TLS Working Group   P. Eronen, Ed.
Internet-Draft   Nokia
Expires: August 22, 2005   H. Tschofenig, Ed.
                                                          Siemens
                                                February 21, 2005

        Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)
                        draft-ietf-tls-psk-06.txt

Status of this Memo

   This document is an Internet-Draft and is subject to all provisions
   of section 3 of RFC 3667. By submitting this Internet-Draft, each
   author represents that any applicable patent or other IPR claims of
   which he or she is aware have been or will be disclosed, and any of
   which he or she become aware will be disclosed, in accordance with
   RFC 3668.

   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 June 17, 2005.

Copyright Notice

Abstract

   This document specifies three sets of new ciphersuites for the
   Transport Layer Security (TLS) protocol to support authentication
   based on pre-shared keys. These pre-shared keys are symmetric keys,
   shared in advance among the communicating parties. The first set of
   ciphersuites uses only symmetric key operations for authentication.
   The second set uses a Diffie-Hellman exchange authenticated with a
   pre-shared key; and the third set combines public key authentication
   of the server with pre-shared key authentication of the client.

1. Introduction

   Usually TLS uses public key certificates [3] or Kerberos [12] for
   authentication. This document describes how to use symmetric keys
   (later called pre-shared keys or PSKs), shared in advance among the
   communicating parties, to establish a TLS connection.

   There are basically two reasons why one might want to do this:

   o First, TLS may be used in performance-constrained environments
      where the CPU power needed for public key operations is not
      available.

   o Second, pre-shared keys may be more convenient from a key
      management point of view. For instance, in closed environments
      where the connections are mostly configured manually in advance,
      it may be easier to configure a PSK than to use certificates.
      Another case is when the parties already have a mechanism for
      setting up a shared secret key, and that mechanism could be used
      to "bootstrap" a key for authenticating a TLS connection.

   This document specifies three sets of new ciphersuites for TLS.
   These ciphersuites use new key exchange algorithms, and re-use
   existing cipher and MAC algorithms from [3] and [2]. A summary of
   these ciphersuites is shown below.

```
    CipherSuite                       Key Exchange  Cipher        Hash

    TLS_PSK_WITH_RC4_128_SHA          PSK           RC4_128       SHA
    TLS_PSK_WITH_3DES_EDE_CBC_SHA     PSK           3DES_EDE_CBC  SHA
    TLS_PSK_WITH_AES_128_CBC_SHA      PSK           AES_128_CBC   SHA
    TLS_PSK_WITH_AES_256_CBC_SHA      PSK           AES_256_CBC   SHA
    TLS_DHE_PSK_WITH_RC4_128_SHA      DHE_PSK       RC4_128       SHA
    TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA DHE_PSK       3DES_EDE_CBC  SHA
    TLS_DHE_PSK_WITH_AES_128_CBC_SHA  DHE_PSK       AES_128_CBC   SHA
    TLS_DHE_PSK_WITH_AES_256_CBC_SHA  DHE_PSK       AES_256_CBC   SHA
    TLS_RSA_PSK_WITH_RC4_128_SHA      RSA_PSK       RC4_128       SHA
    TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA RSA_PSK       3DES_EDE_CBC  SHA
    TLS_RSA_PSK_WITH_AES_128_CBC_SHA  RSA_PSK       AES_128_CBC   SHA
    TLS_RSA_PSK_WITH_AES_256_CBC_SHA  RSA_PSK       AES_256_CBC   SHA
```

   The first set of ciphersuites (with PSK key exchange algorithm),
   defined in Section 2 use only symmetric key algorithms, and are thus
   especially suitable for performance-constrained environments.

   The ciphersuites in Section 3 (with DHE_PSK key exchange algorithm)
   use a PSK to authenticate a Diffie-Hellman exchange. These
   ciphersuites protect against dictionary attacks by passive

eavesdroppers (but not active attackers), and also provide Perfect
Forward Secrecy (PFS).

The third set of ciphersuites (with RSA_PSK key exchange algorithm),
defined in Section 4, combine public key based authentication of the
server (using RSA and certificates) with mutual authentication using
a PSK.

1.1 Applicability statement

The ciphersuites defined in this document are intended for a rather
limited set of applications, usually involving only a very small
number of clients and servers. Even in such environments, other
alternatives may be more appropriate.

If the main goal is to avoid PKIs, another possibility worth
considering is to use self-signed certificates with public key
fingerprints. Instead of manually configuring a shared secret in,
for instance, some configuration file, a fingerprint (hash) of the
other party's public key (or certificate) could be placed there
instead.

It is also possible to use the SRP (Secure Remote Password)
ciphersuites for shared secret authentication [14]. SRP was designed
to be used with passwords, and incorporates protection against
dictionary attacks. However, it is computationally more expensive
than the PSK ciphersuites in Section 2.

1.2 Conventions used in this document

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 [1].

2. PSK key exchange algorithm

   This section defines the PSK key exchange algorithm and associated
   ciphersuites. These ciphersuites use only symmetric key algorithms.

   It is assumed that the reader is familiar with ordinary TLS
   handshake, shown below. The elements in parenthesis are not included
   when PSK key exchange algorithm is used, and "*" indicates a
   situation-dependent message that is not always sent.

```
      Client                                            Server
      ------                                            ------

      ClientHello                  -------->
                                                     ServerHello
                                                    (Certificate)
                                              ServerKeyExchange*
                                             (CertificateRequest)
                                   <--------       ServerHelloDone
      (Certificate)
      ClientKeyExchange
      (CertificateVerify)
      ChangeCipherSpec
      Finished                     -------->
                                                  ChangeCipherSpec
                                   <--------              Finished
      Application Data             <------->      Application Data
```

   The client indicates its willingness to use pre-shared key
   authentication by including one or more PSK ciphersuites in the
   ClientHello message. If the TLS server also wants to use pre-shared
   keys, it selects one of the PSK ciphersuites, places the selected
   ciphersuite in the ServerHello message, and includes an appropriate
   ServerKeyExchange message (see below). The Certificate and
   CertificateRequest payloads are omitted from the response.

   Both clients and servers may have pre-shared keys with several
   different parties. The client indicates which key to use by
   including a "PSK identity" in the ClientKeyExchange message (note
   that unlike in [7], the session_id field in ClientHello message keeps
   its usual meaning). To help the client in selecting which identity
   to use, the server can provide a "PSK identity hint" in the
   ServerKeyExchange message. If no hint is provided, the
   ServerKeyExchange message is omitted. See Section 5 for more
   detailed description of these fields.

   The format of the ServerKeyExchange and ClientKeyExchange messages is
   shown below.

```
    struct {
        select (KeyExchangeAlgorithm) {
            /* other cases for rsa, diffie_hellman, etc. */
            case psk:  /* NEW */
                opaque psk_identity_hint<0..2^16-1>;
        };
    } ServerKeyExchange;

    struct {
        select (KeyExchangeAlgorithm) {
            /* other cases for rsa, diffie_hellman, etc. */
            case psk:   /* NEW */
                opaque psk_identity<0..2^16-1>;
        } exchange_keys;
    } ClientKeyExchange;
```

The premaster secret is formed as follows: if the PSK is N octets long, concatenate an uint16 with the value N, N zero octets, a second uint16 with the value N, and the PSK itself.

Note 1: All the ciphersuites in this document share the same general structure for the premaster secret, namely

```
    struct {
        opaque other_secret<0..2^16-1>;
        opaque psk<0..2^16-1>;
    };
```

Here "other_secret" is either zeroes (plain PSK case), or comes from the Diffie-Hellman or RSA exchange (DHE_PSK and RSA_PSK, respectively). See Sections 3 and 4 for a more detailed description.

Note 2: Using zeroes for "other_secret" effectively means that only the HMAC-SHA1 part (but not the HMAC-MD5 part) of the TLS PRF is used when constructing the master secret. See [8] for a more detailed rationale.

The TLS handshake is authenticated using the Finished messages as usual.

If the server does not recognize the PSK identity, it MAY respond with an "unknown_psk_identity" alert message. Alternatively, if the server wishes to hide the fact that the PSK identity was not known, it MAY continue the protocol as if the PSK identity existed but the key was incorrect: that is, respond with a "decrypt_error" alert.

3. **DHE_PSK key exchange algorithm**

   This section defines additional ciphersuites that use a PSK to
   authenticate a Diffie-Hellman exchange. These ciphersuites give some
   additional protection against dictionary attacks, and also provide
   Perfect Forward Secrecy (PFS). See Section 7 for discussion of
   related security considerations.

   When these ciphersuites are used, the ServerKeyExchange and
   ClientKeyExchange messages also include the Diffie-Hellman
   parameters. The PSK identity and identity hint fields have the same
   meaning as in the previous section (note that the ServerKeyExchange
   message is always sent even if no PSK identity hint is provided).

   The format of the ServerKeyExchange and ClientKeyExchange messages is
   shown below.

```
      struct {
          select (KeyExchangeAlgorithm) {
              /* other cases for rsa, diffie_hellman, etc. */
              case diffie_hellman_psk:  /* NEW */
                  opaque psk_identity_hint<0..2^16-1>;
                  ServerDHParams params;
          };
      } ServerKeyExchange;

      struct {
          select (KeyExchangeAlgorithm) {
              /* other cases for rsa, diffie_hellman, etc. */
              case diffie_hellman_psk:   /* NEW */
                  opaque psk_identity<0..2^16-1>;
                  ClientDiffieHellmanPublic public;
          } exchange_keys;
      } ClientKeyExchange;
```

   The premaster secret is formed as follows. First, perform the
   Diffie-Hellman computation in the same way as for other
   Diffie-Hellman based ciphersuites in [3]. Let Z be the value
   produced by this computation (with leading zero bytes stripped as in
   other Diffie-Hellman based ciphersuites). Concatenate an uint16
   containing the length of Z (in octets), Z itself, an uint16
   containing the length of the PSK (in octets), and the PSK itself.

   This corresponds to the general structure for the premaster secrets
   (see Note 1 in Section 2) in this document, with "other_secret"
   containing Z.

[4](#). RSA_PSK key exchange algorithm

   The ciphersuites in this section use RSA and certificates to
   authenticate the server, in addition to using a PSK.

   As in normal RSA ciphersuites, the server must send a Certificate
   message. The format of the ServerKeyExchange and ClientKeyExchange
   messages is shown below. If no PSK identity hint is provided, the
   ServerKeyExchange message is omitted.

```
   struct {
       select (KeyExchangeAlgorithm) {
           /* other cases for rsa, diffie_hellman, etc. */
           case rsa_psk:  /* NEW */
               opaque psk_identity_hint<0..2^16-1>;
       };
   } ServerKeyExchange;

   struct {
       select (KeyExchangeAlgorithm) {
           /* other cases for rsa, diffie_hellman, etc. */
           case rsa_psk:   /* NEW */
               opaque psk_identity<0..2^16-1>;
               EncryptedPreMasterSecret;
       } exchange_keys;
   } ClientKeyExchange;
```

   The EncryptedPreMasterSecret field sent from the client to the server
   contains a 2-byte version number and a 46-byte random value,
   encrypted using the server's RSA public key as described in [Section
   7.4.7.1](#) of [[3](#)]. The actual premaster secret is formed by both
   parties as follows: concatenate an uint16 with the value 48, the
   2-byte version number and the 46-byte random value, an uint16
   containing the length of the PSK (in octets), and the PSK itself.
   (The premaster secret is thus 52 octets longer than the PSK.)

   This corresponds to the general structure for the premaster secrets
   (see Note 1 in [Section 2](#)) in this document, with "other_secret"
   containing both the 2-byte version number and the 46-byte random
   value.

   Neither the normal RSA ciphersuites nor these RSA_PSK ciphersuites
   themselves specify what the certificates contain (in addition to the
   RSA public key), or how the certificates are to be validated. In
   particular, it is possible to use the RSA_PSK ciphersuites with
   unvalidated self-signed certificates to provide somewhat similar
   protection against dictionary attacks as the DHE_PSK ciphersuites
   defined in [Section 3](#).

5. Conformance requirements

   It is expected that different types of identities are useful for
   different applications running over TLS. This document does not
   therefore mandate the use of any particular type of identity (such as
   IPv4 address or FQDN).

   However, the TLS client and server clearly have to agree on the
   identities and keys to be used. To improve interoperability, this
   document places requirements on how the identity is encoded on the
   wire, and what kinds of identities and keys implementations have to
   support.

   The requirements for implementations are divided to two categories,
   requirements for TLS implementations and management interfaces. In
   this context, "TLS implementation" refers to a TLS library or module
   that is intended to be used for several different purposes, while
   "management interface" would typically be implemented by a particular
   application that uses TLS.

5.1 PSK identity encoding

   The PSK identity MUST be first converted to a character string, and
   then encoded to octets using UTF-8 [5]. For instance,

   o IPv4 addresses are sent as dotted-decimal strings (e.g.,
      "192.0.1.2"), not as 32-bit integers in network byte order.

   o Domain names are sent in their usual text form (e.g.,
      "www.example.com" or "embedded.dot.example.net"), not in DNS
      protocol wire format.

   o X.500 Distinguished Names are sent in their string representation
      [9], not as BER-encoded ASN.1.

   This encoding is clearly not optimal for many types of identities.
   It was chosen to avoid identity type specific parsing and encoding
   code in implementations where the identity is configured by a person
   using some kind of management interface. Requiring such identity
   type specific code would also increase the chances for
   interoperability problems resulting from different implementations
   supporting different identity types.

5.2 Identity hint

   In the absence of an application profile specification specifying
   otherwise, servers SHOULD NOT provide an identity hint and clients
   MUST ignore the identity hint field. Applications that do use this

field MUST specify its contents, how the value is chosen by the TLS
server, and what the TLS client is expected to do with the value.

5.3 Requirements for TLS implementations

TLS implementations supporting these ciphersuites MUST support
arbitrary PSK identities up to 128 octets in length, and arbitrary
PSKs up to 64 octets in length. Supporting longer identities and
keys is RECOMMENDED.

5.4 Requirements for management interfaces

In the absence of an application profile specification specifying
otherwise, a management interface for entering the PSK and/or PSK
identity MUST support entering PSK identities consisting of up to 128
printable ASCII characters, and MUST support entering PSKs up to 64
octets in length as ASCII strings and in hexadecimal encoding.

Supporting as wide character repertoire and as long identities and
keys as feasible is RECOMMENDED, as is processing the character
string with an appropriate stringprep [10] profile.

6. IANA considerations

IANA does not currently have a registry for TLS-related numbers, so
there are no IANA actions associated with this document.

For easier reference in the future, the ciphersuite numbers defined
in this document are summarized below.

```
CipherSuite TLS_PSK_WITH_RC4_128_SHA          = { 0x00, 0x8A };
CipherSuite TLS_PSK_WITH_3DES_EDE_CBC_SHA     = { 0x00, 0x8B };
CipherSuite TLS_PSK_WITH_AES_128_CBC_SHA      = { 0x00, 0x8C };
CipherSuite TLS_PSK_WITH_AES_256_CBC_SHA      = { 0x00, 0x8D };
CipherSuite TLS_DHE_PSK_WITH_RC4_128_SHA      = { 0x00, 0x8E };
CipherSuite TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA = { 0x00, 0x8F };
CipherSuite TLS_DHE_PSK_WITH_AES_128_CBC_SHA  = { 0x00, 0x90 };
CipherSuite TLS_DHE_PSK_WITH_AES_256_CBC_SHA  = { 0x00, 0x91 };
CipherSuite TLS_RSA_PSK_WITH_RC4_128_SHA      = { 0x00, 0x92 };
CipherSuite TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA = { 0x00, 0x93 };
CipherSuite TLS_RSA_PSK_WITH_AES_128_CBC_SHA  = { 0x00, 0x94 };
CipherSuite TLS_RSA_PSK_WITH_AES_256_CBC_SHA  = { 0x00, 0x95 };
```

This document also defines a new TLS alert message,
unknown_psk_identity(115).

7. Security Considerations

As with all schemes involving shared keys, special care should be
taken to protect the shared values and to limit their exposure over
time.

7.1 Perfect forward secrecy (PFS)

The PSK and RSA_PSK ciphersuites defined in this document do not
provide Perfect Forward Secrecy (PFS). That is, if the shared secret
key (in PSK ciphersuites), or both the shared secret key and the RSA
private key (in RSA_PSK ciphersuites), is somehow compromised, an
attacker can decrypt old conversations.

The DHE_PSK ciphersuites provide Perfect Forward Secrecy if a fresh
DH private key is generated for each handshake.

7.2 Brute-force and dictionary attacks

Use of a fixed shared secret of limited entropy (for example, a PSK
that is relatively short, or was chosen by a human and thus may
contain less entropy than its length would imply) may allow an
attacker to perform a brute-force or dictionary attack to recover the
secret. This may be either an off-line attack (against a captured

TLS handshake messages), or an on-line attack where the attacker
attempts to connect to the server and tries different keys.

For the PSK ciphersuites, an attacker can get the information
required for an off-line attack by eavesdropping a TLS handshake, or
by getting a valid client to attempt connection with the attacker (by
tricking the client to connect to wrong address, or intercepting a
connection attempt to the correct address, for instance).

For the DHE_PSK ciphersuites, an attacker can obtain the information
by getting a valid client to attempt connection with the attacker.
Passive eavesdropping alone is not sufficient.

For the RSA_PSK ciphersuites, only the server (authenticated using
RSA and certificates) can obtain sufficient information for an
off-line attack.

It is RECOMMENDED that implementations that allow the administrator
to manually configure the PSK also provide a functionality for
generating a new random PSK, taking [4] into account.

7.3 Identity privacy

The PSK identity is sent in cleartext. While using a user name or
other similar string as the PSK identity is the most straightforward
option, it may lead to problems in some environments since an
eavesdropper is able to identify the communicating parties. Even
when the identity does not reveal any information itself, reusing the
same identity over time may eventually allow an attacker to perform
traffic analysis to identify the parties. It should be noted that
this is no worse than client certificates, since they are also sent
in cleartext.

7.4 Implementation notes

The implementation notes in [11] about correct implementation and use
of RSA (including Section 7.4.7.1) and Diffie-Hellman (including
Appendix F.1.1.3) apply to the DHE_PSK and RSA_PSK ciphersuites as
well.

8. Acknowledgments

The protocol defined in this document is heavily based on work by Tim
Dierks and Peter Gutmann, and borrows some text from [7] and [2].
The DHE_PSK and RSA_PSK ciphersuites are based on earlier work in
[6].

Valuable feedback was also provided by Bernard Aboba, Philip

Ginzboorg, Peter Gutmann, Russ Housley, David Jablon, Nikos
Mavroyanopoulos, Bodo Moeller, Eric Rescorla, and Mika Tervonen.

When the first version of this draft was almost ready, the authors
learned that something similar had been proposed already in 1996
[13]. However, this draft is not intended for web password
authentication, but rather for other uses of TLS.

9. References

9.1 Normative References

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

[2]   Chown, P., "Advanced Encryption Standard (AES) Ciphersuites
        for Transport Layer Security (TLS)", RFC 3268, June 2002.

[3]   Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC
        2246, January 1999.

[4]   Eastlake, D., Crocker, S. and J. Schiller, "Randomness
        Recommendations for Security", RFC 1750, December 1994.

[5]   Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC
        3629, November 2003.

9.2 Informative References

[6]   Badra, M., Cherkaoui, O., Hajjeh, I. and A. Serhrouchni,
        "Pre-Shared-Key key Exchange methods for TLS",
        draft-badra-tls-key-exchange-00 (work in progress), August
        2004.

[7]   Gutmann, P., "Use of Shared Keys in the TLS Protocol",
        draft-ietf-tls-sharedkeys-02 (expired), October 2003.

[8]   Krawczyk, H., "Re: TLS shared keys PRF", message on
        ietf-tls@lists.certicom.com mailing list 2004-01-13,
        http://www.imc.org/ietf-tls/mail-archive/msg04098.html.

[9]   Zeilenga, K., "LDAP: String Representation of Distinguished
        Names", draft-ietf-ldapbis-dn-15 (work in progress), October
        2004.

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

[11]   Dierks, T. and E. Rescorla, "The TLS Protocol Version 1.1",
       draft-ietf-tls-rfc2246-bis-09 (work in progress), December
       2004.

[12]   Medvinsky, A. and M. Hur, "Addition of Kerberos Cipher Suites
       to Transport Layer Security (TLS)", RFC 2712, October 1999.

[13]   Simon, D., "Addition of Shared Key Authentication to Transport
       Layer Security (TLS)", draft-ietf-tls-passauth-00 (expired),
       November 1996.

[14]   Taylor, D., Wu, T., Mavroyanopoulos, N. and T. Perrin, "Using
       SRP for TLS Authentication", draft-ietf-tls-srp-08 (work in
       progress), August 2004.

Authors' and Contributors' Addresses

    Pasi Eronen
    Nokia Research Center
    P.O. Box 407
    FIN-00045 Nokia Group
    Finland
    Email: pasi.eronen@nokia.com

    Hannes Tschofenig
    Siemens
    Otto-Hahn-Ring 6
    Munich, Bayern 81739
    Germany
    Email: Hannes.Tschofenig@siemens.com

    Mohamad Badra
    ENST Telecom
    46 rue Barrault
    75634 Paris
    France
    Email: Mohamad.Badra@enst.fr

    Omar Cherkaoui
    UQAM University
    Montreal (Quebec)
    Canada
    Email: cherkaoui.omar@uqam.ca

    Ibrahim Hajjeh
    ENST Telecom
    46 rue Barrault
    75634 Paris
    France
    Email: Ibrahim.Hajjeh@enst.fr

    Ahmed Serhrouchni
    ENST Telecom
    46 rue Barrault
    75634 Paris
    France
    Email: Ahmed.Serhrouchni@enst.fr

Appendix A. **Changelog**

(This section should be removed by the RFC Editor before
publication.)

Changes from -05 to -06:

o Small clarifications to how the premaster secret is formed.

o Added a section about conformance requirements, and moved existing
   text about identity formats there.

Changes from -04 to -05:

o Omit ServerKeyExchange message (in PSK/RSA_PSK versions) if no
   identity hint is provided.

Changes from -03 to -04:

o Added a note about premaster secret "general structure" in
   Sections 3 and 4.

o Something in the I-D submission procedure had removed all
      circumflexes from -03 version, turning e.g. "2^16" (two-to-
      the sixteenth power) to "216" (two hundred and sixteen).
      Let's try again.

Changes from -02 to -03:

o Aligned the way the premaster secret is derived.

o Specified that identities must be sent as human-readable UTF-8
   strings, not in binary formats. Changed reference to RFC 3629
   from informative to normative.

o Selected ciphersuite and alert numbers, and updated IANA
   considerations section to match this.

o Reworded some text about dictionary attacks in Section 6.2.

Changes from -01 to -02:

o Clarified text about DHE_PSK ciphersuites in Section 1.

o Clarified explanation of HMAC-SHA1/MD5 use of PRF in Section 2.

o Added note about certificate validation and self-signed
   certificates in Section 4.

o Added Mohamad Badra et al. as contributors.

Changes from draft-ietf-tls-psk-00 to -01:

o Added DHE_PSK and RSA_PSK key exchange algorithms, and updated
   other text accordingly

o Removed SHA-1 hash from PSK key exchange premaster secret
   construction (since premaster secret doesn't need to be 48 bytes).

o Added unknown_psk_identity alert message.

o Updated IANA considerations section.

Changes from draft-eronen-tls-psk-00 to draft-ietf-tls-psk-00:

o Updated dictionary attack considerations based on comments from
   David Jablon.

o Added a recommendation about using UTF-8 in the identity field.

o Removed Appendix A comparing this document with
   draft-ietf-tls-sharedkeys-02.

o Removed IPR comment about SPR.

o Minor editorial changes.

Intellectual Property Statement

   The IETF takes no position regarding the validity or scope of any
   Intellectual Property Rights or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; nor does it represent that it has
   made any independent effort to identify any such rights. Information
   on the procedures with respect to rights in RFC documents can be
   found in BCP 78 and BCP 79.

   Copies of IPR disclosures made to the IETF Secretariat and any
   assurances of licenses to be made available, or the result of an
   attempt made to obtain a general license or permission for the use of
   such proprietary rights by implementers or users of this
   specification can be obtained from the IETF on-line IPR repository at
   http://www.ietf.org/ipr.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights that may cover technology that may be required to implement
   this standard. Please address the information to the IETF at
   ietf-ipr@ietf.org.

Disclaimer of Validity

   This document and the information contained herein are provided on an
   "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS
   OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET
   ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED,
   INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
   INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
   WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

   Copyright (C) The Internet Society (2004). This document is subject
   to the rights, licenses and restrictions contained in BCP 78, and
   except as set forth therein, the authors retain all their rights.

Acknowledgment

   Funding for the RFC Editor function is currently provided by the
   Internet Society.