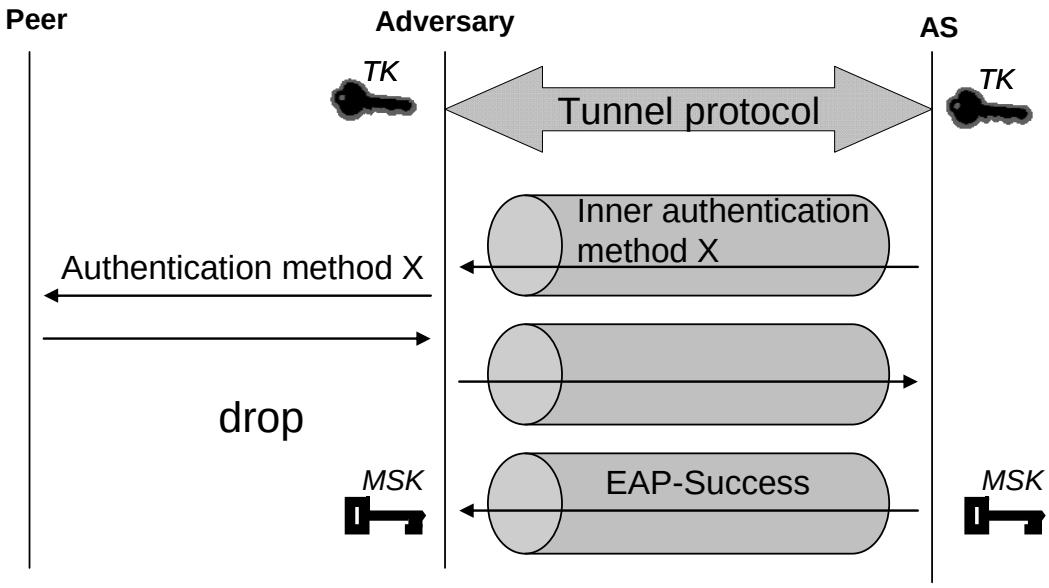

Figure 8: Man-in-the-middle Attack on Tunnel-based EAP methods

The described man-in-the-middle attack can be mitigated by a combination of *cryptographic bindings* provided by the tunnel-based method and server- and/or peer-enforced system requirements. Cryptographic bindings support binding the inner authentication method(s) to the tunnel protocol. This is realized by computing a compound key *CTK* using the tunnel key *TK* and the derived key MK_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 prove that the peer and the authentication server were indeed the endpoints of the tunnel as well as all inner authentication methods. It needs to be emphasized that only inner authentication methods with key establishment contribute a non-zero input to the compound key computations resulting into a non-trivial cryptographic binding. On the other hand, 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 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 [5]. Here, the adversary breaks the key establishment of the authentication method and is thus still able to compute compound key *CTK*.

The possible ways for mitigating the man-in-the-middle attack on tunnel-based EAP methods can be summarized as follows[3]:

1. Only permit tunnel-based EAP methods supporting cryptographic bindings and inner authentication methods with key establishment that 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.
3. Only permit inner authentication methods that do not establish keys or are vulnerable to attacks to be executed within a protective tunnel.

The first mitigation option is the most preferable from a security point of view, because supporting cryptographic bindings can be enforced by the tunnel-based EAP method while ensuring that only key establishing inner authentication methods are executed can be enforced by the authentication server. The exclusive use of inner methods with key establishment guarantees non-trivial cryptographic bindings. The second option is less favorable because the server policy that certain authentication methods can only to be executed within a protective tunnel must be enforced by the peers. Unlike authentication 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. The last option is the least favorable one because it completely relies on the peer to correctly enforce the policy.

For the aforementioned 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 of introducing tunnel-based EAP methods. At the same time it supports cryptographic bindings for key establishing authentication methods facilitating more secure implementations. The same mitigation option is chosen in the current IETF draft "Requirements for a Tunnel Based EAP Method" [34].

From the above discussion about the mitigation of man-in-the-middle attacks, the following requirements for the federal wireless network access authentication apply.

SR-TBEAP-1 Every tunnel-based EAP method **shall** provide cryptographic bindings, i.e. define how a compound key *CTK* is computed from the tunnel key *TK* and the established keys from all inner methods *MK* and how this key is then used to provide mutual authentication.

SR-TBEAP-2 Every tunnel-based EAP method **shall** compute *MSK* and *EMSK* from the compound key *CTK*.

A typical key hierarchy for tunnel-based EAP methods with cryptographic bindings is depicted in Figure 9, in which dashes indicate optional keying material.

[3] The option of only permitting tunnel protocols with mutual authentication to mitigate the attack is not listed here, because tunnel-based EAP methods were introduced to enable mutual authentication without requiring client certificates.

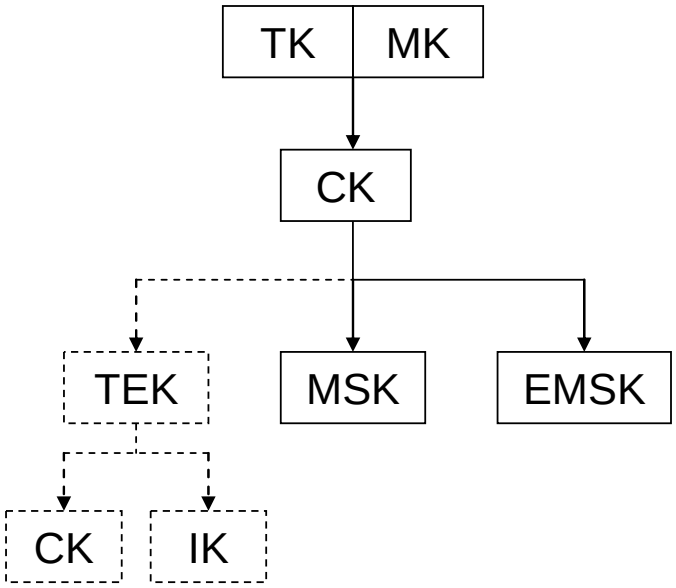

Figure 9: Key Hierarchy of Tunnel-based EAP methods with Cryptographic Binding

Security requirements for the derived key hierarchy of tunnel-based EAP methods with cryptographic binding can be summarized as:

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

As mentioned in Section 4.4, it is permitted per [1] that multiple authentication methods are executed within a protective tunnel during an EAP execution. This Recommendation distinguishes between the *parallel and sequential execution* of authentication methods within the tunnel. Parallel 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 tunnel during the same 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, parallel 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-4] In the case of the parallel execution of n inner authentication methods, an individual compound key CTK_i **should** be computed upon the completion of each inner method i, i.e. $CTK_i = f(MK_i, TK)$ for $0 < i \leq n$.

[SR-TBEAP-5] In the case of the sequential executions of n inner methods, 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. $CTK_i = f(CTK_{i-1}, MK_i)$ 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 authentication server. The established keying material is used to derive a tunnel key *TK*.

SR-TP-1 The key establishment of the tunnel protocols **shall** satisfy all requirements specified in Section 8.3[4].

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 the tunnel protocol.

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 mutual privacy—do not comply with this Recommendation. For example, anonymous DH-key establishment, as defined in some TLS v1.0 [35] ciphersuites and supported by EAP-FAST [15] and EAP-TTLSv0 [11], is non-compliant. Please refer to [5] for a detailed description of risks when bidirectional anonymous tunnels are used in tunnel-based EAP methods.

9.2.1 TLS as 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 tunnel protocol are described. Currently three TLS versions exist, TLS v1.0 [35], TLS v1.1 [10], and TLS v1.2 [37]. 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_key establishment algorithm_WITH _encryption algorithm_message authentication 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. In TLS v1.0 and v1.1 the XOR of the outputs of HMAC-MD5 and HMAC-SHA1 is used as pseudo-random function (PRF), while starting with TLS v1.2 the specified message authentication algorithm is also used as PRF.

Not all TLS ciphersuites are suitable to meet the requirements for tunnel protocols defined in the previous section, i.e. SR-TP-1 and SR-TP-2. As a general rule to comply with this Recommendation, acceptable TLS ciphersuites only consist of approved cryptographic algorithms. However, not all such combinations are automatically secure. NIST SP 800-57, Part 3 [37] specifies which ciphersuites in TLS v1.0, v.1.1, and v1.2 are secure and thus acceptable for federal use. Therefore, in order to comply with this Recommendation,

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

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 acceptable TLS ciphersuite, i.e. a TLS ciphersuite that is listed in Tables 4-1, 4-2, 4-3 or 4-4 in NIST SP 800-57, Part 3 [37], **shall** be supported.

SR-TLS-2 Only acceptable TLS ciphersuites **shall** be the result of a TLS ciphersuite negotiation as part of the tunnel protocol in a tunnel-based EAP method.

Note that requirements SR-TLS-1 and SR-TLS-2 replace requirements SR-TP-1 and SR-TP-2, whenever TLS is used as the tunnel protocol in a tunnel-based EAP method.

9.3 Tunneled EAP Method

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

SR-TEAP-1 Any inner authentication method employed **shall** provide unidirectional authentication from the peer to the authentication 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. For example, tunneled EAP methods do no longer need to employ federally approved cryptographic algorithms for authentication, key establishment, integrity protection and other cryptographic operations.

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

SR-TEAP-2 Every EAP 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 EAP methods **shall not** be executed as an autonomous authentication method outside a protective tunnel.

Under certain conditions sequential execution of multiple authentication methods enable the secure use of methods that are vulnerable to attacks without system requirement SR-TEAP-2. This can be done by the chained compound keys described in [SR-TBEAP-5] as long as at least one authentication method out of the sequence of authentication methods provides key establishment and resist attacks in non-tunneled mode. To ensure that at least one of the authentication methods provides key establishment, the authentication server and peer could first negotiate the sequence of methods, and/or the tunnel-based EAP method aborts with a failure if none of the inner methods derived keying material.

[5] If the tunnel protocol is not secure (i.e. one or both requirements SR-TP-1 and SR-TP-2 are 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.

10. Summary

EAP is widely deployed and currently used to secure a growing number of wireless mobile applications. In such applications, a mobile station attempts to access a network over a wireless link, as is the case in an increasing number of applications used by the U.S. Federal Government. Hence, the security of EAP methods 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 (such as 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 186-2, 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 protection.

This Recommendation distinguishes between non-tunneled and tunneled EAP methods and specifies the necessary requirements to allow a cryptographically weaker EAP method to be executed within a protective tunnel. Attacks, which are possible to mount 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.

The next section demonstrates how this Recommendation may be used to check the compliance of EAP methods with the security requirements listed in this document.

11. Discussion of Selected EAP Methods

This section discusses how this Recommendation may be used to the check the security compliance of EAP methods when used in Federal wireless applications. Compliance checks are provided for a selection of widely known methods (namely EAP-GPSK [9], EAP-TLS [12], EAP-TTLSv0 [11] and EAP-FAST [15]) representing secret key-based, public key-based and tunneled EAP methods, respectively. However, this selection is purely illustrative and does not represent favorable methods for federal use.

In order to check the compliance with this Recommendation, the security requirements are checked for each ciphersuite supported 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 and Table 5 for EAP-TTLS, 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 if the considered EAP method and ciphersuite is in compliance with all shall requirements, this EAP method used with this ciphersuite is in compliance with this Recommendation, and thus safe to use in the considered Federal wireless applications.

Please recall that pre-conditions are EAP method independent and must be checked separately for any system that supports EAP as an authentication and key establishment protocol for access control.

11.1 EAP-GPSK

The EAP Generalized Pre-Shared Key (EAP-GPSK) method is specified in an Internet draft [9] 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 authentication server. The set up of these pairwise secret keys is part of the peer registration, and thus, must be included in the system pre-conditions.

During an EAP-GPSK execution 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 depends on the used key derivation function (KDF) and the length of the exchanged nonces. The EAP-GPSK draft 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}, and 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 communication. GKDF is defined in [9] 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 does meets all shall requirements and is thus in compliance with this Recommendation.

Security Requirement	**Compliance**
SR-CN-1	✓

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

	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	✓	✓
SR-KE-3	N/A	N/A
SR-KE-4	✓	✓
SR-KE-5	✓	✓
SR-KE-6	✓	✓
[SR-KE-7]	✓	✓
[SR-KE-8]	✓	✓
[SR-KE-9]	N/A	N/A
SR-KD-1 a) b)	a) ✓ b) ✓	a) ✓ b) ✓
SR-KD-2	✓	✓

[SR-KD-3]	✓	✓
[SR-KD-4]	○[7]	○
[SR-KD-5]	○	○
[SR-KD-6]	○	○
[SR-CB-1][8]	○	○
SR-MP-1	✓	✓
SR-MP-2	✓	✓
SR-MP-3	✓	✓
SR-MP-4	✓	✓
SR-MP-5	✓	✓
SR-MP-6	✓	✓
SR-MP-7	✓	✓

Table 2: EAP-GPSK Compliance Check

11.2 EAP-TLS

EAP-TLS [12] defines how the TLS protocol can be encapsulated in EAP messages. Per [12], every EAP-TLS implementation must support TLS v1.0 [35] and may support TLS v1.1 [10], TLS v1.2 [37] as well as later versions that might be published in the future, and implementations may support several of the numerous existing TLS ciphersuites. Ciphersuites in EAP-TLS are of the format CS=TLS_KE_WITH_ENC_MAC, i.e. the suite specifies key establishment, encryption and integrity-protection algorithms. Please note that MAC defines the hash function that is used to build a MAC for integrity-protection (i.e. using HMAC). The PRF in EAP-TLS is the PRF used in the implemented TLS version (see Section 9.2.1).

EAP-TLS supports mutual authentication, server authentication, and no authentication. Only ciphersuites 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 NAI is used), and the peer certificate is sent confidentially (i.e. in the tunnel). This technique is in compliance with this Recommendation as long as all security requirements are met.

[7] MSK does not include the identifier of the authenticator.

[8] EAP-GPSK provides an integrity-protected channel but does not define the encoding of channel binding information.

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

CS-TLS-1. TLS_RSA_WITH_3DES_EDE_CBC_ SHA-1.

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

CS-TLS-2. TLS_RSA_WITH_RC4-128_SHA-1
CS-TLS-3. TLS_RSA_WITH_AES-128-CBC_SHA-1

where for EAP servers it is additionally recommended to implement

CS-TLS-3. 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 complies and CS-TLS-2, CS-TLS-3, and CS-TLS-3 **shall not** be implemented.

For the compliance check in this section, the use of CS-TLS-1 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 thus is in compliance with this Recommendation.

Security Requirement	**Compliance**
SR-CN-1	✓
	Ciphersuite in compliance with the Recommendations in NIST SP 800-57 Part 3.
[SR-CN-2]	✓
SR-AUTH-1	✓
SR-AUTH-2	✓
SR-AUTH-3	✓
SR-AUTH-4	✓
SR-AUTH-5	✓
SR-AUTH-6	✓
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-KE-9]	✓
SR-KD-1 a) b)	a) ✓ b) ✓
SR-KD-2	✓
[SR-KD-3]	✓
[SR-KD-4]	○
[SR-KD-5]	○
[SR-KD-6]	○
[SR-CB-1]	○
SR-MP-1	✓
SR-MP-2[9]	N/A
SR-MP-3	✓
SR-MP-4	✓
SR-MP-5	✓

[9] EAP-TLS does not enable the exchange of confidential information as part of the EAP execution, For this reason, EAP-TLS does not provide an encryption algorithm.

SR-MP-6	✓
SR-MP-7	✓

Table 3: EAP-TLS Compliance Check

11.3 EAP-FAST

EAP-FAST [15] is an EAP tunnel-based 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.

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 one of the tables 4-1, 4-2, 4-3 or 4-4 in NIST SP 800-57, Part 3. Hence, none of the mandatory-to-implement ciphersuites are in compliance with this Recommendation. As a consequence, EAP-FAST implementations used in Federal wireless applications **shall** implement additional ciphersuite that comply with this Recommendation.

If and only 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 used in combination with a tunnel protocol (see SR-TEAP-2).

In case an EAP or other authentication method with strong key establishment is used as an inner authentication method, cryptographic bindings need to be computed.

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 and with a recommended choice of TLS ciphersuites, the tunnel protocol meets all requirements too. The compliance of the inner authentication methods cannot be generally checked for EAP-FAST, because EAP-FAST supports any type of authentication method as inner method. Note that SR-TEAP-1 must be met by every inner authentication method and SR-TEAP-2 is a system requirement. Concluding, for federal applications to comply with this Recommendation, EAP-FAST **shall** be only executed using one of the recommended TLS ciphersuites and inner methods that satisfy SR-TEAP-1.

Security Requirement	Compliance
SR-TBEAP-1	✓
SR-TBEAP-2	✓
SR-TBEAP-3	✓[10]
[SR-TBEAP-4]	N/A[11]
[SR-TBEAP-5]	✓
TLS used as tunnel protocol. Ciphersuites in compliance with the Recommendations in NIST SP 800-57 Part 3.	
SR-TLS-1	✓
SR-TLS-2	✓

Table 4: EAP-FAST Compliance Check

11.4 EAP-TTLSv0

EAP-TTLSv0 [11] is an EAP tunnel-based 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 EAP method or a legacy peer authentication protocol is then executed within the tunnel. EAP-TTLSv0 supports TLS v1.0, v1.1, v1.2 and potential later versions. There are no mandatory-to-implement ciphersuites defined in EAP-TTLSv0.

EAP-TTLSv0 provides peer privacy because 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 privacy is unidirectional (i.e. server identifiers are exchanged as part of the tunnel protocol), and the peer subsequently authenticates within the tunnel.

EAP-TTLSv0 does not support cryptographic bindings and method chaining[12].

Table 5 summarizes the requirements check for EAP-TTLSv0. As for EAP-FAST, with a recommended choice of TLS ciphersuites the tunnel protocol meets all requirements, while the compliance of the inner authentication methods cannot be generally checked, because EAP-TTLSv0

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

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

[12] 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.

supports any type of inner authentication method. In summary, this version of EAP-TTLS does not comply with this Recommendation because it does not support cryptographic bindings.

Security Requirement	**Compliance**
SR-TBEAP-1	×
SR-TBEAP-2	✓
SR-TBEAP-3	✓[13]
[SR-TBEAP-4]	N/A[14]
[SR-TBEAP-5]	N/A[15]
TLS used as tunnel protocol. Ciphersuites in compliance with the Recommendations in NIST SP 800-57 Part 3.	
SR-TLS-1	✓
SR-TLS-2	✓

Table 5: EAP-TTLSv0 Compliance Check

Appendix: References (Informative)

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

[2] IEEE Standard 802.11-1999, 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", 1999.

[3] IEEE Standard 802.16-2004, Institute of Electrical and Electronics Engineers, "Standard for Local and metropolitan area networks - specific requirements - part 16: Air Interface for Fixed Broadband Wireless Access Systems", October 2004.

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

[5] K. Hoeper and L. Chen, "Where EAP Security Claims Fail", "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.

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

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

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

[6] RFC 3579, IETF Request for Comments, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", Aboba, B. and P. Calhoun, September 2003.
[7] RFC 4072, IETF Request for Comments, "Diameter Extensible Authentication Protocol (EAP) Application", Eronen, P., Hiller, T. and G. Zorn, August 2005.
[8] FIPS PUB 180-3, "Secure Hash Standard", Federal Information Processing Standards Publication, October 2008.
[9] IETF Internet-Draft, "EAP Generalized Pre-Shared Key (EAP-GPSK)", T. Clancy and H. Tschofenig, Work in progress, draft-ietf-emu-eap-gpsk-17.txt, November 2008.
[10] RFC 4346, IETF Request for Comments, "The Transport Layer Security (TLS) Protocol Version 1.1", Dierks, T. and Rescorla, E., April 2006, [obsoletes RFC 2246].
[11] 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.
[12] RFC 5216, IETF Request for Comments, obsoletes RFC 2716, Simon, D., Aboba, B., and Hurst R., "The EAP TLS Authentication Protocol", March 2008.
[13] IEEE Standard 802.11i, Institute of Electrical and Electronics Engineers, "Supplement to Standard for Telecommunications and Information Exchange between Systems - LAN/MAN Specific Requirements - part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security", July 2004.
[14] IETF Personal draft, Vivek Kamath, Ashwin Palekar, Mark Wodrich, "Microsoft's PEAP version 0 (Implementation in Windows XP SP1)", draft-kamath-pppext-peapv0-00.txt, work in progress, October 2002
[15] 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.
[16] RFC 5247, 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.
[17] 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.
[18] NIST SP 800-108, "Recommendation for Key Derivation and Distribution in Wireless and Mobility Applications", Chen, L., Recommendations of the National Institute of Standards and Technology, NIST Special Publication Draft, November 2008.
[19] RFC 4017, IETF Request for Comments, "EAP Method Requirements for Wireless LANs", Stanley, D., Walker, J. and B. Aboba, March 2005.
[20] NIST SP 800-97, "Guide to IEEE 802.11i: Robust Security Networks, Recommendations of the National Institute of Standards and Technology", NIST Special Publication, November 2006.
[21] FIPS PUB 186-3, "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication, Reaffirmed, Jan. 27, 2000.
[22] NIST SP 800-57, "Recommendation for Key Management – Part 1: General (Revised)", NIST Special Publication, March, 2007
[23] FIPS PUB 198-1, "The Keyed-Hash Message Authentication Code (HMAC)", Federal Information Processing Standards Publication, July 2008.

[24] NIST SP 800-38B, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", NIST Special Publication, May 2005.
[25] NIST SP 800-67, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", NIST Special Publication, revised March 2008.
[26] FIPS PUB 197, "Specification for the ADVANCED ENCRYPTION STANDARD (AES)", Federal Information Processing Standards Publication, November 26, 2001.
[27] FIPS PUB 196, "Entity Authentication Using Public Key Cryptography", Federal Information Processing Standards Publication, February 1997.
[28] NIST SP 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication, March 2007.
[29] NIST SP 800-56B, "DRAFT Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography", NIST Special Publication, December 2008.
[30] RFC 4962, IETF Request for Comments, "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management", R. Housley and B. Aboba, Best Current Practice, July 2007.
[31] RFC 5247, IETF Request for Comments, "Extensible Authentication Protocol (EAP) Key Management Framework". Bernard Aboba, Dan Simon, and P. Eronen, EAP Working Group, August 2008.
[32] IETF Personal Draft, "Channel Binding Support for EAP Methods", Charles Clancy and Katrin Hoeper, work in progress, < draft-clancy-emu-chbind-02.txt>, July 2008.
[33] 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>.
[34] IETF Internet Draft, "Requirements for an Tunnel Based EAP Method", K. Hoeper, S. Hanna, H. Zhou and J. Salowey (Ed.), work in progress, <draft-ietf-emu-eaptunnel-req-01.txt>, October 2008.
[35] RFC 2246, IETF Request for Comments, "The TLS Protocol Version 1.0", T. Dierks, and C. Allen, January 1999, [obsolete by RFC 4346].
[36] 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]
[37] NIST SP 800-57, Part 3 (DRAFT), "Recommendation for Key Management, Part 3 Application-Specific Key Management Guidance", NIST Special Publication, October 2008.