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

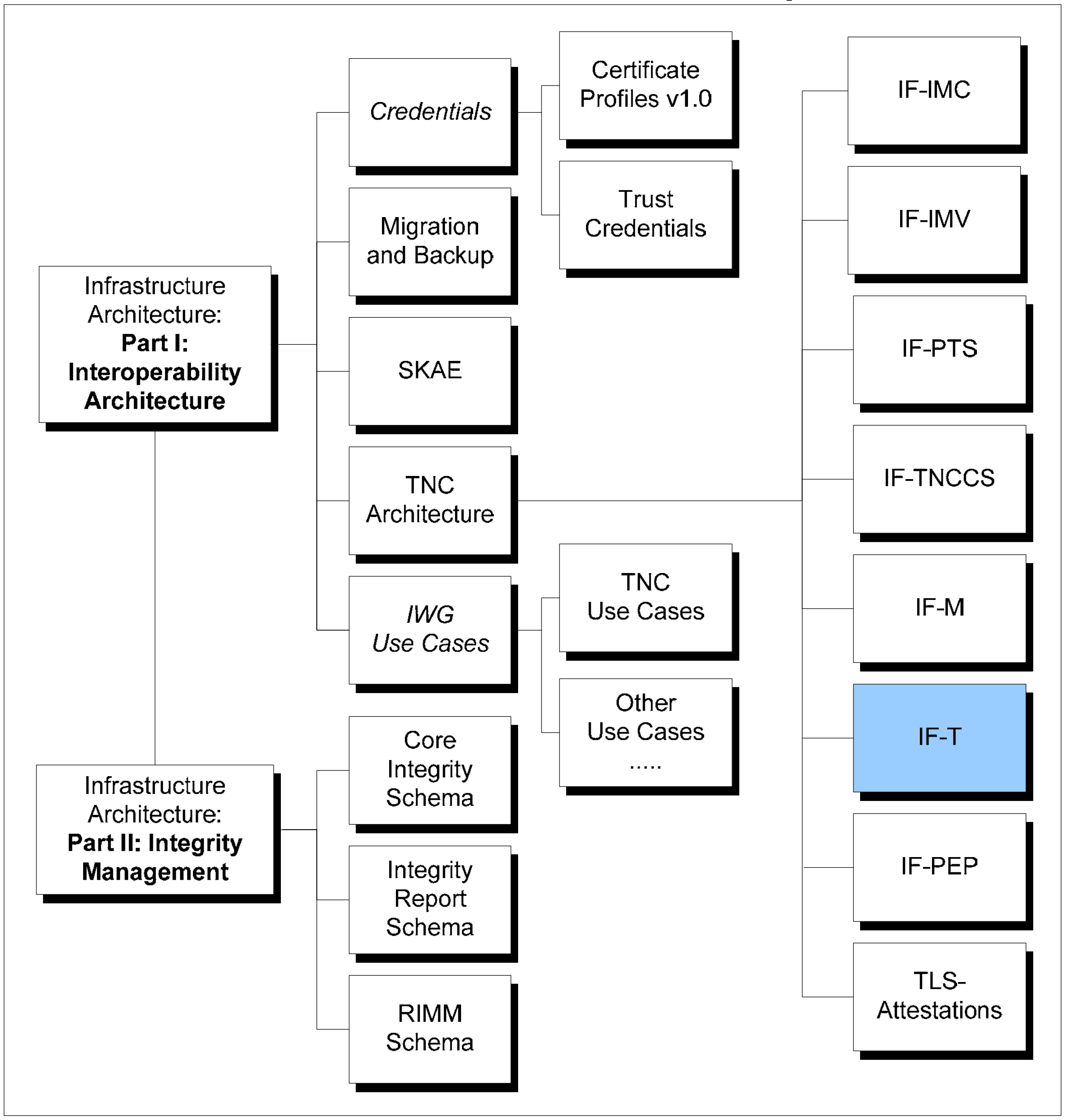
Specification Version 1.0
Revision 3
1 May 2006
Published

Contact: admin@trustedcomputinggroup.org

TCG PUBLISHED

TCG

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

Mahalingam Mani	Avaya
Kazuaki Nimura	Fujitsu Limited
Boris Balacheff	Hewlett-Packard
Mauricio Sanchez	Hewlett-Packard
Diana Arroyo	IBM
Lee Terrell	IBM
Ravi Sahita	Intel Corporation
Ned Smith	Intel Corporation
Barbara Nelson	iPass
Steve Hanna (TNC co-chair)	Juniper Networks, Inc.
Alex Romanyuk	Meetinghouse Data Communications
Gene Chang	Meetinghouse Data Communications
John Vollbrecht	Meetinghouse Data Communications
Joseph Tardo	Nevis Networks
Pasi Eronen	Nokia Corporation
Sandilya Garimella	Motorola
Thomas Hardjono	SignaCert, Inc.
Bryan Kingsford	Symantec Corporation
Paul Sangster(TNC co-chair)	Symantec Corporation
Scott Cochrane	Wave Systems
Greg Kazmierczak	Wave Systems

Table of Contents

1 Scope and Audience

Trusted Network Connect (TNC) is a sub-group of the Infrastructure Working Group (IWG) 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.

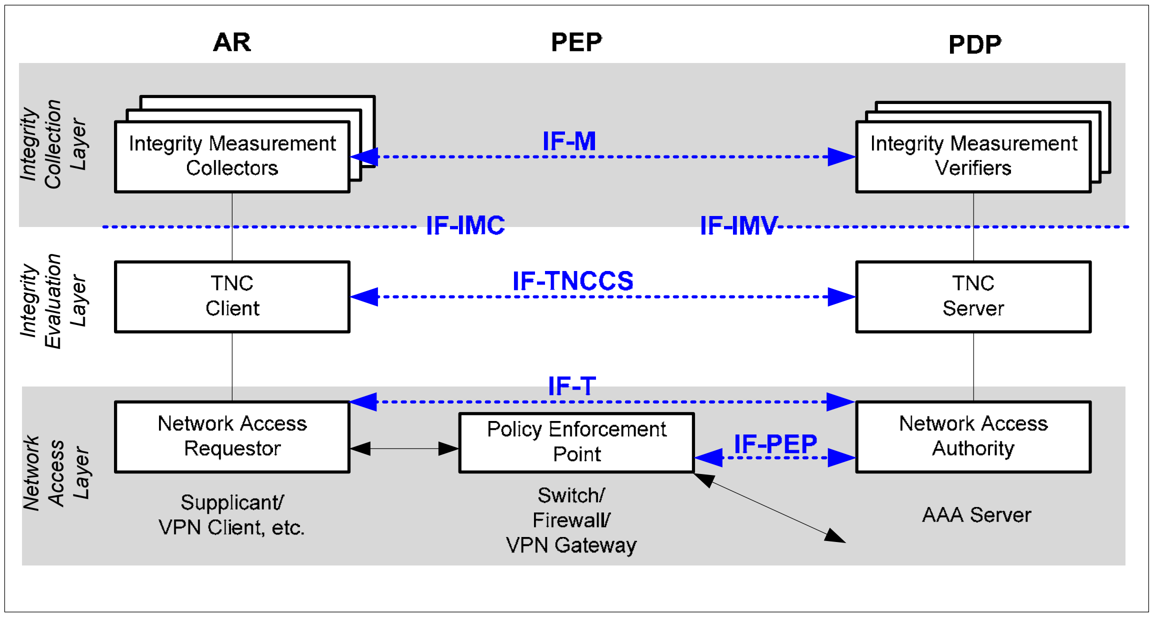

AR
PEP
PDP
Integrity Collection Layer
Integrity Evaluation Layer
Network Access Layer
Integrity Measurement Collectors
IF-M
Integrity Measurement Verifiers
IF-IMC
IF-IMV
TNC Client
IF-TNCCS
TNC Server
IF-T
Network Access Requestor
Policy Enforcement Point
IF-PEP
Network Access Authority
Supplicant/ VPN Client, etc.
Switch/ Firewall/ VPN Gateway
AAA Server

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 protocolsTNC 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.

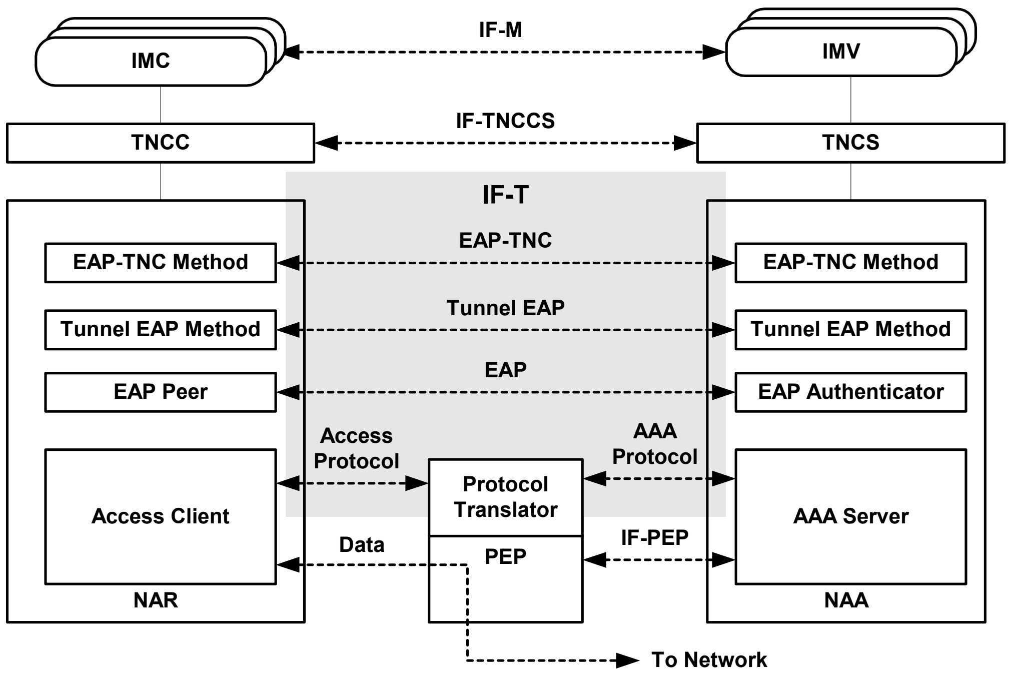


Figure 2. EAP-TNC and EAP Protocol Layers

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 Annex (Section 7) 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 [6].

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 Annex (Section 7).

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.

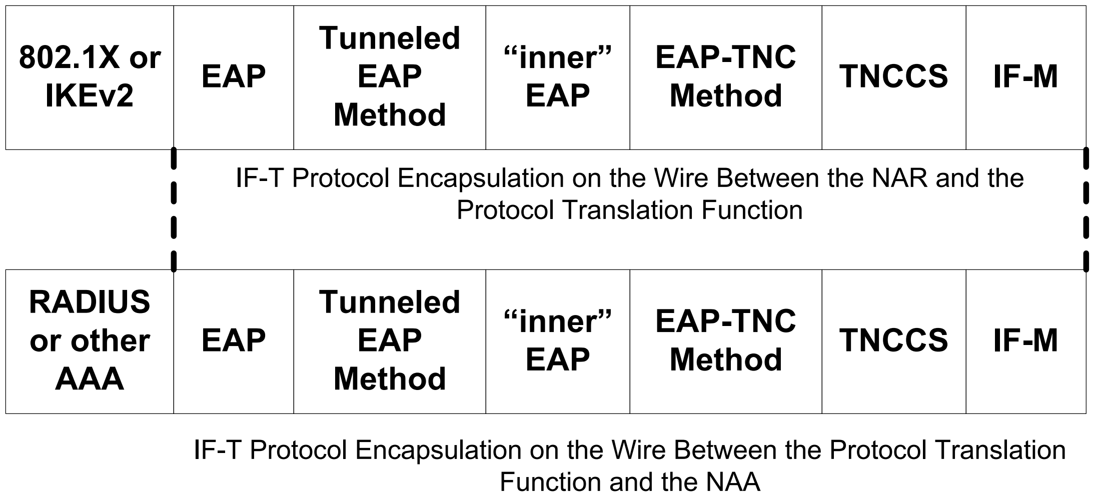

802.1X or IKEv2
EAP
Tunneled EAP Method
“inner” EAP
EAP-TNC Method
TNCCS
IF-M
IF-T Protocol Encapsulation on the Wire Between the NAR and the Protocol Translation Function
RADIUS or other AAA
EAP
Tunneled EAP Method
“inner” EAP
EAP-TNC Method
TNCCS
IF-M
IF-T Protocol Encapsulation on the Wire Between the Protocol Translation Function and the NAA

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 Appendix (Section 7). Figure 4 below shows how EAP-TNC carries an IF-TNCCS handshake.

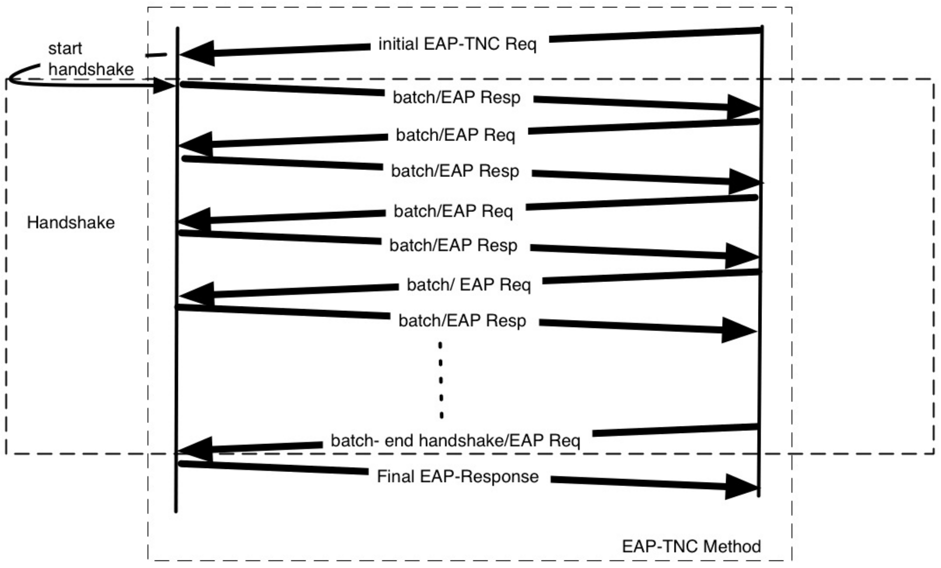


Figure 4. EAP-TNC TNCCS Handshake

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 5 does not show cases where fragmentation of a batch transfer is required. Fragmentation is described in the Annex, Section 7.

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, some tunneled EAP methods allow the peer and authenticator pair to signal each other’s tunnel endpoint using special AVPs. This is specifically true of EAP-FAST and PEAPv2.

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.

EAP-TNC does not provide its own protection mechanisms. It depends upon a tunneled EAP method to provide them. 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.

EAP-TNC does not provide its own user or platform authentication mechanisms. EAP-TNC SHOULD be used in addition to tunneled EAP methods that provide 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 all but one of the Tunnelled 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) [12] and Protected EAP Version 2 (PEAPv2) [19] are both tunneled EAP methods that define how to run multiple inner EAP methods. Because EAP-TNC does not provide key material, an ISK for the method shall be set to zero. In EAP-FAST and PEAPv2, EAP-TNC it 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 [16], 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 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 approach to how this could be done. This section is informative only.

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 a user authentication dialog. For example EAP-TNC could run either before or after MD5 or MSCHAP.
3. Finally. EAP-TNC could be one of a number of dialogs that might include user authentication such as EAP-MD5. .

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

4 Examples (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.

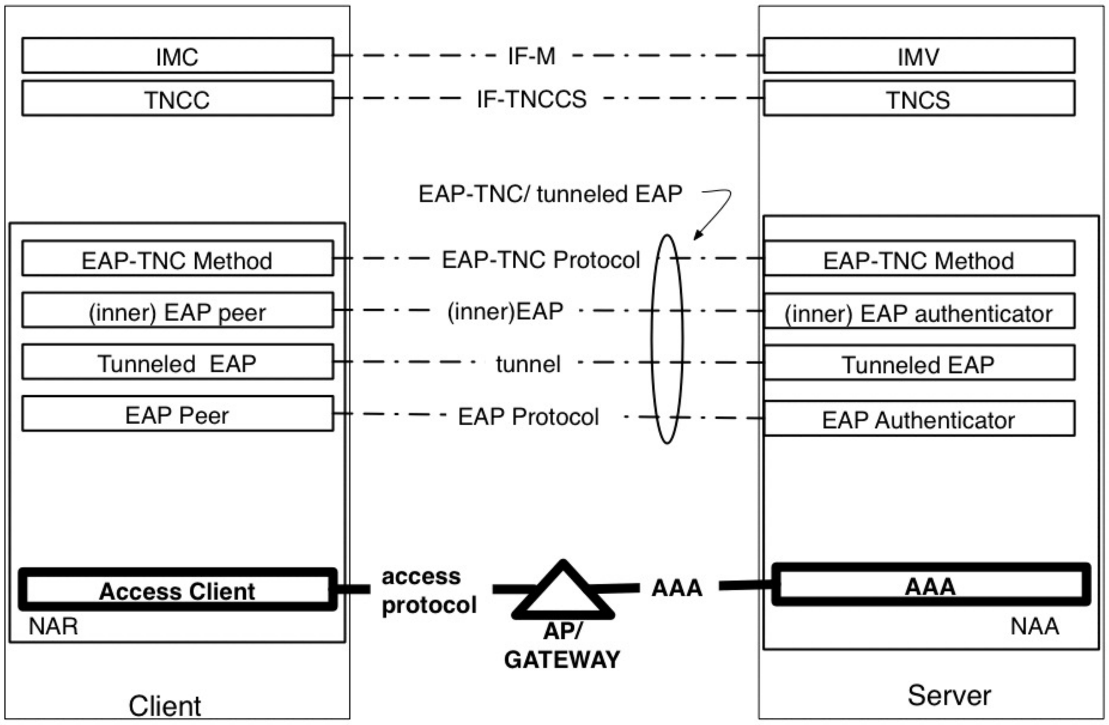


Figure 5. Access Client and Gateway Relationship

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

Access clients are part of the NAR. Other examples of access clients that fit in this model include PPP for Dial Access and 802.16e.

4.1 802.1X

Figure 6 below shows how 802.1X [5] 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.

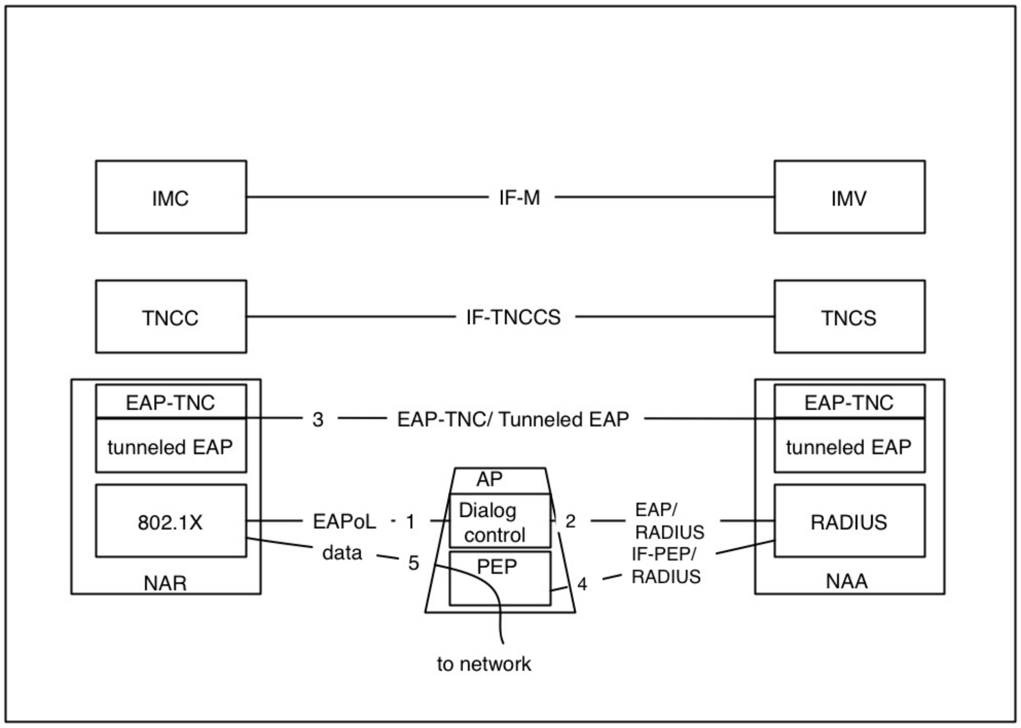


Figure 6. IF-T Over 802.X

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 establish a shared key between an IKE client and an IKEv2 gateway. This key is then used to create an IPsec tunnel between the client and gateway. Figure 7 shows how TNC can be incorporated into this capability.

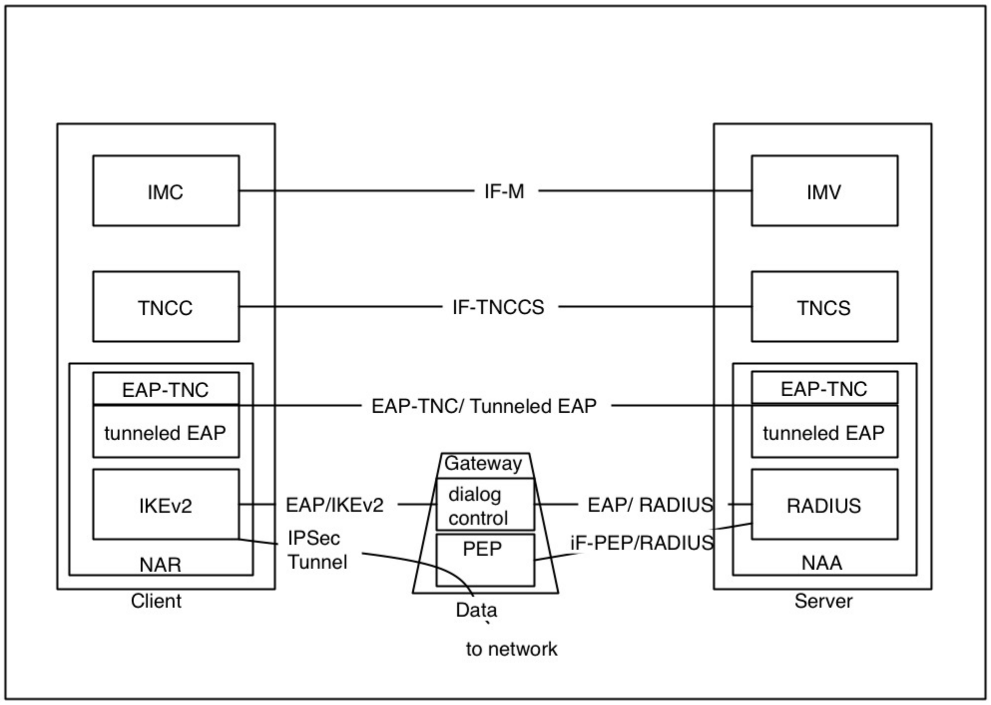


Figure 7. IF-T Over IKEv2

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.

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.

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. The first two messages carry an ephemeral Diffie-Hellman exchange and ciphersuite 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, the IKEv2 gateway, that EAP authentication is requested. If supported and configured, the gateway/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 MUST be used by both the initiator and responder to generate AUTH payloads using the syntax for shared secrets specified in [15]. The shared key from EAP is the field from the EAP specification named 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)

To protect against an attacker exploiting a vulnerability on the client or server, IF-T MAY participate in TCG security protection of the client and server.

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-IMC and PTS-IMV are specified in IF-PTS.

In addition to measurement of modules, IF-T modules may interact directly with Platform Trust Services to utilize cryptographic signing and encryption of messages sent between client and server as described in the IF-PTS specification

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. For this protocol binding, TNC requires 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 limited to that required for an authorization dialog, which is typically small. TNC does not specify what protection to provide, but does require that whatever is provided be equivalent to what is provided in TLS.

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 these to extent that we are aware of them.

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 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. Misrepresenting the TNCS recommendation 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 the transport authentication
4. Cryptographic integrity protection of the message tied to 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[12], PEAPv0/v2[17,18], and TTLS[15,16] all make use of TLS [7] 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.

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. This prevents an attack where a corrupted endpoint performs the user or platform authentication, then proxies an integrity verification to a clean endpoint and uses the keys derived from the user or platform authentication for network access.

5.4.5 Protecting against MiTM attacks against EAP-TNC

Man in the Middle attacks against tunneled EAP methods that use an internal method to support authentication have been described by Asokan, Niemi and Nyberg [10]. Such an attack is possible against EAP-TNC. A description of the attack is given in the Annex in section 8. The attack involves an attacker who wants to attach a compromised platform to a network that requires platform attestation using a TPM. The attacker would divert the TNC dialog from the invalid platform to a valid platform using the responses from the valid platform to allow the invalid platform to gain access.

To prevent this attack the following recommendation is made. When the use of PTS for measurement and reporting of endpoint integrity is combined with user or platform authentication, the authentication must be done with a certificate tied to the TPM and the PDP must verify that the certificate is associated with the same TPM as the one used for the PTS exchange.

Here is a more detailed description of the process. Use a TPM to "certify" the client authentication (signing) key. At the time of key enrollment with a Certificate Authority (CA), the enrollment SHOULD include key certification by a TPM where the TPM Attestation Identity Key (AIK) digitally signs the client authentication key. Evidence of the signing is provided to the CA. The CA MUST include the evidence in the client certificate as an extension. The Subject Key Attestation Evidence (SKAE) extension is documented in [11]. The verifier MUST check that the AIK used to sign the client authentication key matches the AIK contained in the SKAE extension. The AIK certificate must be signed by a trusted issuer.

In the future, TNC plans to enhance the EAP-TNC method to generate a key that will be used to generate bulk data transfer keys. This will provide protection against the MiTM attack without requiring the use of credentials tied to the TPM.

Although a MiTM attack against EAP-TNC could be employed with a network that does not require a TPM, there would be no point in mounting such a sophisticated attack. A compromised endpoint could simply send false measurements.

6 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.

6.1 Normative References

[1] Trusted Computing Group, *TNC Architecture for Interoperability*, Specification Version 1.1, May 2006.

[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.

[4] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", Internet Engineering Task Force RFC3748, June, 2004

6.2 Non-Normative References

[5] 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

[6] 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

[7] B. Aboba, D. Simon "PPP EAP TLS Authentication Protocol", Internet Engineering Task Force RFC2716, October, 1999

[8] B. Aboba, P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", Internet Engineering Task Force RFC3579, September, 2003

[9] 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

[10] 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

[11] Trusted Computing Group, *Subject Key Attestation Evidence Extension,* Specification version 1, revision 7, June 16,2005

6.3 Non-Normative Internet Drafts

[12] 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

[13] Hannes Tschofenig, Pasi Eronen IETF draft-eronen-ipsec-ikev2-eap-auth-04 "Extension for EAP Authentication in IKEv2" , October, 2005

[14] Nancy Cam-Winget IETF draft-cam-winget-eap-fast-provisioning-01 "Dynamic Provisioning using EAP-FAST", July, 2005

[15] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," Internet Engineering Task Force RFC4306, December, 2005

[16] Paul Funk, Simon Blake-Wilson IETF <draft-funk-eap-ttls-v1-00.txt> EAP Tunneled TLS Authentication Protocol Version 1, February, 2005

[17] Paul Funk, Simon Blake-Wilson IETF draft-ietf-pppext-eap-ttls-05.txt EAP Tunneled TLS Authentication Protocol (EAP-TTLS), July, 2004

[18] 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

[19] 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

[20] Vivek Kamath, Ashwin Palekar, Mark Wodrich draft-kamath-pppext-peapv0-00.txt Microsoft's PEAP version 0 (Implementation in Windows XP SP1, October, 2002

[21] Pasi Eronen, Paul Hoffman "IKEv2 Clarification and Implementation Guidelines", draft-eronen-ipsec-ikev2-clarifications-06.txt, October, 2005

[22] Pasi Eronen et al, "Multiple Authentication Exchanges in IKEv2", draft-eronen-ipsec-ikev2-multiple-auth-00.txt, October, 2005

7 Annex: EAP-TNC Protocol Definition (Normative)

This appendix 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. EAP-TNC does not provide any security features on its own.

This appendix is normative.

7.1 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. 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 messages 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.

7.1.1 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 [6], 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.

7.1.2 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.

7.1.3 EAP-TNC format

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

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |   Identifier  |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |   Flags | Ver |     Data Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Data Length           |           Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

Type

38

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

Flags

```
+-+-+-+-+-+
|L M S R R|
+-+-+-+-+-+
```

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.

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.

7.1.4 EAP-TNC Compliance

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

7.1.5 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:	No
Key strength:	N/A
Dictionary attack resistant.:	N/A
Fast reconnect:	No
Crypt. binding:	N/A
Session independence:	N/A
Fragmentation:	Yes
Channel binding:	No

7.2 EAP-TNC Conversation Example

Figure 9 illustrates a representative EAP-TNC message exchange.

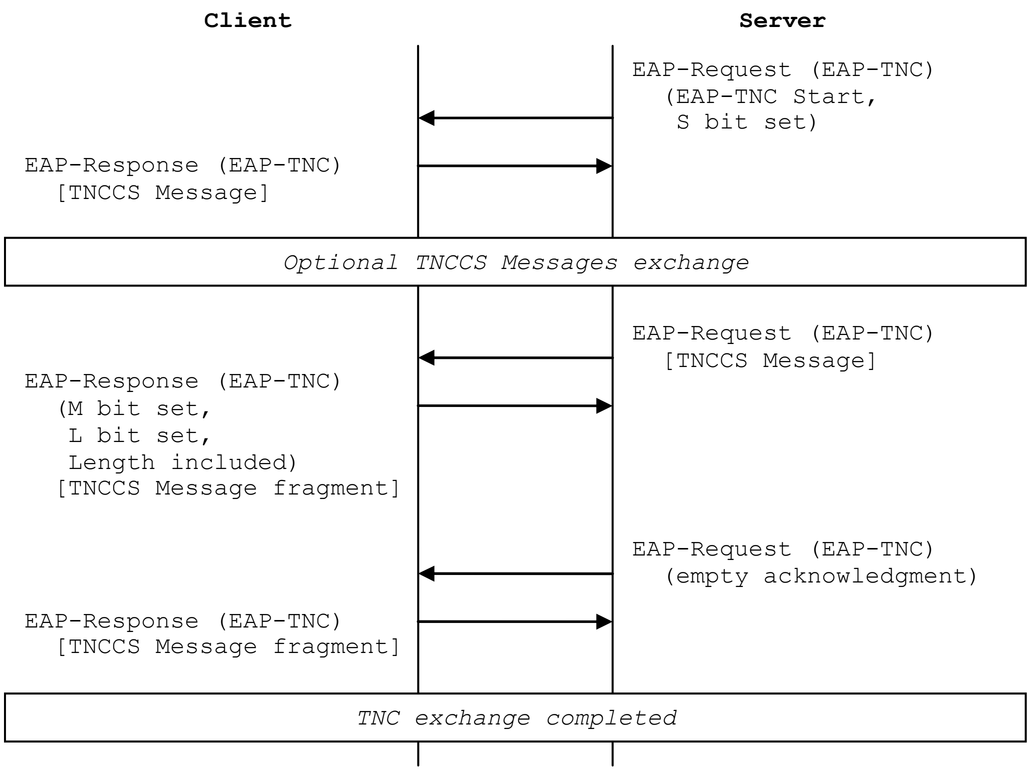
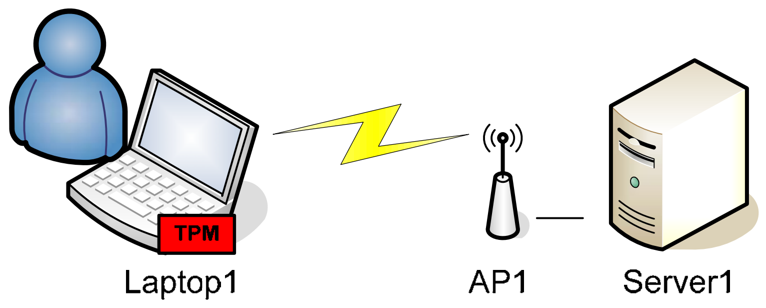


Figure 4. EAP-TNC Conversation Example

8 Appendix A: Man in the Middle Attack on TNC over Tunneled EAP

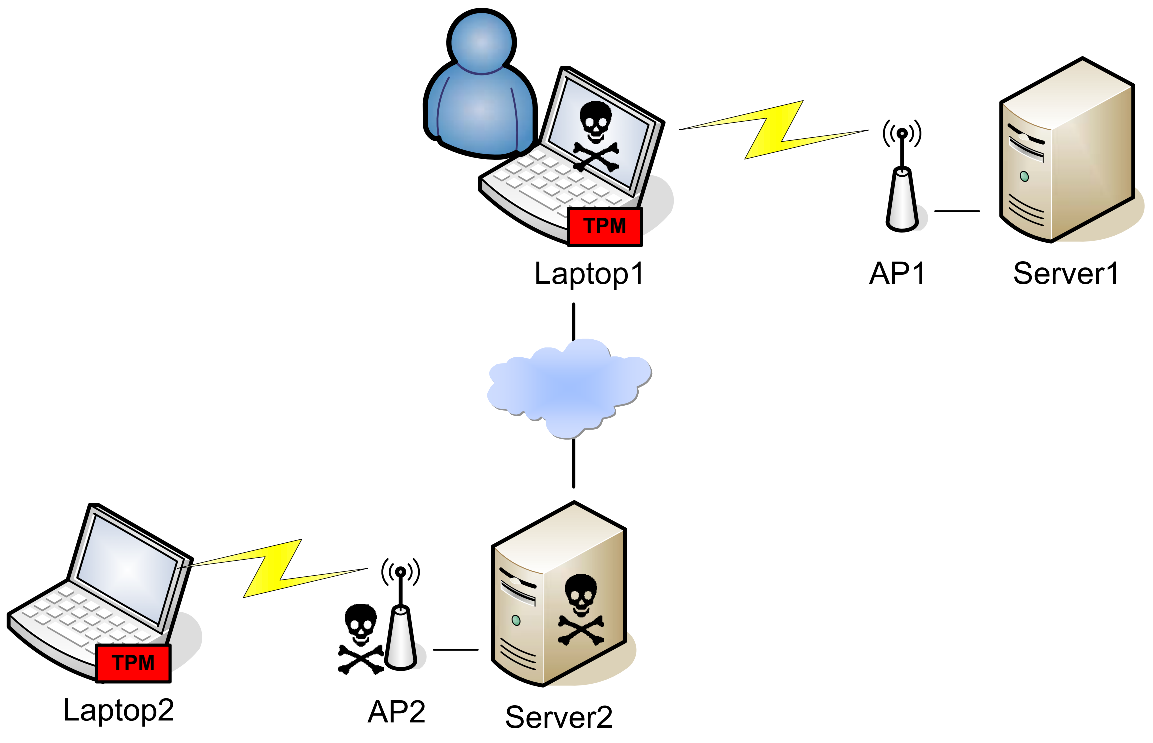
This Appendix describes a potential problem when using tunneled EAP methods to support IF-T in TNC. This is basically the “man-in-the-middle attack against tunneled authentication protocols” described in the 2003 Security Protocols Workshop paper by Asokan, Niemi, and Nyberg [10]. This shows how the problem might be exploited in when doing TNC over tunneled EAP methods.

Let’s consider WLAN access to company intranet (figure below). Server1 requires user authentication using PEAP, and proper integrity measurements, including a “quote” from a valid TPM. Therefore, if Laptop1 gets compromised (with software only, no hardware attacks), it cannot get network access since the quoted PCR values will not match the expected values, and Laptop1 will not allow access.



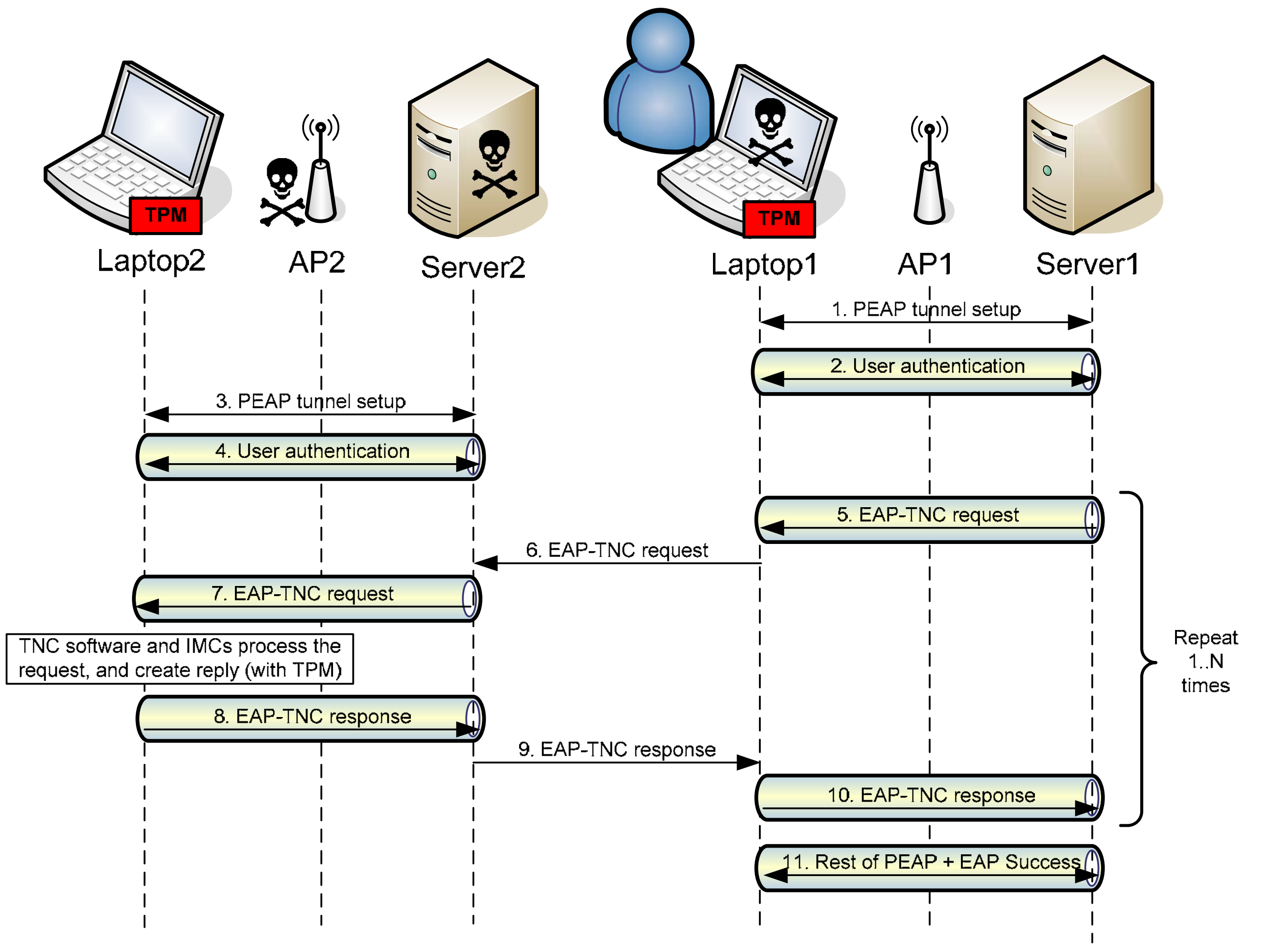
However, let’s assume that Laptop1 does get compromised, and the attacker behind the compromise can somehow communicate with Laptop1 (maybe the laptop has an EV-DO card in addition to WLAN).

The attacker now sets up his own equipment, Laptop2, AP2, and Server2. Laptop2 is configured to match the “good” configuration that would be accepted by Server1. However, Server2 is doing some special things.



Next, the honest user of Laptop1 tries to connect to the network. Eventually, PEAP is started and the user is authenticated (steps 1-2). At the same time, attacker uses Laptop2, starts PEAP with his own server, and does user authentication there (steps 3-4).

Next, the TNC phase starts. 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 (which involves creating a "quote" with the TPM at some point), and the response goes from Server2 to Laptop1 and eventually Server1. The steps 5-10 are repeated as often as needed. The exchange will succeed eventually, since Laptop2 really is in good shape (has all the things required by Server1's policy, and the integrity of everything is protected with the TPM). Thus Server1 will eventually give network access to the compromised Laptop1.



Section 5.4.5 describes how implementations can protect against this attack.