

TCG Trusted Network Connect TNC IF-T: Protocol Bindings for Tunneled EAP Methods

**Specification Version 1.1
Revision 10
21 May 2007
Published**

Contact: admin@trustedcomputinggroup.org

TCG

TCG PUBLISHED

Copyright © TCG 2004-2007

Copyright © 2004-2007 Trusted Computing Group, Incorporated.

Disclaimer

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

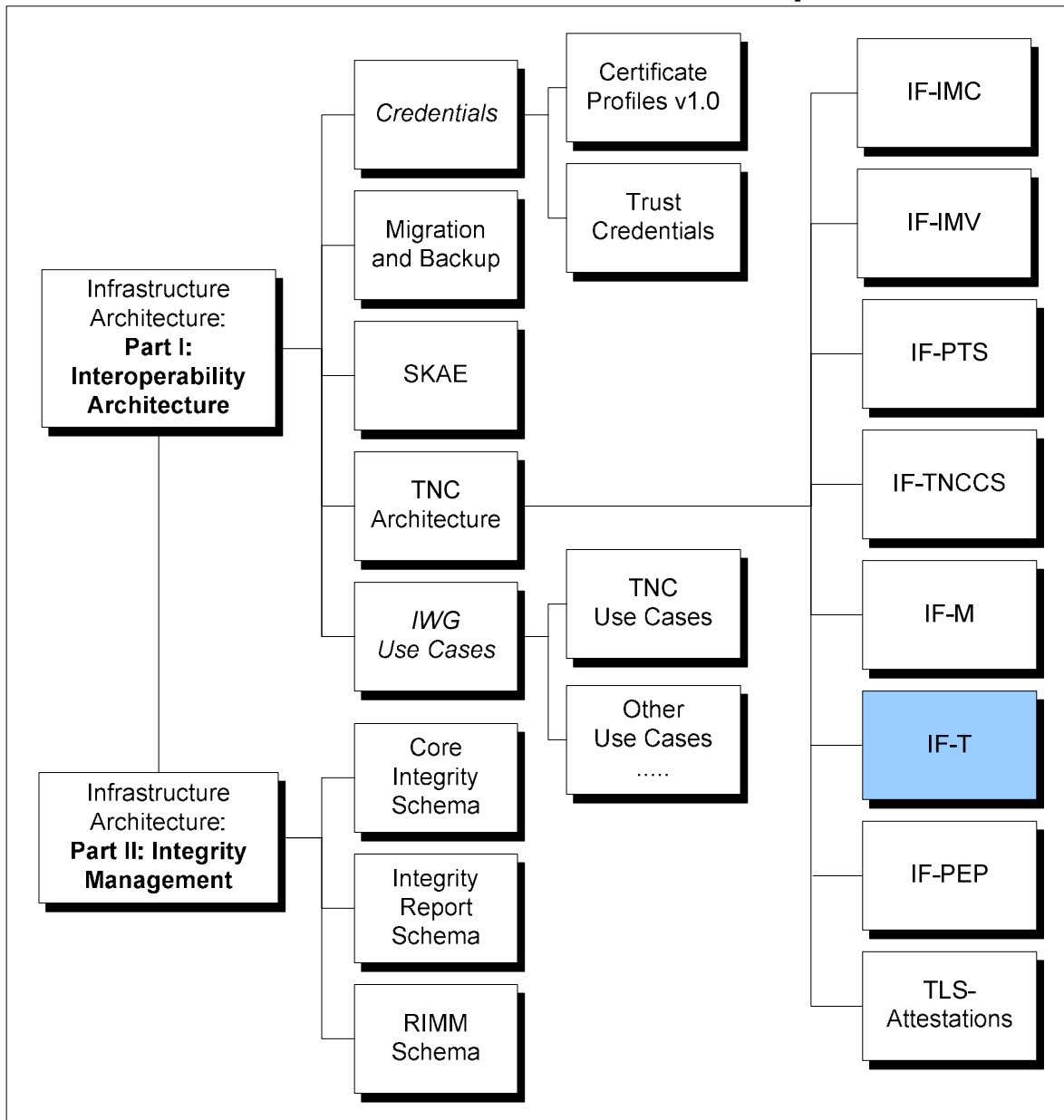
No license, express or implied, by estoppel or otherwise, to any TCG or TCG member intellectual property rights is granted herein.

Except that a license is hereby granted by TCG to copy and reproduce this specification for internal use only.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

IWG TNC Document Roadmap



Acknowledgements

The TCG wishes to thank all those who contributed to this specification. This document builds on work done in several other working groups in the TCG.

Special thanks to the active and previously active members of the TNC who contributed to the discussions and/or wordings used in this document:

Aman Garg	3Com
Bipin Mistry	3Com
Scott Kelly	Aruba Networks
Amit Agarwal	Avaya
Mahalingam Mani	Avaya
Hidenobu Ito	Fujitsu Limited
Sung Lee	Fujitsu Limited
Kazuaki Nimura	Fujitsu Limited
Boris Balacheff	Hewlett-Packard
Mauricio Sanchez	Hewlett-Packard
Diana Arroyo	IBM
Seiji Munetoh	IBM
Lee Terrell	IBM
Chris Hessing	Identity Engines
Morteza Ansari	Infoblox
Stuart Bailey	Infoblox
Ivan Pulley	Infoblox
Uri Blumenthal	Intel Corporation
David Grawrock	Intel Corporation
Ravi Sahita	Intel Corporation
Ned Smith	Intel Corporation
Chris Trytten	iPass
Barbara Nelson	iPass
Steve Hanna (TNC co-chair)	Juniper Networks
PJ Kirner	Juniper Networks
Lisa Lorenzin	Juniper Networks
Dean Sheffield	Juniper Networks
John Jerrim	Lancope
Gene Chang	Meetinghouse Data Communications
Alex Romanyuk	Meetinghouse Data Communications
John Vollbrecht	Meetinghouse Data Communications
Ryan Hurst	Microsoft Corporation
Sandilya Garimella	Motorola
Joseph Tardo	Nevis Networks
Pasi Eronen	Nokia Corporation
Chris Salter	National Security Agency
Jeff Six	National Security Agency
Meenakshi Kaushik	Nortel
Ron Pon	Nortel
Bryan Kingsford	Symantec Corporation
Paul Sangster (TNC co-chair, Editor r1.1)	Symantec Corporation
Curtis Simonson	University of New Hampshire InterOperability Labs
Rod Murchison	Vernier Networks
Michelle Sommerstad	Vernier Networks
Scott Cochrane	Wave Systems
Thomas Hardjono	Wave Systems
Greg Kazmierczak	Wave Systems

Table of Contents

1	Scope and Audience	8
2	Background	9
2.1	Purpose of IF-T and EAP Protocol Bindings	9
2.2	Requirements	9
2.3	Keywords	10
2.4	Features Provided by IF-T	10
3	Use of EAP-TNC with Tunneled EAP Methods	11
3.1	Model.....	11
3.2	EAP-TNC	12
3.3	Inner EAP Peer and Authenticator	13
3.4	Tunneled EAP Methods	14
3.4.1	EAP-FAST and PEAPv2	14
3.4.2	EAP-TTLS	14
3.4.3	PEAPv0/1	14
3.5	EAP-TNC Sequencing	15
4	Access Protocol Bindings (Informative)	16
4.1	802.1X.....	16
4.2	IKEv2.....	17
4.2.1	IKEv2 Dialog.....	19
5	Security Considerations	20
5.1	Threat Model	20
5.1.1	Threats	20
5.2	IF-T Capabilities	20
5.2.1	Interaction with Platform Trust Services (PTS)	20
5.2.2	Authentication Protection	20
5.2.3	Protection of TNC Data	21
5.3	Some Attack Scenarios	21
5.4	Philosophy of Protection	21
5.4.1	Scope of Protection	21
5.4.2	Minimum security Protection	21
5.4.3	Tunneled EAP Minimum Protections	22
5.4.4	Recommended Security Practices	23
5.4.5	Protecting against MiTM attacks against EAP-TNC.....	23
6	EAP-TNC Protocol Definition (Normative)	28
6.1	Protocol overview	28
6.1.1	State Machine.....	28
6.1.2	Fragmentation	28
6.1.3	EAP-TNC format.....	29
6.1.4	EAP-TNC Compliance.....	30
6.1.5	Security claims	30
6.2	EAP-TNC Conversation Example	30
6.3	Diffie-Hellman (D-H) Pre-Negotiation.....	31
6.3.1	Use of D Flag.....	31
6.3.2	D-H Pre-Negotiation Message Syntax	32
6.3.3	Diffie-Hellman Pre-Negotiation Protocol	34
6.3.4	Diffie-Hellman Pre-Negotiation Hash Algorithm Values	36
6.3.5	Diffie-Hellman Group Values.....	37
7	References	39

7.1	Normative References	39
7.2	Non-Normative References	39
7.3	Non-Normative Internet Drafts	40

1 Scope and Audience

Trusted Network Connect (TNC) is a working group within the Trusted Computing Group (TCG). TNC is defining an open solution architecture that enables network operators to enforce policies regarding endpoint integrity when granting access to a network infrastructure. Part of the TNC architecture is IF-T, a standard for mapping the communications between TNC Clients and TNC Servers onto existing protocols. This document defines and specifies IF-T.

IF-T is integral to the TNC reference architecture. The relationship of IF-T to other components of the TNC reference architecture is shown below in Figure 1.

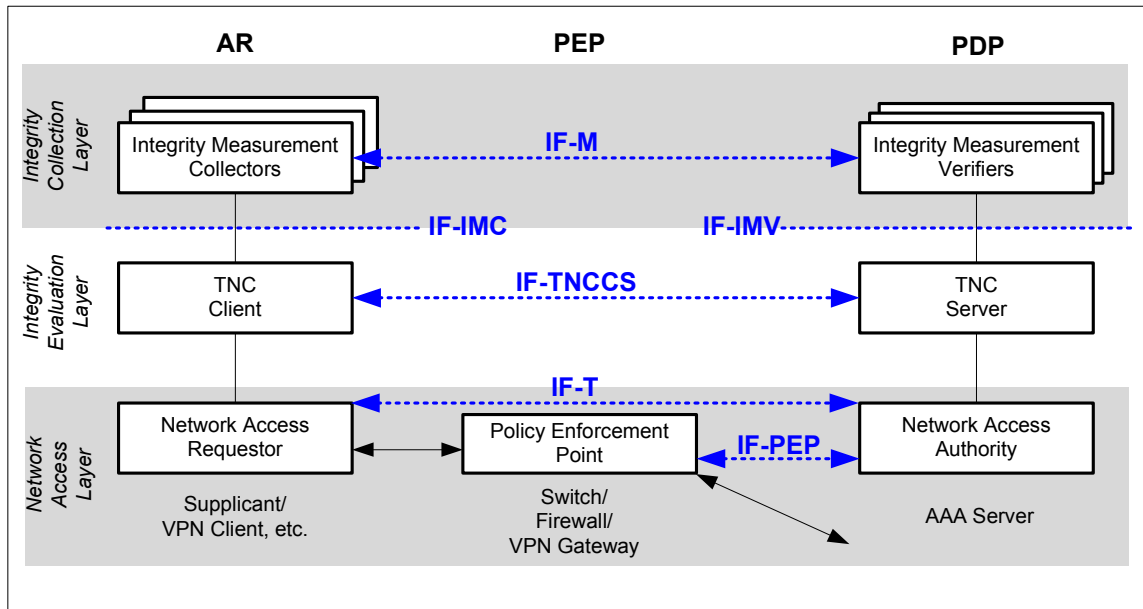


Figure 1. TNC Architecture

Architects, designers, developers, and technologists interested in the development, deployment, and interoperation of trusted systems will find this document necessary in providing specific mechanisms for transporting integrity information.

Before reading this document any further, the reader should review and understand the TNC architecture as described in [1].

2 Background

2.1 Purpose of IF-T and EAP Protocol Bindings

As shown in Figure 1, IF-T is the transport that carries IF-TNCCS messages between Network Access Requestor (NAR) and the Network Access Authenticator (NAA). IF-TNCCS is a TNC specified protocol for carrying Integrity Measurement (IM) protocol messages between Integrity Measurement Collectors (IMC) and Integrity Measurement Verifiers (IMV). This specification is for a protocol mapping of IF-T to a set of Tunneled EAP protocol methods that can be used to provide IF-T transport during access request dialogs.

This protocol binding specification is the initial IF-T binding being provided by TNC. Architecturally, IF-T does not specify a single protocol for transporting IF-TNCCS messages. TNC may provide one or more additional specification for protocol bindings which define how IF-T can be implemented over other lower layer protocols. While the architecture allows IF-TNCCS to be carried over many existing protocols and also over new protocols, TNC is providing this specification of IF-T protocol binding as the initial specification of how IF-T is required to be implemented when using Tunneled EAP methods for the lower layer protocol. As stated above, TNC may define additional IF-T protocol bindings defined as needed.

This document describes and specifies a mapping of IF-T to tunneled Extensible Authentication Protocol (EAP) methods. A tunneled EAP method is one that provides a cryptographically protected wrapper within which other protocol elements can be exchanged. Suitable tunneled EAP methods for IF-T are those able to carry nested EAP exchanges as protected protocol elements. This document further specifies an EAP wrapper for TNCCS, enabling it to be carried as a nested EAP method within a suitable tunneled EAP method.

For interoperability, the protocol bindings specified in this document **MUST** be implemented in any product claiming TNC compliance and providing IF-T using tunneled EAP methods. These tunneled EAP bindings make it possible to implement IF-T over a number of existing access protocols that use EAP at the access level. Some examples of such access protocols include 802.1X for wired and wireless, and IKEv2 for establishing VPNs over IP networks.

2.2 Requirements

Here are the requirements for IF-T.

- Meets the needs of the TNC architecture

IF-T must support all the use cases described in the TNC architecture as they apply to transporting IF-TNCCS messages between the TNCC and TNCS.

- Provide security

The integrity and confidentiality of communications between IMCs and IMVs must be protected. IF-T must specify how to provision secure communications between the TNCC and TNCS to transport IF-TNCCS messages. See the Security Considerations section.

- Be efficient

The TNC architecture delays network access until certain endpoint integrity checks have been performed. To minimize user frustration, it is essential to minimize delays and make IF-T communications as rapid and efficient as possible. Efficiency is also important when you consider that some network endpoints are small and low powered.

- Provide a half duplex message protocol

IF-T guarantees delivery of messages in the order received, and provides reliable transmission of data, handling retransmission and fragmentation of messages if needed.

- Be extensible

IF-T will need to be expanded over time as new features are added to the TNC architecture and new use cases identified.

2.3 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [2]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

2.4 Features Provided by IF-T

The TNC architecture does not specify that any particular protocol be used for IF-T, and in fact specifies that different protocols may be used. This document provides the specification of Protocol Bindings when a Tunneled EAP method is used as the method to carry IF-T. This **MUST** be used when using a version of IF-T that uses a tunneled EAP method. It **MAY** be used for other applications to be described in TNC use cases later.

In particular this document describes the mapping of IF-TNCCS messages to a standard TNC EAP method. It includes specification of the standard EAP method called EAP-TNC. It also describes the method for using three existing tunneled EAP methods to carry EAP-TNC: EAP-FAST, EAP-TTLS, and EAP-PEAP.

The EAP-TNC method specified is an EAP inner method which is compatible with the EAP framework defined by IETF [4]. EAP-TNC **MUST** be used when TNC is used with tunneled EAP methods. EAP-TNC carries the IF-TNCCS messages, and is itself carried as an inner method by one of the tunneled EAP methods.

3 Use of EAP-TNC with Tunneled EAP Methods

3.1 Model

Figure 2 shows the protocol layers that combine to provide IF-T using EAP-TNC over tunneled EAP methods. All of the highlighted layers have components that are part of the IF-T protocol binding for tunneled EAP methods.

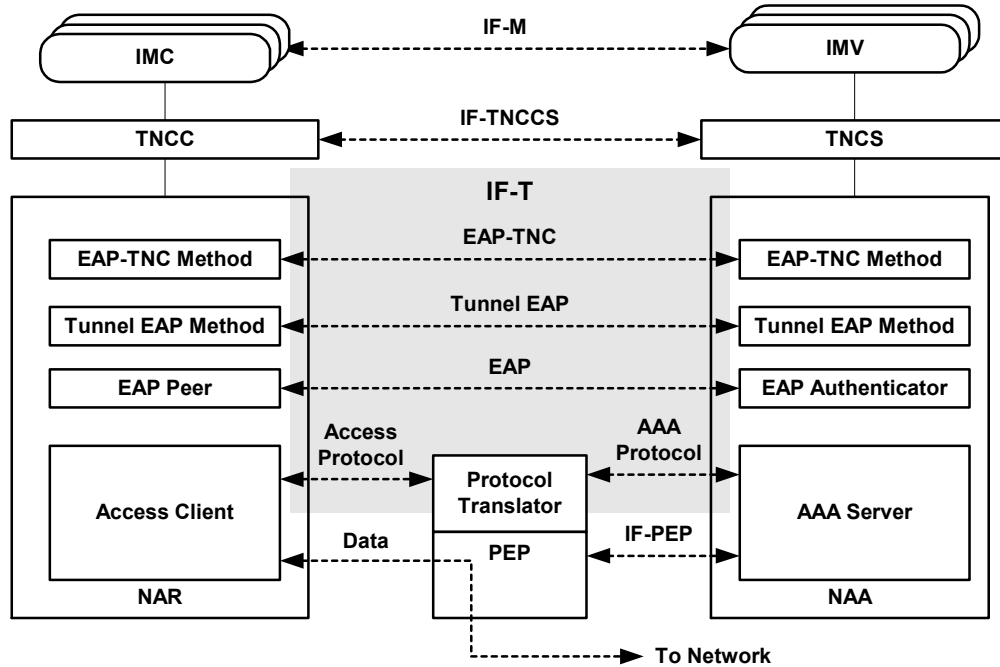


Figure 2. EAP-TNC and EAP Protocol Layers

The Access Requestor consists of the NAR, the TNCC and the IMCs. The PDP consists of the NAA, the TNCS and the IMVs. In Figure 2, the NAR and NAA communicate via four protocol layers which combine to form the IF-T protocol binding for tunneled EAP methods. Each side must send protocol messages that interoperate at each of these layers.

Starting from the top, the EAP-TNC method is a simple EAP method. This method is specified by TNC in the Section 6 of this document. EAP-TNC encapsulates TNCCS messages so that they can be carried over tunneled EAP methods using standard attribute value pairs.

The tunneled EAP Method creates a cryptographically protected tunnel over EAP. It then carries a sequence of EAP frames over the tunnel it has created. The EAP Peer and Authenticator exchange EAP messages and manage EAP negotiation and protocol sequencing. EAP is described in [4], and the EAP state machine in [8].

The access client initiates the access control dialog with the protocol translator. 802.1X and IKEv2 are examples of access protocols. In this document, 802.1X and IKEv2 are shown as example access use cases. However, other access protocols may be used as long as they support EAP authentication.

An AAA protocol is used to communicate between the Protocol Translator and the NAA. This AAA protocol carries EAP messages for IF-T as well as IF-PEP. RADIUS and Diameter are examples of AAA protocols.

3.1.1 Tunneling

IF-TNCCS messages are carried within the EAP-TNC method. The details of the EAP-TNC method are defined in the Section 6.

EAP-TNC can be carried within any EAP tunneled method that supports inner EAP methods as an “inner EAP stream.” This inner EAP stream is carried in different ways depending upon the particular tunneled EAP method. These differences are described below.

Interoperability requires that both sides use the same tunneled EAP method, and that peer and authenticator vendors implement to the same EAP-TNC standard. This allows a client with tunneled method peer from one vendor can communicate with a tunneled method authenticator from a different vendor,

EAP messages are carried by an access protocol (e.g., 802.1X) from the NAR to the Protocol Translator, and by RADIUS (or other AAA protocol) from the Protocol Translator to the NAA. If vendors implement according to EAP, access protocol, and AAA protocol specifications, then a peer from one vendor can talk with an authenticator from another.

Finally the access protocol on the client must work with the protocol translator. This specification provides examples of 802.1X and IKEv2 mapping in Section 4. Additional access protocol mappings will be specified later.

3.1.2 Protocol Encapsulation

The following figures show protocol encapsulation of messages on the client and server side. In both cases messages are exchanged with the protocol translator (e.g. wireless access point, switch, or gateway). The protocol translator removes an EAP message from the access protocol (e.g., 802.1X or IKEv2) and forwards it over the AAA protocol (e.g., RADIUS). It also does the reverse. Figure 3 shows how protocols are encapsulated at the different layers, and how a message “on the wire” looks.

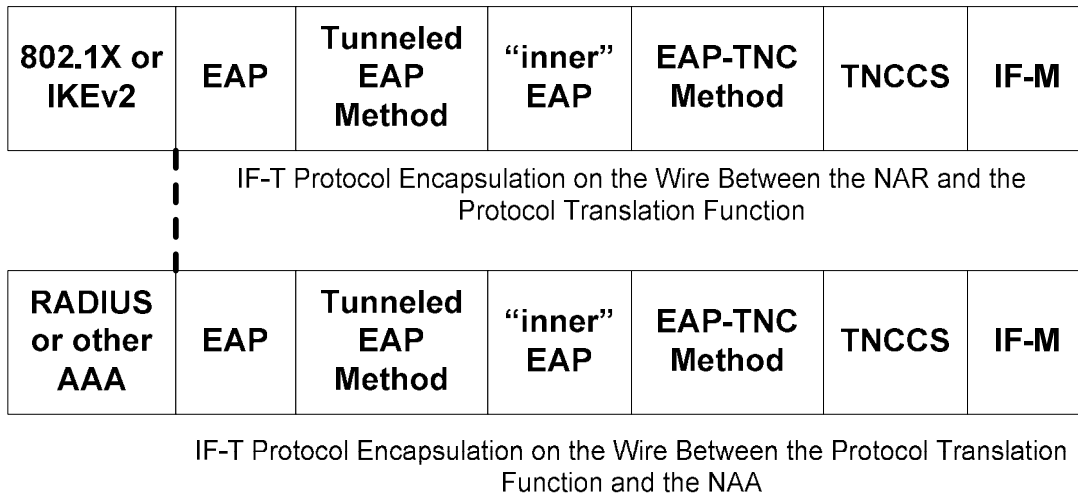


Figure 3. IF-T Protocol Encapsulation on the Wire

3.2 EAP-TNC

EAP-TNC is a very simple EAP method that MUST run over a tunneled EAP method. It is described in the Section 6. Figure 4 below shows how EAP-TNC carries an IF-TNCCS handshake.

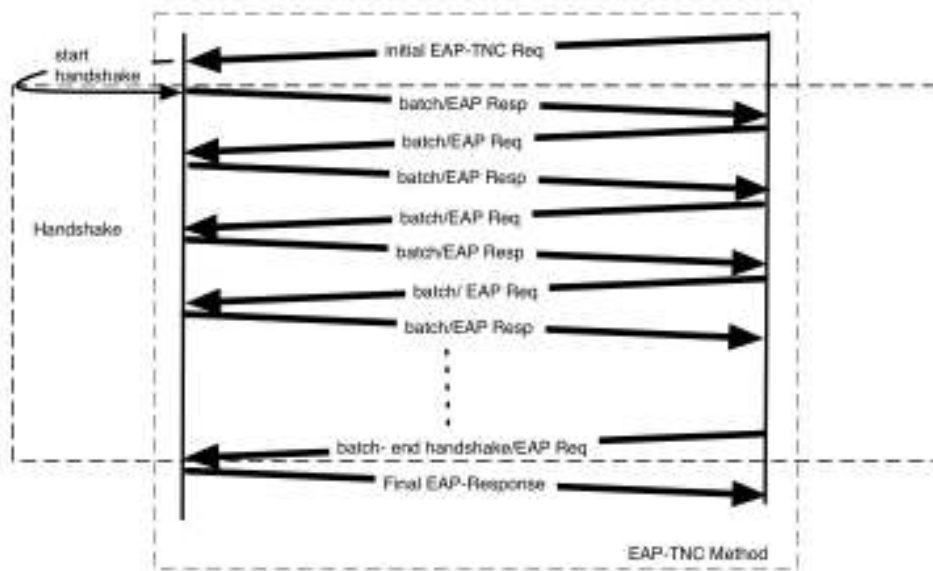


Figure 4. EAP-TNC TNCCS Handshake

EAP-TNC is an “inner” method. EAP messages for the EAP-TNC method are sent over a tunnel created by a tunneled EAP method. It is required that EAP-TNC be carried on a TLS tunnel in order that the conversation between the client and the server be protected. TLS provides integrity and confidentiality between the client and the RADIUS Server.

Note that Figure 4 does not show cases where fragmentation of a batch transfer is required. Fragmentation is described in Section 6.

3.3 Inner EAP Peer and Authenticator

Tunneled EAP methods make it possible to carry one or more “inner” EAP methods over a protected tunnel created in the first phase of the tunneled EAP method. Phase 1 is often called the “outer” method, and methods carried over the tunnel are called “inner” methods. In existing EAP methods a particular protocol element, either a Type-Length-Value (TLV) or Attribute-Value Pair (AVP) is defined for carrying “inner” EAP messages.

Inner EAP messages are sent end-to-end between inner EAP peer and authenticator. The inner peer and authenticator work just like the normal peer and authenticator, with a few exceptions noted below. Tunneled EAP methods that provide security for inner methods are allowed to carry sequences of EAP methods. In addition, most tunneled EAP methods allow the peer and authenticator pair to signal each other’s tunnel endpoint using special AVPs.

At the conclusion of the inner EAP-TNC method, if EAP-TNC has established a shared secret between the parties (such as when Diffie-Hellman Pre-Negotiation is employed), the shared secret (or a derivative thereof) MUST be exported to the outer tunneled method for mixing with its session keys. It’s expected that the outer method will cryptographically mix any existing keys for the session with those exported from each inner method (e.g. EAP-TNC) and at some point after the inner methods have completed perform an operation to assure both parties have computed the same mixed keys. This might be performed by having a final message roundtrip encrypted in the final mixed keys.

3.4 Tunneled EAP Methods

A number of Tunneled EAP methods have been implemented by different vendors. These methods all consist of two phases: the first phase creates a TLS tunnel over EAP; and the second phase carries other information protected using the TLS tunnel. A major intent of these methods has been to provide a secure path over which other authentication or authorization dialogs can be done. The outer tunneled EAP method secures the inner dialogs. This allows otherwise insecure EAP methods to be used securely as long as the outer EAP method meets the security requirements.

Because the outer EAP method provides protection against a wide variety of active attacks, the inner EAP-TNC method largely focuses on reporting of integrity information. Hence, EAP-TNC MUST NOT be used as a stand-alone method. EAP-TNC MUST only be used as an inner method within a protected tunneled EAP created by an outer EAP method.

The currently defined EAP-TNC inner method does not provide its own user or platform authentication mechanisms. EAP-TNC message payloads may carry IF-M messages that include additional authentications, but EAP-TNC does not depend upon and is not aware of, such services occurring. EAP-TNC SHOULD be used within a tunneled EAP method that provides authentication or can carry other authentication methods within the tunnel. If the authentication method is itself a tunneled EAP method, the tunneled EAP method MUST allow a sequence of EAP methods to be carried within it. It is assumed that EAP-TNC may be run in addition to other authentication methods within the tunneled EAP method.

The following sections provide further guidance on using EAP-TNC with specific tunneled EAP methods.

Note: Existing Tunneled EAP methods are not officially recognized by any standards body. The ones discussed here have all been described as Internet Drafts in the IETF. Specifications for these have been submitted to the IETF draft repository. Items in the IETF Draft repository are removed after 6 months if they are not modified. The specifications for some of the Tunneled EAP methods described in this document have been timed out. However, they are available from other Web sites.

3.4.1 EAP-FAST and PEAPv2

EAP Flexible Authentication via Secure Tunneling (FAST) [16] and Protected EAP Version 2 (PEAPv2) [23] are both tunneled EAP methods that define how to run multiple inner EAP methods. In EAP-FAST and PEAPv2, EAP-TNC is carried in an EAP payload TLV.

EAP FAST and PEAPv2 also provide an “over the tunnel” protocol with messages between tunnel endpoints at each end. This protocol manages messages sent over the TLS tunnel created in its initial phase. This “over the tunnel” protocol includes messages such as intermediate success/fail and crypto binding.

3.4.2 EAP-TTLS

In EAP-TTLS [20], EAP-TNC is carried in an EAP AVP. The latest specification for TTLS defines mechanism to perform a sequence of inner EAP methods.

3.4.3 PEAPv0/1

Neither PEAPv0 nor PEAPv1 define how run multiple inner EAP methods. EAP sequences are not prohibited in PEAPv0/1 but are implementation dependent. Hence, in PEAPv0/1, EAP-TNC would be carried directly on the tunnel.

When using PEAPv0/1 with EAP-TNC and a traditional authentication method, the server is responsible for sending the sequence of inner EAP methods and checking results.

The following provides an example as to how this could be done. However, this approach is not recommended by this specification as it would break countermeasures to the MiTM attack described in Section 5.4.5 that require EAP-TNC to be able to export a key that is integrated into

the tunneling methods protective keys. This section is informative only and SHOULD NOT be used without additional protections.

Note, one method of supporting inner EAP sequences that can easily be implemented is to use the same mechanism as defined for TTLS version 0: The authentication server starts the next EAP method by sending EAP-Request with the new method type once the previous method is completed but before sending inner result indication.

3.5 EAP-TNC Sequencing

EAP-TNC is one of a number of possible dialogs that can take place over the tunnel created in the first phase of tunneled EAP methods. TNC does not require any specific ordering of dialogs.

Possible scenarios include

1. EAP-TNC is the only dialog that runs over the tunnel. In this case Phase 1 of the tunneled EAP method provides client and server authentication as needed.
2. EAP-TNC is used in addition to one or more other inner EAP methods which might include a user authentication dialog all within the same EAP outer tunnel. For example EAP-TNC could run either before or after MD5 or MSCHAP allowing for an authenticated identity to be linked to the TNC integrity exchange.

It should also be noted that efficiency should be considered when using and ordering multiple EAP dialogs.

4 Access Protocol Bindings (Informative)

This section shows example protocol bindings that allow access protocols that use EAP authentication to support TNC capabilities using tunneled EAP methods. This section is non-normative.

Figure 5 shows the relationship of Access client and higher level protocols used to provide IF-T.

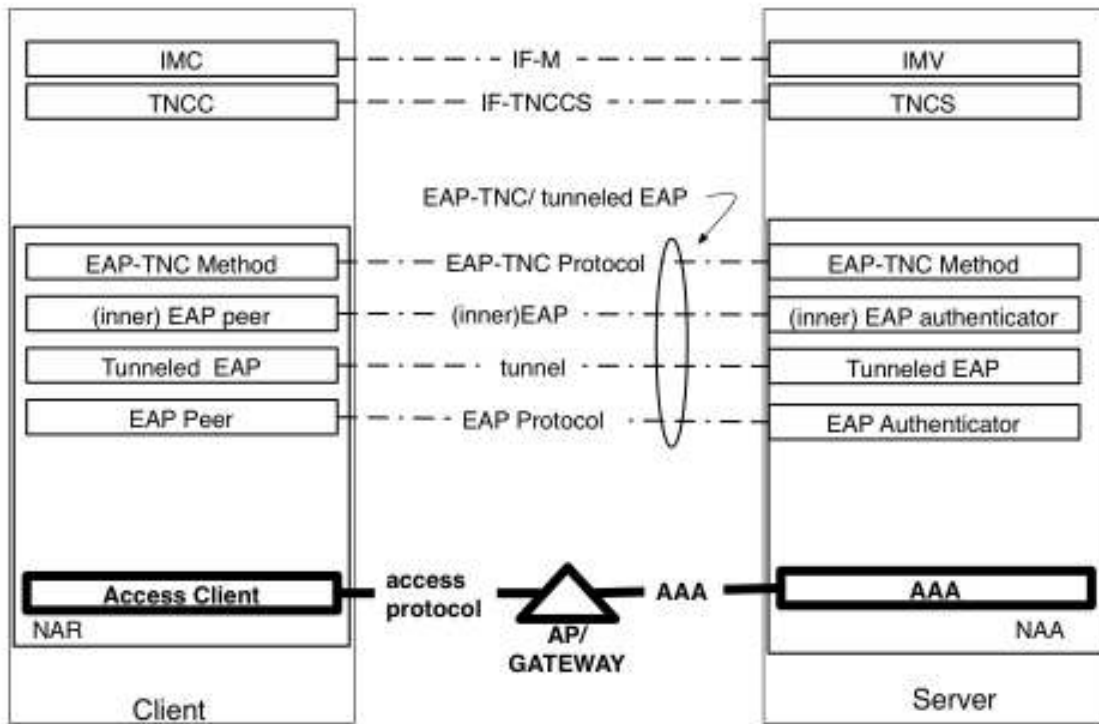


Figure 5. Access Client and Gateway Relationship

The examples in the following sections show how two different access protocols, 802.1X and IKEv2, interface with a tunnel EAP method to provide TNC functionality.

Access protocols are used by the NAR to establish network connectivity (e.g. at the link layer for 802.1X or the network layer for IKEv2 with IPsec.) Other access protocols exist that fit the TNC model including: PPP for Dial Access and 802.16e.

4.1 802.1X

Figure 6 below shows how 802.1X [7] can use EAP-TNC to facilitate incorporating TNC capabilities into access decisions. 802.1X is used to control access to 802.3 (wired Ethernet switch) and 802.11 (wireless) networks.

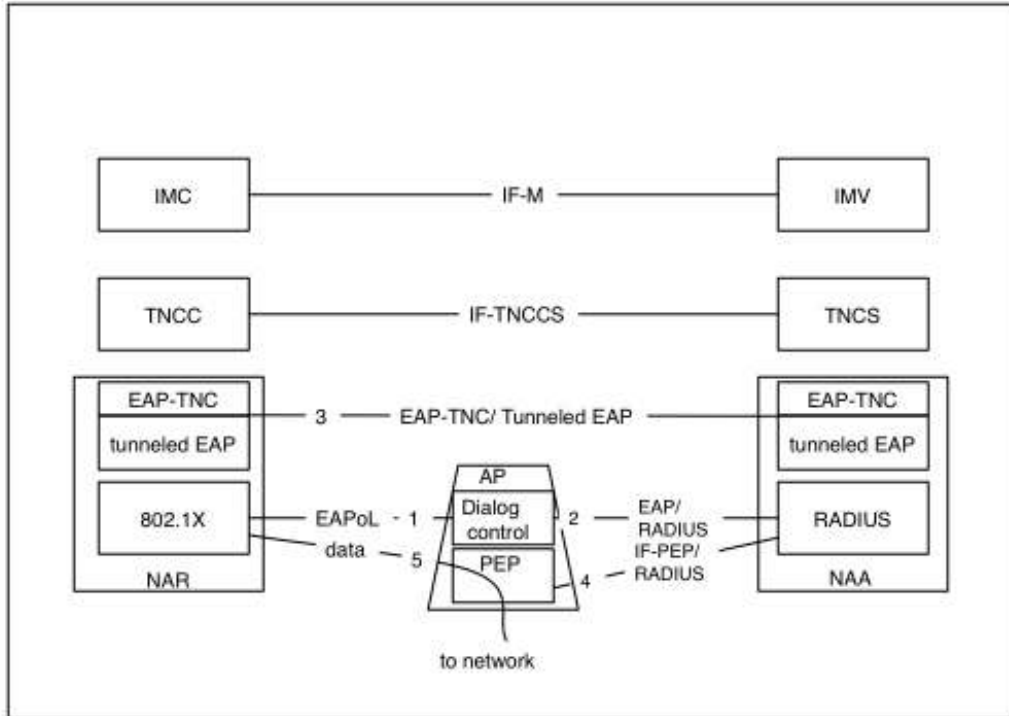


Figure 6. IF-T Over 802.X

This diagram explodes the low level protocol, showing EAPoL (EAP over LAN) being used between the 802.1X supplicant (client) and the 802.1X authenticator (wireless access point or fixed LAN switch), and RADIUS being used between the 802.1X authenticator and the 802.1X authentication server (RADIUS server). EAPoL and RADIUS carry EAP messages, and the authenticator creates an end-to-end EAP path between the client and server by moving EAP messages between the two protocols.

In addition to carrying EAP messages, both EAPoL and RADIUS have other functions and messages. For example, EAPoL includes EAPoL-Start, EAPoL-Logoff, and EAPoL-Key messages, all of which communicate only between the client and AP. On the server side, RADIUS messages typically may contain several attributes in addition to EAP messages.

Thus, at the bottom layer there are actually five dialogs, as shown in Figure 6.

1. The 802.1X dialog between the client and AP to control client access
2. The RADIUS dialog between AP and server to authorize access
3. The EAP dialog between the client and server to authenticate and validate the client
4. The RADIUS dialog between the server and the PEP on the AP. This dialog is actually a command that tells the AP how to respond to the access request
5. Data from the client to the AP which is controlled by the PEP.

4.2 IKEv2

IKEv2 is used to negotiate security settings and ultimately establish shared keys normally used with IPsec to protect subsequent packet exchanges. For example a client system may use IKEv2 to establish keys with a VPN gateway. These keys can then be used to create an encrypted IPsec

tunnel between the client and gateway. Figure 7 shows how TNC can be incorporated into this capability.

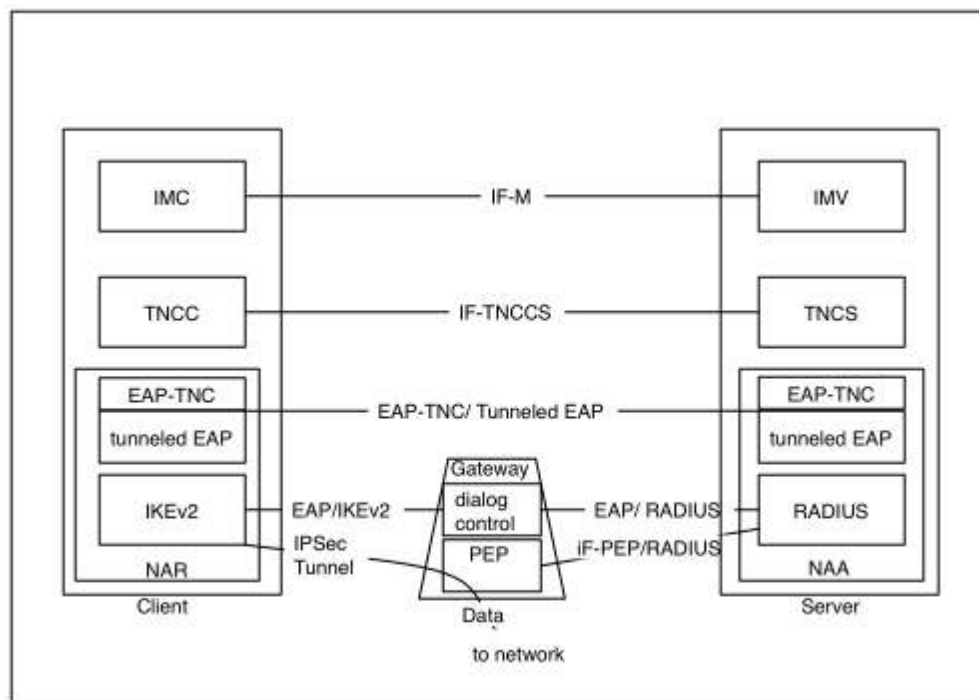


Figure 7. IF-T Over IKEv2

IKEv2 supports use of EAP as an authentication framework as one of the possible standard methods of providing authentication. When EAP is used with IKEv2, it allows the gateway to use RADIUS to request authorization from a remote server, just as an authenticator does for 802.1X. One reason for allowing EAP in IKEv2 is to permit use of legacy authentication mechanisms, such as passwords or OTP one-way schemes, in addition to pre-shared-secret or certificate-based mutual authentication mechanisms supported in IKEv1.

IKEv2 with TNC MUST use EAP-TNC as an inner method of a tunneled EAP method. This is necessary because IKEv2 doesn't support the ability to make an authorization decision based on a sequence of EAP methods. In the case where EAP-TNC and another EAP dialog such as user authentication (e.g., EAP-MD5) run over the outer tunneled EAP method, the outer method provides aggregation of the result of the multiple inner methods.

Thus, at the bottom layer there are the equivalent five dialogs (as shown in Figure 6) using IKE in Figure 7.

1. The IKEv2 dialog between the client and AP to control client access
2. The RADIUS dialog between AP and server to authorize access
3. The EAP dialog (over IKEv2) between the client and server to authenticate and validate the client
4. The RADIUS dialog between the server and the PEP on the AP. This dialog is actually a command that tells the AP how to respond to the access request
5. Data from the client to the AP which is controlled by the PEP.

4.2.1 IKEv2 Dialog

The basic IKEv2 authentication sequence consists of a four-message handshake between the two IKE peers, referred to as the initiator and the responder. In this case the initiator is the endpoint system requesting access and the IPsec gateway might be the responder. The first two messages carry an ephemeral Diffie-Hellman exchange and cipher suite negotiation parameters. In the third message, the initiator proves its identity by sending a signed AUTH payload. In the fourth message, the responder proves its identity with a signed AUTH payload. The signature algorithm may be based on a pre-shared secret or on RSA X.509 certificates containing an RSA public key.

When using EAP for authentication, the remote access client omits the AUTH payload in the third message while simply declaring its identity. This signals the responder, such as an IKEv2 gateway, that EAP authentication is requested. If supported and configured, the responder returns a fourth message containing an EAP request. IKEv2 endpoints then carry EAP messages until the EAP authentication is complete. Note: IKEv2 authentication may be provided by the EAP method, and when doing TNC with IKEv2 it is required to use EAP as the IKEv2 Authentication method.

When doing TNC, the initial outer tunneled EAP-method creates its own shared key. That shared key is used by both the initiator and responder to generate AUTH payloads using the syntax for shared secrets specified in [19]. The shared key from EAP is the field from the EAP specification named Master Session Key (MSK).

5 Security Considerations

5.1 Threat Model

The TNC threat model asserts that there are two parties interested in interoperating (client and server) and a third (attacker) interested in exploiting vulnerabilities. IF-T provides protection between the NAR and the NAA against attacks on the communication path between them. IF-T authenticates the NAR and NAA, and provides a secure channel for carrying TNCCS messages. The security includes integrity protection against data modification and encryption to protect against eavesdropping.

5.1.1 Threats

The attacker's goals with regard to IF-T are assumed to be the following.

1. Exploit a vulnerability on the end system to defeat the protection provided by IF-T
2. Attack the IF-T authentication dialog to enable a spoofing attack
3. Mount a cryptographic attack on IF-T to expose TNC data
4. Mount a Man in the Middle Attack on the initial access attempt

5.2 IF-T Capabilities

5.2.1 Interaction with Platform Trust Services (PTS)

The PTS is a local service that can optionally be leveraged by the TNC architecture to measure and report upon the state of software present on the system which TNC relies upon for its security. The PTS can leverage the TPM and other trusted components on the system in such a way that it could provide for protected evidence and optionally prevention of malware from running on the system. TNC verifiers can request evidence from the client PTS in order to be more assured that the responses from the other IMCs are trustworthy and not subject to subversion by malware running on the system. Such information can be factored into the network connection and subsequent access decisions made by the verifier.

One major benefit of TNC participation in TCG is the ability to provide the client and server a facility to cryptographically verify the integrity of the TNC components of the peer system. IF-T may also be provided proof of cryptographic integrity of all or part of the peer system as a whole.

Cryptographic verification of client modules by PTS is done by either 1) requesting PTS to measure IF-T modules prior to an IF-T request, or 2) registering IF-T modules with PTS and automatically measuring them during the boot process of the platform. Cryptographic measurements of client side IF-T modules may be sent to the Server as part of data sent by PTS-IMC and checked by PTS-IMV. PTS and the IF-M protocol used by PTS IMC and IMV are described in a forthcoming specification. The local IF-PTS interface (used by PTS IMC) is described in the IF-PTS Specification[14]. A set of XML-based schemas used by the PTS to report integrity information can be found at the TCG website[15].

In addition to measurement of modules, IF-T modules may interact directly with other aspects of the TCG Trusted Platform architecture to utilize cryptographic signing and encryption of messages sent between client and server.

5.2.2 Authentication Protection

For this specification of mapping IF-T to tunneled EAP methods, authentication protection is provided by the tunneled EAP method optionally augmented by use of other authenticating inner EAP methods. For this protocol binding, TNC recommends that the outer tunnel method be based on either a secure mutual authentication using symmetric keys or one way or mutual public key authentication.

5.2.3 Protection of TNC Data

TNC data is protected by the tunnel provided by the outer method of the tunneled EAP method. The tunnel is typically provided using TLS. The amount of data in a TNC exchange is likely to be limited to that required for an authorization dialog, which is typically small. See section 5.4.2 for a summary of the required security protections provided around the EAP-TNC data messages by the outer tunneling methods.

5.3 Some Attack Scenarios

IF-T is concerned with the communication channel between the TNCC and TNCS. An attacker may use the following capabilities or techniques. The protocol binding specification protects against all of these types of attack:

1. Eavesdropping
 - a. To learn client vulnerabilities
 - b. To extract client identification to impersonate client in TNCCS handshakes.
2. Modification
 - a. To represent a client as in compliance when not, so the server does not remediate, permitting an exploit against the client
 - b. To represent client as out of compliance, so that server isolates or blocks
 - c. To misrepresent the TNCS recommendation to the client
 - d. To deliver erroneous remediation instructions to the client
3. Impersonation
 - a. Of server, to discover vulnerabilities
 - b. Of client, to obtain or infer reference measurement data

These can all be mounted using a man in the middle (MITM) attack.

5.4 Philosophy of Protection

5.4.1 Scope of Protection

IF-T as mapped to tunneled EAP methods is a network protocol providing a protected transport to the TNCC and TNCS for message exchange. Because it is a protocol, the protections it affords are limited to the network communication channel and do not extend beyond the IF-T interface. Protocols layered on top of IF-T can assume the presence of the mandated security protections for IF-T described in section 5.4.2, but SHOULD provide security for higher layer protocol attacks (e.g. message falsification) that impact their ability to perform the higher layer function.

IF-T does not offer protection from local attacks. If malware has infected a system and is capable of interception, replacement or deletion of IMC or IMV messages before they receive IF-T's protections, IF-T will not be able to detect or prevent this from occurring. Use of proper host-based security protection is necessary to address such attacks and assure the proper operation of the IF-T mechanism. Other TNC architecture specifications such as IF-PTS SHOULD be used to address such attacks.

5.4.2 Minimum security Protection

In order for higher level protocols such as IF-TNCCS and IF-M to make assumptions as to the minimum level of protection that IF-T provides, this section describes the required security

properties that any IF-T MUST meet. All IF-T bindings MUST include an explanation of how these properties will be achieved.

The security requirements described in this section MUST be implemented in any product claiming to be TNC compliant. The decision of whether a particular deployment chooses to use these protections is a deployment issue. A customer may choose to avoid potential deployment issues or performance penalties associated with the use of cryptography when the required protection has been achieved through other mechanisms (e.g. physical isolation). If security mechanisms may be deactivated by policy, an implementation should offer an interface to query how a message will be (or was) protected by IF-T.

Compliant IF-T bindings and products implementing them using tunneled EAP methods MUST support:

1. Cryptographic authentication of the NAA to the NAR
2. NAR authentication and TNC dialog protected by at least a cryptographic transport
3. Encryption of the message stream tied to at least the transport authentication
4. Cryptographic integrity protection of the message tied to at least the transport authentication
5. Protection against replay attack

Having the NAR always authenticate the NAA provides assurance to the NAR that the NAA is authentic (not a rogue or MITM) prior to disclosing secret or potentially privacy sensitive information about what is running or configured on the system. However the NAA's policy may allow for the delay of the authentication of the NAR until a suitable protected channel has been established allowing for non-cryptographic NAR credentials (e.g. username/password) to be used. Whether the communication channel is established with both or one party performing a cryptographic authentication, the resulting channel needs to provide strong integrity and confidentiality protection to its contents. These protections are to be bound to at least the authentication of the NAA, so the session is cryptographically bound to a particular authentication event.

5.4.3 Tunneled EAP Minimum Protections

This section discusses how EAP-TNC used within the tunneled EAP methods described in section 3.4 meets the IF-T requirements from section 5.4.2 above.

EAP-FAST[16], PEAPv0/v2[22,23], and TTLS[20,21] all make use of TLS [9] to protect the transport of information between the NAR and NAA. Each of these has two phases, and in the first phase a TLS tunnel is established between NAR and NAA, and in the second phase the tunnel is used to pass other information. IF-T requires that establishing this tunnel include authentication of the NAA by the NAR.

The phase two dialog may include authentication of the user by doing other EAP methods or in the case of TTLS by using non-EAP authentication dialogs. EAP-TNC is also carried by the phase 2 tunnel. The phase 2 TLS tunnel provides support for requirements 2-5 above.

With all these methods, a cryptographic key is derived from the authentication that may be used to secure later transmissions. For these methods this means that server side certificates are required. Within each tunneled EAP method will exist a set of inner EAP method (or an equivalent using TLVs if inner methods are directly supported.) These inner methods may perform additional security handshakes including more granular authentications or exchanges of integrity information (such as EAP-TNC.) At some point after the conclusion of each inner methods, some of the methods will export the established secret keys to the outer tunneling method. It's expected that the outer method will cryptographically mix these keys into any keys it is currently using to protect the session and perform a final operation to determine whether both

parties have arrived at the same mixed key. This is essential for detection of a number of nested method attacks (see 5.4.5 below for one such attack.)

5.4.4 Recommended Security Practices

In order to enable a strongly protected use of the TNC architecture for endpoint compliance, the following measures are necessary.

1. The NAA SHOULD authenticate the platform integrity of the trusted components on the TNC client to prevent falsification of integrity measurement reports about the current state of the platform. This would involve use of other TCG and TNC components (e.g. TPM and PTS) via IF-PTS.
2. The NAA and NAR SHOULD protect authentication tokens such as: private keys, trust anchor public keys/certs, and traffic encryption keys from unauthorized access. Theft of cryptographic material can be catastrophic to the security of the system since the party could impersonate a party in the session. Countermeasures to this type of attack also may involve use of the platform's TPM or a secure key storage device.
3. To ensure that the endpoint authenticated with IF-T is the same one used for network access, either the cryptographic keys derived from IF-T SHOULD be used to authenticate subsequent network traffic (as with 802.11i) or suitable other protections against this attack SHOULD be employed.
4. When the use of PTS for verification of endpoint integrity is combined with user or platform authentication, the authentication SHOULD be done with credentials tied to the TPM (like SKAE) and the PDP SHOULD verify that the credentials are associated with the same TPM as the one used for the PTS exchange. See Section 5.4.5 for a discussion of the attack and how this countermeasure operates.

5.4.5 Protecting against MiTM attacks against EAP-TNC

The IF-T binding for tunneled EAP methods works on the premise that the tunneling method is capable of carrying (and protecting) various inner methods that perform additional security operations to establish the authenticity and integrity of the NAR. While this model is very flexible since it allows for the variety of existing EAP methods to be leveraged within the tunnel, it may introduce vulnerabilities. One such vulnerability is an attack described in "Man-in-the-Middle Attack against Tunneled Authentication Protocols" described in the 2003 Security Protocols Workshop paper by Asokan, Niemi, and Nyberg [12].

5.4.5.1 Example Attack Against EAP Nested Tunnels

This section describes a TNC oriented example of the Asokan, Niemi and Nyberg attack against an environment where Trusted Platforms are required to join the network. Trusted Platforms include an enabled TPM and a TBB measuring each software component as it is loaded so the network can assess what software is running on the system and detect malware.

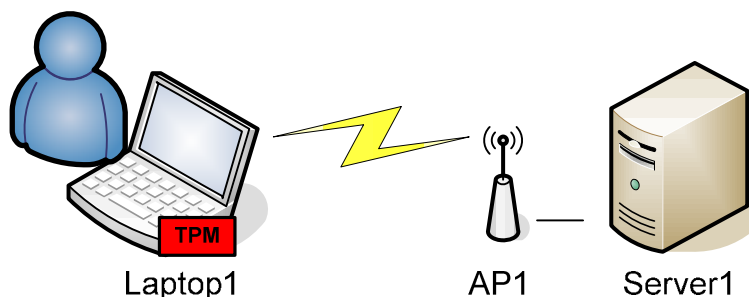


Figure 8. Clean Laptop Accessing Wireless Network

In this example, the NAR is called Laptop1 and the NAA is referred to as Server1 (see figure 8.) Laptop1 contains a Trusted Platform with an operating TPM and TBB and is wishing to gain access to the company intranet via an Access Point (AP1.) Server1 requires user authentication using PEAP, and verification of a PTS generated Integrity Report describing the running software on the laptop. Therefore, if Laptop1 gets compromised via a software malware attack (e.g. rootkit), it cannot get network access since the quoted PCR values in the Integrity Report will not match the expected values, and Server1 will not allow access.

Next, let's assume that Laptop1 does get compromised and can be controlled remotely by the attacker (maybe over the laptop's EV-DO card independent of the WLAN NIC used in this example.) So now sometime after the CRTM and early RTMs have performed their early platform measurements the attacker's malware is loaded and can communicate with the attacker. Now the attacker sets up his own equipment, Laptop2, AP2, and Server2 (see figure 9) on a stub network to aid his attack. Laptop2 is configured to match the "good" configuration that would be accepted by Server1 and even includes an enabled TPM and TBB. However, Server2 contains malware to aid the attacker obtain the Integrity Report from Laptop2.

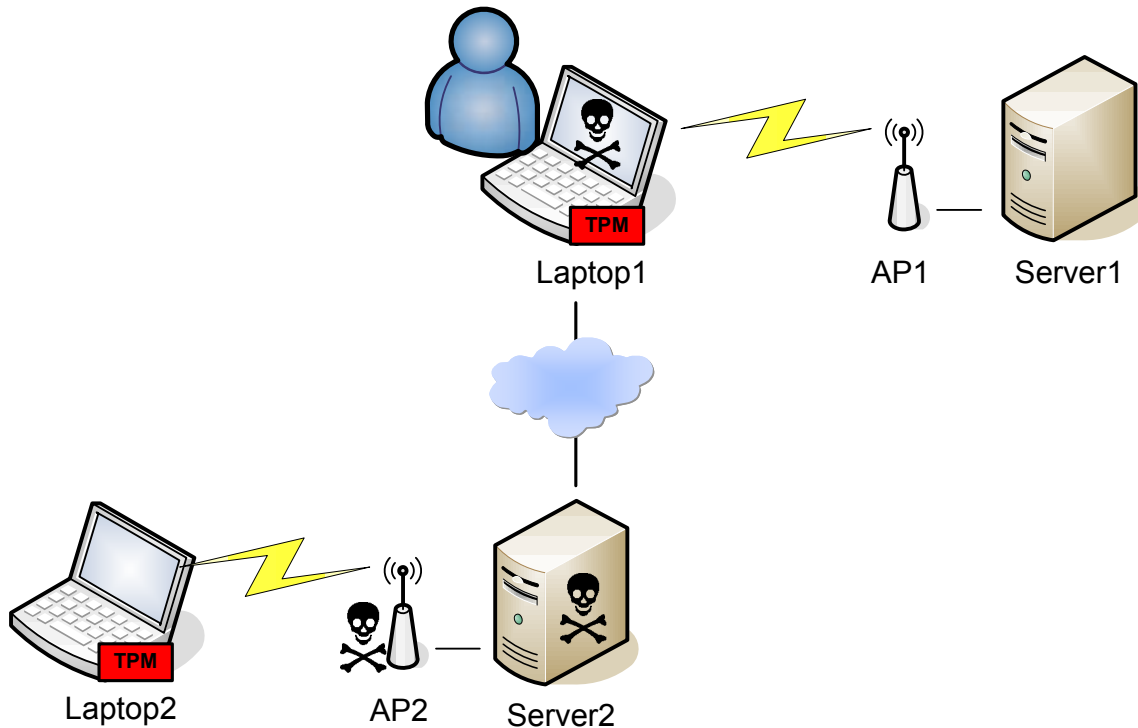


Figure 9. Attacker MiTM Network for Accessing Company Network

Next, the honest (unaware of the malware) user of Laptop1 tries to connect to the network, see figure 10 for flow diagram. Eventually, PEAP is started and the user is authenticated (steps 1-2). At the same time, the attacker uses Laptop2 and starts PEAP with his own server, and does user authentication on the stub network (steps 3-4).

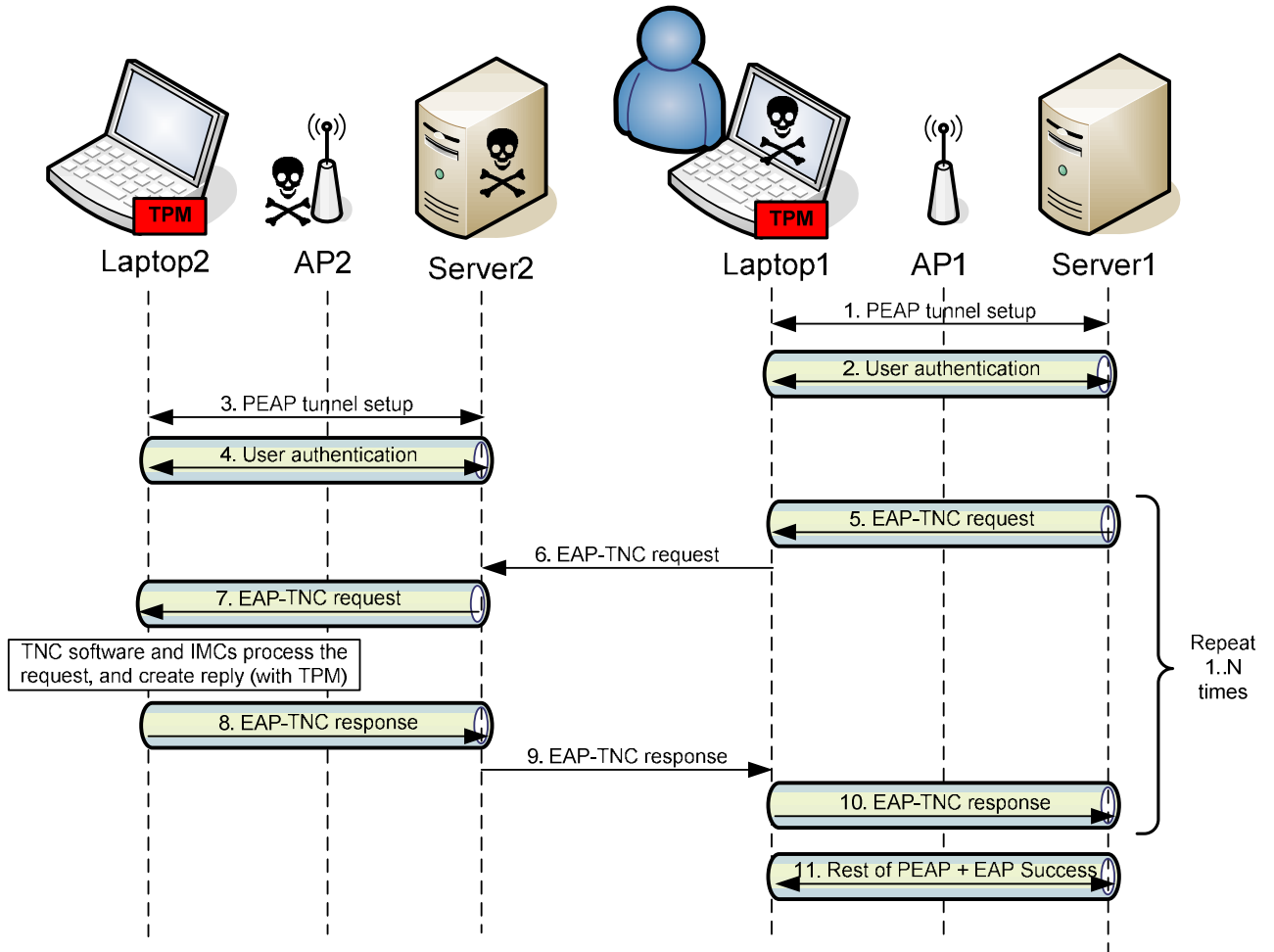


Figure 10. MiTM Attack Flows

Next, the TNC protocols start within the authenticated PEAP tunnel. At this point Server2 begins to interact with the compromised Laptop1. Server1 sends an EAP-TNC request to Laptop1, which being compromised, forwards it to Server2 and eventually Laptop2. Laptop2 creates a response including an Integrity Report containing a TPM quote of the PCRs of Laptop2. The Integrity Report is sent to Server2 which forwards it to Laptop1 who presents it to Server1 as a description of the contents of Laptop1. Steps 5-10 can be repeated as often is as required by Server1.

Eventually the exchanges will succeed because Server1 is unable to tell that the Integrity Report it is receiving do not describe the same system that participated in the authenticated PEAP session, thus compromised Laptop1 gets access to the network despite the requirement for TPM rooted measurements.

5.4.5.2 Countermeasures using the Trusted Platform and TPM

Protection against this form of attack involves providing Server1 with a strong linkage between the party that performed the authentication and outer method with the party providing integrity information via the EAP-TNC inner method. There are several ways this linkage might be established based upon leveraging the secret keys stored within the TPM of the valid (healthy) platform in such a way that the attacker is unable to successfully replay the responses to join the network. Version 1.0 of this specification defined an approach involving the use of certificates identified as containing a public key bound to a TPM resident private key. For backward

compatibility with 1.0, the 1.1 version of this specification leaves this countermeasure while adding a second approach. Implementations of IF-T 1.1 SHOULD support the second countermeasure based on a Diffie-Hellman pre-negotiation if they expect to be used on a system containing a TPM. 1.1 compliant implementations MAY also include the support for special certificates as well.

5.4.5.2.1 SKAE Certificates

The original countermeasure (present in version 1.0 of this specification) uses a non-migratable (or CMK) private key stored within a TPM and paired with a public key present in an X.509 certificate to perform the user or platform authentication of the outer tunneling method (e.g. using TLS). Because a single signed certificate can both identify the authenticated user (or platform) and bind that authentication to a particular TPM, this provides the strong linkage between the authenticating party and the TPM-based Integrity Report required to address this attack. The enrollment description and ASN.1 encoding for certifying that a private key is held within a TPM within an X.509 certificate is described in the SKAE [13] specification.

To understand how an SKAE is beneficial, we need to review how one is created during certificate enrollment. First the client system creates an AIK and obtains an AIK Credential from the Privacy CA. Next the client creates a “client identity authentication” key pair within the TPM and uses the TPM to “certify” that the key is only present within the TPM using the AIK. The TPM will produce evidence of this binding in a signed TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 structure. This structure is included in the proposed SKAE that is sent to the CA during enrollment for inclusion when creating the client identity authentication public key certificate. Now we have an X.509 identity certificate which contains signed evidence that the associated private key is housed in a TPM (thus a binding between a TPM and an identity.)

With SKAE, the verifier is expected to process the certificate as usual but also perform a validation of the SKAE’s evidence using the signing AIK public key. If the client can perform cryptographic operations using the identity private key, the verifier can be trust that it is the same platform with the described TPM and AIK. Similarly if the client system can perform cryptography using the AIK private key, the verifier can trust that is the same user (or platform) as identified by the client identity certificate.

Assuming that Laptop1 or Laptop2 were created with an identity certificate leveraging a TPM resident private key, neither party would be able to sign information as the other party because they wouldn’t have access to the private key. Clean laptop2 (created by the attacker) would likely have a difficult time obtaining an identity certificate in the first place for a legitimate user using TPM resident keys on Laptop2. The attacker would be unable to steal the private key for the identity certificate (with SKAE) on Laptop1 so would be unable to spoof that identity to Server1. Therefore if Laptop1 performed an authentication with this identity certificate to Server1 that understood SKAE (e.g. using a TLS tunneling EAP method) and later a quote came signed using an AIK from a different TPM, the verifier could detect this disparity.

5.4.5.2.2 Diffie-Hellman Pre-Negotiation (D-H PN)

The second approach introduced in version 1.1 of this specification involves the use of a Diffie-Hellman (D-H) Pre-Negotiation exchange immediately before the start of the EAP-TNC dialog to establish shared secret keys used in the Integrity Report and with the EAP outer method. The D-H PN exchange includes D-H public values and nonces to assure freshness of the session (for replay detection.) The resulting secret information derived from the exchanged nonces and D-H public values are known only to the two parties performing the D-H PN. These secrets are used to derive a secret that is used effectively as a per-session nonce during the TPM quote included in the Integrity Report and another value (also based on the contents of the TNC exchange) which is exported to the outer EAP method for subsequent mixing with its session keys. This has

the effect of linking the Integrity Report and the TNC message exchanges with the outer EAP method authentication.

At the conclusion of the EAP-TNC protocol including the new D-H pre-negotiation, a derivative of the established secret keys (described in Section 6.3.6, step 9) MUST be exported to the outer method which MUST cryptographically mix the keys into those used to protect the tunnel in order to address this attack. To assure that a MiTM is not present in this situation, the EAP outer method MUST assure both parties had knowledge of all the keys so likely would perform a final roundtrip exchange using the new mixed keys. The PDP must also check the integrity reports received from the client to make sure they indicate that the secret derived from the D-H PN was included (as ExternalData) within the PCR quote. If the client side of the tunnel is able to complete the exchange (involving use of the mixed key) and the integrity report sent by the client includes the use of the secret derived from the D-H PN, the client has proven that it was not a MiTM and that it reported its local state.

If an active MiTM (Laptop1) takes part in the D-H PN it establishes shared values with Server1 that aren't actually known by Laptop2 so can't be included in Laptop2's attested integrity report. This forces Laptop1 to have to graft on this secret information into the Integrity Report coming from Laptop2 and remove the different secret information computed by Laptop2. This graft is prevented by the cryptographic signature on the Integrity Report and over the requested PCRs and ExternalData (including the D-H PN secret) using keys stored in Laptop2's TPM.

If a MiTM (Laptop1) just forwarded the D-H PN protocol over a tunnel to Server2 so that clean Laptop2 and Server1 were selecting the D-H values and nonces, Laptop1 would be unable to determine the established secret since it lacks knowledge of any D-H private values and would be unable to complete the outer EAP tunnel exchange once the secret was mixed into the session keys.

In order for the MiTM protection to continue during the subsequent communications on the network, the communications SHOULD protect the data exchanges using keys based on the final tunneled EAP method keys that were mixed with the D-H secret keys. At present, 802.1X for wireless use has provisions for such a key to be used. However wired 802.1X lacks the use of keys to protect the communications. These unprotected flows are again vulnerable to a variety of attacks including alteration or replay by a MiTM. It is believed that the use of 802.1AE will address this issue so deployers should consider this if their threat model (especially with respect to wired 802.1X) warrants ongoing protections.

Although a MiTM attack against EAP-TNC could be employed against systems that do not include a TPM, there would be no point in mounting such a sophisticated attack. A compromised endpoint could simply send false measurements (Laptop1 could include an IMC that just sends information it knows would comply with policy even if that information didn't reflect the true state of the system.) Similarly if Laptop1 was not using its TBB to measure boot, the attacker might be able to falsify measurements in the TPM without the need for the stub network.

6 EAP-TNC Protocol Definition (Normative)

This section describes the EAP-TNC method. EAP-TNC is designed to encapsulate IF-TNCCS messages in a very simple EAP method. This allows IF-TNCCS messages to be carried within tunneled EAP methods, such as those previously described. The tunneled EAP methods are assumed to provide privacy and message integrity.

6.1 Protocol overview

EAP-TNC carries IF-TNCCS messages over tunneled EAP methods.

The EAP-TNC method begins with the EAP server sending an EAP Request message with the Start Bit set and with no data. This initial message may also include the D-H PN bit set to indicate a D-H Pre-Negotiation is requested. If the EAP peer on the Client supports and wishes to perform a D-H PN, then this handshake is started and will continue until completion or a party aborts the negotiation. If the D-H PN was not requested (or is undesired, the EAP Peer on the Client responds by sending an EAP Response containing an IF-TNCCS message created by the TNCC. The server may end the EAP method or may continue the handshake by sending an additional EAP Request containing IF-TNCCS message created by the TNCS, to which the client responds as before.

The authenticator (EAP server) may terminate the method after receiving any EAP Response. If the Authenticator decides to end the method, it signals to the "inner" EAP layer that its state is "done." The "inner" EAP layer acts on this in different ways depending on the particular tunneled method. In some cases it may indicate intermediate Success or Fail and in other cases it may do nothing to signal the end of the method, but may start another method or may terminate all "inner" methods.

6.1.1 State Machine

Here are the permissible state values for the EAP-TNC server and peer. These state values are as defined in the EAP State Machine RFC [8], and are included to help developers by incorporating the terminology used in that RFC.

EAP State for the Authenticator can take the values [Init | Continue Done]

EAP State for the Peer can take the values [Init | May Continue]

Note; Continue and Done states are not used in EAP-TNC

EAP Decision for the Authenticator can take the values [Success | Continue]

Note: Fail decision is not used in EAP-TNC.

EAP Decision for the Peer can take the values [UnconditionalSuccess]

Note: Fail and Conditional Success decisions are not used in EAP-TNC Peer.

6.1.2 Fragmentation

In most cases EAP-TNC fragmentation is not required because EAP-TNC is supposed to be used only inside another EAP method (tunneled EAP method) that provides protection for inner methods. All currently widely used tunneled EAP methods (TTLS, PEAP, and FAST) already support a fragmentation mechanism. But in some cases, fragmentation inside EAP-TNC may be required, when either a tunneled EAP method does not support fragmentation (like EAP-PSK) [4], or like in the case of FAST/PEAPv2, where the data length for inner EAP method cannot exceed $2^{16}-1$ bytes and this is not enough to send an IF-TNCCS message. This is because of a 16 bit length field limitation of TLV.

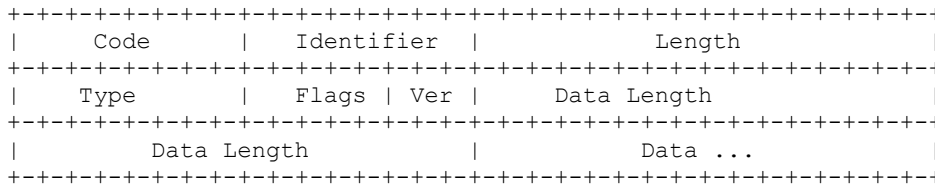
The fragmentation mechanism is defined the same way as for TLS based protocols (e.g., TLS, TTLS, PEAP, FAST). This uses Length included (L), More fragments (M) and Start (S) bits as well as Data Length field. The L flag is set to indicate presence of the Data Length field and

MUST be set only for the first fragment of a fragmented EAP-TNC message. The M flag MUST be set on all but the last fragment and MUST NOT be set on the last fragment. The S flag MUST be set only on the first EAP-TNC start message from the authenticator. The Data Length field defines the total length of the data being fragmented and is used to simplify buffer allocation.

A party that receives an EAP-TNC packet with the M-bit set MUST respond with acknowledgement packet – an EAP-TNC packet with no data. The sending party must wait for the acknowledgement packet before sending the next fragment.

6.1.3 EAP-TNC format

All fields are transmitted from left to right in network byte order.



Code

The Code field is one octet and identifies the type of the EAP packet:
 1 – EAP-Request
 2 – EAP-Response

Identifier

The Identifier field is one octet and aids in matching Responses with Requests.

Length

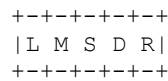
The length field is two octets and indicates the length, in octets, of the EAP packet starting from the Code field. If the EAP packet is fragmented, then this length will cover only the data portion carried in the fragment and thus doesn't provide the overall length of the EAP message.

Type

38

This is only a tentative assignment but implementers SHOULD go ahead and use it.

Flags



- L: Length included
 Indicates the presence of the Data Length field in the EAP-TNC message. It MUST be set for the first fragment of a fragmented EAP-TNC message and only for such a message. This bit MUST NOT be set for non-fragmented messages.
- M: More fragments
 Indicates that more fragments are to follow. MUST be set on all but the last fragment and MUST not be set on the last fragment of a fragmented EAP-TNC message.
- S: Start

Indicates the beginning of an EAP-TNC conversation. MUST be set only for the first message from the authenticator. If Start is set, the EAP message MUST NOT contain data.

D: Diffie-Hellman Pre-Negotiation

Indicates the use of a Diffie-Hellman (D-H) based exchange to protect the session from MiTM forms of attack. See section 6.3 for specifics of when to set this flag. Platforms not supporting this optional feature MUST NOT set this flag and MUST ignore it when included in a message.

R: Reserved

MUST be set to 0.

Version

1

Data Length

The Data Length is four octets optional field. It MUST be present if only if the L-bit is set. When present, it indicates the total length, before fragmentation, of a fragmented IF-TNCCS message. The Data Length MUST be sent in the first such fragment, and MUST NOT be sent in subsequent fragments.

Data

Variable length data. The length of the Data field in a particular EAP-TNC message may be determined by subtracting the length of the header fields from the value of the two octet Length field. Note, however, that this data may only be part of a longer fragmented IF-TNCCS message conveyed in multiple EAP-TNC messages.

6.1.4 EAP-TNC Compliance

Implementations compliant with this specification MUST support the fragmentation mechanism and IF-TNCCS messages up to 100 kilobytes in length.

6.1.5 Security claims

See RFC3748 Section 7.2:

Auth. mechanism:	None
Ciphersuite negotiation:	No
Mutual authentication:	No
Integrity protection:	No
Replay protection:	No
Confidentiality:	No
Key derivation:	Yes
Key strength:	Depends on D-H Group and Hash used
Dictionary attack resistant:	N/A
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	Yes
Channel binding:	No

6.2 EAP-TNC Conversation Example

Figure 4 below illustrates a representative EAP-TNC message exchange without the D-H pre-negotiation.

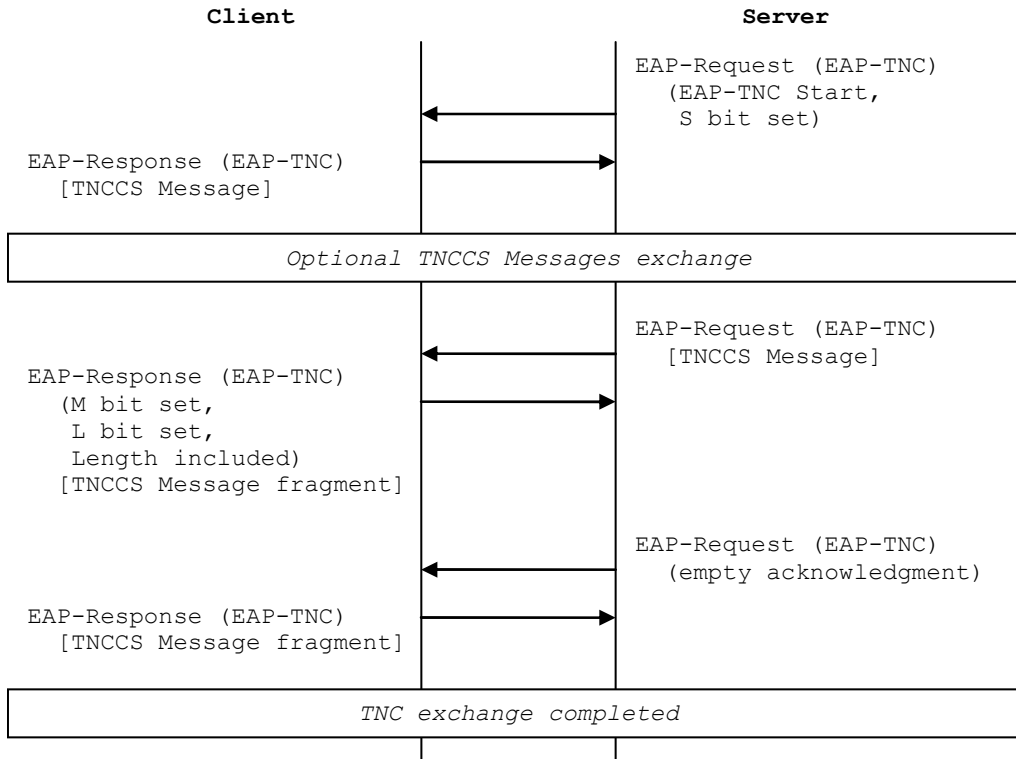


Figure 11. EAP-TNC Conversation Example

6.3 Diffie-Hellman (D-H) Pre-Negotiation

This section describes the Diffie-Hellman Pre-Negotiation (known as D-H PN) intended to establish shared keys in order to detect MiTM replay of TPM-based integrity information sent from access requestors (see section 5.4.5 for attack details.) This section describes the messages sent before the initial IF-TNCCS message. IF-T compliant access requestors SHOULD support D-H PN when used on an enabled Trusted Platform with a TPM and PTS. IF-T compliant authenticators SHOULD support this pre-negotiation even if not running on a Trusted Platform as validation doesn't require a Trusted Platform.

The deployment-time use of D-H PN SHOULD be policy driven. The access requestor MAY support policy allowing for this feature to never be used to support certain privacy policies. The authenticator SHOULD support policy describing when to request the use of D-H PN and whether access requestors responding without this feature are allowed to proceed.

D-H PN was designed to enable it to be added without causing backward compatibility issues. This leverages the fact that clients only supporting 1.0 of this specification MUST ignore the use of the D flag. As a result the authenticator (who initiates a D-H PN request) can not assume the access requestor will pay attention to the D flag before processing the payload (as a 1.0 compliant client would do.) Therefore the authenticator MUST wait until the AR has sent a message indicating it supports D-H PN (D flag set) before sending messages with the D-H PN described below. This decision causes a full roundtrip to occur prior to exchanging D-H PN messages.

6.3.1 Use of D Flag

The use of the "D" flag in the EAP-TNC header MUST follow very strict rules described in Section 6.3. If the D flag is 1 this indicates the data field of the EAP-TNC message MUST only contain the pre-negotiation information or be empty (as in the initial exchange messages) and not IF-TNCCS messages. IF-TNCCS messages MUST NOT be included in messages with the D flag

set to 1. Either entity MAY set the D flag to 0 at any time indicating it does not wish to (or is incapable of) perform the D-H PN exchange. The other party MAY then determine whether to proceed with the dialog without a D-H PN exchange.

6.3.2 D-H Pre-Negotiation Message Syntax

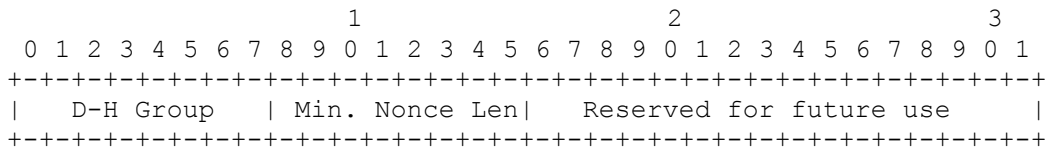
This section describes the format of the data field within each of the four D-H PN messages exchanged. These messages are only present in the data field when the D flag is set to 1. The D flag MUST be set for all D-H PN messages. If the Authenticator or Access Requestor receives a message with the D flag set to 0 in the middle of a D-H PN exchange, it SHOULD interpret this as meaning that the sender has decided to terminate the D-H PN exchange but would like to proceed with the TNC exchange. The receiver MAY proceed or terminate the entire TNC exchange. The messages are presented in the order they would appear in a D-H PN message exchange.

6.3.2.1 D-H PN Hello Request Format (sent by Authenticator)

The data field of the initial D-H PN message from the authenticator MUST be empty for backward compatibility. This message occurs at the start of a session with the start bit set to 1 and the D flag set to 1 (indicating a desire to initiate D-H PN without sending a data field that would be confusing to a pre-1.1 access requestor who might ignore the D flag.)

6.3.2.2 D-H PN Hello Response Format (sent by Access Requestor)

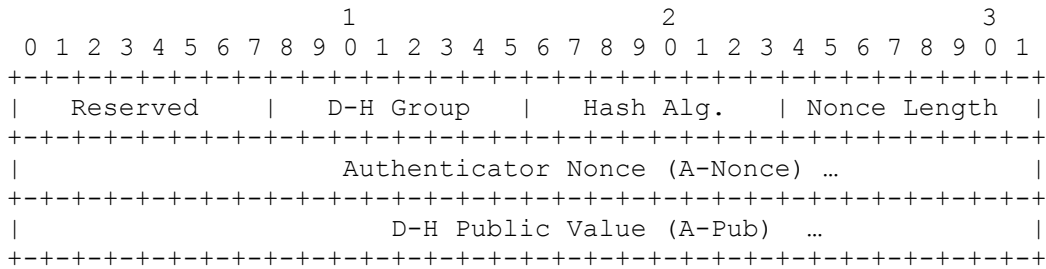
The data field of the initial response from the access requestor allows a 1.1 compliant access requestor to notify the authenticator that it is able and willing to perform a D-H PN (by replying with the D flag set to 1.) The defined protocol expects the Authenticator to lead the negotiation except for the D-H group which needs to be negotiated before the public value exchange can occur (since it affects the size.) In order to reduce the number of messages required this message includes the set of supported/preferred D-H groups and any minimum nonce size.



Field	Description
D-H Group	Bit field indicating the supported D-H groups. See section 6.3.5 for description of the D-H groups and their representation in this field. The access requestor policy MAY dictate what groups are allowable for a particular authenticator.
Min Nonce Len	Access requestor can send a minimum acceptable length for the nonce in bytes. This value should be set to 0 if there is no minimum required.

6.3.2.3 D-H PN Parameters Request Format (sent by Authenticator)

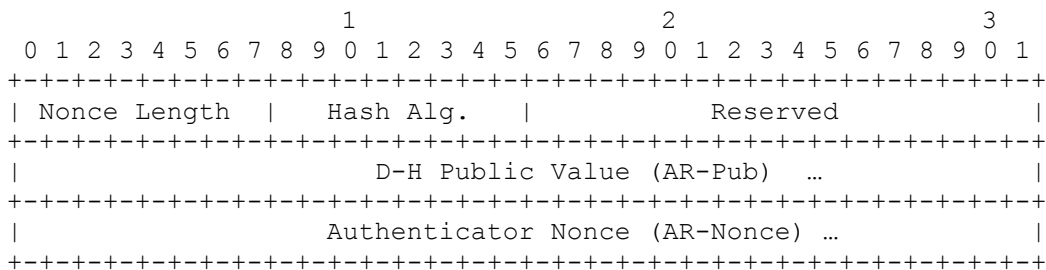
This is the data field of the authenticator's request message trying to finalize the negotiation of the parameters of the D-H PN exchange. This message proposes the authenticator's set of supported hash algorithms. The D-H group MUST be selected from the set offered in the Hello Response. If the Authenticator's policy does not allow the use of any of the D-H groups offered by the access requestor, this MUST result in the unsuccessful termination of the D-H PN. The Authenticator MAY decide to continue with an EAP-TNC exchange without the D-H PN protections by sending a message with no S or D flag and an empty data field. If the Authenticator decides to select an offered D-H group, the authenticator can offer its D-H public value (using the size from the selected group) and includes a nonce for freshness of the exchange in the following D-H PN Parameters Request message.



Field	Description
Reserved	This field MUST be set to 0 and MUST be ignored by 1.1 compliant implementations.
D-H Group	Selected D-H Group (single bit) from set offered in the access requestor's from D-H PN Hello Response message. See section 6.3.5 for description of the D-H groups and their representation in this field.
Hash Algorithm	Bit field indicating the set of supported hash algorithms. See section 6.3.4 for a description of the defined hash algorithms and their representation in this field.
Nonce Length	Length of the nonce field in bytes. This value MUST be greater than 16 and MUST be greater then or equal to the Min Nonce Len specified by the access requestor's D-H PN Hello Response message.
A-Nonce	High entropy random data used to assure the freshness of the session. Nonces MUST NOT be repeated or be predictable by other parties.
D-H Public Value	Authenticator's public value for this D-H exchange. The size of this field is determined by the authenticator selected D-H group to use. See section 6.3.5 for the lengths used for each D-H group.

6.3.2.4 D-H PN Parameters Response Format (sent by Access Requestor)

The data field of the access requestor's parameter response message which should complete the D-H PN exchange. This message establishes the particular hash algorithm for the derivation. Because the D-H group has been established, the access requestor can offer its D-H public value and include a nonce for freshness of the exchange. When the authenticator receives this message, both parties will have everything they need to perform the remaining transforms to derive the share secrets.



Field	Description
Nonce Length	Length of the nonce field in bytes. This value MUST be greater than 16 and SHOULD match the length used by the authenticator's nonce.
Hash Algorithm	Selected hash algorithm (single bit) from offered set for use later in D-H PN. See section 6.3.4 for a description of the defined hash algorithms.

Reserved	This field MUST be set to 0 and MUST be ignored by 1.1 compliant implementations.
D-H Public Value	Access requestor's public value for this D-H exchange. The size of this field is indicated by the selected D-H group.
AR-Nonce	High entropy random data used to assure the freshness of the session (nonces MUST NOT be repeated or be predictable.)

6.3.3 Diffie-Hellman Pre-Negotiation Protocol

This section describes the message exchange protocol which occurs during the D-H pre-negotiation. At any point during the exchange if a party is unwilling to accept the options offered by the other party, it SHOULD set the D flag to 0 indicating it no longer wishes to continue the D-H PN. This MAY result in a fallback to a standard request/response protocol if acceptable by both parties. All D-H PN protocol messages MUST have the D flag set to 1.

The following diagram shows the message sequence when the D-H PN protocol occurs during the EAP protocols.

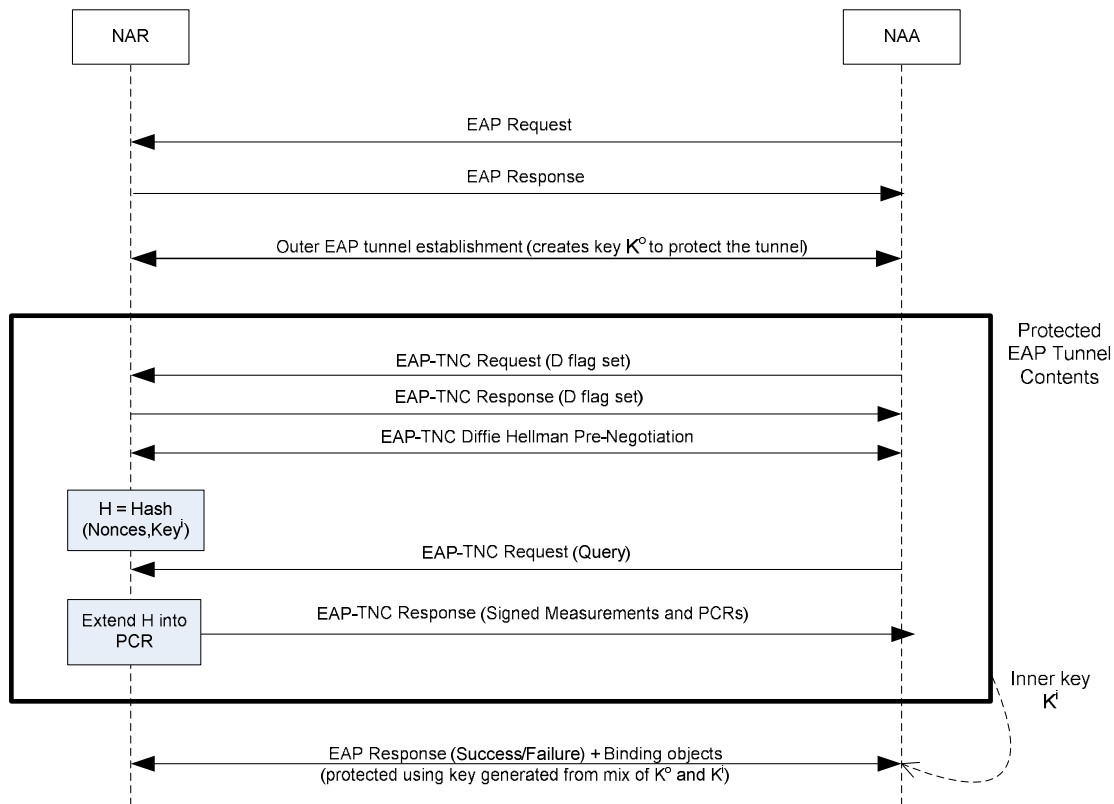


Figure 12. EAP-TNC with D-H PN Example

The following sequence explains the details of the processing of each message and how it provides security against MiTM attacks:

1. Initially, the authenticator sends an EAP-Request with the S (Start) bit set to 1 to indicate the beginning of the session. The authenticator SHOULD check policy to determine if the access requestor should be asked to use the D-H PN and if so set the D (D-H PN) bit. If the D-H PN bit is set the message MUST NOT contain data in the data section and is known as a D-H PN Hello Request message.
2. The access requestor receives the D-H PN Hello Request message. If the access requestor does not support IF-T 1.1, it would ignore the D flag and try to process the data section as an IF-TNCCS request message so must find no data field for backward compatibility. If the access requestor is 1.1 compliant it will perform the following:
 - a. If the D flag is 0, the access requestor MUST NOT respond with a message with the D flag set to 1. Instead, it MAY terminate the exchange if it requires a D-H PN but will usually proceed with EAP-TNC without D-H PN.
 - b. If the D flag is 1, the access requestor MAY consult policy to decide whether to respond with the D-H PN Hello Response message indicating a willingness to perform a D-H PN. If willing to use D-H PN, the access requestor includes a set of acceptable D-H groups and any minimum nonce lengths it requires. The access requestor MAY decline to perform D-H PN by sending an EAP-TNC message without the D bit set. In this case, the authenticator MAY proceed with a TNC exchange unprotected by D-H PN or terminate the entire TNC exchange.
3. If the authenticator receives the D-H PN Hello Response message, this indicates an ability and willingness to perform a D-H PN. The authenticator sends a D-H PN Parameters Request message selecting a D-H group from those offered by the access requestor. This message also indicates its set of supported hash algorithms, and the authenticator's public value and freshness nonce.
4. The access requestor responds with a D-H PN Parameters Response Message. This message MUST select a hash algorithm from the offered set. If no acceptable options were offered the access requestor SHOULD respond with a message with the D flag set to 0 and proceed with an EAP-TNC response (to the Start message) without D-H PN protection. The access requestor also sends its D-H public value corresponding to the selected D-H group and a freshness nonce.
5. The authenticator receives the D-H PN Parameters Response message and assures the response is consistent with its request message and meets its policy.

At this point the access requestor and authenticator compute the shared secret key using the Diffie-Hellman algorithm. Passive MiTM listeners can not determine the key value, although an active MiTM that participates in the D-H PN exchange and acts as a proxy between the true access requestor and authenticator could share keys with each party. In the proxy case, the true access requestor and authenticator do not share a common secret key (they each only share a secret with the MiTM proxy.) To detect this style of attack, the access requestor uses a byproduct (Unique-Value-1) of the secret key and both nonces as the ExternalData (nonce) parameter to the subsequent TPM_Quote that is included the PTS Integrity Report. This cryptographically binds the PCR values quoted to the endpoints of the IF-T session. The MiTM proxy is unable to dictate the ExternalData to use on the healthy access requestor system therefore can not obtain a signed clean PCR set with the Unique-Value-1 included. The authenticator can evaluate the Integrity Report and verify it includes knowledge of the secret key and the nonces to assure the access requestor was valid.

Both parties compute the following:

6. Unique-Value-1 = HASH ("1" | AR-Nonce | A-Nonce | D-H Shared Secret Key)The access requestor saves Unique-Value-1 for later use with the PTS. If the value is >20 bytes (e.g. when the selected HASH is SHA-256) the value MUST be truncated to the 20 most significant bytes. Later when the access requestor is asked to produce an Integrity

Report, this value is passed to the TPM_Quote operation (in the ExternalData argument) along with the desired PCR set so that the resulting AIK signed TPM_QUOTE_INFO contains a binding of the system state to the session state.. The authenticator can also compute Unique-Value-1 and locate it in the Integrity Report's TPM_QUOTE_INFO element. Note that Unique-Value-1 isn't the actual secret key used to protect traffic.

- 7. Next, the access requestor and authenticator MUST compute the following value which will be used later by EAP-TNC:

$$\text{Unique-Value-2} = \text{HASH} ("2" | \text{AR-Nonce} | \text{A-Nonce} | \text{D-H Shared Secret Key})$$

- 8. After the completion of the D-H PN protocol, both entities MUST set the D flag to 0 and then use the data field to exchange IF-TNCCS messages. The Authenticator will start by sending an EAP-TNC message with no Data and the D and S bits set to 0. The access requestor will respond with an EAP-TNC message containing its first IF-TNCCS message.
- 9. When a D-H PN has successfully completed, the authenticator and access requestor MUST compute a running hash (using the selected algorithm) including the complete contents (from the Code field through the Data field, inclusive) of each EAP-TNC message sent/received. This running hash is performed by repeated use of the following after receiving or sending an EAP-TNC Message:

$$\text{Unique-Value-2} = \text{HASH} (\text{Unique-Value-2} | \text{HASH} (\text{EAP-TNC Message}))$$

The result (final Unique-Value-2) is a value which is cryptographically computed from the D-H PN secret key, nonce pair and the contents of all of the messages exchange (thus all the integrity information responses.)

- 10. At the completion of the EAP-TNC exchanges when the D-H PN has been used, the final Unique-Value-2 MUST be exported and mixed into the tunneled EAP method's session keys. An additional tunneled EAP method round trip is required to assure that authenticator and access requestor both computed the same value. If this occurred, then both parties knew the secret D-H PN key, nonce pair and observed the same set of EAP-TNC message (thus allowing for detection of MiTM message tampering.) If both do not compute the same final Unique-Value-2, then the final tunneled EAP method message exchange (cryptographic binding check) will not properly decrypt, so the session MUST be considered compromised.
- 11. Finally after the outer tunneled EAP method completes, it's critical that the subsequent communications continue to be protected from active attacks by a MiTM. This SHOULD be achieved by leveraging keys derived from the Unique-Value-2 known by both parties to encrypt and integrity protect future traffic. Wireless 802.1X has provisions for provisioning a key for this purpose, but wired 802.1X requires an equivalent mechanism (possibly part of 802.1AE.)

6.3.4 Diffie-Hellman Pre-Negotiation Hash Algorithm Values

This section defines the values for the Hash Alg. field for the various hashing algorithms supported by D-H PN. The values are as follows:

```
+-----+
|R R R R R R 2 1|
+-----+
```

- 1 – SHA-1[5]
- 2 – SHA-256[5]
- R – Reserved for future use

Implementations compliant with version 1.1 of this specification MUST ignore bits set that they are unable to support. Such implementations MUST NOT set hash algorithm values that they are unable to support.

6.3.5 Diffie-Hellman Group Values

This section defines the bit values for the D-H Group field used in several D-H PN messages. The values are as follows:

```

+---+---+---+---+---+---+
|R R R R R 3 2 1|
+---+---+---+---+---+---+

```

- 1 – Indicates the use of values based on the group 2 from IKE.
- 2 – Indicates the use of values based on the group 5 from IKE.
- 3 – Indicates the use of values based on the group 14 from IKE.
- R – Reserved for future use

Implementations compliant with version 1.1 of this specification MUST ignore bits set that they are unable to support. Such implementations MUST NOT set D-H Group values that they are unable to support.

6.3.5.1 Diffie-Hellman Group 1 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 1 (bit 1 above) of the D-H PN. This group is taken from group 2 of IKE.

The public values exchanged when using this group MUST be 128 bytes in length.

The Diffie-Hellman generator (g) MUST be 2.

The prime modulus is the 128 byte value:

$$2^{1024} - 2^{960} - 1 + 2^{64} * \{ [2^{894} \text{ pi}] + 129093 \}$$

which has a hexadecimal value of:

```

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08
8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B
302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9
A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6
49286651 ECE65381 FFFFFFFF FFFFFFFF

```

6.3.5.2 Diffie-Hellman Group 2 Definitions

This section defines the Diffie-Hellman algorithm values that MUST be used when using group 2 (bit 2 above) of the D-H PN. This group is based on group 5 from IKE MODP Groups[6].

The public values exchanged when using this group MUST be 192 bytes in length.

The Diffie-Hellman generator (g) MUST be 2.

The prime modulus is the 192 byte value:

$$2^{1536} - 2^{1472} - 1 + 2^{64} * \{ [2^{1406} \text{ pi}] + 741804 \}$$

which has a hexadecimal value of:

```

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1

```

```
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA237327 FFFFFFFF FFFFFFFF
```

6.3.5.3 Diffie-Hellman Group 3 Definitions

This section defines the Diffie-Hellman algorithm values that **MUST** be used when using group 3 (bit 3 above) of the D-H PN. This group is based on group 14 from IKE MODP Groups[6].

The public values exchanged when using this group **MUST** be 256 bytes in length.

The Diffie-Hellman generator (g) **MUST** be 2.

The prime modulus is the 256 byte value:

$$2^{2048} - 2^{1984} - 1 + 2^{64} * \{ [2^{1918} \text{ pi}] + 124476 \}$$

which has a hexadecimal value of:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AACAA68 FFFFFFFF FFFFFFFF
```

7 References

The references are divided as to normative and non-normative. Normative references are those that are required to implement IF-T protocol bindings for tunneled EAP methods. Non-normative references are helpful in understanding use cases covered in this specification, but are not required to implement IF-T protocol bindings for tunneled EAP methods.

IETF Internet Drafts are listed in a separate non-normative section. Internet Drafts are not kept by IETF for over 6 months. However the drafts referenced here have been implemented by a number of organizations. Some of these drafts will never be published as RFCs, while others may eventually become RFCs, with some even “standards track” RFCs. TNC is supportive of IETF efforts to define one or more standard tunneled EAP methods. Finding expired Internet Drafts can generally be done using a search engine on the Web.

7.1 Normative References

- [1] Trusted Computing Group, *TNC Architecture for Interoperability*, Specification Version 1.1, May 2006, <https://www.trustedcomputinggroup.org/specs/TNC> .
- [2] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, Internet Engineering Task Force RFC 2119, March 1997.
- [3] Trusted Computing Group, *TNC IF-TNCC*, Specification Version 1.0, May 2006, <https://www.trustedcomputinggroup.org/specs/TNC>.
- [4] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowitz, Ed., “Extensible Authentication Protocol (EAP)”, Internet Engineering Task Force RFC3748, June, 2004
- [5] NIST, “Secure Hash Standard”, FIPS 180-1, National Institute of Standards and Technology, U.S. Department of Commerce, May 1994, <http://csrc.nist.gov/CryptoToolkit/shs/dfips-180-2.pdf>
- [6] T. Kivinen, M. Kojo, “More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)”, <http://www.ietf.org/rfc/rfc3526.txt>, May, 2003.

7.2 Non-Normative References

- [7] LAN/MAN Standards Committee of the IEEE Computer Society, Standard for Local and Metropolitan Area Networks – Port Based Network Access Control, IEEE Std. 802.1X-2004, December, 2004
- [8] J. Vollbrecht, P. Eronen, N. Petroni, Y. Ohba, “State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator”, Internet Engineering Task Force RFC 4137, August, 2005
- [9] B. Aboba, D. Simon “PPP EAP TLS Authentication Protocol”, Internet Engineering Task Force RFC2716, October, 1999
- [10] B. Aboba, P. Calhoun, “RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)”, Internet Engineering Task Force RFC3579, September, 2003
- [11] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, D. Spence, “AAA Authorization Framework”, Internet Engineering Task Force RFC2904, August ,2000

- [12] N. Asokan, Valtteri Niemi, Kaisa Nyberg, "Man in the Middle Attacks in Tunneled Authentication Protocols", Nokia Research Center, Finland, Nov. 11, 2002, <http://eprint.iacr.org/2002/163.pdf>
- [13] Trusted Computing Group, *Subject Key Attestation Evidence Extension*, Specification version 1, revision 7, https://www.trustedcomputinggroup.org/specs/IWG/IWG_SKAE_Extension_1-00.pdf June 16, 2005.
- [14] Greg Kazmierczak, Ned Smith, "TCG IWG Platform Trust Services Interface Specification (IF-PTS)", https://www.trustedcomputinggroup.org/specs/IWG/IF-PTS_v1.0.pdf, November, 2006.
- [15] TCG Infrastructure WG, Miscellaneous schema documents, <https://www.trustedcomputinggroup.org/specs/IWG>, November 2006.

7.3 Non-Normative Internet Drafts

- [16] Joseph Salowey, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", Internet Engineering Task Force Internet Draft draft-cam-winget-eap-fast-03, October, 2005
- [17] Hannes Tschofenig, Pasi Eronen IETF draft-eronen-ipsec-ikev2-eap-auth-04 "Extension for EAP Authentication in IKEv2", October, 2005
- [18] Nancy Cam-Winget IETF draft-cam-winget-eap-fast-provisioning-01 "Dynamic Provisioning using EAP-FAST", July, 2005
- [19] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," Internet Engineering Task Force RFC4306, December, 2005
- [20] Paul Funk, Simon Blake-Wilson IETF <draft-funk-eap-ttls-v1-00.txt> EAP Tunneled TLS Authentication Protocol Version 1, February, 2005
- [21] Paul Funk, Simon Blake-Wilson IETF draft-ietf-pppext-eap-ttls-05.txt EAP Tunneled TLS Authentication Protocol (EAP-TTLS), July, 2004
- [22] H. Andersson, S. Josefsson, Glen Zorn, Dan Simon, Ashwin Palekar IETF draft-josefsson-pppext-eap-tls-eap-05.txt Protected EAP Protocol (PEAP), September, 2002
- [23] Ashwin Palekar, Dan Simon, Joe Salowey, Hao Zhou, Glen Zorn, S. Josefsson draft-josefsson-pppext-eap-tls-eap-15 Protected EAP Protocol (PEAP) Version 2, October, 2004
- [24] Vivek Kamath, Ashwin Palekar, Mark Wodrich draft-kamath-pppext-peapv0-00.txt Microsoft's PEAP version 0 (Implementation in Windows XP SP1, October, 2002
- [25] Pasi Eronen, Paul Hoffman "IKEv2 Clarification and Implementation Guidelines", <draft-eronen-ipsec-ikev2-clarifications-06.txt>, October, 2005
- [26] Pasi Eronen et al, "Multiple Authentication Exchanges in IKEv2", <draft-eronen-ipsec-ikev2-multiple-auth-00.txt>, October, 2005