Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 20, 2020

M. Msahli, Ed.
Telecom Paris
N. Cam-Winget, Ed.
Cisco
A. Serhrouchni, Ed.
H. Labiod , Ed.
Telecom Paris
W. Whyte, Ed.
Qualcomm

November 17, 2019

TLS Authentication using IEEE 1609.2 certificate
draft-msahli-ise-ieee1609-03

Abstract

The IEEE and ETSI have specified end-entity certificates. This docment defines an experimental change to TLS to support IEEE/ETSI certificate types to authenticate TLS entities.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The TLS protocol [RFC8446] [RFC5246] uses X.509 certificates and Raw Public Key in order to authenticate servers and clients. This document describes an experimental extension following the [RFC7250] to support use of the certificate format specified by the IEEE in [IEEE1609.2] and profiled by the European Telecommunications Standards Institute (ETSI) in [TS103097]. These standards specify secure communications in vehicular environments. These certificates are referred to in this document as Intelligent Transportation Systems (ITS) Certificates.

The certificate types are optimized for bandwidth and processing time to support delay-sensitive applications, and also to provide both authentication and authorization information to enable fast access control decisions in ad hoc networks such as are found in Intelligent Transportation Systems (ITS). The standards specify different types

of certificate to support a full Public Key Infrastructure (PKI) specification; the certificates to be used in this context are end-entity certificates, i.e. certificates that have the IEEE 1609.2 appPermissions field present.

Use of ITS certificates is becoming widespread in the ITS setting. ITS communications in practice make heavy use of 10 MHz channels with a typical throughput of 6 Mbps. (The 802.11OCB modulation that gives this throughput is not the one that gives the highest throughput, but it provides for a robust signal over a range up to 300-500 m, which is the "sweet spot" for communications range for ITS operations like collision avoidance). The compact nature of ITS certificates as opposed to X.509 certificates makes them appropriate for this setting.

The ITS certificates are also suited to the M2M ad hoc network setting, because their direct encoding of permissions (see Security Considerations, section 7.4) allows a receiver to make an immediate accept/deny decision about an incoming message without having to refer to a remote identity and access management server. The EU has committed to the use of ITS certificates in Cooperative Intelligent Transportation Systems deployments. A multi-year project developed a certificate policy for the use of ITS certificates, including a specification of how different root certificates can be trusted across the system (hosted at https://ec.europa.eu/transport/themes/its/c-its_en, direct link at https://ec.europa.eu/transport/sites/transport/files/c-its_certificate_policy_release_1.pdf).

The EU has committed funding for the first five years of operation of the top-level Trust List Manager entity, enabling organizations such as motor vehicle OEMs and national road authorities to create root CAs and have them trusted. In the US, the US Department of Transportation (USDOT) published a proposed regulation, which at the time of writing is active though not rapidly progressing, which would have required all light vehicles in the US to implement V2X communications including the use of ITS certificates (available from https://www.federalregister.gov/documents/2017/01/12/2016-31059/federal-motor-vehicle-safety-standards-v2v-communications). As of 2019, ITS deployments across the US, Europe and Australia were using ITS certificates. Volkswagen have committed to deploying V2X next year using ITS certificates. New York, Tampa and Wyoming are deploying traffic management systems using ITS certificates. GM deployed V2X in their Cadillac CTSes using ITS certificates.

ITS certificates are also used in a number of standards that build on top of the foundational IEEE and ETSI standards, particularly the SAE J2945/x series of standards for applications and ISO 21177, which

builds a framework for exchanging multiple authentication tokens on top of the TLS variant specified in this document.

1.1. Experiment Overview

This document describes an experimental extension to the TLS security model. It uses a form of certificate that has not previously been used in the Internet. Systems using this Experimental approach are segregated from system using standard TLS by the use of a new Certificate Type value, reserved through IANA (see Section 9). An implementation of TLS that is not involved in the Experiment will not recognise this new Certificate Type and will not be able to interact with an Experimental implementation: TLS sessions will fail to be established.

This extension has been encouraged by stakeholders in the Cooperative ITS community in order to support the ITS use cases deployment and it is anticipated that its use will be widespread.

2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Extension Overview

For TLS 1.2 [RFC5246], the "extension_data" field SHALL follow the [RFC7250]. In case of TLS 1.3, the "extension_data" field SHALL contain a list of supported certificate types proposed by the client as provided in the figure below:

```
     /* Managed by IANA */
      enum {
          X509(0),
          RawPublicKey(2),
          1609Dot2(3),
          (255)
      } CertificateType;

      struct {
          select (certificate_type) {

              /* certificate type defined in this document.*/
               case 1609Dot2:
               opaque cert_data<1..2^24-1>;

               /* RawPublicKey defined in RFC 7250*/
              case RawPublicKey:
              opaque ASN.1_subjectPublicKeyInfo<1..2^24-1>;

              /* X.509 certificate defined in RFC 5246*/
              case X.509:
              opaque cert_data<1..2^24-1>;

               };

             Extension extensions<0..2^16-1>;
         } CertificateEntry;
```

In case where the TLS server accepts the described extension, it selects one of the certificate types. Note that a server MAY authenticate the client using other authentication methods.

4. TLS Client and Server Handshake

The "client_certificate_type" and "server_certificate_type" extensions MUST be sent in handshake phase as illustrated in figure 1 below.

```
  Client                                           Server

 Key  ^ ClientHello
 Exch | + server_certificate_type*
      | + client_certificate_type*
      | + key_share*
      v + signature_algorithms*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  v Exch
                                        {EncryptedExtensions}  ^ Server
                                   {+ server_certificate_type*}| Params
                                   {+ client_certificate_type*}|
                                        {CertificateRequest*}  v
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                               <-------   [Application Data*]
      ^ {Certificate*}
 Auth | {CertificateVerify*}
      v {Finished}             -------->
        [Application Data]     <------->   [Application Data]

               +  Indicates noteworthy extensions sent in the
                  previously noted message.

               *  Indicates optional or situation-dependent
                  messages/extensions that are not always sent.

               {} Indicates messages protected using keys
                  derived from a [sender]_handshake_traffic_secret.

               [] Indicates messages protected using keys
                  derived from [sender]_application_traffic_secret_N.
```

Figure 1: Message Flow with certificate type extension for Full TLS 1.3 Handshake

In case of TLS 1.3 and in order to negotiate the support of IEEE 1609.2 or ETSI TS 103097 certificate-based authentication, the clients and the servers MAY include the extension of type "client_certificate_type" and "server_certificate_type" in the extended Client Hello and "EncryptedExtensions". In case of TLS 1.2, used extensions are in Client Hello and Server Hello.

4.1. Client Hello

In order to indicate the support of IEEE 1609.2 or ETSI TS 103097 certificates, a client MUST include an extension of type "client_certificate_type" or "server_certificate_type" in the extended Client Hello message as described in Section 4.1.2 of TLS 1.3 [RFC8446].

The extension 'client_certificate_type' sent in the Client Hello MAY carry a list of supported certificate types, sorted by client preference. It is a list in the case where the client supports multiple certificate types.

In both TLS 1.2 and 1.3, the rules if client Certificate and CertificateVerify messages appear is as follows:

- Client Certificate message is present if and only if server sent a CertificateRequest message.

- Client CertificateVerify message is present if and only if the Client Certificate message is present and contains non-empty certificate list.

All implementations SHOULD be prepared to handle extraneous certificates and arbitrary orderings from any TLS version, with the exception of the end-entity certificate which MUST be first.

4.2. Server Hello

When the server receives the Client Hello containing the client_certificate_type extension and/or the server_certificate_type extension, the following options are possible:

- The server supports the extension described in this document. It selects a certificate type from the client_certificate_type field in the extended Client Hello and SHALL take into account the client authentication list priority.

- The server does not support any of the proposed certificate type and terminates the session with a fatal alert of type "unsupported_certificate".

- The server does not support the extension defined in this document. In this case, the server returns the Server Hello without the extensions defined in this document.

- The server supports the extension defined in this document, but it does not have any certificate type in common with the client.

Then, the server terminates the session with a fatal alert of type "unsupported_certificate".

- The server supports the extensions defined in this document and has at least one certificate type in common with the client. In this case, the server MAY include the client_certificate_type extension in the Server Hello for TLS 1.2 or in Encrypted Extension for TLS 1.3. Then, the server requests a certificate from the client (via the certificate_request message)

The TLS client or server public keys can be obtained from an online repository. In fact, the repository is used to retrive the certificate chain. All PKI requests and responses are indicated in ETSI[ETSI102941].

5. Certificate Verification

Verification of an IEEE 1609.2/ ETSI TS 103097 certificates or certificate chain is described in section 5.1 of [IEEE1609.2]. In the case of TLS 1.3 and when the certificate_type is 1609Dot2, the CertificateVerify contents and processing are different than for the CertificateVerify message specified for other values of certificate_type in [RFC8446]. In this case, the CertificateVerify message contains a Canonical Octet Encoding Rules [ITU-TX.696] -encoded IEEE1609Dot2Data of type signed as specified in [IEEE1609.2], [IEEE1609.2b], where:

Payload contains an extDataHash containing the SHA-256 hash of the data and the signature is calculated over. This is identical to the data, the signature is calculated over in standard TLS, which is reproduced below for clarity.

Psid indicates the application activity that the certificate is authorizing.

generationTime is the time at which the data structure was generated.

PduFunctionalType (as specified in [IEEE1609.2b]) is present and is set equal to tlsHandshake (1).

All other fields in the headerInfo are omitted. The certificate appPermissions field shall be present and shall permit (as defined in IEEE1609.2) signing of PDUs with the PSID indicated in the HeaderInfo of the SignedData. If the application specification for that PSID requires Service Specific Permissions (SSP) for signing a pduFunctionalType of tlsHandshake, this SSP shall also be present.

For more details on the use of PSID and SSP, see [IEEE1609.2] clauses 5.1.1 and 5.2.3.3.3. All other fields in the headerInfo are omitted.

The certificate appPermissions field shall be present and shall permit (as defined in IEEE 1609.2) signing of PDUs with the PSID indicated in the HeaderInfo of the SignedData. If the application specification for that PSID requires Service Specific Permissions (SSP) for signing a pduFunctionalType of tlsHandshake, this SSP shall also be present.

The message input to the signature calculation is the usual message input for TLS 1.3, as specified in [RFC8446] section 4.4.3, consisting of pad, context string, separator and content, where content is Transcript- Hash(Handshake Context, Certificate).

The signature and verification are carried out as specified in [IEEE1609.2].

The message input to the signature calculation is the usual message input for TLS 1.3, as specified in [RFC8446] section 4.4.3, consisting of pad, context string, separator and content, where content is Transcript- Hash(Handshake Context, Certificate).

The signature and verification are carried out as specified in [IEEE1609.2].

6. Examples

Some of exchanged messages examples are illustrated in Figures 2 and 3.

6.1. TLS Server and TLS Client use the 1609Dot2 Certificate

This section shows an example where the TLS client as well as the TLS server use the IEEE 1609.2 certificate. In consequence, both the server and the client populate the client_certificate_type and server_certificate_type with extension IEEE 1609.2 certificates as mentioned in figure 2.

```
      Client                                               Server

   ClientHello,
   client_certificate_type=1609Dot2,
   server_certificate_type=1609Dot2,   -------->     ServerHello,
                                            {EncryptedExtensions}
                               {client_certificate_type=1609Dot2}
                               {server_certificate_type=1609Dot2}
                                             {CertificateRequest}
                                                    {Certificate}
                                              {CertificateVerify}
                                                       {Finished}
     {Certificate}           <-------          [Application Data]
     {CertificateVerify}
     {Finished}              -------->
     [Application Data]      <------->         [Application Data]
```

Figure 2: TLS Client and TLS Server use the IEEE 1609.2 certificate

6.2. TLS Client uses the IEEE 1609.2 certificate and TLS Server uses the X.509 certificate

This example shows the TLS authentication, where the TLS Client populates the server_certificate_type extension with the X.509 certificate and Raw Public Key type as presented in figure 3. the client indicates its ability to receive and to validate an X.509 certificate from the server. The server chooses the X.509 certificate to make its authentication with the Client.

```
   Client                                               Server
   ClientHello,
   client_certificate_type=(1609Dot2),
   server_certificate_type=(1609Dot2,
   X509,RawPublicKey),         ----------->         ServerHello,
                                           {EncryptedExtensions}
                              {client_certificate_type=1609Dot2}
                                  {server_certificate_type=X509}
                                                   {Certificate}
                                             {CertificateVerify}
                                                      {Finished}
                               <---------     [Application Data]
   {Finished}                  --------->
   [Application Data]          <-------->     [Application Data]
```

Figure 3: TLS Client uses the IEEE 1609.2 certificate and TLS Server uses the X.509 certificate

7. Security Considerations

This section provides an overview of the basic security considerations which need to be taken into account before implementing the necessary security mechanisms. The security considerations described throughout [RFC8446] regarding the supported groups and signature algorithms apply here as well.

7.1. Securely Obtaining Certificates from an Online Repository

The certificates used to establish a secure connection may be obtained from an online repository in particular, an online repository may be used to obtain the CA certificates in the chain of either participant in the secure session. ETSI TS 102 941 [ETSI102941] provides a mechanism that can be used to securely obtain ITS certificates.

7.2. Expiry of Certificates

Conventions around certificate lifetime differ between ITS certificates and X.509 certificates, and in particular ITS certificates may be relatively short-lived compared with typical X.509 certificates. A party to a TLS session that accepts ITS certificates MUST check the expiry time in the received ITS certificate and SHOULD terminate a session when the certificate received in the handshake expires. We can consider the TLS renegotiation as specified in [RFC8446] and [RFC5246], but an implementation of proposed extension could favor terminating the session on expiry of the the certificate.

7.3. Algorithms and Cryptographic Strength

All ITS certificates use public-key cryptographic algorithms with an estimated strength of at least 128 bits specifically, Elliptic Curve Cryptography (ECC) based on curves with keys of length 256 bits or longer. An implementation of the techniques specified in this document SHOULD require that if X.509 certificates are used by one of the parties to the session, those certificates are associated with cryptographic algorithms with (pre-quantum-computer) strength of at least 128 bits.

7.4. Interpreting ITS Certificate Permissions

ITS certificates in TLS express the certificate holders permissions using two fields: a Provider Service Identifier (PSID), also known as an ITS Application Identifier (ITS-AID), which identifies a broad set of application activities which provide a context for the certificate holders permissions, and a Service Specific Permissions (SSP) field

associated with the PSID, which identifies which specific application activities the certificate holder is entitled to carry out within the broad set of activities identified by that PSID. For example, SAE [SAEJ29453] uses PSID 0204099 to indicate activities around reporting weather and managing weather response activities, and an SSP that states whether the certificate holder is a Weather Data Management System (WDMS, i.e. a central road manager), an ordinary vehicle, or a vehicle belonging to a managed road maintenance fleet. For more information about PSIDs, see [IEEE160912] and for more information about the development of SSPs, see [SAEJ29455]

The assumption in this document is that a party that accepts ITS certificates will do it in the context of an access control policy that states what PSIDs and SSPs are to be accepted in the handshake, and what activities are permitted within the session based on the PSIDs and SSPs presented in the handshake. [ISO21177] provides a generalization of this where additional certificates may be presented within the context of a TLS session to provide a more complete picture of the permissions of the counterparty within the session, allowing that counterparty to demonstrate its entitlement to a broader range of permissions than those indicated within the single certificate presented within the handshake. An implementation that accepts ITS certificates MUST do so in the context of an access policy of this type.

7.5. Psid and Pdufunctionaltype in Certificateverify

The CertificateVerify message for TLS 1.3 is an Ieee1609Dot2Data of type signed, signed using a ITS certificate. This certificate may include multiple PSIDs. When a CertificateVerify message of this form is used, the HeaderInfo within the Ieee1609Dot2Data MUST have the pduFunctionalType field present and set to tlsHandshake. The background to this requirement is as follows. A ITS certificate may (depending on the definition of the application associated with its PSID(s)) be used to directly sign messages, or to sign TLS CertificateVerify messages, or both. To prevent the possibility that a signature generated in one context could be replayed in a different context i.e., that a message signature could be replayed as a CertificateVerify, or vice versa the pduFunctionalType field provides a statement of intent by the signer as to the intended use of the signed message. If the pduFunctionalType field is absent, the message is a directly signed message for the application and MUST NOT be interpreted as a CertificateVerify. If the pduFunctionalType field is present and set equal to tlsHandshake, the message is a CertificateVerify and MUST NOT be interpreted as a directly signed message for the application.

Note that each PSID is owned by an owning organization that has sole rights to define activities associated with that PSID. If an application specifier wishes to expand activities associated with an existing PSID (for example, to include activities over a secure session such as specified in this document), that application specifier must negotiate with the PSID owner to have that functionality added to the official specification of activities associated with that PSID. For new application activities, PSIDs can be requested via IEEE or via ISO TC 204. In particular, note that there is currently no PSID associated to the extension, although such a PSID could be reserved in future if it were found to be useful.

8. Privacy Considerations

For privacy considerations in a vehicular environment the use of IEEE 1609.2/ETSI TS 103097 certificate is used for many reasons:

In order to address the risk of a personal data leakage, messages exchanged for V2V communications are signed using IEEE 1609.2/ETSI TS 103097 pseudonym certificates

The purpose of these certificates is to provide privacy relying on geographical and/or temporal validity criteria, and minimizing the exchange of private data

9. IANA Considerations

IANA maintains the "Transport Layer Security (TLS) Extensions" registry with a subregistry called "TLS Cetificate Types".

IIANA has previously assigned an entry (value 3) for "1609Dot2" with reference set to draft-tls-certieee1609. IANA is requested to update that entry to reference the RFC number of this document when it is published.

10. Acknowledgements

The authors wish to thank Adrian Farrel , Eric Rescola , Russ Housley and Ilari Liusvaara for their feedback and suggestions on improving this document. Thanks are due to Sean Turner for his valuable and detailed comments. Special thanks to Panos Kampanakis, Jasja Tijink and Bill Lattin for their guidance and support of the draft.

11. Normative References

[ETSI102941]
"ETSI TS 102 941 : Intelligent Transport Systems (ITS); Security; Trust and Privacy Management", 2018.

[IEEE1609.2]
"IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages", 2016.

[IEEE1609.2b]
"IEEE Standard for Wireless Access in Vehicular Environments--Security Services for Applications and Management Messages - Amendment 2--PDU Functional Types and Encryption Key Management", 2019.

[IEEE160912]
"IEEE Standard for Wireless Access in Vehicular Environments Identifier Allocations", December 2016.

[ISO21177]
"Intelligent transport systems -- ITS station security services for secure session establishment and authentication between trusted devices".

[ITU-TX.696]
"Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree", July 2011.

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

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

[RFC7250] Wouters, P., Tschofenig, H., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", June 2014.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", May 2017.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", August 2018.

[SAEJ29453]
"Requirements for V2I Weather Applications".

[SAEJ29455]
"Service Specific Permissions and Security Guidelines for Connected Vehicle Applications".

[TS103097]
"ETSI TS 103 097 : Intelligent Transport Systems (ITS); Security; Security header and certificate formats".

Authors' Addresses

Mounira Msahli (editor)
Telecom Paris
France

EMail: mounira.msahli@telecom-paris.fr

Nancy Cam-Winget (editor)
Cisco
USA

EMail: ncamwing@cisco.com

Ahmed Serhrouchni (editor)
Telecom Paris
France

EMail: ahmed.serhrouchni@telecom-paris.fr

Houda Labiod (editor)
Telecom Paris
France

EMail: houda.labiod@telecom-paris.fr

William Whyte (editor)
Qualcomm
USA

EMail: wwhyte@qti.qualcomm.com