Network Working Group
Internet-Draft
Obsoletes: 5216 (if approved)
Intended status: Standards Track
Expires: January 4, 2018

J. Mattsson
M. Sethi
Ericsson

July 3, 2017

The EAP-TLS Authentication Protocol
draft-mattsson-eap-tls13-00

Abstract

Extensible Authentication Protocol (EAP) provides support for multiple authentication methods. Transport Layer Security (TLS) provides mutual authentication, integrity-protected cipher suite negotiation, and key exchange between two endpoints. This document specifies an EAP authentication method to provide support for certificate-based mutual authentication and key derivation using the version 1.3 of the TLS protocol. This document obsoletes RFC5216.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP) [RFC3748], provides a standard mechanism for supporting multiple authentication methods. Authentication methods such as Generalized Pre-Shared Key (GPSK) [RFC5433] and Protected One-Time Password Protocol (POTP) [RFC4793] have been defined using the EAP framework. Specifications for how EAP messages are carried over a variety of lower layers such as Point-to-Point Protocol (PPP) [RFC1661], IEEE 802 wired networks [IEEE-802.1X], and wireless technologies such as IEEE 802.11 [IEEE-802.11] and IEEE 802.16 [IEEE-802.16e] also exist.

[RFC5216] had defined TLS based mutual authentication for EAP. This document obsoletes [RFC5216] and specifies EAP-TLS that is based on TLS version 1.3. TLS 1.3 obsoletes the older version 1.2 and introduces a number of changes such as encrypting all messages after ServerHello and adding a 0-RTT mode that saves a round-trip at connection setup. For a complete list of updates see

[I-D.ietf-tls-tls13]. This document does not request a new EAP method type assignment.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In addition, this document frequently uses the following terms as defined in :

authenticator The entity initiating EAP authentication.

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

server The entity that terminates the EAP authentication method with the peer. In the case where no backend authentication server is used, the EAP server is part of the authenticator. In the case where the authenticator operates in pass-through mode, the EAP server is located on the backend authentication server.

NAI A Network Access Identifier [RFC7542]. It is the user identifier submitted by the peer/supplicant prior to accessing resources.

Master Session Key (MSK) Keying material that is derived between the EAP peer and server and exported by the EAP method.

Extended Master Session Key (EMSK) Additional keying material derived between the EAP peer and server that is exported by the EAP method.

3. Protocol Overview

The EAP-TLS conversation begins with the authenticator and the peer negotiating EAP. The authenticator then sends an EAP-Request/ Identity packet to the peer. The peer then responds with its Network

Access Identifier (NAI) in an EAP-Response/Identity packet. From this point onwards, although nominally the EAP conversation occurs between the EAP peer and the EAP authenticator, the EAP server is the ultimate endpoint conversing with the EAP peer. The authenticator MAY act as a pass-through to a backend EAP server. In the case where no backend authentication server is used, the EAP server is part of the authenticator.

3.1. Base Case

After receiving the peer's Identity (NAI), the EAP server MUST respond with an EAP-TLS/Start packet. This is an EAP-Request packet with EAP-Type=EAP-TLS, the Start (S) bit set, and no data. The EAP-TLS conversation will then begin. The EAP-TLS conversation consists of EAP-Response and EAP-Request packets with EAP-Type=EAP-TLS and with the data field encapsulating one or more TLS records in TLS record layer format. The formating and processing of the TLS handshake SHALL be done as specified by TLS 1.3 [I-D.ietf-tls-tls13]. This document only lists additional requirements, restrictions, and processing compared to TLS 1.3.

The peer responds to the EAP-Request with EAP-Response packet with EAP-Type=EAP-TLS. The data field in the response encapsulates one or more TLS records in TLS record layer format, containing a TLS ClientHello handshake message. The ClientHello message contains the peer's legacy_version that MUST be set to 0x0303, a random number, a legacy_session_id that MUST be set as a zero length vector (i.e., a single zero byte length field), a set of ciphersuites supported by the peer, a legacy_compression_methods vector field that MUST contain exactly one byte set to zero. The ClientHello must include the supported_versions extension. Peers can request additional functionality using extensions in the ClientHello message.

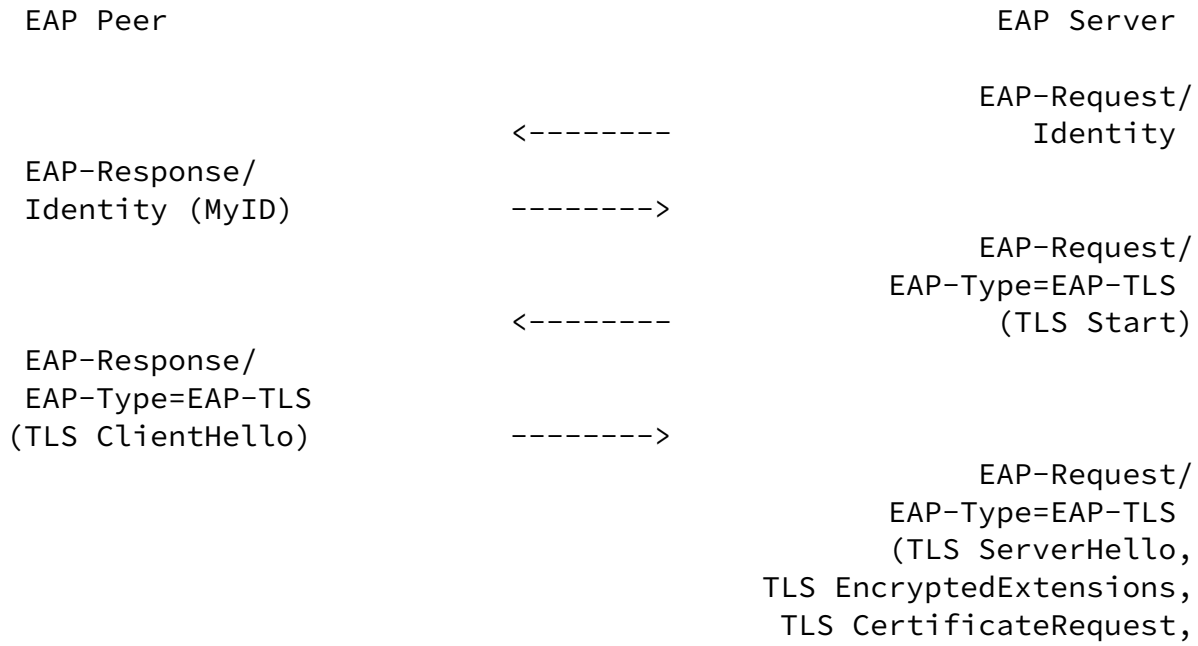
After receiving the EAP-Response containing the ClientHello, the EAP-Server sends an EAP-Request packet with EAP-Type=EAP-TLS. The data field of this packet contains one or more TLS records for TLS ServerHello, Encrypted extensions, a CertificateRequest, the server

Certificate along with an explicit proof of the server identity (CertificateVerify), followed by finished handshake message.

The ServerHello contains the version of TLS. EAP Servers MUST select a version from the list in ClientHello's supported_versions extension. For this version of the specification, the version is 0x0304. The ServerHello also contains a random number, a single cipher_suites selected by the server from the list in the ClientHello, and the "key_share" extension which specifies the cryptographic parameters such as the named group for the key being exchanged.

After receiving the EAP-Response containing the ServerHello, the EAP-Server sends an EAP-Request packet with EAP-Type=EAP-TLS. The data field of this packet contains one or more TLS records for TLS Certificate, CertificateVerify, followed by finished handshake message.

After the TLS handshake has completed, the EAP server sends EAP Success. In the case where the EAP-TLS with mutual authentication is successful, the conversation will appear as shown in Figure 1.



```
                                                  TLS Certificate,
                                            TLS CertificateVerify,
                              <--------                 TLS Finished)
 EAP-Response/
 EAP-Type=EAP-TLS
(TLS Certificate,
 TLS CertificateVerify,
 TLS Finished)                --------->
                              <--------                 EAP-Success
```

Figure 1: EAP-TLS base case - Mutual authentication

While the EAP server SHOULD require peer authentication, this is not mandatory, since there are circumstances in which peer authentication will not be needed (e.g., emergency services, as described in [RFC7406]), or where the peer will authenticate via some other means.

If the EAP Server does not desire the peer to authenticate itself, the CertificateRequest is omitted, and the EAP Peer therefore does not send Certificate and CertificateVerify. The message flow is shown in Figure 2.


```
 EAP Peer                                          EAP Server

                                                  EAP-Request/
                              <--------              Identity
 EAP-Response/
 Identity (MyID)              --------->
                                                  EAP-Request/
                                                EAP-Type=EAP-TLS
                              <--------           (TLS Start)
 EAP-Response/
 EAP-Type=EAP-TLS
(TLS ClientHello)             --------->
                                                  EAP-Request/
                                                EAP-Type=EAP-TLS
                                                (TLS ServerHello,
```

```
                                        TLS EncryptedExtensions,
                                                TLS Certificate,
                                          TLS CertificateVerify,
                          <--------                 TLS Finished)
 EAP-Response/
 EAP-Type=EAP-TLS
(TLS Finished)            -------->
                          <--------                  EAP-Success
```

Figure 2: EAP-TLS base case - Server authentication only

3.2. Resumption

The purpose of resumption is to allow for improved efficiency in the case where a peer repeatedly attempts to authenticate to an EAP server within a short period of time.

Once a TLS handshake has completed, the EAP Server can send the EAP Peer a PSK identity (TLS NewSessionTicket) that corresponds to a key derived from the handshake. It is left up to the EAP Server whether to support resumption.

An initial authentication, where both sides authenticate successfully and the EAP Server sends a TLS NewSessionTicket is shown in Figure 3.


```
    EAP Peer                                          EAP Server

                                                    EAP-Request/
                          <--------                    Identity
    EAP-Response/
    Identity (MyID)       -------->
                                                    EAP-Request/
                                                EAP-Type=EAP-TLS
```

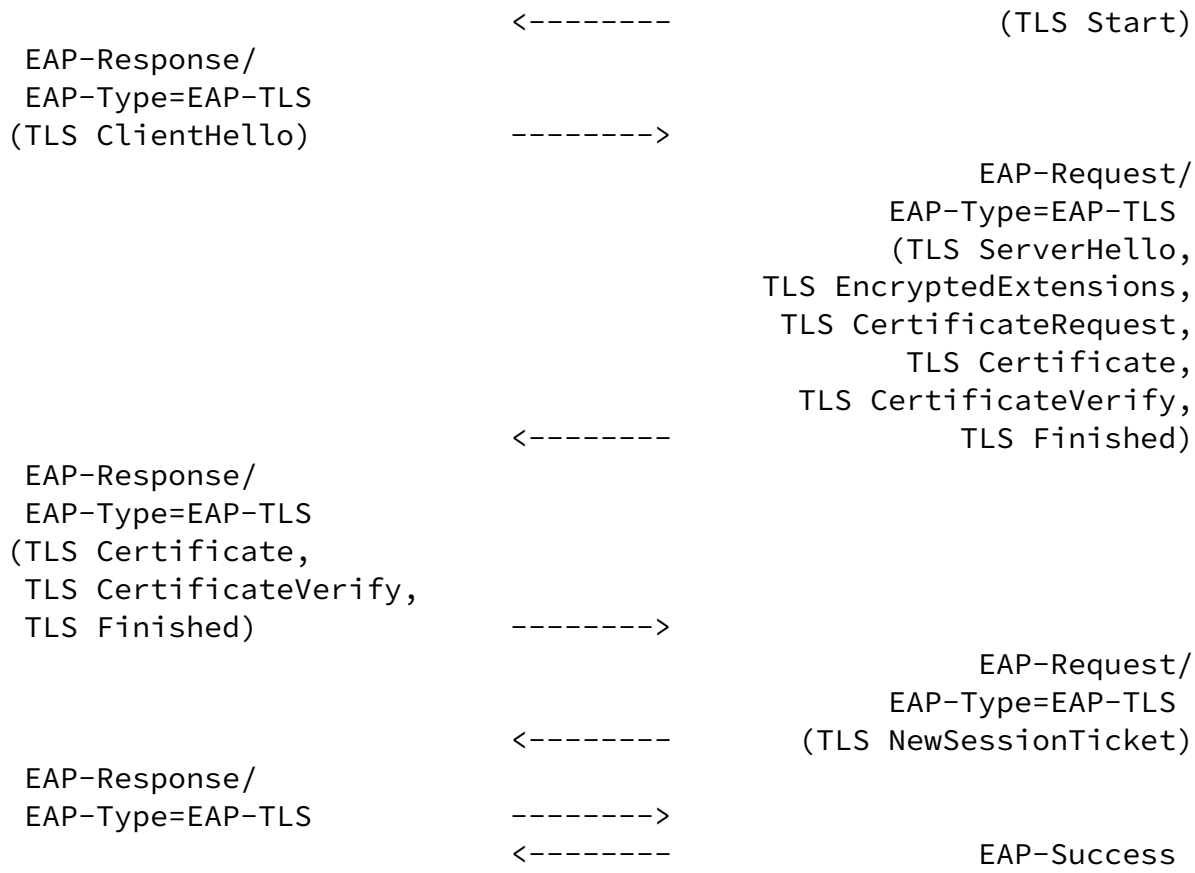


Figure 3: EAP-TLS resumption - Initial authentication

The EAP Peer can then use the PSK identity received in TLS NewSessionTicket to negotiate use of the PSK in future authentications. If the server accepts it, then the security context of the new connection is tied to the original connection and the key derived from the initial handshake is used to bootstrap the cryptographic state instead of a full handshake.

It is up to the peer whether to attempt resumption. Typically, a the peer's decision will be made based on the time elapsed since the previous authentication attempt to that EAP server. Based on the the time elapsed since the previous full authentication, the EAP server will decide whether to allow resumption or require a full authentication.

An subsequent authentication using resumption, where both sides

authenticate successfully is shown in Figure 4..

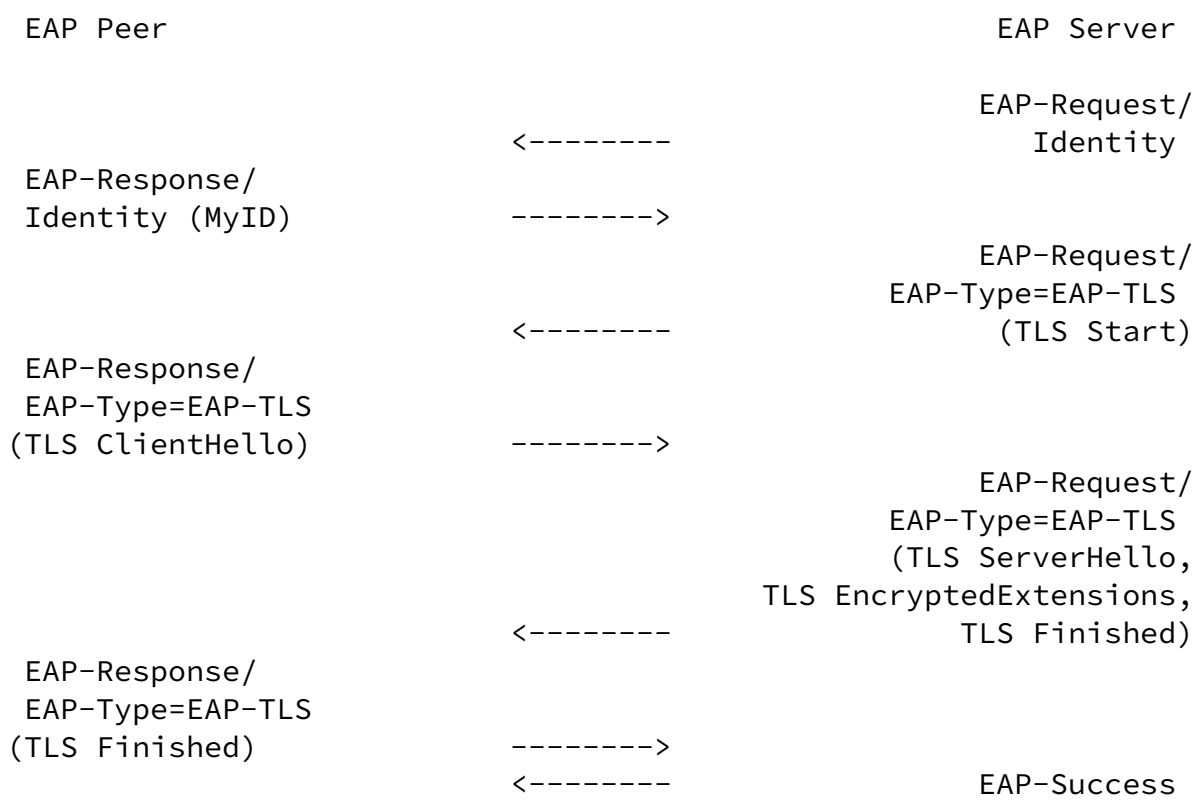


Figure 4: EAP-TLS resumption - Subsequent authentication

3.3. Termination

3.4. Fragmentation

A single TLS record may be up several thousand octets in length and a TLS message may consiste of multiple TLS records. TLS certificate message may of the order of 10s of Megabytes. The group of EAP-TLS messages sent in a single round may thus be larger than the MTU size or the maximum Remote Authentication Dail-In User Service (RADIUS) packet size of 4096 octets. As a result, an EAP-TLS implementation MUST provide its own support for fragmentation and reassembly.

In order to protect against reassembly lockup and denial-of-service attacks, it may be desirable for an implementation to set a maximum size for one such group of TLS messages. Since a single certificate is rarely longer than a few thousand octets, and no other field is likely to be anywhere near as long, a reasonable choice of maximum acceptable message length might be 64 Kilobyte as suggested in [RFC5216]

This specification reuses the mechanism of fragmentation and reassembly specified in [RFC5216]. The fragmentation support is provided through addition of a flags octet within the EAP-Response and EAP-Request packets, as well as a TLS Message Length field of four octets. The three 1-bit flags included are:

- Length included (L) bit flag: is set to indicate the presence of the four-octet TLS Message Length field. It MUST be set for the first fragment of a fragmented TLS message or set of messages.

- More fragments (M) bit flag: The M flag is set on all but the last fragment.

- EAP-TLS Start (S) bit flag: The S flag is set only within the EAP-TLS start message sent from the EAP server to the peer.

The remaining 5 bits in the flags octet are reserved. The TLS Message Length field is four octets, and provides the total length of the TLS message or set of messages that is being fragmented. This simplifies buffer allocation.

When an EAP-TLS peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-TLS and no data. This serves as an acknowledgment for the fragment. The EAP server MUST wait until it receives the EAP-Response (acknowledging the previous fragment) before sending another fragment.

As specified in [RFC3748] each EAP packet has an Identifier field. The EAP server MUST increment the Identifier field for each fragment contained within an EAP-Request, and the peer MUST include this Identifier value in the EAP-Response that acknowledges the fragment.

Similarly, when the EAP Peer needs to fragment a large message, it sends an EAP-Response with the M bit set. The EAP Server MUST respond to this with an EAP-Request with EAP-Type=EAP-TLS and no data. This acts as an acknowledgment for the fragment received from the EAP peer. The EAP peer MUST wait until it receives the EAP-Request before sending another fragment. Even when the EAP Peer fragments messages over several EAP-Response messages, it is the EAP Server that MUST increment the Identifier value for each fragment acknowledgment in the EAP-Request, and the peer MUST include this Identifier value in the subsequent fragment within the EAP-Response.

3.5. Key Heirarchy

...

3.6. Ciphersuite and Compression Negotiation

EAP-TLS implementations MUST support TLS v1.3.

To ensure interoperability, EAP-TLS peers and servers MUST support the TLS mandatory-to-implement ciphersuite: TLS_AES_128_GCM_SHA256 [GCM] and SHOULD implement the TLS_AES_256_GCM_SHA384 [GCM] and TLS_CHACHA20_POLY1305_SHA256 [RFC7539] cipher suites.

During the EAP-TLS conversation the EAP peer and server MUST NOT request or negotiate compression.

4. Security Considerations

4.1. EAP security claims

EAP security claims are defined in section 7.2.1 of [RFC3748]. The security claims for EAP-TLS are listed in Table 1.

Security property	EAP-TLS claim
Authentication mechanism	Certificates
Protected cryptosuite negotiation	yes
Mutual authentication	yes
Integrity protection	yes
Replay protection	yes
Key derivation	yes
Key strength	...
Dictionary attack protection	yes
Fast reconnect	yes
Cryptographic binding	not applicable
Session independence	yes
Fragmentation	no
Channel binding	...

Table 1: EAP security claims

4.2. Certificate Validation and Revocation Checks

In contrast to the EAP-TLS server, the EAP-TLS peer may not have Internet connectivity. Therefore, the EAP-TLS server SHOULD provide its entire certificate chain minus the root to facilitate certificate validation by the peer. The EAP-TLS peer SHOULD support validating the server certificate using RFC6818 [RFC6818] compliant path validation.

A EAP server MUST NOT request that a EAP Peer present an OCSP response with its certificate, i.e., the EAP server MUST NOT send an empty "status_request" extension in its CertificateRequest message.

5. IANA considerations

6. Acknowledgements

7. References

7.1. Normative References

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D , November 2007.

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.

[IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004. , December 2004.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,

<http://www.rfc-editor.org/info/rfc2119>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <http://www.rfc-editor.org/info/rfc3748>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <http://www.rfc-editor.org/info/rfc5216>.

[RFC6818] Yee, P., "Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 6818, DOI 10.17487/RFC6818, January 2013, <http://www.rfc-editor.org/info/rfc6818>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <http://www.rfc-editor.org/info/rfc7539>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <http://www.rfc-editor.org/info/rfc7542>.

7.2. Informative references

[IEEE-802.11]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", November 1997.

[IEEE-802.16e]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems", April 2002.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994, <http://www.rfc-editor.org/info/rfc1661>.

[RFC4793] Nystroem, M., "The EAP Protected One-Time Password

Protocol (EAP-POTP)", RFC 4793, DOI 10.17487/RFC4793, February 2007, <http://www.rfc-editor.org/info/rfc4793>.

[RFC5433] Clancy, T. and H. Tschofenig, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", RFC 5433, DOI 10.17487/RFC5433, February 2009, <http://www.rfc-editor.org/info/rfc5433>.

[RFC7406] Schulzrinne, H., McCann, S., Bajko, G., Tschofenig, H., and D. Kroeselberg, "Extensions to the Emergency Services Architecture for Dealing With Unauthenticated and Unauthorized Devices", RFC 7406, DOI 10.17487/RFC7406, December 2014, <http://www.rfc-editor.org/info/rfc7406>.

Authors' Addresses

John Mattsson
Ericsson
Farogatan 6
Kista 16480
Sweden

Email: john.mattsson@ericsson.com

Mohit Sethi
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: mohit@piuha.net