TLS Working Group
Internet-Draft
Expires: October 1, 2004

N. Mavroyanopoulos

April 2, 2004

# Using OpenPGP keys for TLS authentication
# draft-ietf-tls-openpgp-keys-05

## Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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 October 1, 2004.

## Copyright Notice



## Abstract

This memo proposes extensions to the TLS protocol to support the OpenPGP trust model and keys. The extensions discussed here include a certificate type negotiation mechanism, and the required modifications to the TLS Handshake Protocol.

Table of Contents

## 1. Introduction

At the time of writing, TLS [1] uses the PKIX [6] infrastructure, to provide certificate services. Currently the PKIX protocols are limited to a hierarchical key management and as a result, applications which follow different - non hierarchical - trust models, like the "web of trust" model, could not be benefited by TLS.

OpenPGP keys (sometimes called OpenPGP certificates), provide security services for electronic communications. They are widely deployed, especially in electronic mail applications, provide public key authentication services, and allow distributed key management.

This document will extend the TLS protocol to support OpenPGP keys and trust model using the existing TLS cipher suites. In brief this would be achieved by adding a negotiation of the certificate type in addition to the normal handshake negotiations. Then the required modifications to the handshake messages, in order to hold OpenPGP keys as well, will be described. The the normal handshake procedure with X.509 certificates will not be altered, to preserve compatibility with existing TLS servers and clients.

This document uses the same notation used in the TLS Protocol specification.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

## 2. Extension Type

A new value, "cert_type(7)", is added to the enumerated ExtensionType, defined in TLSEXT [3]. This value is used as the extension number for the extensions in both the client hello message and the server hello message. This new extension type will be used for certificate type negotiation.

## 3. Changes to the Handshake Message Contents

This section describes the changes to the TLS handshake message contents when OpenPGP keys are to be used for authentication.

### 3.1 Client Hello

In order to indicate the support of multiple certificate types clients will include an extension of type "cert_type" to the extended client hello message. The hello extension mechanism is described in TLSEXT [3].

This extension carries a list of supported certificate types the client can use, sorted by client preference. This extension MAY be omitted if the client only supports X.509 certificates. The "extension_data" field of this extension will contain a CertificateTypeExtension structure.

```
enum { client, server } ClientOrServerExtension;

enum { X.509(0), OpenPGP(1), (255) } CertificateType;

struct {
   select(ClientOrServerExtension) {
      case client:
         CertificateType certificate_types<1..2^8-1>;
      case server:
         CertificateType certificate_type;
   }
} CertificateTypeExtension;
```

### 3.2 Server Hello

Servers that receive an extended client hello containing the "cert_type" extension, and have chosen a cipher suite that supports certificates, then they MUST select a certificate type from the certificate_types field in the extended client hello, or terminate the connection with a fatal alert of type "unsupported_certificate".

The certificate type selected by the server, is encoded in a CertificateTypeExtension structure, which is included in the extended server hello message, using an extension of type "cert_type". Servers that only support X.509 certificates MAY omit including the "cert_type" extension in the extended server hello.

### 3.3 Server Certificate

The contents of the certificate message sent from server to client and vice versa are determined by the negotiated certificate type and the selected cipher suite's key exchange algorithm.

If the OpenPGP certificate type is negotiated then it is required to present an OpenPGP key in the Certificate message. The OpenPGP key must contain a public key that matches the selected key exchange algorithm, as shown below.

| Key Exchange Algorithm | OpenPGP Key Type |
|---|---|
| RSA | RSA public key which can be used for encryption. |
| DHE_DSS | DSS public key. |
| DHE_RSA | RSA public key which can be used for signing. |

An OpenPGP public key appearing in the Certificate message will be sent using the binary OpenPGP format. The term public key is used to describe a composition of OpenPGP packets to form a block of data which contains all information needed by the peer. This includes public key packets, user ID packets and all the fields described in "Transferable Public Keys" section in OpenPGP [2].

The option is also available to send an OpenPGP fingerprint, instead of sending the entire key. The process of fingerprint generation is described in OpenPGP [2]. The peer shall respond with a "certificate_unobtainable" fatal alert if the key with the given key fingerprint cannot be found. The "certificate_unobtainable" fatal alert is defined in section 4 of TLSEXT [3].

If the key is not valid, expired, revoked, corrupt, the appropriate fatal alert message is sent from section A.3 of the TLS specification. If a key is valid and neither expired nor revoked, it is accepted by the protocol. The key validation procedure is a local matter outside the scope of this document.

```
enum {
     key_fingerprint (0), key (1), (255)
} PGPKeyDescriptorType;

opaque PGPKeyFingerprint<16..20>;

opaque PGPKey<0..2^24-1>;

struct {
     PGPKeyDescriptorType descriptorType;
     select (descriptorType) {
          case key_fingerprint: PGPKeyFingerprint;
          case key: PGPKey;
     }
} Certificate;
```

### 3.4 Certificate request

The semantics of this message remain the same as in the TLS specification. However the structure of this message has been modified for OpenPGP keys. The PGPCertificateRequest structure will only be used if the negotiated certificate type is OpenPGP.

```
enum {
    rsa_sign(1), dss_sign(2), (255)
} ClientCertificateParamsType;

struct {
    ClientCertificateParamsType certificate_params_types<1..2^8-1>;
} PGPCertificateRequest;
```

The certificate_params_types is a list of accepted client certificate parameter types, sorted in order of the server's preference.

### 3.5 Client certificate

This message is only sent in response to the certificate request message. The client certificate message is sent using the same formatting as the server certificate message and it is also required to present a certificate that matches the negotiated certificate type. If OpenPGP keys have been selected, and no key is available from the client, then a Certificate that contains an empty PGPKey should be sent. The server may respond with a "handshake_failure" fatal alert if client authentication is required. This transaction follows the TLS specification.

### 3.6 Server key exchange

The server key exchange message for OpenPGP keys is identical to the TLS specification.

### 3.7 Certificate verify

The certificate verify message for OpenPGP keys is identical to the TLS specification.

### 3.8 Finished

The finished message for OpenPGP keys is identical to the description in the specification.

## 4. Cipher suites

No new cipher suites are required to use OpenPGP keys. OpenPGP keys can be combined with existing cipher suites defined in TLS [1], except the ones marked as "Exportable". Exportable cipher suites SHOULD NOT be used with OpenPGP keys.

Some additional cipher suites are defined here in order to support algorithms which are defined in OpenPGP [2], and are always available in OpenPGP implementations but are not present in TLS [1].

```
CipherSuite TLS_DHE_DSS_WITH_3DES_EDE_CBC_RMD    = { 0x00, 0x72 };

CipherSuite TLS_DHE_DSS_WITH_AES_128_CBC_RMD     = { 0x00, 0x73 };

CipherSuite TLS_DHE_DSS_WITH_AES_256_CBC_RMD     = { 0x00, 0x74 };

CipherSuite TLS_DHE_RSA_WITH_3DES_EDE_CBC_RMD    = { 0x00, 0x77 };

CipherSuite TLS_DHE_RSA_WITH_AES_128_CBC_RMD     = { 0x00, 0x78 };

CipherSuite TLS_DHE_RSA_WITH_AES_256_CBC_RMD     = { 0x00, 0x79 };

CipherSuite TLS_RSA_WITH_3DES_EDE_CBC_RMD        = { 0x00, 0x7C };

CipherSuite TLS_RSA_WITH_AES_128_CBC_RMD         = { 0x00, 0x7D };

CipherSuite TLS_RSA_WITH_AES_256_CBC_RMD         = { 0x00, 0x7E };
```

All of the above cipher suites use either the AES [5] and 3DES block ciphers in CBC mode. The choice of hash is the RIPEMD-160 [4] algorithm. Implementations are not required to support the above cipher suites.

## 5. Internationalization Considerations

All the methods defined in this document are represented as machine readable structures. As such issues of human internationalization and localization are not introduced.

## 6. Security Considerations

As with X.509 ASN.1 formatted keys, OpenPGP keys need specialized parsers. Care must be taken to make those parsers safe against maliciously modified keys, that may crash or modify the application's memory.

Security considerations about the use of the web of trust or the verification procedure are outside the scope of this document, since they are considered a local policy matter.

## Normative References

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

[2] Callas, J., Donnerhacke, L., Finey, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[3] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and T. Wright, "TLS Extensions", RFC 3546, June 2003.

[4] Dobbertin, H., Bosselaers, A. and B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD", April 1996.

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

## Informative References

[6] Housley, R., Ford, W., Polk, W. and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[7] "Recommendation X.509: The Directory - Authentication Framework", 1988.

## Author's Address

Nikos Mavroyanopoulos

EMail: nmav@gnutls.org
URI: http://www.gnutls.org/

## Appendix A. Acknowledgements

The author wishes to thank Werner Koch, David Taylor and Timo Schulz for their suggestions on improving this document.

## Intellectual Property Statement



## Full Copyright Statement

## Acknowledgment

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