              Password Authenticated Connection Establishment
          with the Internet Key Exchange Protocol version 2 (IKEv2)

Abstract

   The Internet Key Exchange protocol version 2 (IKEv2) does not allow
   secure peer authentication when using short credential strings, i.e.,
   passwords. Several proposals have been made to integrate password-
   authentication protocols into IKE. This document provides an
   adaptation of Password Authenticated Connection Establishment (PACE)
   to the setting of IKEv2 and demonstrates the advantages of this
   integration.

Status of This Memo

   This document is not an Internet Standards Track specification; it is
   published for examination, experimental implementation, and
   evaluation.

   This document defines an Experimental Protocol for the Internet
   community. This document is a product of the Internet Engineering
   Task Force (IETF). It represents the consensus of the IETF
   community. It has received public review and has been approved for
   publication by the Internet Engineering Steering Group (IESG). Not
   all documents approved by the IESG are a candidate for any level of
   Internet Standard; see Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6631.

Copyright Notice

Table of Contents

1. Introduction

   PACE [TR03110] is a security protocol that establishes a mutually
   authenticated (and encrypted) channel between two parties based on
   weak (short) passwords. PACE provides strong session keys that are
   independent of the strength of the password. PACE belongs to a
   family of protocols often referred to as Zero-Knowledge Password
   Proof (ZKPP) protocols, all of which amplify weak passwords into
   strong session keys. This document describes the integration of PACE
   into IKEv2 [RFC5996] as a new authentication mode, analogous to the
   existing certificate and Pre-Shared Key (PSK) authentication modes.

   Some of the advantages of our approach, compared to the existing
   IKEv2, include the following:

   o The current best practice to implement password authentication in
      IKE involves certificate-based authentication of the server plus
      some Extensible Authentication Protocol (EAP) method to
      authenticate the client. This involves two non-trivial
      infrastructure components (PKI and EAP/AAA). Moreover,
      certificate authentication is hard to get right and often depends
      on unreliable user behavior for its security.

   o Alternatively, native IKEv2 shared secret authentication can be
      used with passwords. However, this usage is insecure;
      specifically, it is vulnerable to active attackers.

     o Some newer EAP methods can be used for mutual authentication and,
        combined with [RFC5998], can be well integrated into IKEv2. This
        is certainly an option in some cases, but the current proposal may
        be simpler to implement.

   Compared to other protocols aiming at similar goals, PACE has several
   advantages. PACE was designed to allow for a high level of
   flexibility with respect to cryptographic algorithms; e.g., it can be
   implemented based on Modular Diffie-Hellman as well as Elliptic Curve
   Diffie-Hellman without any restrictions on the mathematical group to
   be used, other than the requirement that the group be
   cryptographically secure. The protocol itself is also proven to be
   cryptographically secure [PACEsec]. The PACE protocol is currently
   used in an international standard for digital travel documents
   [ICAO].

   The integration aims at keeping IKEv2 unchanged as much as possible;
   e.g., the mechanisms used to establish Child security associations
   (SAs) as provided by IKEv2 would be maintained with no change.

   The Password-Authenticated Key Exchange (PAKE) framework document
   [RFC6467] defines a set of payloads for different types of PAKE
   methods within IKEv2. This document reuses this framework. Note
   that the current document is self-contained; i.e., all relevant
   payloads and semantics are redefined here.

1.1. 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].

   The following notation is used in this document:

       E()   Symmetric encryption
       D()   Symmetric decryption
       KA()   Key agreement
       Map()   Mapping function
       Pwd   Shared password
       SPwd   Stored password
       KPwd   Symmetric key derived from a password Pwd
       G   Static group generator
       GE   Ephemeral group generator
       ENONCE   Encrypted nonce
       PKEi   Ephemeral public key of the initiator
       SKEi   Ephemeral secret key of the initiator

         PKEr   Ephemeral public key of the responder
         SKEr   Ephemeral secret key of the responder
         AUTH   Authentication payload

   Any other notation used here is defined in [RFC5996].

2. Overview

   At a high level, the following steps are performed by the initiator
   and the responder. They result in a two-round IKE_AUTH exchange,
   described in Section 3 below.

   1. The initiator randomly and uniformly chooses a nonce s, encrypts
       the nonce using the password, and sends the ciphertext

           ENONCE = E(KPwd, s)

       to the responder. The responder recovers the plaintext nonce s
       with the help of the shared password Pwd.

   2. The nonce s is mapped to an ephemeral generator

           GE = G^s * SASharedSecret,

       where G is the generator of the selected Modular Exponential
       (MODP) group and SASharedSecret is a shared secret that has been
       generated in the IKE_SA_INIT step.

   3. Both the initiator and the responder each calculate an ephemeral
       key pair

           (SKEi, PKEi = GE^SKEi) and (SKEr, PKEr=GE^SKEr),

       respectively, based on the ephemeral generator GE, and exchange
       the public keys.

   4. Finally, they compute the shared secret

           PACESharedSecret = PKEi^SKEr = PKEr^SKEi

       and generate, exchange, and verify the IKE authentication token
       AUTH using the shared secret PACESharedSecret.

   The encryption function E() must be carefully chosen to prevent
   dictionary attacks that would otherwise allow an attacker to recover
   the password. Those restrictions are described in Section 4.1.
   Details on the mapping function, including the elliptic curve
   variant, can be found in Section 4.2.

To avoid the risks inherent in storing a short password (e.g., the
fact that passwords are often reused for different applications),
this protocol allows the peers to jointly convert the password into a
cryptographically stronger shared secret. This shared secret can
then be stored by both peers, in lieu of the original password or its
salted variants.

3. Protocol Sequence

The protocol consists of three round trips -- an IKE_SA_INIT exchange
and a 2-round IKE_AUTH exchange -- as shown in the next figure. An
optional Informational exchange may follow (see Section 3.5).

```
   Initiator                        Responder
   ---------                        ---------

   IKE_SA_INIT:

   HDR, SAi1, KEi, Ni, N(SECURE_PASSWORD_METHODS)  ->

                 <- HDR, SAr1, KEr, Nr, N(SECURE_PASSWORD_METHODS)

   IKE_AUTH round #1:

   HDR, SK{IDi, [IDr,], SAi2,
         TSi, TSr, GSPM(ENONCE), KEi2} ->

                                        <- HDR, SK{IDr, KEr2}

   IKE_AUTH round #2:

   HDR, SK{AUTH [, N(PSK_PERSIST)] } ->

                 <- HDR, SK{AUTH, SAr2, TSi, TSr [, N(PSK_PERSIST)] }

                    Figure 1: IKE SA Setup with PACE
```

3.1. The IKE_SA_INIT Exchange

The initiator sends a SECURE_PASSWORD_METHODS notification that
indicates its support of this extension and its wish to authenticate
using a password. The following text assumes that the responder sent
back a SECURE_PASSWORD_METHODS notification that indicates its
preference for PACE.

If PACE was chosen, the algorithms negotiated in SAi1 and SAr1 are
also used for the execution of PACE, i.e., the key agreement protocol
(Modular Diffie-Hellman or Elliptic Curve Diffie-Hellman), the group
to be used, and the encryption algorithm.

3.2. The IKE_AUTH Exchange, Round #1

This is the first part of the PACE authentication of the peers. This
exchange MUST NOT be used unless both peers indicated support of this
protocol.

The initiator selects a random nonce s and encrypts it to form ENONCE
using the password Pwd, as described in Section 4.1. Then, the
initiator maps the nonce to an ephemeral generator GE of the group as
described in Section 4.2, chooses randomly and uniformly an ephemeral
key pair (SKEi,PKEi) based on the ephemeral generator, and finally
generates the payloads GSPM(ENONCE) containing the encrypted nonce
and KEi2 containing the ephemeral public key.

The responder decrypts the received encrypted nonce s = D(KPwd,
ENONCE), performs the mapping, and randomly and uniformly chooses an
ephemeral key pair (SKEr,PKEr) based on the ephemeral generator GE.
The responder generates the KEr2 payload containing the ephemeral
public key.

The request is equivalent to the IKE_AUTH request in a normal IKEv2
exchange; i.e., any payload that is valid in an IKE_AUTH request is
valid (with the same semantics) in this round's request. In
particular, certificate-related payloads are allowed, even though
their use may not be practical within this mode.

3.3. The IKE_AUTH Exchange, Round #2

This is the second part of the PACE authentication of the peers.

The initiator and the responder calculate the shared secret
PACESharedSecret

    PACESharedSecret = KA(SKEi, PKEr, GE) = KA(SKEr, PKEi, GE),

where KA denotes the Diffie-Hellman key agreement, e.g., (for MODP
groups), modular exponentiation. Then, they calculate the
authentication tokens AUTHi and AUTHr.

The initiator calculates

    AUTHi = prf(prf+(Ni | Nr, PACESharedSecret),
    <InitiatorSignedOctets> | PKEr)

See Section 2.15 of [RFC5996] for the definition of signed octets.

The responder calculates

    AUTHr = prf(prf+(Ni | Nr, PACESharedSecret),
    <ResponderSignedOctets> | PKEi)

Both AUTH payloads MUST indicate as their authentication method the
Generic Secure Password Authentication Method [RFC6467], whose value
is 12. The authentication tokens are exchanged, and each of them
MUST be verified by the other party. The behavior when this
verification fails is unchanged from [RFC5996].

Each of the peers MAY generate a long-term credential at this point,
after it has verified the opposite peer's identity. The shared
secret is

    LongTermSecret = prf(Ni | Nr, "PACE Generated PSK" |
    PACESharedSecret),

where the literal string is ASCII-encoded, with no zero terminator.
The generated secret MUST be persisted to stable memory before
sending the response. See Section 3.5 for more details about this
facility.

This round's response is equivalent to the IKE_AUTH response in a
normal IKEv2 exchange; i.e., any payload that is valid in an IKE_AUTH
response is valid (with the same semantics) in the second round's
response.

Following authentication, all temporary values MUST be deleted by the
peers, including in particular s, the ephemeral generator, the
ephemeral key pairs, and PACESharedSecret.

3.4. Public Key Validation

The security of the protocol relies on the entanglement of a weak
password with cryptographically strong shared secrets, SASharedSecret
and PACESharedSecret, mutually and randomly generated by the
initiator and the responder. If an attacker can influence the
randomness of those shared secrets, the confidentiality of the
password may be directly affected.

Implementations MUST therefore verify that the shared secrets
SASharedSecret and PACESharedSecret are random elements of the group
generated by G to prevent small subgroup attacks. This can be
achieved by a validation of the public keys (i.e., KEi, PKEi, and
KEr, PKEr).

First of all, each party MUST check that the public keys PKEi, PKEr, KEi, and KEr differ. Otherwise, it MUST abort the protocol.

For each received public key PK, the following tests SHOULD be performed. Any failure in the validation MUST be interpreted as an attack, and the protocol SHALL be aborted.

o Verify that PK is an element of the Diffie-Hellman Group.

   * For Modular Diffie-Hellman, check that PK lies within the interval [2,p-2].

   * For Elliptic Curve Diffie-Hellman, check that PK is a point on the Elliptic Curve and not the point at infinity.

o Verify that PK is an element of the cryptographic subgroup of order q.

   * For Modular Diffie-Hellman, check that $PK^q = 1 \pmod p$.

   * For Elliptic Curve Diffie-Hellman, check that q * PK = 0.

Note that for most of the MODP groups, the order $q = (p-1)/2$. This applies in particular to the standard groups #2, #5, and #14, commonly used in IKE. For ECP and MODP groups not based on safe primes, the order q is explictly stated in the parameters.

As an alternative to the public key validation, the compatible cofactor exponentiation/multiplication may be used, which is often more efficient but requires changes to the implementation of the key agreement. Details on the implementation can be found in [RFC2785] and in [TR03111] for Modular Diffie-Hellman and Elliptic Curve Diffie-Hellman, respectively.

3.5. Creating a Long-Term Shared Secret

To reduce the time that the peers store a hashed password, it is RECOMMENDED that the password be replaced by a dedicated shared secret, according to the method described in this section. See Appendix B for more discussion of the security threats involved.

Both peers generate the value LongTermSecret during round #2 of IKE_AUTH, as shown above. Later on, they exchange a PERSIST_PSK notification. Assume that both peers support this mechanism (e.g., the IKE implementation is able to modify its own credential store). Then, each of the peers, when receiving the notification, permanently

deletes the stored password and replaces it with LongTermSecret.
These credentials are stored in the Peer Authorization Database (PAD)
[RFC4301] and are associated with the identity of the opposite peer.

This solution is designed as a two-phase commitment, so that failure
at any time cannot result in the peers not having any shared secret.

```
   Initiator                          Responder
   ---------                          ---------

   IKE_AUTH round #2:

   HDR, SK{..., N(PSK_PERSIST)} ---------->
                              Responder computes and stores PSK

                    <------- HDR, SK{..., N(PSK_PERSIST)}

   Initiator computes and stores PSK

   HDR, SK{N(PSK_CONFIRM)} -------------->

                              Responder deletes the short password

                    <-------------- HDR, SK{N(PSK_CONFIRM)}

   Initiator deletes the short password
```

          Figure 2: IKE SA Setup with PACE and PSK Generation

In the second round of IKE_AUTH, the initiator MAY send a PSK_PERSIST
notification if it wishes to use this mechanism. If the responder
agrees, and only after it has authenticated the initiator, it MUST
generate a new PSK, save it to stable storage (e.g., to disk), and
MUST respond with a PSK_PERSIST notification. Otherwise, it simply
does not include the notification in its reply. When receiving the
reply, and after authenticating the responder, the initiator MUST
also generate the PSK and save it in stable storage.

If the peers have negotiated this mechanism, the initiator MUST send
the PSK_CONFIRM notification in an Informational exchange shortly
after the IKE SA has been set up. When the responder receives it, it
MUST delete the stored short password from its credential database
and respond with a PSK_CONFIRM notification. Upon receiving this
notification, the initiator deletes its copy of the short password.

If not saved to persistent storage, the LongTermSecret MUST be
deleted when the IKE SA is rekeyed or when it is torn down. It
SHOULD be deleted 1 hour after the initial IKE SA has been set up.

3.6. Using the Long-Term Shared Secret

   The LongTermSecret MUST be used as a regular IKE Pre-Shared Key
   (PSK), rather than with PACE or any other password-based
   authentication method.

   Normally, at the completion of this protocol, both peers will have
   either a shared password or a shared PSK. The protocol is designed
   so that the peers will have a shared credential, regardless of any
   protocol failures. However, in some failure cases, the initiator may
   find itself with both a short password and a PSK for a particular
   peer. In that case, it MUST first try to authenticate with a
   password and, upon success, MUST attempt to convert it to a PSK. If
   password authentication fails, it MUST use the PSK and upon
   successful setup of the IKE SA MUST permanently delete the password.

4. Encrypting and Mapping the Nonce

4.1. Encrypting the Nonce

   The shared password is not used as is. Instead, it SHOULD be
   converted into a "stored password" SPwd, so that the plaintext
   password does not need to be stored for long periods. SPwd is
   defined as

      SPwd = prf("IKE with PACE", Pwd),

   where the literal string consists of ASCII characters with no zero
   terminator. If the negotiated pseudorandom function (prf) requires a
   fixed-size key, the literal string is either truncated or padded with
   zero octets on the right, as needed. Multiple copies of SPwd MAY be
   stored, if the prf function is not known in advance.

      KPwd = prf+(Ni | Nr, SPwd),

   where Ni and Nr are the regular IKE nonces, stripped of any headers.
   If the negotiated prf takes a fixed-length key and the lengths of Ni
   and Nr do not add up to that length, half the bits must come from Ni
   and half from Nr, taking the first bits of each. "prf+" is defined
   in Section 2.13 of [RFC5996]. The length of KPwd is determined by
   the key length of the negotiated encryption algorithm.

   A nonce s is randomly selected by the initiator (see Section 6.4 for
   additional considerations). The length of s MUST be exactly
   32 octets.

KPwd is now used with the encryption transform to encrypt the nonce:

    ENONCE = E(KPwd, s)

If an Initialization Vector (IV) is required by the cipher, it MUST
be included in the GSPM(ENONCE) payload. It is RECOMMENDED that the
IV be chosen both randomly and uniformly distributed, even though
this condition is not necessary for the cryptographic security of the
protocol.

Note: Padding MUST NOT be used when encrypting the nonce. The size
of the nonce has been chosen such that it can be encrypted with block
ciphers having block sizes of 32, 64, and 128 bits without any
padding.

If an authenticated encryption cipher [RFC5282] has been negotiated
for the IKE SA, it MUST NOT be used as is because such use would be
vulnerable to dictionary attacks. Instead, the corresponding
unauthenticated mode MUST be used. All Galois/Counter Mode (GCM) and
all Counter with CBC-MAC (CCM) encryption algorithms are mapped to
the corresponding counter-mode algorithm. For example, if the
negotiated encryption algorithm (Transform Type 1) is "AES-GCM with
an 8-octet Integrity Check Value (ICV)", then ENCR_AES_CTR (with the
same key length) is used to encrypt the nonce. If such a mapping
does not exist for a particular cipher, then it MUST NOT be used
within the current protocol.

4.2. Mapping the Nonce

The mapping is based on a second anonymous Diffie-Hellman key
agreement protocol to create a shared secret that is used together
with the exchanged nonce to calculate a common secret generator of
the group.

While in [TR03110] the generation of the shared secret is part of the
mapping, in the setting of IKEv2, a shared secret SASharedSecret has
already been generated as part of the IKE_SA_INIT step. Using the
notation of [RFC5996],

    SASharedSecret = g^ir

Let G and GE be the generator of the negotiated Diffie-Hellman group,
and the calculated ephemeral generator, respectively. The following
subsections describe the mapping for different Diffie-Hellman
variants.

4.2.1. Modular Diffie-Hellman

   The function Map:G->GE is defined as GE = G^s * SASharedSecret.

   Note that the protocol will fail if G^s = 1/SASharedSecret. If s is
   chosen randomly, this event occurs with negligible probability. In
   implementations that detect such a failure, the initiator SHOULD
   choose s again.

4.2.2. Elliptic Curve Diffie-Hellman

   The function Map:G->GE is defined as GE = s*G + SASharedSecret.

   Note that the protocol will fail if s*G = -SharedSecret. If s is
   chosen randomly, this event occurs with negligible probability. In
   implementations that detect such a failure, the initiator SHOULD
   choose s again.

5. Protocol Details

5.1. Password Processing

   The input password string SHOULD be processed according to the rules
   of the [RFC4013] profile of [RFC3454]. A password SHOULD be
   considered a "stored string" per [RFC3454]; therefore, unassigned
   code points are prohibited. The output is the binary representation
   of the processed UTF-8 character string. Prohibited output and
   unassigned codepoints encountered in SASLprep preprocessing SHOULD
   cause a preprocessing failure, and the output SHOULD NOT be used. A
   compliant implementation MUST NOT apply any other form of processing
   to the input password, other than as described in this section.

   See Section 3 of [RFC4013] for examples of SASLprep processing.

5.2. The SECURE_PASSWORD_METHODS Notification

   [RFC6467] defines a new type of Notify payload to indicate support
   for Secure Password Methods (SPMs) in the IKE_SA_INIT exchange. The
   SPM Notify payload is defined as follows:

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Next Payload  |C|  RESERVED   |         Payload Length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Protocol ID  |   SPI Size    |      Notify Message Type      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~               Security Parameter Index (SPI)                  ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                      Notification Data                        ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
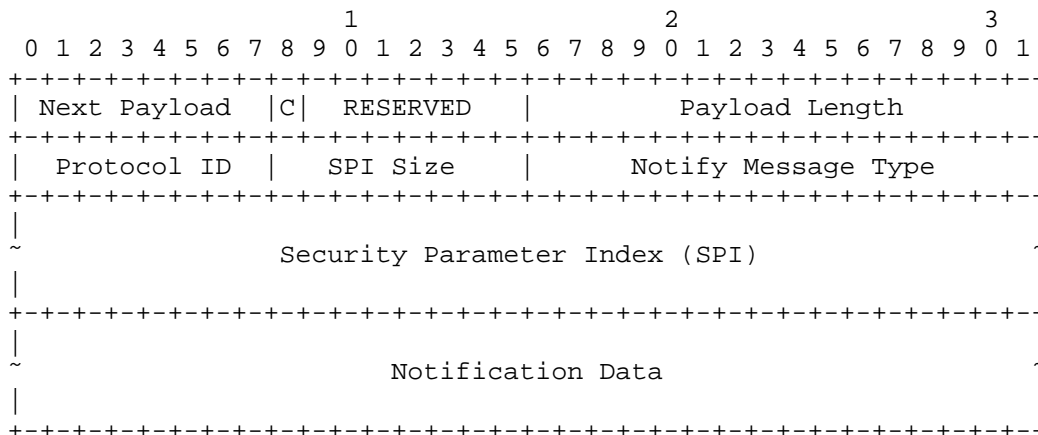
            Figure 3: SECURE_PASSWORD_METHODS Payload Structure

   The Protocol ID is zero, and the SPI Size is also zero, indicating
   that the SPI field is empty. The Notify Message Type is
   SECURE_PASSWORD_METHODS (value 16424).

   The Notification Data contains the list of the 16-bit secure password
   method numbers:

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Secure Password Method #1     | Secure Password Method #2     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Secure Password Method #3     | ...                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

             Figure 4: SECURE_PASSWORD_METHODS Payload Data

   For the current method, the list of proposed methods MUST include the
   value PACE (1).

5.3. The PSK_PERSIST Notification

   This document defines the PSK_PERSIST notification type, whose value
   is 16425. This notification MUST be sent with no data. However, for
   future extensibility, the receiver MUST ignore any notification data
   if such data is present.

5.4. The PSK_CONFIRM Notification

   This document defines the PSK_CONFIRM notification type, whose value
   is 16426. This notification MUST be sent with no data. However, for
   future extensibility, the receiver MUST ignore any notification data
   if such data is present.

5.5. The GSPM(ENONCE) Payload

   This protocol defines the ENONCE (encrypted nonce) payload, which
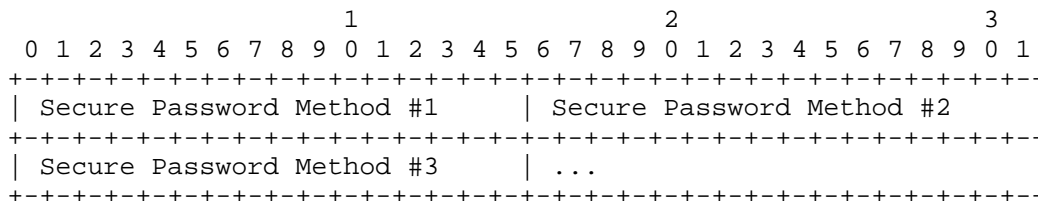   reuses the Generic SPM (GSPM) payload type [RFC6467] (value 49). Its
   format is as follows:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Next Payload  |C|  RESERVED   |         Payload Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | PACE-RESERVED |    Initialization Vector                      |
   +-+-+-+-+-+-+-+-+                                               +
   |     (optional, length depends on the encryption algorithm)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Encrypted Nonce                         ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
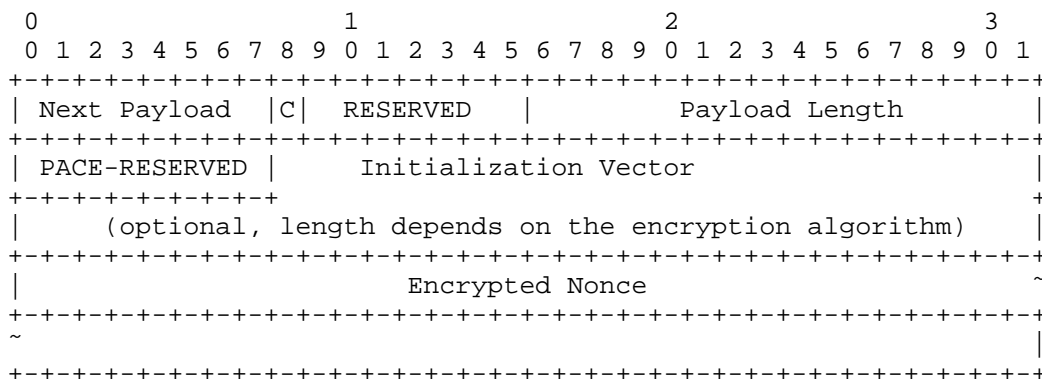
                   Figure 5: ENONCE Payload Structure

   See Section 4.1 for further details about the encrypted nonce. Note
   that the protocol -- and in particular this payload's format -- does
   not support any padding of the encrypted data.

   The PACE-RESERVED field must be sent as zero, and it must be rejected
   by the receiver if it is not 0.

5.6. The KE (KEi2/KEr2) Payloads in PACE

   PACE reuses the Key Exchange (KE) payload for its Diffie-Hellman
   exchange, with the new payloads being sent within the IKE_AUTH
   exchange. Since only one Diffie-Hellman group is negotiated, the
   group denoted by these payloads MUST be identical to the one used in
   the "regular" KE payloads in IKE_SA_INIT.

5.7. PACE and Session Resumption

   A session resumption [RFC5723] ticket may be requested during the
   IKE_AUTH exchange. The request MUST be sent in the request of the
   first round, and any response MUST be sent in the response of the
   second one.

   PACE should be considered an "authentication method", in the sense of
   Section 5 of [RFC5723], which means that its use MUST be noted in the
   protected ticket. The format of the ticket is not standardized;
   however, it is RECOMMENDED that this indication distinguish between
   the different secure password authentication methods defined for IKE.

   Note that even if the initial authentication used PACE and its
   extended IKE_AUTH, session resumption will still include the normal
   IKE_AUTH exchange.

6. Security Considerations

   A major goal of this protocol has been to maintain the level of
   security provided by IKEv2. What follows is an analysis of this
   protocol. The reader is referred to [RFC5996] for the generic IKEv2
   security considerations.

6.1. Credential Security Assumptions

   This protocol makes no assumption on the strength of the shared
   credential. Best common practices regarding minimal password length,
   use of multiple character classes, etc. SHOULD be followed.

6.2. Vulnerability to Passive and Active Attacks

   The protocol is secure against both passive and active attackers.
   See Section 6.8 for a security proof.

   While not attacking the cryptography, an attacker can still perform a
   standard password-guessing attack. To mitigate such attacks, an
   implementation MUST include standard protections, such as rate-
   limiting the number of allowed password-guessing attempts, possibly

   locking identities out after a certain number of failed attempts,
   etc. Note that the protocol is symmetric; therefore, this guidance
   applies to client-side implementations as well.

6.3. Perfect Forward Secrecy

   The key derivation for the IKE SA and any Child SAs is performed as
   part of IKEv2 and remains unchanged. It directly follows that
   perfect forward security is provided independent of the
   authentication additionally performed by PACE.

6.4. Randomness

   The security of this protocol depends on the quality generation of
   random quantities; see Section 5 of [RFC5996] for more details.
   Specifically, any deviation from randomness of the nonce s might
   compromise the password. Therefore, it is strongly RECOMMENDED that
   the initiator pass the raw random material through a strong prf to
   ensure the statistical qualities of the nonce.

6.5. Identity Protection

   This protocol is identical to IKEv2 in the quality of identity
   protection it provides. Both peers' identities are secure from
   passive attackers, and both peers' identities are exposed to active,
   man-in-the-middle attackers.

6.6. Denial of Service

   We are not aware of any new denial-of-service attack vector enabled
   by this protocol.

6.7. Choice of Encryption Algorithms

   Any transforms negotiated for IKEv2 may be used by this protocol.
   Please refer to Section 4.1 for the considerations regarding
   authenticated encryption ("combined mode") algorithms.

6.8. Security Model and Security Proof

   PACE is cryptographically proven secure in [PACEsec] in the model of
   Bellare, Pointcheval, and Rogaway [BPRmodel]. The setting in which
   PACE is proven secure is, however, slightly different from the
   setting used in IKEv2. The differences are described in the
   following:

   o Part of the mapping is already performed within IKEv2 before PACE
      is started. This rearrangement does not affect the proof, as the
      resulting PACESharedSecret remains close to uniformly distributed
      in the group generated by G.

   o The keys for the IKE SA and any Child SAs are already generated
      within IKEv2 before PACE is started. While those session keys
      could also be derived in PACE, only the keys for the
      authentication token are considered in the proof, which explicitly
      recommends a separate key for this purpose.

   o IKEv2 allows the negotiation of a stream cipher for PACE, while
      the proven variant always uses a block cipher. The ideal cipher
      is replaced in the proof by a lazy-sampling technique that is
      similarly applicable to the stream-cipher-based construction.

   The differences in the setting therefore have no impact on the
   validity of the proof.

6.9. Long-Term Credential Storage

   This protocol does not require that peers store the plaintext
   password. Instead, the value SPwd SHOULD be stored by both peers.

   In addition, the protocol allows both peers to replace the password
   by a crypto-strength shared secret. This solution improves the
   system's security (since passwords are often used for multiple
   applications), but at the cost of implementation complexity. In
   particular, if this optional mechanism is to be used, the credential
   database would need to be writable by the key management subsystem.

   See Appendix B for alternatives to this approach.

7. IANA Considerations

   IANA has allocated the following values:

   o A PACE value of 1 from the "IKEv2 Secure Password Methods"
      registry [RFC6467].

   o A PSK_PERSIST value of 16425 and a PSK_CONFIRM value of 16426 from
      the "IKEv2 Notify Message Types - Status Types" registry. We note
      that these notification types are generic and that other password
      authentication methods may also choose to use them.

8. Acknowledgments

   We would like to thank Dan Harkins for pointing out a security issue
   with our use of combined-mode algorithms in a previous version of the
   protocol. We thank Tero Kivinen for his generic framework document,
   and for a thorough and fruitful review. Hugo Krawczyk proposed that
   the password be amplified into a persistent shared secret.

9. References

9.1. Normative References

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

   [RFC2785]   Zuccherato, R., "Methods for Avoiding the "Small-
                   Subgroup" Attacks on the Diffie-Hellman Key Agreement
                   Method for S/MIME", RFC 2785, March 2000.

   [RFC3454]   Hoffman, P. and M. Blanchet, "Preparation of
                   Internationalized Strings ("stringprep")", RFC 3454,
                   December 2002.

   [RFC4013]   Zeilenga, K., "SASLprep: Stringprep Profile for User
                   Names and Passwords", RFC 4013, February 2005.

   [RFC4301]   Kent, S. and K. Seo, "Security Architecture for the
                   Internet Protocol", RFC 4301, December 2005.

   [RFC5996]   Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen,
                   "Internet Key Exchange Protocol Version 2 (IKEv2)",
                   RFC 5996, September 2010.

9.2. Informative References

   [BPRmodel]   Bellare, M., Pointcheval, D., and P. Rogaway,
                  "Authenticated Key Exchange Secure against Dictionary
                  Attacks", EUROCRYPT 2000, LNCS 1807, pp. 139-155,
                  Springer-Verlag, 2000, <http://www.iacr.org/cryptodb/
                  archive/2000/EUROCRYPT/18070139.pdf>.

   [ICAO]   ISO/IEC JTC1 SC17 WG3/TF5 for the International Civil
                  Aviation Organization (ICAO), "Supplemental Access
                  Control for Machine Readable Travel Documents",
                  Version 1.01, November 2010.

   [IKEv2-CONS]   Harkins, D., "Password-Based Authentication in IKEv2:
                  Selection Criteria and Considerations", Work
                  in Progress, October 2010.

   [PACEsec]   Bender, J., Fischlin, M., and D. Kuegler, "Security
                  Analysis of the PACE Key-Agreement Protocol",
                  LNCS 5735, pp. 33-48, Springer-Verlag (the extended
                  abstract appeared in Information Security Conference
                  (ISC) 2009), December 2009,
                  <http://eprint.iacr.org/2009/624>.

   [RFC5282]   Black, D. and D. McGrew, "Using Authenticated
                  Encryption Algorithms with the Encrypted Payload of the
                  Internet Key Exchange version 2 (IKEv2) Protocol",
                  RFC 5282, August 2008.

   [RFC5723]   Sheffer, Y. and H. Tschofenig, "Internet Key Exchange
                  Protocol Version 2 (IKEv2) Session Resumption",
                  RFC 5723, January 2010.

   [RFC5998]   Eronen, P., Tschofenig, H., and Y. Sheffer, "An
                  Extension for EAP-Only Authentication in IKEv2",
                  RFC 5998, September 2010.

   [RFC6467]   Kivinen, T., "Secure Password Framework for Internet
                  Key Exchange Version 2 (IKEv2)", RFC 6467,
                  December 2011.

   [TR03110]   BSI, "TR-03110, Advanced Security Mechanisms for
                  Machine Readable Travel Documents, Part 2 - Extended
                  Access Control Version 2 (EACv2), Password
                  Authenticated Connection Establishment (PACE), and
                  Restricted Identification (RI)", Version 2.10,
                  March 2012.

   [TR03111]   BSI, "TR-03111, Elliptic Curve Cryptography",
                  Version 1.11, April 2009.

Appendix A. Protocol Selection Criteria

   To support the selection of a password-based protocol for inclusion
   in IKEv2, a number of criteria are provided in [IKEv2-CONS]. In the
   following sections, those criteria are applied to the PACE protocol.

A.1. Security Criteria

   SEC1: PACE is a zero-knowledge protocol.

   SEC2: The protocol supports perfect forward secrecy and is resistant
          to replay attacks.

   SEC3: The identity protection provided by IKEv2 remains unchanged.

   SEC4: Any cryptographically secure Diffie-Hellman group can be used.

   SEC5: The protocol is proven secure in the Bellare-Pointcheval-
          Rogaway model.

   SEC6: Strong session keys are generated.

   SEC7: A transform of the password can be used instead of the
          password itself.

A.2. Intellectual Property Criteria

   IPR1: The first version of [TR03110] was published on May 21, 2007.

   IPR2: BSI has developed PACE aiming to be free of patents. BSI has
          not applied for a patent on PACE.

   IPR3: The protocol itself is believed to be free of IPR.

A.3. Miscellaneous Criteria

   MISC1:   One additional exchange is required.

   MISC2:   The protocol requires the following operations per entity:

            * one key derivation from the password,

            * one symmetric encryption or decryption,

            * one multi-exponentiation for the mapping,

　　　　　　　* one exponentiation for the key pair generation,

　　　　　　　* one exponentiation for the shared secret calculation, and

　　　　　　　* two symmetric authentications (generation and
　　　　　　　 verification).

　　MISC3:   The performance is independent of the type/size of password.

　　MISC4:   Internationalization of character-based passwords is
　　　　　　　supported.

　　MISC5:   The protocol uses the same group as that negotiated for
　　　　　　　IKEv2.

　　MISC6:   The protocol fits into the request/response nature of IKE.

　　MISC7:   The password-based symmetric encryption must be additionally
　　　　　　　negotiated.

　　MISC8:   Neither trusted third parties nor clock synchronization are
　　　　　　　required.

　　MISC9:   Only general cryptographic primitives are required.

　　MISC10: Any secure variant of Diffie-Hellman (e.g., Modular or
　　　　　　　Elliptic Curve) can be used.

　　MISC11: The protocol can be implemented easily based on existing
　　　　　　　cryptographic primitives.

Appendix B. Password Salting

　　This protocol requires that passwords not be stored in plaintext.
　　Instead, we store a hash of the password with a fixed hash. This
　　value is then used in the ZKPP protocol, replacing the original
　　password and acting as a "password equivalent". The main benefit of
　　this solution is that a system administrator or an undetermined
　　attacker does not get immediate access to the passwords. We believe
　　this is sufficiently secure for the main usage scenario of the
　　protocol.

However, the common practice of password salting is clearly more
powerful, and this appendix presents a few ideas on how password
salting can be applied and/or adapted to fit into a symmetric
protocol such as IKE. First, let us list the threats that we expect
salting to handle, as well as the non-threats:

o The plain password should not be visible to a casual onlooker, as
   noted above. It is assumed that very often the same password is
   used for multiple applications, and so a password exposed allows
   an attacker a starting point for further attacks.

o An attacker must not be able to construct lookup tables (such as
   the famous "rainbow tables") that enable her to discover the plain
   password.

o IKE is a symmetric protocol, in the sense that any of the peers
   might initiate an IKE exchange to another peer. As a result, all
   peers must have stored credentials (passwords or password
   equivalents) that would enable them to set up an IKE exchange.
   So, an attacker that reaches the credential store would in fact be
   able to impersonate IKE to another peer. We believe that this
   reduces, but does not invalidate, the importance of salting,
   because of the other threats that remain.

Below we present different scenarios and solutions that support
password salting in this setting.

We assume that each credential is used to authenticate exactly two
peers to one another; i.e., (as per the best practice), group
credentials are not allowed.

B.1. Solving the Asymmetric Case with Symmetric Cryptography

Despite the protocol's symmetry, there are use cases that are
somewhat asymmetric. Consider the case of an organization that
consists of a headquarters and branches, using a hub-and-spoke
architecture. Communication sessions can be initiated by the center
or by any of the branches, but only the center holds a large
credential database.

Here it would be possible to use traditional password salting,

    stored password = hash(salt, password),

where the hash function is a symmetric hash (e.g., HMAC-SHA-256,
using the salt as its key), and the salt is picked at random for each
password. The salt would need to be sent in the first exchange of
the protocol, regardless of which side initiates the session. Unlike

the normal use of salted passwords, here it is the stored password,
rather than the original password, that is used by the follow-on ZKPP
protocol.

B.2. Solving the Fully Symmetric Case with Asymmetric Cryptography

   For the fully symmetric case, we propose a salting method based on a
   commutative one-way function. This is essentially a novel variant of
   the RSA protocol. Using this solution, all protocol peers can store
   the password in a salted form.

   The implementation proposed here requires a composite number n that
   is common to all peers. The composite number n can be generated by a
   trusted (third) party as n = p * q, where p and q are strong primes
   (i.e., p = 2 * p' + 1 and q = 2 * q' + 1, where p' and q' are also
   primes), and the trusted party promises not to retain a copy of the
   primes. Alternatively, n can be chosen randomly and tested for
   "small" prime factors. In the latter case, it is certainly not
   guaranteed that n is composed of only two primes. While this has the
   advantage that no one knows the factorization of n, the disadvantage
   is that n is likely to be significantly easier to factor.

   Each peer then chooses a public encryption key "e". In a simple
   implementation, the encryption key is generated randomly by each
   peer, picking a different value for each of the passwords that it
   stores.

   Note that although the pair (n,e) is similar to an RSA public key,
   the usual rules for generating "e" for the RSA protocol do not apply
   here, and a random "e" is sufficient. The password is hashed by a
   symmetric hash function H (e.g., SHA-256). Each peer i stores the
   two values

      e_i, $H(P)^{e_i} \pmod{n}$,

   where P is the original password. The values e_i are exchanged by
   the peers before the ZKPP protocol commences (in IKEv2-PACE, this
   would be in IKE_SA_INIT), and the following value is used in the ZKPP
   protocol run that follows, in lieu of the original password:

      $H(P) ^ (e_i * e_j) \pmod{n}$.

   This transformation is used as a salting mechanism only, and the
   salted values themselves are never sent on the wire.

This scheme can be enhanced by basing the value "e" on each peer's
identity (IDi, IDr), e.g., making it a simple hash of the identity.
This eliminates the need to send "e" explicitly and additionally
binds the identity of the peer with its secret.

B.3. Generation of a Strong, Long-Term, Shared Secret

An alternative to salting is to store the plain passwords, but only
for a short while. As soon as the first IKE SA is set up between two
peers, the peers exchange nonces and generate a strong shared secret,
based on IKE's SK_d. They now destroy the short password and replace
it with the new secret.

This method has been added to the current protocol as an optional
mechanism.

Authors' Addresses

Dennis Kuegler
Bundesamt fuer Sicherheit in der Informationstechnik (BSI)
Postfach 200363
Bonn 53133
Germany

EMail: dennis.kuegler@bsi.bund.de

Yaron Sheffer
Porticor

EMail: yaronf.ietf@gmail.com