EMU Working Group   T. Clancy
Internet-Draft   LTS
Intended status: Standards Track   H. Tschofenig
Expires: January 7, 2008   Nokia Siemens Networks
                                                            July 6, 2007

EAP Generalized Pre-Shared Key (EAP-GPSK)
draft-ietf-emu-eap-gpsk-06

Status of this Memo

This Internet-Draft will expire on January 7, 2008.

Copyright Notice

Abstract

This Internet Draft defines an Extensible Authentication Protocol method called EAP Generalized Pre-Shared Key (EAP-GPSK). This method is a lightweight shared-key authentication protocol supporting mutual authentication and key derivation.

Table of Contents

1. Introduction

EAP Generalized Pre-Shared Key (EAP-GPSK) is an EAP method defining a generalized pre-shared key authentication technique. Mutual authentication is achieved through a nonce-based exchange that is secured by a pre-shared key.

EAP-GPSK addresses a large number of design goals with the intention of being applicable in a broad range of usage scenarios.

The main design goals of EAP-GPSK are

Simplicity:

EAP-GPSK should be easy to implement.

Security Model:

EAP-GPSK has been designed in a threat model where the attacker has full control over the communication channel. This is the EAP threat model that is presented in Section 7.1 of [RFC3748].

Efficiency:

EAP-GPSK does not make use of public key cryptography and fully relies of symmetric cryptography. The restriction on symmetric cryptographic computations allows for low computational overhead. Hence, EAP-GPSK is lightweight and well suited for any type of device, especially those with processing power, memory and battery constraints. Additionally it seeks to minimize the number of round trips.

Flexibility:

EAP-GPSK offers cryptographic flexibility. At the beginning, the EAP server selects a set of cryptographic algorithms and key sizes, a so called ciphersuite. The current version of EAP-GPSK comprises two ciphersuites, but additional ones can be easily added.

Extensibility:

The design of EAP-GPSK allows to securely exchange information between the EAP peer and the EAP server using protected data fields. These fields might, for example, be used to exchange channel binding information or to provide support for identity confidentiality.

2. Terminology

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. 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].

This section describes the various variables and functions used in the EAP-GPSK method.

Variables:

CSuite_List: An octet array listing available ciphersuites (variable length)

CSuite_Sel: Ciphersuite selected by the peer (6 octets)

ID_Peer: Peer NAI [RFC4282]

ID_Server: Server identity as an opaque blob.

KS: Integer representing the key size in octets of the selected ciphersuite CSuite_Sel. The key size is one of the ciphersuite parameters.

PD_Payload: Data carried within the protected data payload

PD_Payload_Block: Block of possibly multiple PD_Payloads carried by a GPSK packet

PL: Integer representing the length of the PSK in octets (2 octets)

RAND_Peer: Random integer generated by the peer (32 octets)

RAND_Server: Random integer generated by the server (32 octets)

Operations:

A || B: Concatenation of octet strings A and B

A**B: Integer exponentiation

truncate(A,B): Returns the first B octets of A

ENC_X(Y): Encryption of message Y with a symmetric key X, using a defined block cipher

KDF_X(Y): Key Derivation Function that generates an arbitrary number of octets of output using secret X and seed Y

length(X): Function that returns the length of input X in octets, encoded as a 2-octet integer in network byte order

MAC_X(Y): Keyed message authentication code computed over Y with symmetric key X

SEC_X(Y): SEC is a function that provides integrity protection based on the chosen ciphersuite. The function SEC uses the algorithm defined by the selected ciphersuite and applies it to the message content Y with key X. In short, SEC_X(Y) = Y || MAC_X(Y).

X[A..B]: Notation representing octets A through B of octet array X

The following abbreviations are used for the keying material:

EMSK: Extended Master Session Key is exported by the EAP method (64 octets)

MK: Master Key between the peer and EAP server from which all other EAP method session keys are derived (KS octets)

MSK: Master Session Key exported by the EAP method (64 octets)

PK: Session key generated from the MK and used during protocol exchange to encrypt protected data (KS octets)

PSK: Long-term key shared between the peer and the server (PL octets)

SK: Session key generated from the MK and used during protocol exchange to demonstrate knowledge of the PSK (KS octets)

3. Overview

The EAP framework (see Section 1.3 of [RFC3748]) defines three basic steps that occur during the execution of an EAP conversation between the EAP peer, the Authenticator and the EAP server.

1. The first phase, discovery, is handled by the underlying protocol.
2. The EAP authentication phase with EAP-GPSK is defined in this document.
3. The secure association distribution and secure association phases are handled differently depending on the underlying protocol.

EAP-GPSK performs mutual authentication between EAP peer ("Peer") and EAP server ("Server") based on a pre-shared key (PSK). The protocol consists of four message exchanges (GPSK-1, ..., GPSK-4), in which both sides exchange nonces and their identities, compute and exchange a Message Authentication Code (MAC) over the previously exchanged values, keyed with the pre-shared key. This MAC is considered as proof of possession of the pre-shared key.

A successful protocol exchange is shown in Figure 1.

```
+--------+                                     +--------+
|        |                EAP-Request/Identity |        |
|  EAP   |<------------------------------------|  EAP   |
|  peer  |                                     | server |
|        | EAP-Response/Identity               |        |
|        |------------------------------------>|        |
|        |                                     |        |
|        |                  EAP-Request/GPSK-1 |        |
|        |<------------------------------------|        |
|        |                                     |        |
|        | EAP-Response/GPSK-2                 |        |
|        |------------------------------------>|        |
|        |                                     |        |
|        |                  EAP-Request/GPSK-3 |        |
|        |<------------------------------------|        |
|        |                                     |        |
|        | EAP-Response/GPSK-4                 |        |
|        |------------------------------------>|        |
|        |                                     |        |
|        |          EAP-Success                |        |
|        |<------------------------------------|        |
+--------+                                     +--------+
```

Figure 1: EAP-GPSK: Successful Exchange

The full EAP-GPSK protocol is as follows:

GPSK-1:

ID_Server, RAND_Server, CSuite_List

GPSK-2:

SEC_SK(ID_Peer, ID_Server, RAND_Peer, RAND_Server, CSuite_List, CSuite_Sel, [ENC_PK(PD_Payload_Block)])

GPSK-3:

SEC_SK(RAND_Peer, RAND_Server, CSuite_Sel, [ENC_PK(PD_Payload_Block)])

GPSK-4:

SEC_SK([ENC_PK(PD_Payload_Block)])

The EAP server begins EAP-GPSK by selecting a random number RAND_Server and by encoding the supported ciphersuites into CSuite_List. A ciphersuite consists of an encryption algorithm, a key derivation function and a message authentication code.

In GPSK-1, the EAP server sends its identity ID_Server, a random number RAND_Server and a list of supported ciphersuites CSuite_List. The decision which ciphersuite to offer and which ciphersuite to pick is policy- and implementation-dependent and therefore outside the scope of this document.

In GPSK-2, the peer sends its identity ID_Peer and a random number RAND_Peer. Furthermore, it repeats the received parameters of the GPSK-1 message (ID_Server, RAND_Server, CSuite_List) and the selected ciphersuite. It computes a Message Authentication Code over all the transmitted parameters.

The EAP server verifies the received Message Authentication Code. In case of successful verification, the EAP server computes a Message Authentication Code over the session parameter and returns it to the peer (within GPSK-3). Within GPSK-2 and GPSK-3, peer and EAP server have the possibility to exchange encrypted protected data parameters.

The peer verifies the received Message Authentication Code. If the verification is successful, GPSK-4 is prepared. This message can optionally contain the peer's protected data parameters.

Upon receipt of GPSK-4, the server processes any included PD_Payload_Block. Then, the EAP server sends an EAP Success message

to indicate the successful outcome of the authentication.

4. Key Derivation

EAP-GPSK provides key derivation in compliance to the requirements of [RFC3748] and [I-D.ietf-eap-keying]. Note that this section provides an abstract description for the key derivation procedure that needs to be instantiated with a specific ciphersuite.

The long-term credential shared between EAP peer and EAP server SHOULD be a strong pre-shared key PSK of at least 16 octets, though its length and entropy is variable. While it is possible to use a password or passphrase, doing so is NOT RECOMMENDED as it would make EAP-GPSK vulnerable to dictionary attacks.

During an EAP-GPSK authentication, a Master Key MK, a Session Key SK and a Protected Data Encryption Key PK (if using an encrypting ciphersuite) are derived using the ciphersuite-specified KDF and data exchanged during the execution of the protocol, namely 'RAND_Peer || ID_Peer || RAND_Server || ID_Server' referred as inputString as its short-hand form.

In case of successful completion, EAP-GPSK derives and exports an MSK and EMSK both in length of 64 octets.

The following notation is used: KDF-X(Y, Z)[A..B], whereby
X is the length, in octets, of the desired output,
Y is a secret key,
Z is the inputString,
[A..B] extracts the string of octets starting with octet A finishing with octet B from the output of the KDF function.

This keying material is derived using the ciphersuite-specified KDF as follows:

o inputString = RAND_Peer || ID_Peer || RAND_Server || ID_Server
o zero = 0x00 || 0x00 || ... || 0x00 (KS times)

o MK = KDF-KS(zero, PL || PSK || CSuite_Sel || inputString)[0..KS-1]
o MSK = KDF-{128+2*KS}(MK, inputString)[0..63]
o EMSK = KDF-{128+2*KS}(MK, inputString)[64..127]
o SK = KDF-{128+2*KS}(MK, inputString)[128..127+KS]
o PK = KDF-{128+2*KS}(MK, inputString)[128+KS..127+2*KS] (if using an encrypting ciphersuite)

Additionally, the EAP keying framework [I-D.ietf-eap-keying] requires the definition of a Method-ID, Session-ID, Peer-ID, and Server-ID.

These values are defined as:

- o zero = 0x00 || 0x00 || ... || 0x00 (KS times)
- o Method-ID = KDF-16(zero, "Method ID" || EAP_Method_Type || CSuite_Sel || inputString)[0..15]
- o Session-ID = Type_Code || Method_ID
- o Peer-ID = ID_Peer
- o Server-ID = ID_Server

EAP_Method_Type refers to the integer value of the IANA allocated EAP Type code.

Figure 2 depicts the key derivation procedure of EAP-GPSK.

```
   +-------------+     +-------------------------------+
   |   PL-octet  |     | RAND_Peer || ID_Peer ||       |
   |     PSK     |     | RAND_Server || ID_Server      |
   +-------------+     +-------------------------------+
          |                            |            |
          |     +------------+         |            |
          |     | CSuite_Sel |         |            |
          |     +------------+         |            |
          |           |                |            |
          v           v                v            |
   +--------------------------------------------+   |
   |                    KDF                     |   |
   +--------------------------------------------+   |
                             |                      |
                             v                      |
                      +-------------+               |
                      |   KS-octet  |               |
                      |     MK      |               |
                      +-------------+               |
                             |                      |
                             v                      v
   +---------------------------------------------------+
   |                      KDF                          |
   +---------------------------------------------------+
        |             |             |            |
        v             v             v            v
   +---------+   +---------+  +----------+  +----------+
   | 64-octet|   | 64-octet|  | KS-octet |  | KS-octet |
   |   MSK   |   |  EMSK   |  |    SK    |  |   PK     |
   +---------+   +---------+  +----------+  +----------+
```

Figure 2: EAP-GPSK Key Derivation

5. Ciphersuites

The design of EAP-GPSK allows cryptographic algorithms and key sizes, called ciphersuites, to be negotiated during the protocol run. The ability to specify block-based and hash-based ciphersuites is offered. Extensibility is provided with the introduction of new ciphersuites; this document specifies an initial set. The CSuite/Specifier column in Figure 3 uniquely identifies a ciphersuite.

For a vendor-specific ciphersuite the first three octets are the vendor-specific Object Identifier (OID) contains the IANA assigned "SMI Network Management Private Enterprise Codes" value (see [RFC3232]), encoded in network byte order. The last three octets are vendor assigned for the specific ciphersuite.

The following ciphersuites are specified in this document:

CSuite/ Specifier	KS	Encryption	Integrity / KDF MAC	Key Derivation Function
0x000001	16	AES-CBC-128	AES-CMAC-128	GKDF
0x000002	32	NULL	HMAC-SHA256	GKDF

Figure 3: Ciphersuites

Ciphersuite 1, which is based on AES as a cryptographic primitive, is mandatory to implement. This document specifies also a second ciphersuite, but its support is optional.

Each ciphersuite needs to specify a key derivation function. The ciphersuites defined in this document make use of the Generalized Key Distribution Function (GKDF) which utilizes the MAC function defined in the ciphersuite. Future ciphersuites can use any other formally specified KDF that takes as arguments a key and a seed value, and produces at least 128+2*KS octets of output.

GKDF has the following structure:

GKDF-X(Y, Z)

```
X  length, in octets, of the desired output
Y  secret key
Z  inputString


GKDF-X (Y, Z)
{
  n = ceiling integer of ( X / KS );
     /* determine number of output blocks */

  M_0 = "";
  result = "";

  for i = 1 to n {
    M_i = MAC_Y (i || Z);
    result = result || M_i;
  }

  return truncate(result, X)
}
```

Note that the variable 'i' in M_i is represented as a 2-octet value in network byte order.

6. Ciphersuites Processing Rules

6.1. Ciphersuite #1

6.1.1. Encryption

With this ciphersuite all cryptography is built around a single cryptographic primitive, AES-128 ([AES]). Within the protected data frames, AES-128 is used in Cipher Block Chaining (CBC) mode of operation (see [CBC]). This EAP method uses encryption in a single payload, in the protected data payload (see Section 7.4).

In a nutshell, the CBC mode proceeds as follows. The IV is XORed with the first plaintext block before it is encrypted. Then for successive blocks, the previous ciphertext block is XORed with the current plaintext, before it is encrypted.

6.1.2. Integrity

Ciphersuite 1 uses CMAC as Message Authentication Code. CMAC is recommended by NIST. Among its advantages, CMAC is capable to work with messages of arbitrary length. A detailed description of CMAC can be found in [CMAC].

The following instantiation is used: AES-CMAC-128(SK, Input) denotes the MAC of Input under the key SK.

where Input refers to the following content:

o Value of SEC_SK(Value) in message GPSK-2
o Value of SEC_SK(Value) in message GPSK-3
o Value of SEC_SK(Value) in message GPSK-4

6.1.3. Key Derivation

This ciphersuite instantiates the KDF in the following way:

inputString = RAND_Peer || ID_Peer || RAND_Server || ID_Server

zero = 0x00 || 0x00 || ... || 0x00 (16 times)

MK = GKDF-16 (zero, PL || PSK || CSuite_Sel || inputString)

MSK = GKDF-160 (MK, inputString)[0..63]

EMSK = GKDF-160 (MK, inputString)[64..127]

SK = GKDF-160 (MK, inputString)[128..143]

PK = GKDF-160 (MK, inputString)[144..159]

Method-ID = GKDF-16 (zero, "Method ID" || EAP_Method_Type || CSuite_Sel || inputString)

6.2. Ciphersuite #2

6.2.1. Encryption

Ciphersuite 2 does not include an algorithm for encryption. With a NULL encryption algorithm, encryption is defined as:

E_X(Y) = Y

When using this ciphersuite, the data exchanged inside the protected data block is not encrypted. Therefore this mode MUST NOT be used if confidential information appears inside the protected data block.

6.2.2. Integrity

Ciphersuite 2 uses the keyed MAC function HMAC, with the SHA256 hash algorithm (see [RFC4634]).

For integrity protection the following instantiation is used:

HMAC-SHA256(SK, Input) denotes the MAC of Input under the key SK where Input refers to the following content:

- o Value of SEC_SK(Value) in message GPSK-2
- o Value of SEC_SK(Value) in message GPSK-3
- o Value of SEC_SK(Value) in message GPSK-4

6.2.3. Key Derivation

This ciphersuite instantiates the KDF in the following way:

inputString = RAND_Peer || ID_Peer || RAND_Server || ID_Server

zero = 0x00 || 0x00 || ... || 0x00 (32 times)

MK = GKDF-32 (zero, PL || PSK || CSuite_Sel || inputString)

MSK = GKDF-160 (MK, inputString)[0..63]

EMSK = GKDF-160 (MK, inputString)[64..127]

SK = GKDF-160 (MK, inputString)[128..159]

Method-ID = GKDF-16 (zero, "Method ID" || EAP_Method_Type || CSuite_Sel || inputString)

7. Packet Formats

This section defines the packet format of the EAP-GPSK messages.

7.1. Header Format

The EAP-GPSK header has the following structure:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    OP-Code    |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         Payload                           ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5

The Code, Identifier, Length, and Type fields are all part of the EAP header, and defined in [RFC3748]. IANA has allocated EAP Method Type XX for EAP-GPSK, thus the Type field in the EAP header MUST be XX.

The OP-Code field is one of four values:

- o 0x01 : GPSK-1
- o 0x02 : GPSK-2
- o 0x03 : GPSK-3
- o 0x04 : GPSK-4
- o 0x05 : GPSK-Fail
- o 0x06 : GPSK-Protected-Fail

All other values of this OP-Code field are available via IANA registration.

7.2. Ciphersuite Formatting

Ciphersuites are encoded as 6-octet arrays. The first four octets indicate the CSuite/Vendor field. For vendor-specific ciphersuites, this represents the vendor Object Identifier (OID) contains the IANA assigned "SMI Network Management Private Enterprise Codes" value (see [RFC3232]), encoded in network byte order. The last two octets indicate the CSuite/Specifier field, which identifies the particular ciphersuite. The 4-octet CSuite/Vendor value 0x00000000 indicates ciphersuites allocated by the IETF.

Graphically, they are represented as

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        CSuite/Vendor = 0x00000000 or OID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      CSuite/Specifier         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6

CSuite_Sel is encoded as a 6-octet ciphersuite CSuite/Vendor and CSuite/Specifier pair.

CSuite_List is a variable-length octet array of ciphersuites. It is encoded by concatenating encoded ciphersuite values. Its length in octets MUST be a multiple of 6.

7.3. Payload Formatting

Payload formatting is based on the protocol exchange description in Section 3.

The GPSK-1 payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       length(ID_Server)       |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Server                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   32-octet RAND_Server                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      length(CSuite_List)      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                        CSuite_List                        ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: GPSK-1 Payload

The GPSK-2 payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         length(ID_Peer)       |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Peer                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        length(ID_Server)      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                         ID_Server                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                     32-octet RAND_Peer                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    32-octet RAND_Server                   ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       length(CSuite_List)     |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                        CSuite_List                        ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           CSuite_Sel                          |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |   length(PD_Payload_Block)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                 optional PD_Payload_Block                 ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   KS-octet payload MAC                    ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: GPSK-2 Payload

If the optional protected data payload is not included, then length(PD_Payload_Block)=0 and the PD payload is excluded.

The GPSK-3 payload is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    32-octet RAND_Peer                     ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    32-octet RAND_Server                     ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          CSuite_Sel                           |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |   length(PD_Payload_Block)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                 optional PD_Payload_Block                   ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                   KS-octet payload MAC                      ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: GPSK-3 Payload

If the optional protected data payload is not included, then length(PD_Payload_Block)=0 and the PD payload is excluded.

The GPSK-4 payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   length(PD_Payload_Block)    |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
...                  optional PD_Payload_Block                ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    KS-octet payload MAC                   ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: GPSK-4 Payload

If the optional protected data payload is not included, then length(PD_Payload_Block)=0 and the PD payload is excluded. The MAC MUST always be included, regardless of the presence of PD_Payload_Block.

The GPSK-Fail payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Failure-Code                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: GPSK-Fail Payload

The GPSK-Protected-Fail payload format is defined as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Failure-Code                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                    KS-octet payload MAC                   ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: GPSK-Protected-Fail Payload

The Failure-Code field is one of three values, but can be extended:

- o 0x00000001: PSK Not Found
- o 0x00000002: Authentication Failure
- o 0x00000003: Authorization Failure

All other values of this field are available via IANA registration.

"PSK Not Found" indicates a key for a particular user could not be located, making authentication impossible. "Authentication Failure" indicates a MAC failure due to a PSK mismatch. "Authorization Failure" indicates that while the PSK being used is correct, the user is not authorized to connect.

7.4. Protected Data

The protected data blocks are a generic mechanism for the peer and server to securely exchange data. If the specified ciphersuite has a NULL encryption primitive, then this channel only offers authenticity, and not confidentiality.

These payloads are encoded as the concatenation of type-length-value (TLV) triples called PD_Payloads.

Type values are encoded as a 6-octet string and represented by a 4-octet vendor and 2-octet specifier field. The vendor field indicates the type as either standards-specified or vendor-specific. If these three octets are 0x00000000, then the value is standards-specified, and any other value represents a vendor-specific Object Identifier (OID).

The specifier field indicates the actual type. For vendor field 0x00000000, the specifier field is maintained by IANA. For any other vendor field, the specifier field is maintained by the vendor.

Length fields are specified as 2-octet integers in network byte order, and reflect only the length of the value, and do not include the length of the type and length fields.

Graphically, this can be depicted as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   PData/Vendor                 |             ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         PData/Specifier        |         PData/Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                        PData/Value                        ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Protected Data Payload (PD_Payload) Formatting

These PD_Payloads are concatenated together to form a PD_Payload_Block. The If the CSuite_Sel includes support for encryption, then the PD_Payload_Block includes fields specifying an initialization vector (IV), and the necessary padding. This can be depicted as follows:

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Initialization Vector                    |
...       (length is block size for encryption algorithm)     ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                        PD_Payload                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                 optional PD_Payload, etc                  ...
|                                                               |
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |             Padding (0-255 octets)            |
+-+-+-+-+-+-+-+-+                               +-+-+-+-+-+-+-+-+
|                                               |  Pad Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Protected Data Block (PD_Payload_Block) Formatting if Encryption Supported

The Initialization Vector is a randomly chosen value whose length is equal to the block length of the underlying encryption algorithm.

Recipients MUST accept any value. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. Senders MUST NOT use the same value for each message, use a sequence of values with low hamming distance (e.g., a sequence number), or use ciphertext from a received message.

The concatenation of PD_Payloads along with the padding and padding length are all encrypted using the negotiated block cipher. If no block cipher is specified, then these fields are not encrypted.

The Padding field MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the concatenation of PD_Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

The Pad Length field is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the PD_Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

If the negotiated ciphersuite does not support encryption, then the padding field MUST be of length zero. The padding length field MUST still be present, and contain the value zero. This is depicted in the following figure.

```
--- bit offset --->
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                        PD_Payload                         ...
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
...                 optional PD_Payload, etc    +-+-+-+-+-+-+-+-+
|                                               |      0x00     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Protected Data Block (PD_Payload_Block) Formatting Without Encryption

For PData/Vendor field 0x000000, the following PData/Specifier fields are defined:

o 0x000000 : Reserved
o 0x000001 : Protected Results Indication
All other values of this field are available via IANA registration.

7.4.1. Protected Results Indication

Based on the PData/Specifier allocation the following 8-bit payload is specified to be placed in the PD_Payload Value to provide the functionality of protected results indication.

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|I|R|R|R|R|R|R|R|
+-+-+-+-+-+-+-+-+
```

I: Result Indicator

The bits have the following meaning:

(0): Success
(1): Failure

R: Reserved
These bits are used for padding.

The 8 bits of protected results indication functionality MUST only be sent in GPSK-3 from the EAP server to the EAP peer.

8. Packet Processing Rules

This section defines how the EAP peer and EAP server MUST behave when received packet is deemed invalid.

Any EAP-GPSK packet that cannot be parsed by the EAP peer or the EAP server MUST be silently discarded. An EAP peer or EAP server receiving any unexpected packet (e.g., an EAP peer receiving GPSK-3 before receiving GPSK-1 or before transmitting GPSK-2) MUST silently discard the packet.

GPSK-1 contains no MAC protection, so provided it properly parses, it MUST be accepted by the peer. Note that the ciphersuite list provided by the EAP server in CSuite_List MUST always include the mandatory-to-implement ciphersuite defined in this document. Hence, there is always at least one ciphersuite in common between the EAP peer and the EAP server. If the EAP peer decides the ID_Server is

that of a AAA server to which it does not wish to authenticate, the EAP peer should respond with an EAP-NAK.

For GPSK-2, if ID_Peer is for an unknown user, the EAP server MUST send either a "PSK Not Found" GPSK-Fail message, or an "Authentication Failure" GPSK-Fail, depending on its policy, and discard the received packet. If the MAC validation fails, the server MUST transmit a GPSK-Fail message specifying "Authentication Failure" and discard the received packet. If the RAND_Server or CSuite_List field in GPSK-2 does not match the values in GPSK-1, the server MUST silently discard the packet. If server policy determines the peer is not authorized and the MAC is correct, the server MUST transmit a GPSK-Protected-Fail message indicating "Authorization Failure" and discard the received packet.

A peer receiving a GPSK-Fail / GPSK-Protected-Fail message in response to a GPSK-2 message MUST replay the received GPSK-Fail / GPSK-Protected-Fail message. Then, the EAP server returns an EAP-Failure after receiving the GPSK-Fail / GPSK-Protected-Fail message to correctly finish the EAP conversation. If MAC validation on a GPSK-Protected-Fail packet fails, then the received packet MUST be silently discarded.

For GPSK-3, a peer MUST silently discard messages where the RAND_Peer, the RAND_Server, or the CSuite_Sel fields do match those transmitted in GPSK-2. An EAP peer MUST silently discard any packet whose MAC fails.

For GPSK-4, a server MUST silently discard any packet whose MAC fails validation.

If a decryption failure of a protected payload is detected, the recipient MUST silently discard the GPSK packet.

9. Example Message Exchanges

This section shows a couple of example message flows.

A successful EAP-GPSK message exchange is shown in Figure 1.

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/EAP-NAK                |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

    EAP-GPSK: Unsuccessful Exchange (Unacceptable AAA server identity;
                                ID_Server)

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-2                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        | EAP-Request/GPSK-3 (GPSK-Fail       |        |
   |        | (PSK Not Found or Authentication    |        |
   |        | Failure))                           |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-4 (GPSK-Fail      |        |
   |        | (PSK Not Found or Authentication    |        |
   |        | Failure))                           |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

EAP-GPSK: Unsuccessful Exchange (Unknown user)

```
   +--------+                                     +--------+
   |        |                EAP-Request/Identity |        |
   |  EAP   |<------------------------------------|  EAP   |
   |  peer  |                                     | server |
   |        | EAP-Response/Identity               |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |                  EAP-Request/GPSK-1 |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-2                 |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        | EAP-Request/GPSK-3 (GPSK-Fail       |        |
   |        | (Authentication Failure))           |        |
   |        |<------------------------------------|        |
   |        |                                     |        |
   |        | EAP-Response/GPSK-4 (GPSK-Fail      |        |
   |        | (Authentication Failure))           |        |
   |        |------------------------------------>|        |
   |        |                                     |        |
   |        |          EAP-Failure                |        |
   |        |<------------------------------------|        |
   +--------+                                     +--------+
```

EAP-GPSK: Unsuccessful Exchange (Invalid MAC in GPSK-2)

```
+--------+                                     +--------+
|        |                 EAP-Request/Identity |        |
|  EAP   |<------------------------------------|  EAP   |
|  peer  |                                     | server |
|        | EAP-Response/Identity               |        |
|        |------------------------------------>|        |
|        |                                     |        |
|        |                   EAP-Request/GPSK-1 |        |
|        |<------------------------------------|        |
|        |                                     |        |
|        | EAP-Response/GPSK-2                 |        |
|        |------------------------------------>|        |
|        |                                     |        |
|        | EAP-Request/GPSK-3                  |        |
|        | GPSK-Protected-Fail                 |        |
|        | (Authorization Failure)             |        |
|        |<------------------------------------|        |
|        |                                     |        |
|        | EAP-Request/GPSK-4                  |        |
|        | GPSK-Protected-Fail                 |        |
|        | (Authorization Failure)             |        |
|        |------------------------------------>|        |
|        |                                     |        |
|        |          EAP-Failure                |        |
|        |<------------------------------------|        |
+--------+                                     +--------+
```

EAP-GPSK: Unsuccessful Exchange (Authorization failure)

10. Security Considerations

[RFC3748] highlights several attacks that are possible against EAP since EAP itself does not provide any security.

This section discusses the claimed security properties of EAP-GPSK as well as vulnerabilities and security recommendations in the threat model of [RFC3748].

10.1. Mutual Authentication

EAP-GPSK provides mutual authentication.

The server believes that the peer is authentic because it can calculate a valid MAC and the peer believes that the server is authentic because it can calculate another valid MAC.

The key used for mutual authentication is computed again based on the

long-term secret PSK that has to provide sufficient entropy and therefore sufficient strength. In this way EAP-GPSK is not different than other authentication protocols based on pre-shared keys.

10.2. Protected Result Indications

EAP-GPSK offers the capability to exchange protected result indications using the protected data payloads.

10.3. Integrity Protection

EAP-GPSK provides integrity protection based on the ciphersuites suggested in this document.

10.4. Replay Protection

EAP-GPSK provides replay protection of its mutual authentication part thanks to the use of random numbers RAND_Server and RAND_Peer. Since RAND_Server is 32 octets long, one expects to have to record 2**64 (i.e., approximately 1.84*10**19) EAP-GPSK successful authentication before an protocol run can be replayed. Hence, EAP-GPSK provides replay protection of its mutual authentication part as long as RAND_Server and RAND_Peer are chosen at random, randomness is critical for replay protection.

10.5. Reflection attacks

EAP-GPSK provides protection against reflection attacks in case of an extended authentication because the messages are constructed in a different fashion.

10.6. Dictionary Attacks

EAP-GPSK relies on a long-term shared secret (PSK) that MUST be based on at least 16 octets of entropy to guarantee security against dictionary attacks. Users who use passwords are not guaranteed security against dictionary attacks. Derivation of the long-term shared secret from a password is strongly discouraged.

10.7. Key Derivation

EAP-GPSK supports key derivation as shown in Section 4.

10.8. Denial of Service Resistance

Denial of Service (DoS) resistance has not been a design goal for EAP-GPSK.

It is however believed that EAP-GPSK does not provide any obvious and avoidable venue for such attacks.

It is worth noting that the server has to maintain some state when it engages in an EAP-GPSK conversation, namely to generate and to remember the 32-octet RAND_Server. This should however not lead to resource exhaustion as this state and the associated computation are fairly lightweight.

It is recommended that EAP-GPSK does not allow EAP notifications to be interleaved in its dialog to prevent potential DoS attacks. Indeed, since EAP Notifications are not integrity protected, they can easily be spoofed by an attacker. Such an attacker could force a peer that allows EAP Notifications to engage in a discussion which would delay his authentication or result in the peer taking unexpected actions (e.g., in case a notification is used to prompt the peer to do some "bad" action).

It is up to the implementation of EAP-GPSK or to the peer and the server to specify the maximum number of failed cryptographic checks that are allowed.

10.9. Session Independence

Thanks to its key derivation mechanisms, EAP-GPSK provides session independence: passive attacks (such as capture of the EAP conversation) or active attacks (including compromise of the MSK or EMSK) do not enable compromise of subsequent or prior MSKs or EMSKs. The assumption that RAND_Peer and RAND_Server are random is central for the security of EAP-GPSK in general and session independence in particular.

10.10. Exposition of the PSK

EAP-GPSK does not provide perfect forward secrecy. Compromise of the PSK leads to compromise of recorded past sessions.

Compromise of the PSK enables the attacker to impersonate the peer and the server and it allows the adversary to compromise future sessions.

EAP-GPSK provides no protection against a legitimate peer sharing its PSK with a third party. Such protection may be provided by appropriate repositories for the PSK, which choice is outside the scope of this document. The PSK used by EAP-GPSK must only be shared between two parties: the peer and the server. In particular, this PSK must not be shared by a group of peers communicating with the same server.

The PSK used by EAP-GPSK must be cryptographically separated from keys used by other protocols, otherwise the security of EAP-GPSK may be compromised.

10.11. Fragmentation

EAP-GPSK does not support fragmentation and reassembly since the message size is small.

10.12. Channel Binding

This document enables the ability to exchange channel binding information. It does not, however, define the encoding of channel binding information in the document.

10.13. Fast Reconnect

EAP-GPSK does not provide the fast reconnect capability since this method is already at (or close to) the lower limit of the number of roundtrips and the cryptographic operations.

10.14. Identity Protection

Identity protection is not specified in this document. Extensions can be defined that enhance this protocol to provide this feature.

10.15. Protected Ciphersuite Negotiation

EAP-GPSK provides protected ciphersuite negotiation via the indication of available ciphersuites by the server in the first message and a confirmation by the peer in the subsequent message.

10.16. Confidentiality

Although EAP-GPSK provides confidentiality in its protected data payloads, it cannot claim to do so as per Section 7.2.1 of [RFC3748].

10.17. Cryptographic Binding

Since EAP-GPSK does not tunnel another EAP method, it does not implement cryptographic binding.

11. IANA Considerations

This document requires IANA to allocate a new EAP Type for EAP-GPSK.

This document requires IANA to create a new registry for

ciphersuites, protected data types, failure codes and op-codes. IANA is furthermore instructed to add the specified ciphersuites, protected data types, failure codes and op-codes to these registries as defined in this document. Values can be added or modified with informational RFCs defining either block-based or hash-based ciphersuites, protected data payloads, failure codes and op-codes. Each ciphersuite needs to provide processing rules and needs to specify how the following algorithms are instantiated: encryption, integrity, key derivation and key length.

Figure 3 represents the initial ciphersuite CSuite/Specifier registry setup. The CSuite/Specifier field is 16 bits long. All other values are available via IANA registration.

The following is the initial protected data PData/Specifier registry setup:

- 0x000000 : Reserved
- 0x000001 : Protected Results Indication

The PData/Specifier field is 24 bits long and all other values are available via IANA registration.

The following layout represents the initial Failure-Code registry setup:

- 0x00000001: PSK Not Found
- 0x00000002: Authentication Failure
- 0x00000003: Authorization Failure

The Failure-Code field is 32 bits long and all other values are available via IANA registration.

The following layout represents the initial OP-Code registry setup:

- 0x01 : GPSK-1
- 0x02 : GPSK-2
- 0x03 : GPSK-3
- 0x04 : GPSK-4
- 0x05 : GPSK-Fail
- 0x06 : GPSK-Protected-Fail

The OP-Code field is 8 bits long and all other values are available via IANA registration.

12. Contributors

This work is a joint effort of the EAP Method Update (EMU) design team of the EMU Working Group that was created to develop a mechanism based on strong shared secrets that meets RFC 3748 [RFC3748] and RFC 4017 [RFC4017] requirements. The design team members (in alphabetical order) were:

- o Jari Arkko
- o Mohamad Badra
- o Uri Blumenthal
- o Charles Clancy
- o Lakshminath Dondeti
- o David McGrew
- o Joe Salowey
- o Sharma Suman
- o Hannes Tschofenig
- o Jesse Walker

Finally, we would like to thank Thomas Otto for his draft reviews, feedback and text contributions.

13. Acknowledgments

We would like to thank

- o Jouni Malinen and Bernard Aboba for their early draft comments in June 2006. Jouni Malinen developed the first prototype implementation. It can be found at: http://hostap.epitest.fi/releases/snapshots/
- o Lakshminath Dondeti, David McGrew, Bernard Aboba, Michaela Vanderveen and Ray Bell for their input to the ciphersuite discussions between July and August 2006.
- o Lakshminath Dondeti for his detailed draft review (sent to the EMU ML on the 12th July 2006).
- o Based on a review requested from NIST Quynh Dang suggested changes to the GKDF function (December 2006).
- o Jouni Malinen and Victor Fajardo for their review in January 2007.
- o Jouni Malinen for his suggestions regarding the examples and the key derivation function in February 2007.
- o Bernard Aboba and Jouni Malinen for their review in February 2007.
- o Vidya Narayanan for her review in March 2007.
- o
- o Joe Salowey, the EMU working group chair, provided a document review in April 2007. Jouni Malinen also reviewed the document during the same month.

14. References

14.1. Normative References

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

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

14.2. Informative References

[I-D.ietf-eap-keying] Aboba, B., "Extensible Authentication Protocol (EAP) Key Management Framework", draft-ietf-eap-keying-18 (work in progress), February 2007.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

[AES] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)", Federal Information Processing Standards (FIPS) 197, November 2001.

[CMAC] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", Special Publication (SP) 800-38B, May 2005.

[CBC] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Encryption. Methods and Techniques.", Special Publication (SP) 800-38A, December 2001.

[RFC3232] Reynolds, J., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, January 2002.

Authors' Addresses

T. Charles Clancy
DoD Laboratory for Telecommunications Sciences
8080 Greenmead Drive
College Park, MD 20740
USA

Email: clancy@ltsnet.net

Hannes Tschofenig
Nokia Siemens Networks
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

Email: Hannes.Tschofenig@nsn.com
URI: http://www.tschofenig.com

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).