Independent Submission
Request for Comments: 8902
Category: Experimental
ISSN: 2070-1721

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

September 2020

TLS Authentication Using Intelligent Transport System (ITS) Certificates

Abstract

The IEEE and ETSI have specified a type of end-entity certificate. This document defines an experimental change to TLS to support IEEE/ETSI certificate types to authenticate TLS entities.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8902.

Table of Contents

1. Introduction

The TLS protocol [RFC8446] allows the use of X.509 certificates and raw public keys to authenticate servers and clients. This document describes an experimental extension following the procedures laid out by [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 Transport 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 found in Intelligent Transport Systems (ITS). The standards specify different types of certificates 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" 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 machine-to-machine (M2M) ad hoc network setting because their direct encoding of permissions (see 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 Transport 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 original equipment manufacturers (OEMs) and national road authorities to create root certificate authorities (CAs) and have them trusted. In the US, the US Department of Transportation (USDOT) published a proposed regulation, active as of late 2019 though not rapidly progressing, requiring all light vehicles in the US to implement vehicle-to-everything (V2X) communications, including the use of ITS certificates (available at <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 has committed to deploying V2X using ITS certificates. New York, Tampa, and Wyoming are deploying traffic management systems using ITS certificates. GM deployed V2X in the Cadillac CTS, 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 Society of Automobile Engineers (SAE) J2945/x series of standards for applications and ISO 21177 [ISO21177], 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 systems 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 recognize this new certificate type and will not participate in the experiment; TLS sessions will either negotiate the use of existing X.509 certificates or fail to be established.

This extension has been encouraged by stakeholders in the Cooperative ITS community in order to support ITS use-case 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

The TLS extensions "client_certificate_type" and "server_certificate_type" [RFC7250] are used to negotiate the type of Certificate messages used in TLS to authenticate the server and, optionally, the client. Using separate extensions allows for mixed deployments where the client and server can use certificates of different types. It is expected that ITS deployments will see both peers using ITS certificates due to the homogeneity of the ecosystem, but there is no barrier at a technical level that prevents mixed certificate usage. This document defines a new certificate type, 1609Dot2, for usage with TLS 1.3. The updated CertificateType enumeration and corresponding addition to the CertificateEntry structure are shown below. CertificateType values are sent in the "server_certificate_type" and "client_certificate_type" extensions, and the CertificateEntry structures are included in the certificate chain sent in the Certificate message. In the case of TLS 1.3, the "client_certificate_type" SHALL contain a list of supported certificate types proposed by the client as provided in the figure below:

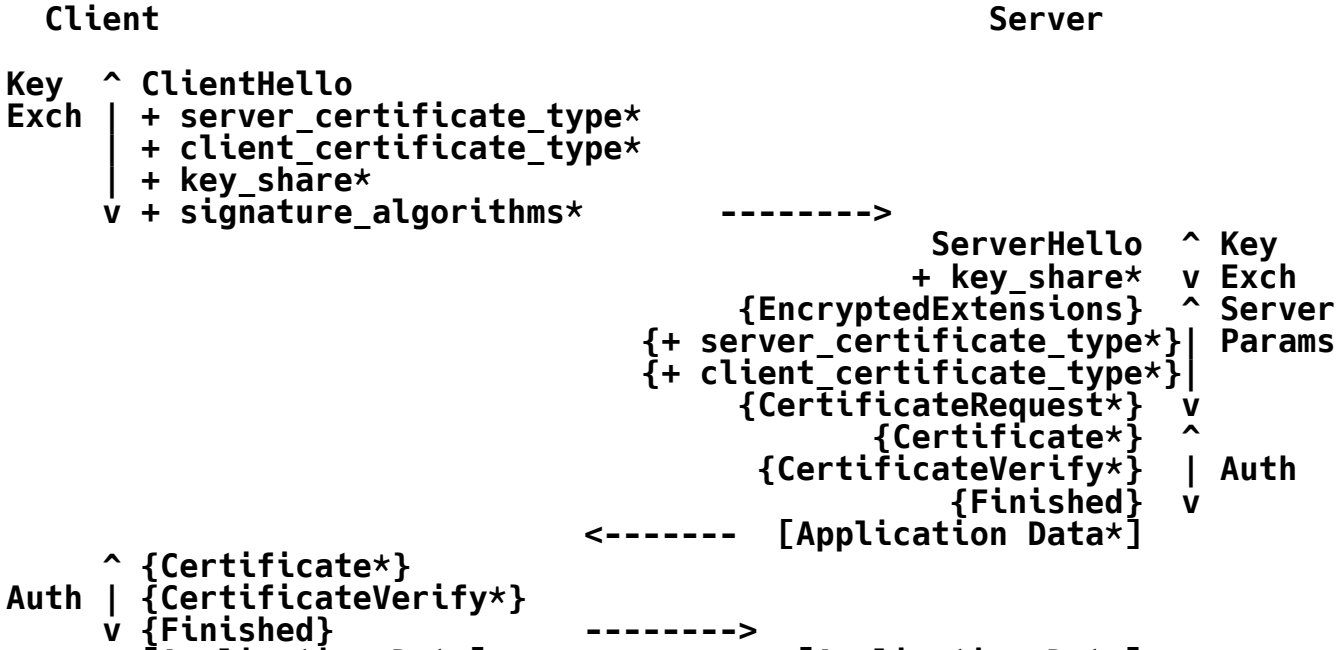
```
  /* Managed by IANA */
   enum {
       X509(0),
       RawPublicKey(2),
       1609Dot2(3),
       (255)
   } CertificateType;

   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 8446*/
           case X.509:
           opaque cert_data<1..2^24-1>;

            };

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

As per [RFC7250], the server processes the received [endpoint]_certificate_type extension(s) and selects one of the offered certificate types, returning the negotiated value in its EncryptedExtensions (TLS 1.3) message. Note that there is no requirement for the negotiated value to be the same in client_certificate_type and server_certificate_type extensions sent

in the same message.

4. TLS Client and Server Handshake

Figure 1 shows the handshake message flow for a full TLS 1.3 handshake negotiating both certificate types.



+ 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 the case of TLS 1.3, in order to negotiate the support of ITS certificate-based authentication, clients and servers include the extension of type "client_certificate_type" and "server_certificate_type" in the extended Client Hello and "EncryptedExtensions".

4.1. Client Hello

In order to indicate the support of ITS 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 [RFC8446] (TLS 1.3).

For TLS 1.3, the rules for when the Client Certificate and CertificateVerify messages appear are as follows:

* The client's Certificate message is present if and only if the server sent a CertificateRequest message.

* The client's CertificateVerify message is present if and only if the client's Certificate message is present and contains a non-empty certificate_list.

For maximum compatibility, all implementations SHOULD be prepared to handle "potentially" 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 scenarios are possible:

* If both the client and server indicate support for the ITS certificate type, the server MAY select the first (most preferred) certificate type from the client's list that is supported by both peers.

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

* The server supports the certificate types specified in this document. In this case, it MAY respond with a certificate of this type. It MAY also include the client_certificate_type extension in Encrypted Extension. Then, the server requests a certificate from the client (via the CertificateRequest message).

The certificates in the TLS client or server certificate chain MAY be sent as part of the handshake, MAY be obtained from an online repository, or might already be known to and cached at the endpoint. If the handshake does not contain all the certificates in the chain, and the endpoint cannot access the repository and does not already know the certificates from the chain, then it SHALL reject the other endpoint's certificate and close the connection. Protocols to support retrieving certificates from a repository are specified in ETSI [TS102941].

5. Certificate Verification

Verification of an ITS certificate 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 1609.2, 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 an Ieee1609Dot2Data encoded with Canonical Octet Encoding Rules (OER) [ITU-TX.696] of type signed as specified in [IEEE1609.2] and [IEEE1609.2b], where:

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

* headerInfo.psid indicates the application activity that the certificate is authorizing.

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

* headerInfo.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 [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.

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

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

6. Examples

Some of the message-exchange examples are illustrated in Figures 2 and 3.

6.1. TLS Server and TLS Client Use the ITS Certificate

This section shows an example where the TLS client as well as the TLS server use ITS certificates. In consequence, both the server and the client populate the client_certificate_type and server_certificate_type extension with the IEEE 1609 Dot 2 type 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 ITS Certificate

6.2. TLS Client Uses the ITS 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 validate an X.509 certificate from the server. The server chooses the X.509 certificate to make its authentication with the client. This is applicable in the case of a raw public key supported by the server.

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

Figure 3: TLS Client Uses the ITS Certificate and TLS Server Uses the X.509 Certificate

7. Security Considerations

This section provides an overview of the basic security considerations that need to be taken into account before implementing the necessary security mechanisms. The security considerations described throughout [RFC8446] apply here as well.

7.1. Securely Obtaining Certificates from an Online Repository

In particular, the certificates used to establish a secure connection MAY be obtained from an online repository. 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 [TS102941] 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.

7.3. Algorithms and Cryptographic Strength

All ITS certificates use public-key cryptographic algorithms with an estimated strength on the order of 128 bits or more, 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 PSID, also known as an ITS Application Identifier (ITS-AID), which identifies a broad set of application activities that provide a context for the certificate holder's permissions, and a Service Specific Permissions (SSP) field associated with that 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 [IEEE1609.12], and for more information about the development of SSPs, see [SAEJ29455].

7.5. Psid and Pdufunctionaltype in CertificateVerify

The CertificateVerify message for TLS 1.3 is an Ieee1609Dot2Data of type signed, where the signature contained in this Ieee1609Dot2Data was generated using an 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: an 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.

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.

8. Privacy Considerations

For privacy considerations in a vehicular environment, the ITS certificate is used for many reasons:

* In order to address the risk of a personal data leakage, messages exchanged for vehicle-to-vehicle (V2V) communications are signed using ITS pseudonym certificates.

* The purpose of these certificates is to provide privacy and minimize the exchange of private data.

9. IANA Considerations

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

Value 3 was previously assigned for "1609Dot2" and included a reference to draft-tls-certieee1609. IANA has updated this entry to reference this RFC.

10. Normative References

[IEEE1609.12]
IEEE, "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Identifier Allocations", IEEE 1609.12-2016, December 2016.

[IEEE1609.2]
IEEE, "IEEE Standard for Wireless Access in Vehicular Environments -- Security Services for Applications and Management Messages", IEEE Standard 1609.2-2016, DOI 10.1109/IEEESTD.2016.7426684, March 2016, <https://doi.org/10.1109/IEEESTD.2016.7426684>.

[IEEE1609.2b]
IEEE, "IEEE Standard for Wireless Access in Vehicular

Environments--Security Services for Applications and Management Messages - Amendment 2--PDU Functional Types and Encryption Key Management", IEEE 1609.2b-2019, June 2019.

[ISO21177] ISO, "Intelligent transport systems - ITS station security services for secure session establishment and authentication between trusted devices", ISO/TS 21177:2019, August 2019.

[ITU-TX.696] ITU-T, "Information technology - ASN.1 encoding rules: Specification of Octet Encoding Rules (OER)", Recommendation ITU-T X.696, August 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[SAEJ29453] SAE, "Requirements for V2I Weather Applications", J2945/3, June 2017.

[SAEJ29455] SAE, "Service Specific Permissions and Security Guidelines for Connected Vehicle Applications", J2945/5_202002, February 2020.

[TS102941] ETSI, "Intelligent Transport Systems (ITS); Security; Trust and Privacy Management", ETSI TS 102 941, 2018.

[TS103097] ETSI, "Intelligent Transport Systems (ITS); Security; Security header and certificate formats", ETSI TS 103 097, 2017.

Acknowledgements

The authors wish to thank Adrian Farrel, Eric Rescola, Russ Housley, Ilari Liusvaara, and Benjamin Kaduk 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 document.

Authors' Addresses

Mounira Msahli (editor)
Telecom Paris
France

Email: mounira.msahli@telecom-paris.fr

Nancy Cam-Winget (editor)
Cisco
United States of America

Email: ncamwing@cisco.com

William Whyte (editor)
Qualcomm
United States of America

Email: wwhyte@qti.qualcomm.com

Ahmed Serhrouchni
Telecom Paris
France

Email: ahmed.serhrouchni@telecom-paris.fr

Houda Labiod
Telecom Paris
France

Email: houda.labiod@telecom-paris.fr