TLS Working Group
Internet-Draft
Expires: February 16, 2005

P. Eronen
Nokia
H. Tschofenig
Siemens
August 18, 2004

Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)
draft-ietf-tls-psk-01.txt

Status of this Memo

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 [11] 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:

- First, TLS may be used in performance-constrained environments where the CPU power needed for public key operations is not available.
- 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 give some additional protection against dictionary

attacks, 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 [13]. 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.

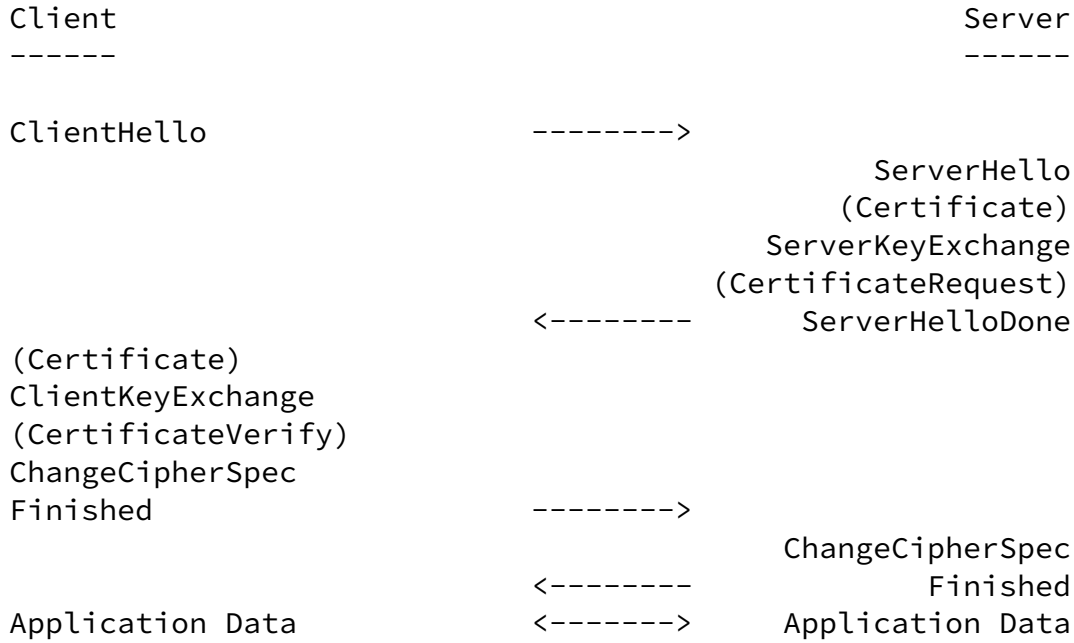
1.2 Conventions used in this document

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.



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 [6], 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 (note that if no hint is provided, a ServerKeyExchange message is still sent).

This document does not specify the format of the PSK identity or PSK identity hint; neither is specified how exactly the client uses the hint (if it uses it at all). The parties have to agree on the

identities when the shared secret is configured (however, see Section 6 for related security considerations). In situations where the identity is entered by a person, it is RECOMMENDED that the input is processed using an appropriate stringprep [9] profile and encoded in octets using UTF-8 encoding [14]. One possible stringprep profile is described in [8].

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 N zero octets and the PSK.

Note: This effectively means that only the HMAC-SHA1 part of the TLS PRF is used, and the HMAC-MD5 part is not used. See [7] 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 6 for discussion of

related security considerations.

When these ciphersuites are used, the ServerKeyExchange and ClientKeyExchange also include the Diffie-Hellman parameters. The PSK identity and identity hint fields have the same meaning as in the previous section.

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: concatenate the value produced by the Diffie-Hellman exchange (with leading zero bytes stripped as in other Diffie-Hellman based ciphersuites) and the PSK, in this order.

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.


```
    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 premaster secret is formed as follows: concatenate the 48-byte value generated by the client (and sent to the server in ClientKeyExchange message) and the PSK, in this order.

5. IANA considerations

(This depends on whether this document is published before or after TLS 1.1.)

(If after 1.1) This document does not create any new namespaces to be maintained by IANA, but it requires new values in the ciphersuite namespace defined in TLS 1.1 specification.

(If before 1.1) 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, 0xTBD };
   CipherSuite TLS_PSK_WITH_3DES_EDE_CBC_SHA     = { 0x00, 0xTBD };
   CipherSuite TLS_PSK_WITH_AES_128_CBC_SHA      = { 0x00, 0xTBD };
   CipherSuite TLS_PSK_WITH_AES_256_CBC_SHA      = { 0x00, 0xTBD };
   CipherSuite TLS_DHE_PSK_WITH_RC4_128_SHA      = { 0x00, 0xTBD };
   CipherSuite TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA = { 0x00, 0xTBD };
   CipherSuite TLS_DHE_PSK_WITH_AES_128_CBC_SHA  = { 0x00, 0xTBD };
   CipherSuite TLS_DHE_PSK_WITH_AES_256_CBC_SHA  = { 0x00, 0xTBD };
   CipherSuite TLS_RSA_PSK_WITH_RC4_128_SHA      = { 0x00, 0xTBD };
   CipherSuite TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA = { 0x00, 0xTBD };
   CipherSuite TLS_RSA_PSK_WITH_AES_128_CBC_SHA  = { 0x00, 0xTBD };
   CipherSuite TLS_RSA_PSK_WITH_AES_256_CBC_SHA  = { 0x00, 0xTBD };
```

This document also defines a new TLS alert message,

unknown_psk_identity(TBD). Since IANA does not maintain a registry of TLS alert messages, no IANA action is needed for this.

6. 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.

6.1 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.

6.2 Brute-force and dictionary attacks

Use of a fixed shared secret of limited entropy (such as a password) 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.

6.3 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.

6.4 Implementation notes

The implementation notes in [10] 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.

7. Acknowledgments

The protocol defined in this document is heavily based on work by Tim Dierks and Peter Gutmann, and borrows some text from [6] and [2]. Valuable feedback was also provided by Philip Ginzboorg, Peter Gutmann, David Jablon, Nikos Mavroyanopoulos, Bodo Moeller, 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 [12]. However, this draft is not intended for web password authentication, but rather for other uses of TLS.

The DHE_PSK and RSA_PSK ciphersuites borrow heavily from [5].

8. Open issues

- Identity privacy could be provided (in DHE_PSK/RSA_PSK versions) by encrypting the psk_identity payload with keys derived from the DH value/RSA-encrypted random (but not PSK). But perhaps this would be an unnecessary complication.

- The way the PSK is combined with DH value (and is then used to calculate the Finished message) is not exactly the traditional

way. It should be OK with TLS-PRF, though.

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.

9.2 Informative References

[5] 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.

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

[7] 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.

[8] Zeilenga, K., "SASLprep: Stringprep profile for user names and passwords", draft-ietf-sasl-saslprep-10 (work in progress), July 2004.

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

[10] Dierks, T. and E. Rescorla, "The TLS Protocol Version 1.1", draft-ietf-tls-rfc2246-bis-08 (work in progress), August 2004.

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

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

[13] Taylor, D., Wu, T., Mavroyanopoulos, N. and T. Perrin, "Using

SRP for TLS Authentication", draft-ietf-tls-srp-07 (work in progress), June 2004.

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

Authors' 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

Appendix A. Changelog

(This section should be removed by the RFC Editor before

publication.)

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

- Added DHE_PSK and RSA_PSK key exchange algorithms, and updated other text accordingly
- Removed SHA-1 hash from PSK key exchange premaster secret construction (since premaster secret doesn't need to be 48 bytes).
- Added unknown_psk_identity alert message.
- Updated IANA considerations section.

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

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

- Added a recommendation about using UTF-8 in the identity field.
- Removed Appendix A comparing this document with draft-ietf-tls-sharedkeys-02.
- Removed IPR comment about SPR.
- Minor editorial changes.

Intellectual Property Statement

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

Copyright Statement

Acknowledgment

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