Internet Draft
expires in six months
September 17, 2001
Expires March 2002

Romain Berrendonner (SAGEM SA)
Herve Chabanne (SAGEM SA)

PPP EAP MAKE Mutual Authentication Protocol
<draft-berrendo-chabanne-pppext-eapmake-00.txt>

Status of this Memo

Abstract

This memo describes EAP-MAKE protocol, an authentication protocol based on EAP which emphasizes on simplicity and scalability. Authentication is provided through a mechanism derived from the Diffie-Hellman scheme, and it is possible to derive and check a common symmetric key for the purpose of privacy. Scalability is provided by the underlying support of legacy PKI systems.

1. Document Conventions

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. PPP EAP MAKE Mutual Authentication Protocol

2.1. Introduction

This document describes the PPP EAP MAKE protocol where MAKE stands for Mutual Authentication protocol with Key Exchange.

PPP EAP MAKE protocol borrows a lot to "PPP EAP KEA Public Key Authentication Protocol" (see Acknowledgments), namely :
- both protocols are aimed to define an authentication mechanism within PPP EAP [1] and both permits the creation of a session key for encryption of data on the PPP link,
- PPP EAP MAKE protocol takes the two 2-pass EAP request-response of PPP EAP KEA and their format for its own, in the sequel, the first 2-pass is called EAP MAKE1 and the second one EAP MAKE2,
- the underlying cryptographic property which allows mutual authentication is essentially the same and consists in supplying the prover and the verifier with a private/public Diffie-Hellman key pair.

Nevertheless, EAP MAKE protocol has its own specificities which are :
- the flow of operations is asymmetric in the sense that there is a prover and a verifier, that prover and verifier do not perform the same computations ; most notably only partial "prover side" forward secrecy is wanted,
- the general format of a message of the PPP EAP MAKE protocol is "some data" followed by the HMAC [2] of these data under the common prover/verifier Diffie-Hellman key.

2.2. Prerequisites

Before describing the PPP EAP MAKE protocol, some elements have to be established.

The hypothesis is here made that both have exchanged some data before the use of the PPP EAP MAKE protocol. In particular they MUST agree on a way of identifying each other. For instance,this could be by their distinguished name. But some other ways can also be imagined. In the following, the verifier is identified as "A" , the prover as "B" .

By certificate, X509 certificate as defined for instance in [4] are here understood. Other choices are possible. In any cases, it is assumed that B SHOULD (resp. A MUST) be able to retrieve and check the validity of the certificate of their correspondent.

They also agree on a Diffie-Hellman group. In this memo by Diffie-Hellman we understand Diffie-Hellman computations made modulo a big prime. Other choices are possible as working in finite fields of characteristic 2 or over elliptic curves. In any cases, A and B MUST share the knowledge of the underlying group required for Diffie-Hellmann common key computation.

Then they MUST agree on an encryption algorithm, an hash algorithm and

an HMAC algorithm.

Finally, a counter which provides a basic but efficient anti-replay mechanism is maintained. This counter is incremented at each attempt of identification. The initial value of this counter is 0. In the following, LID stands for the value of this counter. Both A and B MUST store LID. In what follows, LID is 4 bytes long, but this can be easily changed.

2.3. PPP EAP MAKE protocol in a nutshell

Let's call this Diffie-Hellman common key between A and B, KDH, p the "big prime" which defines the group where Diffie-Hellman keys are computed, privA (resp. privB)/ pubA (resp. pubB) the private / public Diffie-Hellman key pair of A (resp. of B).

We write
- ENCRYPT(D ; K) for the encryption of data D under key K,
- H(D) for the computation of the hash of data D,
- HMAC(D1, D2, ... , Dn ; K) for the computation of the HMAC of concatenated data D1, D2, ..., Dn under key K.

To make things completely clear, let say that
- p is 128 bytes long,
- HMAC is performed using SHA-1 as an hash function. Its output is truncated to 16 bytes [2],
- H is computed according to SHA-1. [5],
- ENCRYPT corresponds to 3DES E-D-E in CBC mode.

MAKE1 Request :
- A initiates the protocol by sending to B its identity.

MAKE1 Response :
- B checks A's certificate validity
- B computes KDH
- B increments LID
- B chooses at random r, a private Diffie-Hellman key
- B computes R the public key corresponding to r
- B computes HMAC(B, LID, R, A ; KDH)
- B sends to A : B, LID, R, HMAC(B, LID, R, A ; KDH)

MAKE2 Request :
- A checks validity of LID, B's certificate
- A computes KDH
- A checks validity of HMAC(B, LID, R, A ; KDH)
- A computes KS a session key as
  Ks = HMAC(A, B, LID, KDH ; R**privA mod p)
- A chooses at random K which is going to encrypt data on the PPP link
- A chooses at random n a nonce

```
 - A encrypts K under Ks, set
       K' = ENCRYPT(K ; Ks)
 - A encrypts n under K, set
       n' = ENCRYPT(n ; K)
 - A computes HMAC (A, B, LID, K', n' ; KDH)
 - A sends to B : K', n', HMAC (B, LID, R, A, K', n' ; KDH)

MAKE2 Response :
 - B checks validity of HMAC (B, LID, R, A, K', n' ; KDH)
 - B computes Ks = HMAC(A, B, LID, KDH ; pubA**r mod p)
 - B retrieves K and n
 - B computes H(n)
 - B sends to A : H(n)

Finally :
 - A checks validity of H(n)
 - A updates LID
```

Figure 1 : General description of EAP MAKE Protocol

3. EAP MAKE Packet Format

Four packets are exchanged in order to perform the authentication, first from A to B, and then from B to A, then repeating that sequence.

Both the EAP Response and Request packets for the MAKE1 and MAKE2 Subtype have the following format :

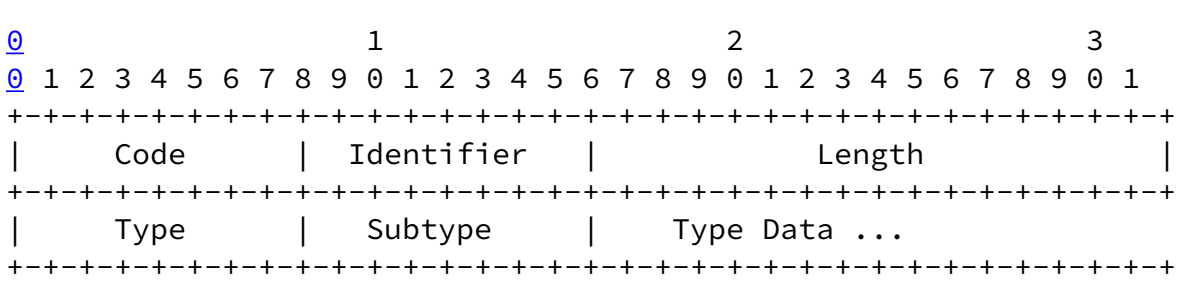


Figure 2 : The PPP EAP packet format

Code

1 (Request)
2 (Response)

Identifier

The Identifier field is one octet and aids in matching responses with requests. The Identifier MUST be changed for each new Request message sent and MUST NOT be changed on retransmission o a given message. The Identifier in the Response message MUST match the corresponding Request. The verifier MUST discard non-matching Response messages.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, Subtype and Subtype-Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception. Truncated packets (with Length greater than the link layer reported length) MUST be silently discarded.

Type

The Type field will carry the value 21 (EAP MAKE).
The Type field in the
Response SHOULD carry the corresponding value unless the peer wishes to send a Nak Type to indicate that it is incapable of handling MAKE authentication.

Subtype

1 - MAKE1
2 - MAKE2

The Subtype field is one octet and must contain the value 1, 2. If any other subtype value is encountered in an EAP MAKE Request message, then the prover SHOULD return an EAP Response with the Type field set to Nak. In EAP MAKE Response messages, the Subtype value MUST be copied from the corresponding Request message. The verifier SHOULD treat unknown Subtype values in Response messages as malformed packets and silently discard.

Subtype Data

The contents and formats of the remainder of the packet differ for each of the four packet types:
1) MAKE1 Request,
2) MAKE1 Response,
3) MAKE2 Request,
4) MAKE2 Response.

The following sections define the format of the request and response where the information is formatted in a length-value format. No explicit type field is necessary because all fields are required and are in a determinate order. The last element does not include a length field because its length can be determined from the overall length. When a packet is ill-formatted, an EAP failure packet MUST be send.

3.1. EAP MAKE1 Request Packet

The EAP MAKE1 Request packet has the following overall format:

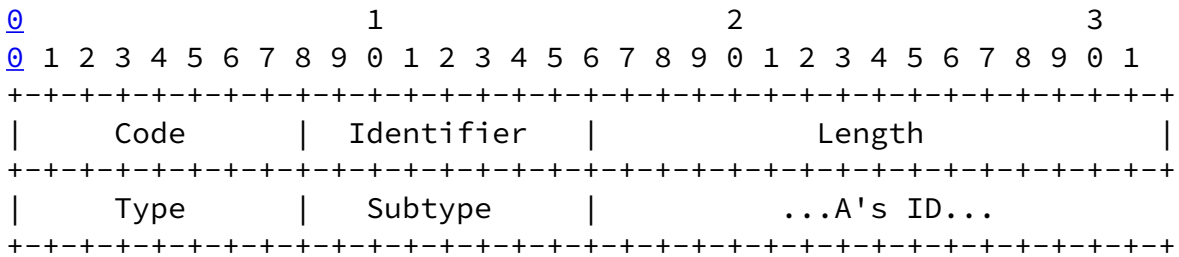


Figure 3 : EAP MAKE1 Request Packet Format

Code : 1 (Request)

Identifier : ID1 (random value)

Length : length of
Code
Identifier
Length
Type
Subtype
A's Identity

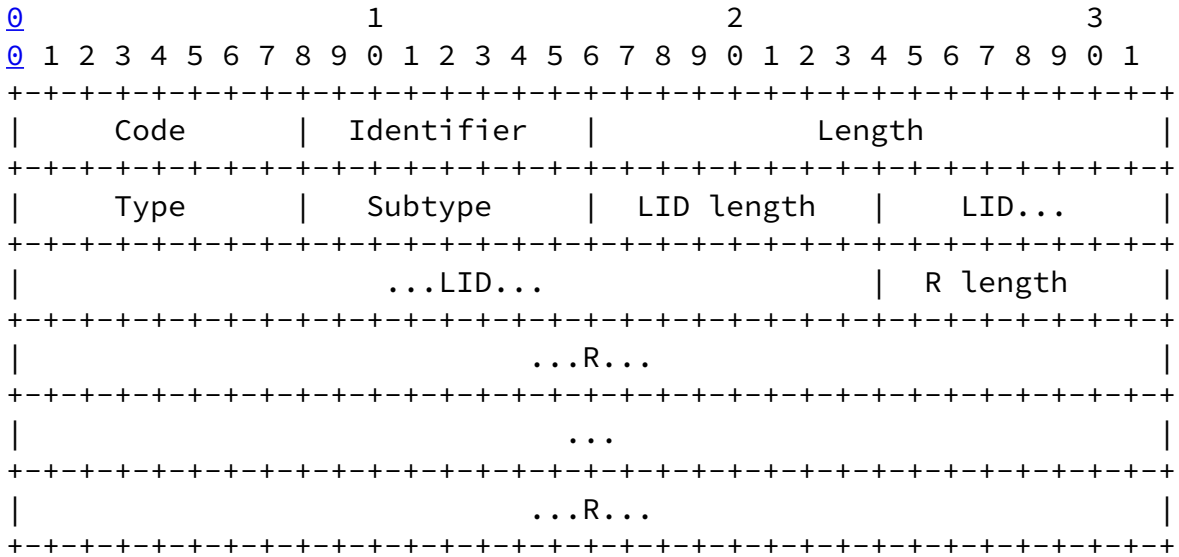
Type : 21 (MAKE)

Subtype : 1 (MAKE1)

A's ID : B SHOULD check A's certificate validity.
If not valid, B SHOULD send an
EAP Failure packet.

3.2. EAP MAKE1 Response Packet

The EAP MAKE1 Response packet has the following overall format:



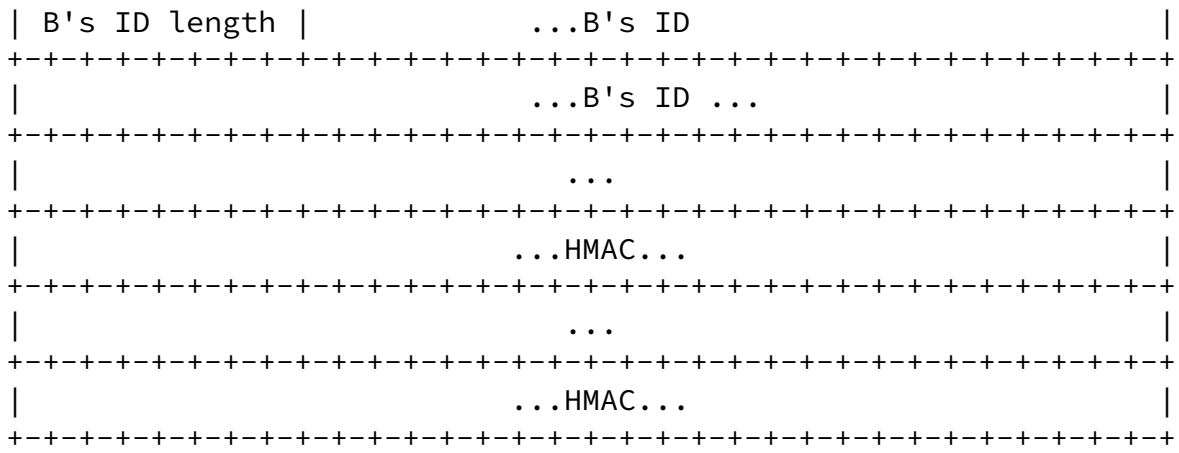


Figure 4 : EAP MAKE1 Response Packet Format

Code : 2 (Response)

Identifier : ID1 (The identifier field MUST match the Identifier field from the corresponding request, i.e. ID1)

Length : length in bytes of
Code
Identifier
Length
Type
Subtype
LID length
LID
R length
R
B's ID length
B's ID
HMAC

Type : 21 (MAKE)

Subtype : 1 (MAKE1)

LID length : 4

LID : actual value of LID, A MUST check that there is an increment with regard to its stored value. If not if MUST send an EAP Failure packet.

B's ID length : length of B's ID will vary accordingly to B's ID form

B's ID : A MUST check B's certificate validity. If not valid, A MUST send an EAP Failure packet.

HMAC : A MUST check HMAC. If not valid,
A MUST send an EAP Failure packet.

3.3. MAKE2 Request Packet

The EAP MAKE2 Request packet has the following overall format:

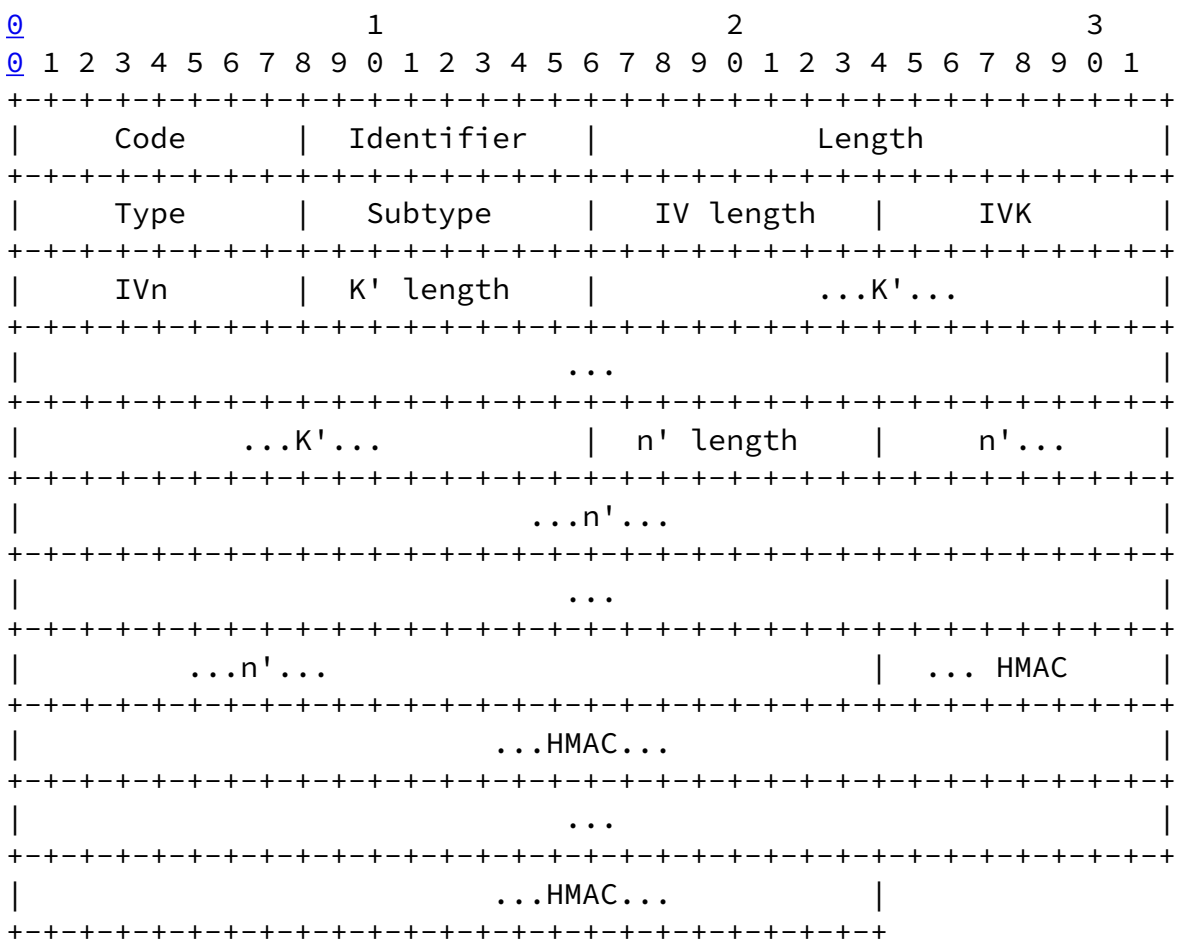


Figure 5 : EAP MAKE2 Request Packet Format

Code : 1 (Request)

Identifier : ID2 (random value)

Length : length in bytes of
Code
Identifier
Length
Type
Subtype
IV length
IVK
IVn
K' length
K'
n' length

n'
HMAC

Type : 21 (MAKE)

Subtype : 2 (MAKE2)

IV length : we here consider that the encryption of K and n is pe
using the same algorithm in the same mode of operatio
IV length gives the length of the Initialization Vect
for these encryptions

IVK : Initialization Vector used to encrypt K

IVn : Initialization Vector used to encrypt n

HMAC : B MUST check HMAC. If not valid,
B MUST send an EAP Failure packet.

3.4. MAKE2 Response Packet

The EAP MAKE2 Response packet has the following overall format:

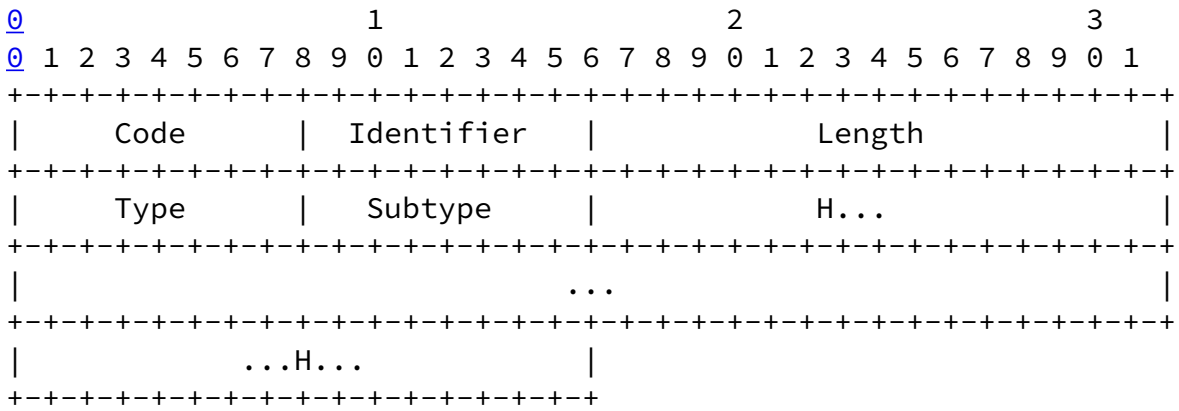


Figure 6 : EAP MAKE2 Response Packet Format

Code : 2 (Response)

Identifier : ID2 (The identifier field MUST match the
Identifier field from the corresponding
request, i.e. ID2)

Length : 25 length in bytes of
Code
Identifier
Length
Type
H

Type : 21 (MAKE)

```
          Subtype      : 2    (MAKE2)

          H            : A MUST check HMAC. If not valid,
                         A MUST send an EAP Failure packet.
```

3.5. PPP EAP MAKE and MAKE-VALIDATE Protocol Processing

Figure 7 depicts the operation of the EAP MAKE and MAKE-VALIDATE protocol. In this figure depicting PDU exchanges, the curly braces ({, }) denote items in Length-Value representation.

```
               A                              B

=> EAP Request (ID1,
                MAKE,
                MAKE1,
                {A})

                                <= EAP Response (ID1,
                                                 MAKE,
                                                 MAKE1,
                                                 {LID, R, B, HMAC})

=> EAP Request (ID2,
                MAKE,
                MAKE2,
                {IVK, IVn, K', n', HMAC})

                                <= EAP Response (ID2,
                                                 MAKE,
                                                 MAKE2
                                                 {H(n)})

=> EAP Success Packet(ID3)
```

Figure 7 : PPP EAP MAKE and MAKE-VALIDATE Protocol processing

4. Security Considerations

When the verifier A is a server from which the prover B is the client, A has some advantages to secure its database in confidentiality too, allowing the storage of pre-computed KDH. Doing so, some Denial of Service attacks should be more difficult to achieve.
Note also that besides its anti-replay role, the counter LID may avoid to the verifier some extras computations against a malevolent prover.

It should be noted that the HMAC computation is performed over data in plaintext as well as in encrypted format (see [3] for a treatment of

this subject).

Finally, please note that most prover's computations might be
carried out off-line. This is especially true when the verifier is known.

References:

[1] " PPP Extensible Authentication Protocol (EAP) "
L. Blunk, J.Vollbrecht, RFC 2284, March 1998

[2] " HMAC : Keyed-Hashing for Message Authentication "
H. Krawczyk, M. Bellare, R. Canetti, RFC 2401, February 1997

[3] " Authenticated encryption : Relations among notions
and analysis of the generic composition paradigm " (full version)
M. Bellare, C. Mamprempre, September 2000

[4] "Internet X.509 Public Key Infrastructure Certificate and CRL Profile",
R. Housley, W. Ford, W. Polk, D. Solo, RFC 2459, January 1999

[5] "FIPS PUB 180-1: Secure Hash Standard"
NIST, April 1995

Acknowledgments:

Authors wish to express their gratitude to W. Nace, P. Yee, J. Zmuda
for their stimulating work " PPP EAP KEA Public Key Authentication
Protocol " , which appears as an Internet Draft in November 1997.
This memo shares more than a simple resemblance with their work.

They also want to thank warmly S. Tramoni for her fruitful
contributions to the subject.

Author's Address:

Romain Berrendonner
SAGEM SA Centre de recherches d'Eragny
Avenue du Gros-Chene
F-95610 ERAGNY
Romain.Berrendonner@SAGEM.com

Herve Chabanne
SAGEM SA Centre de recherches d'Eragny
Avenue du Gros-Chene
F-95610 ERAGNY
Herve.Chabanne@SAGEM.com