Extensible Authentication Protocol Method for Trusted Computing Groups

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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

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

This Internet-Draft will expire on September 4, 2009.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info).

Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

This document describes an Extensible Authentication Protocol (EAP) [RFC3748] method for identity distribution, authentication and session key distribution using the Trusted Computing Group’s (TCG) Trusted Platform Module (TPM). The TPM has been defined by the TCG in order to establish a root of trust and measurements in (consumer) computers. It provides several cryptographic functions and a secure storage for keys and hashes. There is also a TPM specification for mobile devices called Mobile Trusted Module (MTM), which can also be used for EAP-TPM. This new EAP method allows network authentication, which also supports user anonymity, the usage of different user identities for the authentication with different network operators, result indication, and a fast re-authentication.

Table of Contents

1. Introduction .................................................. 3
2. IANA Considerations ........................................... 3
3. Terms and Abbreviations ...................................... 3
4. Motivation ....................................................... 4
5. TPM Certificates ............................................... 4
6. Authentication ................................................... 6
   6.1. Authentication without Zero-Configuration ............... 7
   6.2. Authentication with Zero-Configuration ................. 9
7. Failure, Fragmentation and Fast Re-Authentication .......... 12
8. Key Derivation .................................................. 12
9. Security Considerations ...................................... 12
10. Acknowledgements ............................................. 12
11. Normative References ........................................ 13
Authors’ Addresses ................................................. 13
1. Introduction

This document specifies a new Extensible Authentication Protocol (EAP) [RFC3748] method based on Trusted Platform Modules (TPMs). TPMs are hardware chips attached to the motherboard of the majority of newly shipped computers. They provide small secure storage, cryptographic functions and a root of trust and measurement. In addition to TPMs there are also Mobile Trusted Modules (MTMs) that provide a subset of the TPMs functionality and are meant to be built into mobile handsets like mobile phones.

TPMs/MTMs can be identified uniquely all over the world and may obtain an identity certificate that proofs that it comes from a genuine TPM/MTM. Therefore, TPMs/MTMs are perfectly suited for certificate based authentication schemes. The only problem is the fact that identity certificates cannot be taken for TLS protocols. In order to overcome this issue, EAP-TPM defines new certificate types as will be shown in Section 5.

EAP-TPM provides support for different user identities which allows the user to hide its original identity at the authenticator as requested in [RFC4017]. Furthermore, it provides a zero-configuration mode where the user does not need to request any identity before authenticating to an EAP-TPM secured authenticator.

EAP-TPM should be understood as a more comfortable but not less secure EAP-TLS.

2. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

3. Terms and Abbreviations

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 [RFC2119].

Furthermore, this document uses the following terms and abbreviations:

Authenticator
The EAP authentication server, that does the real authentication of the peer.

Client

EAP-Peer - the one who wants to be authenticated.

MTM

Mobile Trusted Module as specified in [MTMSpec]

TPM

Trusted Platform Module as specified in [TCGMainSpec]

User

EAP-Peer - the one who wants to be authenticated.

4. Motivation

EAP-TPM has the goal to make a secure authentication protocol like EAP-TLS more userfriendly without weakening its security. EAP-TLS [RFC5216] as it is provides mutual authentication but requires the client to request its own valid X.509 certificates. Two problems arise with that approach: First of all, the user has to be capable to request a certificate, which is a non-trivial task; second the user has to know the acceptable Certificate Authorities (CAs) in advance which is a strong constraint for a real world setup. Therefore, the motivation for EAP-TPM was to develop an EAP method that is as secure, scalable, and automatable as EAP-TLS and comfortable in its usage for a naive "normal" user.

5. TPM Certificates

The process of retrieving TPM certificates starts with retrieving an identity certificate as specified in the TCG Main Specification [TCGMainSpec]:

1. The TPM has to create a new identity certificate request using TPM_MakeIdentity, which generates a new identity key that has to be signed by a Privacy CA.

2. The Trusted Subsystem (TSS) has to collect all the information needed by the Privacy CA to certify the formerly created identity key. According to the TPM Main Specification, that function is
called TSS_CollateIdentityRequest.

3. This request will be sent to the Privacy CA, which verifies all the data and certifies the key and replies with an identity certificate.

4. The TPM will now activate the new identity using TPM_ActivateIdentity.

5. Finally, the TSS has to retrieve a plain text copy of the new identity certificate using TSS_RecoverTPMIdentity.

The detailed process is described in [TCGMainSpec] and not part of this document.

According to [TCGMainSpec] those certificates are special purpose certificate and not meant to be usable for TLS purposes. In fact, they are restricted to SHA-1 signing and MUST have the CA:false constraint. However, the TSS provides a method to certify new keys using an identity key. This method is called TPM_CertifyKey and returns a TCPA_CERTIFY_INFO structure:

typedef struct tdTPM_CERTIFY_INFO {
    TPM_STRUCT_VER version;
    TPM_KEY_USAGE keyUsage;
    TPM_KEY_FLAGS keyFlags;
    TPM_AUTH_DATA_USAGE authDataUsage;
    TPM_KEY_PARMS algorithmParms;
    TPM_DIGEST pubkeyDigest;
    TPM_NONCE data;
    BOOL parentPCRStatus;
    UINT32 PCRInfoSize;
    [size_is(pcrInfoSize)] BYTE* PCRInfo;
} TPM_CERTIFY_INFO;

The following new certificate type for TPMs reflects this structure and MUST be used for EAP-TPM.
Certificate ::= SEQUENCE {  
  parentSerialNumber CertificateSerialNumber,  
  pubKey OCTET STRING,  
  tpmCertificate TPMCertificate,  
  tpmSigLen INTEGER,  
  signatureValue BIT STRING  
}

TPMCertificate ::= SEQUENCE {  
  versionMajor OCTET,  
  versionMinor OCTET,  
  versionRevMajor OCTET,  
  versionRevMinor OCTET,  
  keyUsage OCTET STRING,  
  keyFlags OCTET STRING,  
  authDataUsage OCTET,  
  algorithmID OCTET STRING,  
  encScheme OCTET STRING,  
  sigScheme OCTET STRING,  
  parmSize INTEGER,  
  parms [0] OCTET STRING OPTIONAL,  
    --If not present, parmSize MUST be 0--  
  pubkeyDigest OCTET STRING,  
  nonce OCTET STRING,  
  parentPCRStatus BOOLEAN,  
  PCRInfoSize [1] INTEGER OPTIONAL,  
    --If not present, parentPCRStatus MUST be FALSE--  
  PCRInfo [2] OCTET STRING OPTIONAL,  
    --If not present, parentPCRStatus MUST be FALSE--  
}

This certificate will be used as client certificate in EAP-TPM. It MAY be generated silently on the client machine after having requested the identity certificate.

The TPMCertificate provides not only the option to proof the client’s identity, it also allows to send some integrity measurements of the client’s platform within the PCRInfo parameter. Setting this parameter enables network access control in EAP-TPM if the authentication server is also able to analyze and interpret that value.

6. Authentication

Authentication in EAP-TPM can be divided into authentication without zero-configuration, where the user MUST request his certificate(s) before connecting to an EAP-TPM authenticator, and authentication
with zero-configuration, where the user will get a certificate during the authentication process. The first scenario described in Section 6.1 is more suitable for an operator controlled setup with accounting as it allows to register certificates with users, whereas the second scenario described in Section 6.2 is more suitable for corporate environments or environment without accounting in general.

6.1. Authentication without Zero-Configuration

---
| Peer | | Authenticator |
---

EAP-Request/Identity

EAP-Response/Identity
(includes user’s NAI)

EAP-Request/TPM/Start

EAP-Response/TPM/
TPM client_hello

EAP-Request/TPM/
TPM server_hello,
TPM certificate,
[TPM server_key_exchange,]
TPM certificate_request,
TPM server_hello_done

EAP-Response/TPM/
TPM certificate,
TPM client_key_exchange,
TPM certificate_verify,
TPM change_cipher_spec,
TPM finished

EAP-Request/TPM/
TPM change_cipher_spec,
TPM finished

EAP-Response/TPM

EAP-Success

Figure 1: Successful EAP-TPM Authentication With Existing Certificates
Figure 1 shows the full authentication using EAP-TPM in case the peer could be authenticated successfully. The authentication with already existing certificates is very similar to EAP-TLS [RFC5216]:

The authentication will start with the authenticator asking the peer for its identity sending EAP-Request/Identity. The peer will answer with EAP-Response/Identity containing his Network Address Identifier (NAI) [RFC4282]. Afterwards the authenticator sends the EAP-Request/TPM/Start message to indicate the real beginning of EAP-TPM, which will be followed by an EAP-Response/TPM/TPM client_hello sent by the peer. The client_hello message is the same as within EAP-TLS, which means it contains the client’s TLS version number, a random number, a session id (if needed) and a set of cipher suites that are supported by the client. The authenticator then answers with EAP-Request/TPM/TPM server_hello, TPM certificate, TPM server_key_exchange, TPM certificate_request, TPM server_hello_done. Analogue to the client_hello, the server_hello is the same as in EAP-TLS and contains the server’s TLS version number, a random number, a session id (either the session id offered by the client, or a new session id when establishing a new session) and the set of cipher suites supported by the server. The TPM certificate message on the server side is still the same as in EAP-TLS since the server usually works with X.509 certificates. In case the server also uses a TPM certificate, the message will consist of the server’s TPM certificate followed by one or more X.509 certificates that verify the server certificate:

\[
\text{struct } \{ \\
\quad \text{ASN.1TPMCert;} \\
\quad \text{ASN.1Cert } \text{certificate}_\text{list}<1..2^24-1>; \\
\} \text{ Certificate;} \\
\text{opaque } \text{ASN.1TPMCert}<2^12>; \\
\text{opaque } \text{ASN.1Cert}<2^24-1>;
\]

As it is not determined, which certificate the server will send, the client SHOULD support both certificate types, but MUST support X.509 certificates. In case the client only supports X.509 certificates and the server sends a TPM certificate, the authentication procedure MUST stop with a failure. The TPM server_key_exchange, TPM certificate_request and TPM server_hello_done messages are the same as in EAP-TLS. Afterwards it is up to the client to finish the TLS handshake with a EAP-Response/TPM/TPM certificate, TPM certificate_verify, TPM change_cipher_spec, TPM finished. He sends its TPM certificate in Certificate message as described above. All the other messages are the same as in EAP-TLS. The authenticator then ends the TLS handshake with EAP-Request/TPM/TPM change_cipher_spec, TPM finished, which is not special for EAP-TPM, but the same as in EAP-TLS. Finally the EAP authentication is closed.
with EAP-Response/TPM sent by the client and EAP-Success sent by the authenticator.

6.2. Authentication with Zero-Configuration

Section 6.1 requires valid certificates on the client before starting the authentication. This section deals with authentication without existing certificates. The peer tries to authenticate to an authenticator starting as described in Section 6.1. During the authentication, the authenticator has to send a TPM certificate_request message in order to request the peer’s certificate. TLS [RFC5246] allows to include acceptable certificate authority in this certificate_request message:

```
struct{
    CertificateType certificate_types<1..2^8-1>;
    DistinguishedName certificate_authorities<3..2^16-1>;
} certificate_request;
```

In EAP-TPM with zero configuration, the authenticator has to specify acceptable Privacy CAs (PCAs) within the certificate_authorities field in the certificate_request message. After receiving the EAP-Request/TPM, TPM server_hello, TPM certificate, [TPM server_key_exchange,] TPM certificate_request, TPM server_hello_done messages, the peer has to check whether it has a valid certificate from one of the PCAs specified in TPM certificate_request->certificate_authorities or not. In case it has such a certificate, the authentication goes on as shown in figure Figure 1. In case the peer does not possess a valid certificate from one of the acceptable PCAs, it has to answer with EAP-Response/TPM/TPM no_such_certificate, TPM need_certificate, where no_such_certificate is an alert with level warning(1) and description no_certificate(41) [RFC5246]:

```
struct {
    AlertLevel level;
    AlertDescription description;
} no_such_certificate;
```

This message will be followed by TPM need_certificate, which specifies the PCA the peer wants to ask for a certificate:

```
struct {
    opaque privacy_ca<1..2^16-1>;
} need_certificate;
```

The PCA specified in the need_certificate message MUST be one of the PCAs proposed in TPM certificate_request. The authenticator will now
ask the peer for its certificate request message and request the
certificate at the desired PCA on behalf of the peer. He starts the
certificate request process with an EAP-Request/TPM/TPM
request_certificate:

struct {
    ConnectionAllowed allowed;
    opaque privacy_ca<1..2^16-1>;
} request_certificate

where
enum {
    true(1), false(2)
} ConnectionAllowed

The peer MUST check the privacy_ca value. If it does not match the
PCA specified in need_certificate, the peer MUST close the
authentication immediately. If the privacy_ca value matches and
allowed is set to true(1), the peer is allowed to request a new
certificate as shown in figure Figure 2. It sends its
certificate_request described in Section 5 inside EAP-Response/TPM/
TPM certificate_request to the authenticator, which will then request
the peer’s certificate at the PCA. The PCA sends the peer’s
certificate back to the authenticator, who will forward the
certificate to the peer in the EAP-Request/TPM/TPM client_certificate
message. Afterwards, the authentication goes on as shown in figure
Figure 1.

<table>
<thead>
<tr>
<th>Peer</th>
<th>----------------</th>
<th>---------------</th>
</tr>
</thead>
</table>
|EAP-Response/Identity
(incl. user’s NAI)|----------------|
<table>
<thead>
<tr>
<th>EAP-Request/TPM/Start</th>
</tr>
</thead>
</table>
|EAP-Response/TPM/TPM
client_hello|
|---------------------|
|EAP-Request/TPM/TPM
server_hello,
TPM certificate, |
[TPM server_key_exchange,]
TPM certificate_request, |
TPM server_hello_done|

<-------------------

EAP-Response/TPM/
TPM no_such_certificate,
TPM need_certificate
------------------------------------->
EAP-Request/TPM/
TPM request_certificate
<br><br>
EAP-Response/TPM/
TPM certificate_request
------------------------------------->
certificate_request
<br><br>
EAP-Response/TPM/
TPM client_certificate
<br><br>
--------------
<p>| Generate TPM |</p>
<table>
<thead>
<tr>
<th>certificate</th>
</tr>
</thead>
</table>

EAP-Response/TPM/
TPM certificate,
TPM client_key_exchange,
TPM certificate_verify,
TPM change_cipher_spec,
TPM finished
------------------------------------->
EAP-Request/TPM/
TPM change_cipher_spec,
TPM finished
<br><br>
EAP-Response/TPM
------------------------------------->
EAP Success
<br><br>
Figure 2: Successful EAP-TPM Authentication With Zero Configuration

After having requested the identity certificate, the peer has to generate a TPM certificate as described in Section 5. Depending on the peer’s device, that might take some time and causing a TLS connection timeout. That means, the peer MUST start a new attempt to authenticate after having finished its TPM certificate, but this second try will succeed as the peer now has a valid and acceptable certificate.
7. Failure, Fragmentation and Fast Re-Authentication

Failed authentication requests, fragmentation and fast re-authentication are handled in exactly the same way as in EAP-TLS [RFC5216].

8. Key Derivation

Key derivation in EAP-TPM occurs similar to key derivation in EAP-TLS [RFC5216]. The encryption keys are calculated using a pseudo random function (PRF) that takes the master secret obtained during the TLS handshake and a random number which is the concatenation out of the random value in client_hello and server_hello as argument. The initialisation vector (IV) which may be used for symmetric encryption will be calculated out of a PRF using an empty string and the random number mentioned above as argument.

9. Security Considerations

EAP-TPM as described in this document fulfills the mandatory and recommended requirements for wireless LANs specified in [RFC4017]. The mandatory criteria "Generation of symmetric keying material", "Key strength", "Shared state equivalence", "Resistance to dictionary attacks", "Protection against man-in-the-middle attacks" and "Protected cipher suite negotiation" are fulfilled like in EAP-TLS since there is no difference to EAP-TLS regarding those points. The "Mutual authentication support" is also fulfilled by definition since EAP-TPM is only made for mutual authentication. The recommended requirements of [RFC4017] are also fulfilled: "Fragmentation" is provided as in EAP-TLS and "End-user identity hiding" is provided by the fact that the user may use different identities for every authentication.

Furthermore, the request of an identity certificate MUST be acknowledged by the user in order to ensure that he is informed about the identities of his device.

10. Acknowledgements

The authors want to thank Bernhard Hoeneisen and Alan DeKok for their discussions regarding administrative matters and for the comments and suggestions about how to improve the specification.
11. Normative References


Authors’ Addresses

Carolin Latze
University of Fribourg
Boulevard de Perolles 90
Fribourg, FR 1700
Switzerland

Email: carolin.latze@unifr.ch
Ulrich Ultes-Nitsche
University of Fribourg
Boulevard de Perolles 90
Fribourg, FR  1700
Switzerland

Email: uun@unifr.ch

Florian Baumgartner
Swisscom Schweiz AG
Ostermundigenstrasse 93
Bern, BE  3006
Switzerland

Email: florian.baumgartner@swisscom.com