Network Working Group
Internet-Draft
Intended status: Experimental
Expires: February 7, 2011

C. Latze
U. Ultes-Nitsche
University of Fribourg
F. Baumgartner
89grad GmbH

August 6, 2010

Transport Layer Security (TLS) Extensions for the Trusted Platform Module (TPM)
draft-latze-tls-tpm-extns-02

Abstract

Trusted Platform Modules (TPMs) become more and more widespread in modern desktop and laptop computers and provide secure storage and cryptographic functions. As one nice feature of TPMs is that they can be identified uniquely, they provide a good base for device authentication in protocols like TLS. This document specifies a TLS extension that allows to use TPM certified keys with TLS in order to allow for a secure and comfortable device authentication in TLS.

Table of Contents

1. Introduction

This document aims at specifying a new TLS extension that allows to use TPM certified keys directly with TLS [RFC5246]. TPM is short for Trusted Platform Module and describes a trusted module that provides secure storage and some cryptographic functions and has been specified in [TPMMainP1].

TPMs come with the possibility to create so called attestation identity keys (AIKs) that will be signed by a certain certificate authority called Privacy CA resulting in an AIK certificate. The process of signing by the Privacy CA includes verifying that those keys have been generated by a genuine TPM. The private part of the AIK never leaves the TPM, which means that the AIK certificates prove that they belong to a genuine, given TPM. The AIK certificate is a valid X.509 certificate, which can be verified using the standard X.509 verification functions. So if those certificates are used in authentication protocols one can authenticate devices (using their TPM). But, unfortunatly, those keys have some restrictions which makes it impossible to use them in TLS directly:

1. They are restricted to SHA-1 signing, which is a problem to TLS prior to 1.2

2. They only sign data that originates in the TPM, which means that in order to use them with TLS, every data that has to be signed has to originate in the TPM, which would require a lot of changes to the TLS handshake.

In order to be able to use something like the AIKs in authentication, the TPM standard propose to define new keys (with the property that the private part never leaves the TPM) and sign them using the AIK. That signing process is defined in [TPMMainP3] and results in a special structure called TCPA_CERTIFY_INFO. That structure proves that a certified key has been signed by a valid AIK, which itself can be verified using standard X.509 verification techniques. Unfortunatly, TCPA_CERTIFY_INFO does not have the same format like X.509. The certified key could be used in TLS directly, but verifying that this key belongs to a given TPM remains an open question. There are several solutions that all rely on the fact, that the AIK certificate proves that this certificate key belongs to a genuine, given TPM:

1. Make it an X.509 certificate signed by the AIK, which would result in a complete X.509 chain, that is easy to verify. This is not possible since the AIK certificate carries the CA:false constraint.

2. Make it an X.509 proxy certificate of the AIK. However that is not possible since the subject of the AIK is NULL.

Those would be the easy possibilities, but since they do not work, the TCG defined a new X.509 extension to carry the TCPA_CERTIFY_INFO structure called SKAE [SKAE]. Again two possibilities arise to make use of that extension:

1. Generate a new X.509 certificate request around the certified key, include the SKAE extension, and send the request to a CA to sign it. This results in a standard X.509 certificate that can be used for TLS. But if the server wants to ensure that this certificate is bound (which means the private key never leaves the TPM) to a certain genuine TPM, he has to verify the SKAE extension too. That means he has to get the AIK certificate too, which is not part of the chain. The SKAE extension carries a hint where to find that certificate. There has to be some kind of storage for the AIK certificates, that either sends the AIK certificates to all the servers or receives SKAE extensions from servers to validate them. That approach will result in scalability issues.

2. Generate an X.509 self-signed certificate around the certified key including the SKAE extension, use it in the handshake and send the AIK certificate that verifies the SKAE extension within the supplemental data handshake message. This is the approach specified in this document.

This document will first explain how the AIK certificates are retrieved in order to allow the reader to fully understand the idea behind AIK certificates. Afterwards the TLS TPM extension as well as the required supplemental data handshake message type will be described.

2. 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, the document uses the following terms and abbreviations:

AIK - Attestation Identity Key

CA - Certificate Authority

Entity - One of the communication end points, be it either client or

server.

PCA - Privacy CA

TLS - Transport Layer Security

TPM - Trusted Platform Module

3. Certification Process

This section describes the process of creating and requesting all the certificates necessary to be used with the TLS TPM extension specified in the next sections.

First of all an TPM equipped entity has to request its AIK as specified in [TPMMainP1]. Afterwards, a new non-migratable key has to be created and certified using the AIK. The details about the certification process can be found in [TPMMainP3]. Some details will be repeated here for convenience.

The certificate is done using either TPM_CertifyKey or TPM_CertifyKey2 (for details about when to use which function, have a look at [TPMMainP3]). Depending on the key properties, those functions result in either TCPA_CERTIFY_INFO or TCPA_CERTIFY_INFO2 structure, whereas the first is compatible to the TPM 1.1 standard [TCGMainSpec].

Now that the entity has an AIK and a certified key structure, a self-signed certificate around the certified key has to be created. As the TCPA_CERTIFY_INFO (or TCPA_CERTIFY_INFO2) structure is needed to verify the binding between AIK and the certified key, that self-signed certificate has to include the Subject Key Attestation Evidence (SKAE) extension defined in [SKAE]. The SKAE extension is an X.509 extension that has been defined to carry the certify info structure returned by TPM_CertifyKey (or TPM_CertifyKey2).

The self-signed certificate will be sent during the TLS handshake in the Certificate message whereas the AIK MUST be announced with the TLS TPM extension type and sent in the supplemental data handshake message.

4. TLS TPM Extension Type

The general TLS extension format has been defined in [RFC5246] and will be repeated here for convenience:

```
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
}
```

The new extension types for TPM enabled entities are called client_aik and server_aik:

```
enum {
   client_aik(TBD), server_aik(TBD), (65535)
} ExtensionType
```

This extensions MAY be used in full handshakes as well as in session resumption handshakes. Although the latter does not require a certificate exchange it might happen that the server refuses to accept a resumed session and runs a full handshake instead. In order to be able to do that without interruption, the extensions SHOULD be included also in the session resumption handshake.

The extension includes the certify info type the client is able to create and verify:

```
enum {
   tpm_certify_info(0), tpm_certify_info2(1), (255)
} CertifyInfoType
```

The client includes client_aik in order to indicate that he wants to use a self-signed certified key during the handshake and send the AIK in the supplemental data handshake message. If the server receives client_aik, he MUST respond with same client_aik - possibly removing unsupported certify info types or omit the extension in case it is not supported by the server.

In case the client wants to authenticate the server also using TPM certified keys, he MUST include server_aik in its extended hello message.The server_aik contains all the certify info types the client is able to verify. If the server receives server_aik and accepts it, he MUST respond with the same server_aik - possibly removing certify info types he cannot create. Otherwise the server omits server_aik.

5. TLS TPM Supplemental Data Handshake Message

The TLS supplemental data handshake message as defined in [RFC4680] allows to send additional application data during the TLS handshake if it has been announced in a TLS extension.

This document defines a new supplemental data type:

```
enum {
   aik_data(TBD), (65535)
}

with
struct {
    SupplementalDataType supplemental_data_type;
    select(SupplementalDataType) {
       case aik_data: AikData;
    }
} SupplementalData

and
opaque ASN.1Cert<2^24-1>;

struct {
   ASN.1Cert certificate_list<0..2^24-1>;
} AikData;
```

AikData carries the entity's AIK chain.

6. TLS Handshake Using The TPM Extensions

Figure Figure 1 shows the full TLS handshake with a TPM equipped client:

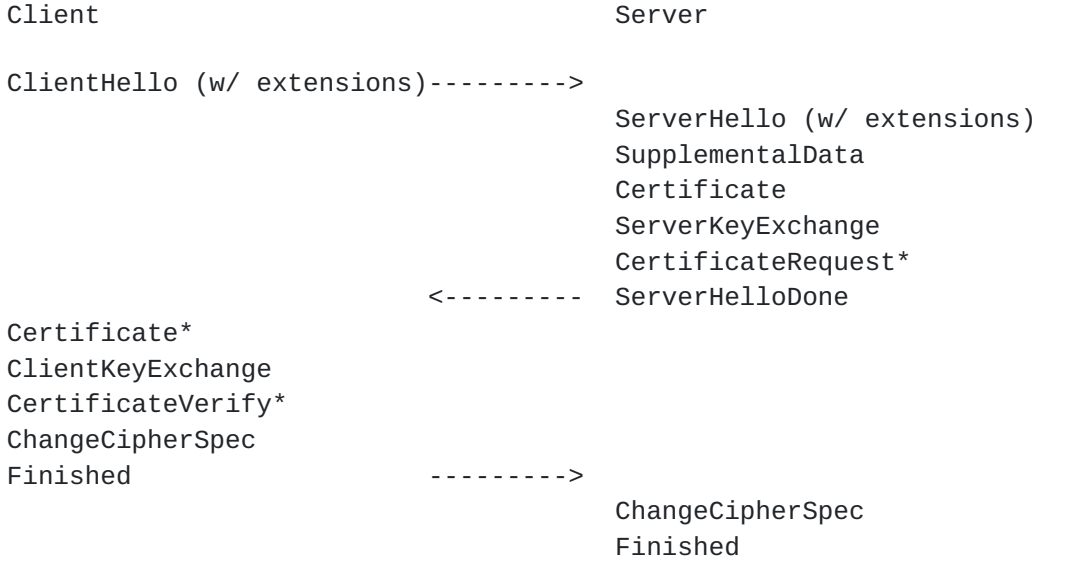
```
Client                                  Server

ClientHello (w/ extensions)--------->
                                        ServerHello (w/ extensions)
                                        Certificate
                                        ServerKeyExchange
                                        CertificateRequest*
                           <---------   ServerHelloDone
SupplementalData
Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished                   --------->
                                        ChangeCipherSpec
                                        Finished
```

* indicates optional or situation dependant messages

Figure 1: Full TLS Handshake With a TPM Equipped Client

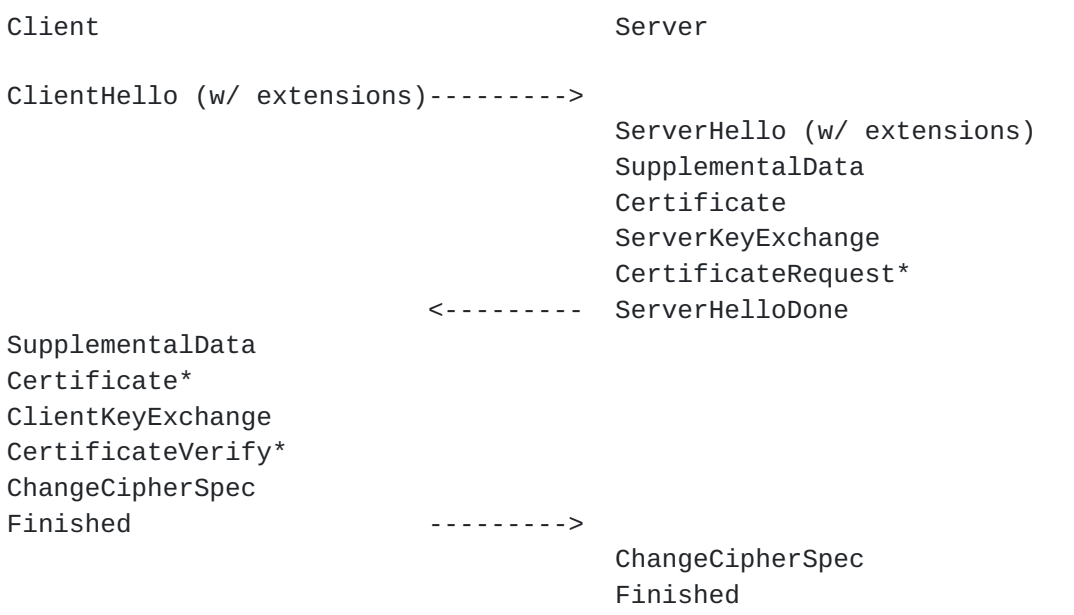
Figure Figure 2 shows the full handshake with a TPM equipped server:



* indicates optional or situation dependant messages

Figure 2: Full TLS Handshake With a TPM Equipped Server

Finally, figure Figure 3 shows the TLS handshakes if both sides make use of certified keys:



* indicates optional or situation dependant messages

Figure 3: Full TLS Handshake With TPM Equipped Client and Server

The authentication of either client or server is done by verifying the self-signed certificate as well as by verifying the binding between the AIK and the certified key in order to ensure that the key used is really protected by a given TPM. In order to verify the binding, the SKAE extension of the self-signed certificate has to be evaluated using the AIK.

There is no need for additional TLS alerts since all the existing certificate related alerts cover possible problems during the entity verification.

7. IANA Considerations

This document makes the following IANA requests:

1. A new registry for certify info types needs to be maintained by IANA. The first two types include tpm_certify_info(0) and tpm_certify_info2(1). Certify info types with values in the inclusive range of 0 to 63 (decimal) are assigned using RFC 5226 [RFC5226] Standards Action, whereas values from the inclusive range of 64 to 223 (decimal) are using RFC 2434 Specification Required. Values in the inclusive range of 224 to 255 (decimal) are reserved for RFC 2434 Private Use.

2. The values client_aik(TBD) and server_aik(TBD) are assigned from TLS Extension Type Registry [RFC5246].

3. The value aik_data(TBD) is assigned from TLS Supplemental Data Type registry [RFC4680].

8. Security Considerations

If an entity certified several keys with the same AIK, somebody who has the AIK and all of the certified keys is able to track that identity. Therefore, the AIK might be seen as sensitive information forcing an implementation to use the double handshake technique. The first handshake requires one or both entities to accept the self-signed certificate since the binding can only be verified during the second protected handhake.

9. Acknowledgements

The basic idea to use the supplemental data handshake message to supply the AIK was supplied by Sam Hartmann.

Furthermore the authors want to thank Bernie Hoeneisen, Kyle Hamilton, Joe Salowey, and Nikos Mavrogiannopoulos for their valuable input, review, and discussions.

10. Normative References

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

[RFC4680] Santesson, S., "TLS Handshake Message for Supplemental Data", RFC 4680, October 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[SKAE] The Trusted Computing Group, "Subject Key Attestation Evidence Extension", TCG Infrastructure Workinggroup, June 2005.

[TCGMainSpec] The Trusted Computing Group, "TCPA Main Specification Version 1.1b", TCG Trusted Platform Module, February 2002.

[TPMMainP1] The Trusted Computing Group, "TPM Main Part 1 Design Principles", TCG Trusted Platform Module, July 2007.

[TPMMainP3] The Trusted Computing Group, "TPM Main Part 3 Commands", TCG Trusted Platform Module, October 2006.

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
89grad GmbH
Holligenstrasse 101
Bern, BE 3008
Switzerland

Email: florian.baumgartner@89grad.ch