An EAP Authentication Method Based on the EKE Protocol
draft-sheffer-emu-eap-eke-00

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 2, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

The Extensible Authentication Protocol (EAP) describes a framework that allows the use of multiple authentication mechanisms. This document defines an authentication mechanism for EAP called EAP-EKE, based on the Encrypted Key Exchange (EKE) protocol. This method provides mutual authentication through the use of a short, easy to remember password.

Table of Contents

1. Introduction .................................................. 3
2. Terminology ................................................... 3
3. Protocol ....................................................... 3
   3.1. Protocol Overview ........................................ 3
   3.2. Message Flows ........................................... 4
4. Packet Formats ................................................ 5
   4.1. EAP-EKE Header ............................................ 6
   4.2. EAP-EKE Payloads .......................................... 6
   4.3. EAP-EKE-ID ............................................... 6
   4.4. EAP-EKE-Commit .......................................... 8
   4.5. EAP-EKE-Confirm ......................................... 9
   4.6. EAP-EKE-Failure and EAP-EKE-Protected-Failure ........... 9
5. Cryptographic Operations ...................................... 11
   5.1. Generating Keying Material ............................... 11
   5.2. EAP-EKE-Commit/Request .................................. 12
   5.3. EAP-EKE-Commit/Response ................................ 12
   5.4. EAP-EKE-Confirm/Request ................................ 13
   5.5. EAP-EKE-Confirm/Response ................................ 14
   5.6. MSK and EMSK ............................................. 14
   5.7. Mandatory Algorithms .................................... 14
6. IANA Considerations ........................................... 14
7. Security Considerations ....................................... 16
8. Acknowledgements ............................................. 19
9. References .................................................... 19
   9.1. Normative References .................................... 19
   9.2. Informative References .................................. 20
Appendix A. Change Log ......................................... 21
   A.1. -00 ..................................................... 21
Appendix B. Design Options ...................................... 21
   B.1. Number of Round Trips ................................... 21
   B.2. Fragmentation ............................................. 21
Authors’ Addresses ............................................. 22
1. Introduction

The predominant access method for the Internet today is that of a human using a username and password to authenticate to a computer enforcing access control. Proof of knowledge of the password authenticates the human to the computer.

Typically, these passwords are not stored on a user’s computer for security reasons and must be entered each time the human desires network access. Therefore, the passwords must be ones that can be repeatedly entered by a human with a low probability of error. They will likely not possess high entropy and it may be assumed that an adversary with access to a dictionary will have the ability to guess a user’s password. It is therefore desirable to have a robust authentication method that is secure even when used with a weak password in the presence of a strong adversary.

EAP-EKE is an EAP method [RFC3748] that addresses the problem of password-based authenticated key exchange, using a possibly weak password for authentication and to derive an authenticated and cryptographically strong shared secret. This problem was first described by Bellovin and Merritt in [BM92] and [BM93]. Subsequently, a number of other solution approaches have been proposed, for example [JAB96], [LUC97], [BMP00], and others.

This proposal is based on the original Encrypted Key Exchange (EKE) proposal, as described in [BM92]. None of the subsequent improvements have been incorporated. However, we have used only the subset of [BM92] (namely the variant described in Section 3.1) which has withstood the test of time and is believed secure as of this writing.

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].

3. Protocol

3.1. Protocol Overview

EAP is a two-party protocol spoken between an EAP peer and an EAP server (also known as "authenticator"). An EAP method defines the specific authentication protocol being used by EAP. This memo defines a particular method and therefore defines the messages sent
between the EAP server and the EAP peer for the purpose of authentication and key derivation.

3.2. Message Flows

EAP-EKE defines three message exchanges: an Identity exchange, a Commit exchange and a Confirm exchange. A successful authentication is shown in Figure 1.

The peer and server use the EAP-EKE Identity exchange to learn each other’s identities and to agree upon a ciphersuite to use in the subsequent exchanges. In the Commit exchange the peer and server exchange information to generate a shared key and also to bind each other to a particular guess of the password. In the Confirm exchange the peer and server prove liveness and knowledge of the password by generating and verifying verification data.

Figure 1: A Successful EAP-EKE Exchange

Schematically, the original exchange as described in [BM92] (and with the roles reversed) is:
The current protocol extends the basic cryptographic protocol, and the regular successful exchange becomes:

EAP-EKE-ID/Request: S -> P
   ID_S, CryptoProposals
EAP-EKE-ID/Response: S <- P
   ID_P, CryptoSelection
EAP-EKE-Commit/Request: S -> P
   E(Password, Y_S))
EAP-EKE-Commit/Response: S <- P
   E(Password, Y_P), E(SharedSecret, Nonce_P)
EAP-EKE-Confirm/Request: S -> P
   E(SharedSecret, Nonce_S | Nonce_P), Auth_S
EAP-EKE-Confirm/Response: S <- P
   E(SharedSecret, Nonce_S), Auth_P

As shown in the exchange above, the following information elements are added to the original protocol: identity values for both protocol parties, negotiation of cryptographic protocols, and signature fields to protect the integrity of the negotiated parameters.

4. Packet Formats
4.1. EAP-EKE Header

The EAP-EKE header consists of the standard EAP header (see Section 4 of [RFC3748]), followed an EAP-EKE exchange type. The header has the following structure:

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   EKE-Exch    |              Data            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: EAP-EKE Header

The Code, Identifier, Length, and Type fields are all part of the EAP header, and defined in [RFC3748]. The Type field in the EAP header MUST be the value allocated by IANA for EAP-EKE version 1.

The EKE-Exch field identifies the type of EAP-EKE payload encapsulated in the Data field. This document defines the following values for the EKE-Exch field:

- 0x00: Reserved
- 0x01: EAP-EKE-ID exchange
- 0x02: EAP-EKE-Commit exchange
- 0x03: EAP-EKE-Confirm exchange
- 0x04: EAP-EKE-Failure exchange
- 0x05: EAP-EKE-Protected-Failure exchange

Further values of this EKE-Exch field are available via IANA registration.

4.2. EAP-EKE Payloads

EAP-EKE payloads all contain the EAP-EKE header and encoded information, which differs for the different exchanges.

4.3. EAP-EKE-ID
The EAP-EKE-ID payload contains the following fields:

ProposalsNr:

The ProposalNr field contains the number of proposals subsequently contained in the Proposal field. In the EAP-EKE-ID/Request the ProposalNr field MUST NOT be set to zero (0) and in the EAP-EKE-ID/Response message the ProposalNr field MUST be set to one (1). The offered proposals in the Request are listed contiguously in priority order, most preferable first. The selected proposal in the Response MUST be fully identical with one of the offered proposals.

Proposal:

Each proposal consists of four one-octet fields, in this order:

Group Description:

This field’s value is taken from the IANA registry for Diffie-Hellman groups defined in Section 6.4.

Encryption:

This field’s value is taken from the IANA registry for encryption algorithms defined in Section 6.1.

PRF:

This field’s value is taken from the IANA registry for pseudo random functions defined in Section 6.2.
MAC:

This field’s value is taken from the IANA registry for keyed message digest algorithms defined in Section 6.3 used to provide integrity protection.

IDType

Denotes the Identity type. This is taken from the IANA registry defined in Section 6. The server and the peer MAY use different identity types.

The Identity field is always printable, and its meaning depends on the values of the Code and IDType fields.

- EAP-EKE-ID/Request: server ID
- EAP-EKE-ID/Response: peer ID

The server SHOULD assert its own identity (e.g. its host name), or it MAY use the peer’s identity if it knows it before the protocol starts.

The length of the Identity is computed from the Length field in the EAP header.

4.4. EAP-EKE-Commit

In this exchange both parties send their encrypted ephemeral public key, and the peer also includes a Challenge.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         DHComponent                                           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Commit_P                                            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: EAP-EKE-Commit Payload
DHComponent:

This field contains the password-encrypted Diffie-Hellman public key, see Section 5.2.

Commit_P:

This field only appears in the response, and contains the encrypted challenge value sent by the peer. See Section 5.3.

4.5. EAP-EKE-Confirm

In this exchange both parties complete the authentication by generating a shared temporary key, authenticating the entire protocol, and generating key material for the EAP consumer protocol.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Confirm_?                                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ~                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ~                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Auth_?                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ~                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ~                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: EAP-EKE-Confirm Payload

Confirm_?:

This field contains the encrypted response to the other peer’s challenge, see Section 5.4 and Section 5.5.

Auth_?

This field signs the Identity and the negotiated fields, to prevent downgrade attacks. See Section 5.4 and Section 5.5.

4.6. EAP-EKE-Failure and EAP-EKE-Protected-Failure

The EAP-EKE-Failure message format is defined as follows:
EAP-EKE-Failure Payload

The EAP-EKE-Protected-Failure payload format is defined as follows:

The MAC field contains the keyed message digest computed with the MAC algorithm selected during the initial exchange computed over the Failure-Code using the MAC key (see Section 5 on how this key is derived). A protected failure response can only be returned once the MAC key has been derived.

Currently the following Failure-Code values are defined:
- 0x00000000: Reserved
- 0x00000001: No Error
- 0x00000002: Protocol Error
- 0x00000003: Password Not Found
- 0x00000004: Authentication Failure
- 0x00000005: Authorization Failure
- 0x00000006: No Proposal Chosen

Additional values of this field are available via IANA registration.

"No Error" is used for failure acknowledgement, see below. "Protocol Error" indicates a failure to parse or understand a protocol message or one of its payloads. "Password Not Found" indicates a password for a particular user could not be located, making authentication impossible. "Authentication Failure" indicates failure in the cryptographic computation most likely caused by an incorrect
password, or an inappropriate identity type. "Authorization Failure" indicates that while the password being used is correct, the user is not authorized to connect. "No Proposal Chosen" indicates that the peer is unwilling to select any of the cryptographic proposals offered by the server.

When the peer encounters an error situation, it MUST respond with either EAP-EKE-Failure or EAP-EKE-Protected-Failure, depending on whether it believes a common MAC key has been agreed upon. The server MUST send an EAP-Failure message to end the exchange.

When the server encounters an error situation, it MUST respond with either EAP-EKE-Failure or EAP-EKE-Protected-Failure, depending on whether it believes a common MAC key has been agreed upon. The peer MUST send back either EAP-EKE-Failure or EAP-EKE-Protected-Failure (corresponding to the server’s selection), containing a "No Error" failure code. Then the server MUST send an EAP-Failure message to end the exchange.

5. Cryptographic Operations

5.1. Generating Keying Material

Keying material will always be derived as the output of the negotiated prf algorithm. Since the amount of keying material needed may be greater than the size of the output of the prf algorithm, we will use the prf iteratively. We will use the terminology prf+ to describe the function that outputs a pseudo-random stream based on the inputs to a prf as follows: (where | indicates concatenation)

prf+ (K,S) = T1 | T2 | T3 | T4 | ...

where:

T1 = prf (K, S | 0x01)
T2 = prf (K, T1 | S | 0x02)
T3 = prf (K, T2 | S | 0x03)
T4 = prf (K, T3 | S | 0x04)

continuing as needed to compute all required keys. The keys are taken from the output string without regard to boundaries (e.g., if the required keys are a 256-bit Advanced Encryption Standard (AES) key and a 160-bit HMAC key, and the prf function generates 160 bits, the AES key will come from T1 and the beginning of T2, while the HMAC key will come from the rest of T2 and the beginning of T3).

The constant concatenated to the end of each string feeding the prf is a single octet. prf+ in this document is not defined beyond 255
times the size of the prf output.

5.2. EAP-EKE-Commit/Request

The server computes

\[ DHValue_S = g^x \mod N, \]

where ‘x’ is a randomly chosen number in the range 2 .. N-1. The randomly chosen number is the private key, and the calculated field is the corresponding public key. The calculated value MUST NOT be zero modulo N. If the peer receives a bad value for this field, it MUST take action to disconnect or disable the link. Each of the peers MUST use a fresh, random value for this field on each run of the protocol.

The server transmits

\[ DHComponent_S = E(prf+(password, "EAP-EKE Password"), DHValue_S), \]

encrypted using the algorithm negotiated during the previous exchange. If required by the encryption algorithm/mode, the encrypted field is preceded by an Initialization Vector (IV), whose length depends on the algorithm.

In many cases (e.g. CBC mode) it may be necessary to pad DHValue_S on the right, to fit the encryption algorithm’s block size. In such cases, random padding MUST be used, and this randomness is critical to the security of the protocol. Randomness recommendations can be found in [RFC4086]. When decrypting this field, the real length of DHValue_S is determined according to the negotiated Diffie Hellman group.

If the password needs to be stored on the server, it is RECOMMENDED to store the randomized password value, i.e. prf+(password, ...), as a password-equivalent, rather than the cleartext password.

5.3. EAP-EKE-Commit/Response

The peer computes

\[ DHValue_P = g^y \mod N \]

and sends

\[ DHComponent_P = E(prf+(password, "EAP-EKE Password"), DHValue_P) \]

If the password is non-ASCII, it SHOULD be normalized by the peer.
before the EAP-EKE message is constructed. The normalization method is SASLprep, [RFC4013]. Note that the password is not null-terminated.

Both sides calculate

$$DHShared = g^{x*y} \mod N$$

The encryption key is computed:

$$Ke = prf+(DHShared, "EAP-EKE Ke" | ID_S | ID_P)$$

The MAC key is computed:

$$MAC = prf+(DHShared, "EAP-EKE MAC" | ID_S | ID_P)$$

And the peer generates

$$Challenge_P = E(Ke, Nonce_P),$$

where Nonce_P is a randomly generated binary string. Nonce_P has length equal to the block size of E for block ciphers, or 32 octets if E is a stream cipher.

5.4. EAP-EKE-Confirm/Request

The server sends:

$$Commit_S = E(Ke, Nonce_P | Nonce_S),$$

where Nonce_S is a randomly generated string, similar to Nonce_P.

It computes:

$$Ka = prf+(DHShared, "EAP-EKE Ka" | ID_S | ID_P | Nonce_P | Nonce_S)$$

And sends:

$$Auth_S = prf(Ka, "EAP-EKE server" | EAP-EKE-ID/Request | EAP-EKE-ID/Response | EAP-EKE-Commit_S | EAP-EKE-Commit_P),$$

where the literal string is encoded using ASCII with no zero terminator. The messages are included in full, starting with the EAP header, and including any possible future extensions.
5.5. EAP-EKE-Confirm/Response

The peer computes $K_a$, and sends:

$Commit_P = E(Ke, \text{Nonce}_S)$
$Auth_P = prf(Ka, \text{"EAP-EKE peer" | EAP-EKE-ID/Request | EAP-EKE-ID/Response | EAP-EKE-Commit_S | EAP-EKE-Commit_P})$

5.6. MSK and EMSK

Following the last message of the protocol, both sides compute and export the shared keys, each 512 bits in length:

$MSK = prf+(DHShared, \text{"EAP-EKE MSK" | ID_S | ID_P | Nonce_P | Nonce_S})$
$EMSK = prf+(DHShared, \text{"EAP-EKE EMSK" | ID_S | ID_P | Nonce_P | Nonce_S})$

5.7. Mandatory Algorithms

To facilitate interoperability, the following algorithms are mandatory to implement:

- ENCR_AES128_CBC (encryption algorithm)
- PRF_HMAC_SHA1 (pseudo random function and keyed message digest)
- DHGROUP_14 (DH-group)

6. IANA Considerations

This document allocates an EAP method type, for "EAP-EKE Version 1".

This document requires IANA to create the registries described in the subsequent sections. Values can be added or modified in these registries per Specification Required [RFC5226].

6.1. Encryption Algorithm Registry

This section defines an IANA registry for encryption algorithms:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ENCR_AES128_CBC</td>
<td>1</td>
<td>AES with a 128-bit key, CBC mode</td>
</tr>
<tr>
<td></td>
<td>2-127</td>
<td>Available for allocation via IANA</td>
</tr>
<tr>
<td></td>
<td>128-255</td>
<td>Reserved for private use</td>
</tr>
</tbody>
</table>
### 6.2. Pseudo Random Function Registry

This section defines an IANA registry for pseudo random function algorithms:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PRF_HMAC_SHA1</td>
<td>1</td>
<td>HMAC SHA-1, as defined in [RFC2104]</td>
</tr>
<tr>
<td></td>
<td>2-127</td>
<td>Available for allocation via IANA</td>
</tr>
<tr>
<td></td>
<td>128-255</td>
<td>Reserved for private use</td>
</tr>
</tbody>
</table>

### 6.3. Keyed Message Digest Registry

This section defines an IANA registry for keyed message digest algorithms:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PRF_HMAC_SHA1</td>
<td>1</td>
<td>HMAC SHA-1, as defined in [RFC2104]</td>
</tr>
<tr>
<td></td>
<td>2-127</td>
<td>Available for allocation via IANA</td>
</tr>
<tr>
<td></td>
<td>128-255</td>
<td>Reserved for private use</td>
</tr>
</tbody>
</table>

### 6.4. Diffie-Hellman Group Registry

This section defines an IANA registry for Diffie-Hellman groups:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DPKGROUP_14</td>
<td>1</td>
<td>2048-bit MODP Group (#14), as defined in [RFC3526]</td>
</tr>
<tr>
<td></td>
<td>2-127</td>
<td>Available for allocation via IANA</td>
</tr>
<tr>
<td></td>
<td>128-255</td>
<td>Reserved for private use</td>
</tr>
</tbody>
</table>
6.5. Identity Type Registry

In addition, an identity type registry is defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ID_OPAQUE</td>
<td>1</td>
<td>A printable string whose format is undefined</td>
</tr>
<tr>
<td>ID_NAI</td>
<td>2</td>
<td>A Network Access Identifier, as defined in [RFC4282]</td>
</tr>
<tr>
<td>ID_IPV4</td>
<td>3</td>
<td>An IPv4 address, in binary format</td>
</tr>
<tr>
<td>ID_IPV6</td>
<td>4</td>
<td>An IPv6 address, in binary format</td>
</tr>
<tr>
<td>ID_FQDN</td>
<td>5</td>
<td>A fully qualified domain name (mandatory to implement)</td>
</tr>
<tr>
<td>6-127</td>
<td></td>
<td>Available for allocation via IANA</td>
</tr>
<tr>
<td>128-255</td>
<td></td>
<td>Reserved for private use</td>
</tr>
</tbody>
</table>

6.6. Failure-Code Registry

This section defines an IANA registry for the Failure-Code registry, a 32-bit long code. Initial values are defined in Section 4.6. All values up to 0xffffffff are available for allocation via IANA. The remaining values up to 0xffffffff are available for private use.

7. Security Considerations

Any protocol that claims to solve the problem of password-authenticated key exchange must be resistant to active, passive and dictionary attack and have the quality of forward secrecy. These characteristics are discussed further in the following paragraphs.

Resistance to Passive Attack A passive attacker is one that merely relays messages back and forth between the peer and server, faithfully, and without modification. The contents of the messages are available for inspection, but that is all. To achieve resistance to passive attack, such an attacker must not be able to obtain any information about the password or anything about the resulting shared secret from watching repeated runs of the protocol. Even if a passive attacker is able to learn the
password, she will not be able to determine any information about
the resulting secret shared by the peer and server.

Resistance to Active Attack An active attacker is able to modify,
add, delete, and replay messages sent between protocol
participants. For this protocol to be resistant to active attack,
the attacker must not be able to obtain any information about the
password or the shared secret by using any of its capabilities.
In addition, the attacker must not be able to fool a protocol
participant into thinking that the protocol completed
successfully. It is always possible for an active attacker to
deny delivery of a message critical in completing the exchange.
This is no different than dropping all messages and is not an
attack against the protocol.

Resistance to Dictionary Attack For this protocol to be resistant to
dictionary attack any advantage an adversary can gain must be
directly related to the number of interactions she makes with an
honest protocol participant and not through computation. The
adversary will not be able to obtain any information about the
password except whether a single guess from a single protocol run
is correct or incorrect.

Forward Secrecy Compromise of the password must not provide any
information about the secrets generated by earlier runs of the
protocol.

[RFC3748] requires that documents describing new EAP methods clearly
articulate the security properties of the method. In addition, for
use with wireless LANs, [RFC4017] mandates and recommends several of
these. The claims are:
1. Mechanism: password.
2. Claims:
   * Mutual authentication: the peer and server both authenticate
each other by proving possession of a shared password. This
is REQUIRED by [RFC4017].
   * Forward secrecy: compromise of the password does not reveal
the secret keys (MSK and EMSK) from earlier runs of the
protocol.
   * Replay protection: an attacker is unable to replay messages
from a previous exchange either to learn the password or a key
derived by the exchange. Similarly the attacker is unable to
induce either the peer or server to believe the exchange has
successfully completed when it hasn’t.
   * Key derivation: a shared secret is derived by performing a
group operation in a finite cyclic group (e.g. exponentiation)
using secret data contributed by both the peer and server. An
MSK and EMSK are derived from that shared secret. This is
REQUIRED by [RFC4017].
* Dictionary attack resistance: an attacker can only make one password guess per active attack. The advantage she can gain is through interaction not through computation. This is REQUIRED by [RFC4017].

* Session independence: this protocol is resistant to active and passive attacks and does not enable compromise of subsequent or prior MSKs or EMSKs from either passive or active attacks.

* Denial of Service resistance: it is possible for an attacker to cause a server to allocate state and consume CPU. Such an attack is gated, though, by the requirement that the attacker first obtain connectivity through a lower-layer protocol (e.g. 802.11 authentication followed by 802.11 association, or 802.3 "link-up") and respond to two EAP messages--the EAP-ID/Request and the EAP-EKE-ID/Request.

* Man-in-the-Middle Attack resistance: this exchange is resistant to active attack, which is a requirement for launching a man-in-the-middle attack. This is REQUIRED by [RFC4017].

* Shared state equivalence: upon completion of EAP-EKE the peer and server both agree on MSK, EMSK. The peer has authenticated the server based on the Server_ID and the server has authenticated the peer based on the Peer_ID. This is due to the fact that Peer_ID, Server_ID, and the generated shared secret are all combined to make the authentication element which must be shared between the peer and server for the exchange to complete. This is REQUIRED by [RFC4017].

* Fragmentation: this protocol does not define a technique for fragmentation and reassembly.

* Resistance to "Denning-Sacco" attack: learning keys distributed from an earlier run of the protocol, such as the MSK or EMSK, will not help an adversary learn the password.

3. Key strength: the strength of the resulting key depends on the finite cyclic group chosen. For example, [RFC5114] defines new groups available for use with this protocol. Using groups from [RFC5114] the strength can vary from 80 bits (for the 1024-bit MODP with 160-bit Prime Subgroup) to 256 bits (for the 521-bit Random ECP Group). Other groups can be defined and the strength of those groups depends on their definition. This is REQUIRED by [RFC4017].

4. Key hierarchy: MSKs and EMSKs are derived from the secret values generated during the protocol run, using a negotiated pseudo-random function.

5. Vulnerabilities (note that none of these are REQUIRED by [RFC4017]):

* Protected ciphersuite negotiation: the ciphersuite proposal made by the server is not protected from tampering by an active attacker. However if a proposal was modified by an active attacker it would result in a failure to confirm the
message sent by the other party, since the proposal is bound by each side into its Confirm message, and the protocol would fail as a result.

* Confidentiality: none of the messages sent in this protocol are encrypted.
* Integrity protection: all messages in this protocol are integrity protected.
* Channel binding: this protocol does not enable the exchange of integrity-protected channel information that can be compared with values communicated via out-of-band mechanisms.
* Fast reconnect: this protocol does not provide a fast reconnect capability.
* Cryptographic binding: this protocol is not a tunneled EAP method and therefore has no cryptographic information to bind.
* Identity protection: the EAP-EKE-ID exchange is not protected. An attacker will see the server’s identity in the EAP-EKE-ID/Request and see the peer’s identity in EAP-EKE-ID/Response.

8. Acknowledgements

Much of this document was unashamedly picked from [I-D.harkins-emu-eap-pwd] and [I-D.ietf-pppext-eap-srp-03], and we would like to acknowledge the authors of these documents: Dan Harkins, Glen Zorn, James Carlson, Bernard Aboba and Henry Haverinen.

9. References

9.1. Normative References


9.2. Informative References


Appendix A.  Change Log

A.1.  -00

Initial version.

Appendix B.  Design Options

B.1.  Number of Round Trips

We have looked at three options: 2 round trips, 3 round trips, and a normal run of 2 round trips with an optional third. Some of the decision factors include:

- Performance (latency).
- Crypto-agility, the ability to negotiate cryptographic algorithms. Ideally this applies to both the symmetric and asymmetric algorithms.
- Complexity of the protocol state machine, when some messages are optional.
- Dependence on a higher-level protocol sending the peer’s identity before EAP-EKE starts, so that the correct password can be used.

The initial version of this protocol has 3 round trips, primarily for simplicity.

B.2.  Fragmentation

While similar documents ([I-D.harkins-emu-eap-pwd]) provide fragmentation support at the level of the EAP method, we have decided that such support is unnecessary given the expected size of messages in EAP-EKE.
Authors’ Addresses

Yaron Sheffer
Check Point Software Technologies Ltd.
5 Hasolelim St.
Tel Aviv  67897
Israel

Email: yaronf@checkpoint.com

Glen Zorn
Network Zen
1310 East Thomas Street
#306
Seattle, Washington  98102
USA

Phone: +1 (206) 377-9035
Email: gwz@net-zen.net

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo  02600
Finland

Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI:  http://www.tschofenig.priv.at