TLS Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

A. KAISER
IRT SystemX
H. LABIOD
Telecom Paristech
B. LONC
Renault
M. MSAHLI
Telecom Paristech
A. SERHROUCHNI
Telecom ParisTech

March 13, 2017

Transport Layer Security (TLS) Authentication using ITS ETSI and IEEE certificates
draft-serhrouchni-tls-certieee1609-01.txt

Abstract

This document specifies the use of two new certificate types to authenticate TLS entities. The first type enables the use of a certificate specified by the Institute of Electrical and Electronics Engineers (IEEE) [IEEE-ITS] and the second by the European Telecommunications Standards Institute (ETSI) [ETSI103097].

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Introduction 2
2. Requirements Terminology 3
3. Extension Overview 3
4. Security Considerations 4
5. IANA Considerations 5
6. Message Flow 5
6.1. Client Hello 5
7. Certificate Verification 6
7.1. IEEE 1609.2 certificates 6
7.2. ETSI TS 103 097 certificates 6
8. References 6
8.1. Normative References 6
8.2. Informative References 7
Appendix A. Certificates comparison 7
A.1. ETSI vs IEEE 7
A.2. ETSI vs X.509 8
Appendix B. ETSI Encoding Example 9
Appendix C. IEEE Encoding Example 13
Authors' Addresses 15

1. Introduction

At present, TLS protocol uses X509 [RFC5246] and OpenPGP digital certificates [RFC6091] in order to authenticate servers and clients. This document describes the use of certificates specified either by the Institute of Electrical and Electronics Engineers (IEEE) [IEEE-ITS] or the European Telecommunications Standards Institute (ETSI) [ETSI103097]. These standards were defined in order to secure communications in vehicular environments. Existing certificates, such as X509 and OpenPGPG, are designed for Internet use, particularly for flexibility and extensibility, and are not optimized for bandwidth and processing time to support delay-sensitive applications. This is why size-optimized certificates that meet the ITS requirements were designed and standardized.

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

Two new values referring the previously mentioned certificated are added to the "cert_type" extension defined in [RFC6091].

2. Requirements Terminology

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 [RFC2119].

3. Extension Overview

In order to negotiate the support of IEEE or ETSI certificate-based authentication, clients MAY include an extension of type "cert_type" in the extended client hello. The "extension_data" field of this extension SHALL contain a list of supported certificate types proposed by the client, where:

```
enum {
    X.509(0), OpenPGP(1), RawPublicKey(2),
    IEEE(TBD), ETSI(TBD), (255)
}CertificateType;
```

In case where the TLS server accepts the described extension, it selects one of the certificate types in the extension described here. The same extension type and structure will be used for the server's response to the extension described here. Note that a server MAY send no certificate type if it either does not support it or wishes to authenticate the client using other authentication methods. The client MAY at its discretion either continue the handshake, or respond with a fatal message alert.

The end-entity certificate's public key has to be compatible with one of the certificate types listed in extension described here.

Servers aware of the extension described here but not wishing to use it, SHOULD gracefully revert to a classical TLS handshake or decide not to proceed with the negotiation.

4. 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 [RFC5246] apply here as well.

For security considerations in a vehicular environment, the minimal use of any TLS extensions is recommended such as :

- The "cert_type" [IANA value 9] extension who's purpose was previously described in Section 3.
- The "elliptic_curves" [IANA value 10] extension which indicates the set of elliptic curves supported by the client.
- The "SessionTicket" [IANA value 35] extension for session resumption.

In addition, servers SHOULD not support renegotiation [RFC5746] which presented Man-In-The-Middle (MITM) type attacks over the past years.

The ETSI and IEEE Standards propose the use of secp256r1 (aka NIST P-256) recommended by the NIST FIPS 186-4 standard [FIPS186].

Elliptic curve algorithms require significantly shorter public keys to achieve the same security strength. ECC is the digital signature algorithm of choice in the IEEE 1609.2 standard that specifies security services and procedures designed for vehicle communications. The ECDSA is specified in American National Standard (ANS) X9.62 . NIST approved the use of ECDSA and specified additional requirements in the FIPS Publication 186-4.

ECDSA also produces smaller signatures than RSA. The smaller key sizes and signature sizes of ECDSA mean lower message overheads when transporting ECDSA public keys over wireless networks compared with transporting RSA or DSA public keys. This is important in a large vehicle network where vehicles may often have to exchange their public keys over bandwidth - limited wireless channels. The smaller ECDSA key lengths can also translate into savings on computing power, storage and memory space, and energy required to achieve the same security strength [KARGL] [SCHUTZE] [PETIT] [ICSI]. This makes ECDSA attractive for resource - constrained mobile devices, such as vehicle on-board communication units.

The Standard defines ECIES as the encryption algorithm. Seen that this RFC aims to client authentication, the use of this algorithm can be optional for future use but not required.

AES-CCM provides both authentication and confidentiality (encryption and decryption) and uses as its only primitive the AES encrypt block cipher operation. This makes it amenable to compact implementations, which are advantageous in constrained envrionments. Adoption outside of constrained environments is necessary to enable interoperability, such as that between web clients and embedded servers, or between embedded clients and web servers.

5. IANA Considerations

Existing IANA references have not been updated yet to point to this document.

IANA is asked to register two new values in the "TLS Certificate Types" registry of Transport Layer Security (TLS) Extensions [TLS-Certificate-Types-Registry], as follows:

- Value: TBD Description: IEEE Reference: [THIS RFC]

- Value: TBD Description: ETSI Reference: [THIS RFC]

6. Message Flow

6.1. Client Hello

In order to indicate the support of IEEE or ETSI certificates, clients MUST include an extension of type "cert_type" to the extended client hello message. The hello extension mechanism is described in Section 7.4.1.4 of TLS 1.2 [RFC5246].

The extension 'cert_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 a vehicular environment, privacy is important. In order to preserve anonymity, a client MUST include IEEE or ETSI certificate types in the "cert_type" extension. certificates.

A TLS client that proposes ECC algorithms in its ClientHello message SHOULD include "elliptic_curves" extension [RFC4492].

Clients respond along with their certificates by sending a "Certificate" message immediately followed by the "ClientKeyExchange" message. The premaster secret is generated according to the cipher algorithm selected by the server in the ServerHello.cipher_suite.

7. Certificate Verification

7.1. IEEE 1609.2 certificates

Verification of an IEEE 1609.2 certificate or certificate chain is described in section 5.5.2 of [IEEE-ITS].

7.2. ETSI TS 103 097 certificates

Verification of ETSI TS 103 097 certificate or certificate chain is described in annex F of [ETSI102941].

8. References

8.1. Normative References

[ETSI103097] ETSI, , "ETSI TS 103 097 v1.1.1 (2013-04): Intelligent Transport Systems (ITS); Security; Security header and certificate formats", April 2013.

[IEEE-ITS] IEEE 1609.2, , "IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages", 2013.

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

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", May 2006.

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

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension"", February 2010.

[RFC6091] Mavrogiannopoulos, N. and D. Gillmor, "Using OpenPGP Keys for Transport Layer Security (TLS) Authentication", February 2011.

[RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", June 2014.

[ETSI102941]
ETSI, , "ETSI TS 102 941 v1.1.9 (2016-04): Intelligent Transport Systems (ITS); Security; Trust and Privacy Management", April 2016.

8.2. Informative References

[FIPS186] FIPS 186-4, , "Digital Signature Standard", July 2013.

[KARGL] Kargl, F., Papadimitratos, P., Buttyan, L., Muter, M., Schoch, E., Wiedersheim, B., Thong, T.-V., Calandriello, G., Held, A., Kung, A., and J.-P. Hubaux, "Secure Vehicular Communications: Implementation, Performance, and Research Challenges", November 2008.

[SCHUTZE] Schutze, T., "Automotive security: Cryptography for Car2X communication", March 2011.

[PETIT] Petit, J., "Analysis of ECDSA authentication processing in VANETs", December 2009.

[ICSI] ICST project, , "Analysis of timeliness of communication for IEEE 1609.2", 2013.

[X696] ITU-T X.696, , "Information Technology - ASN.1 encoding rules: Specification of Octet Encoding Rules (OER)", august 2014.

Appendix A. Certificates comparison

A.1. ETSI vs IEEE

The ETSI and IEEE 1609.2 represent the active standardization groups in Europe and U.S those dealing with the security of vehicular communications. Although defined for the same purpose, the different security requirements have led to the definition of different certificate formats.

ETSI Certificate	IEEE Certificate
Version	Version
	Type
Signer_info	IssuerIdentifier
Subject_info	

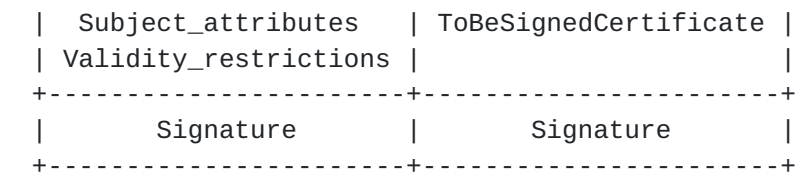

Figure 1: Certificates comparison

As given in Figure 1, the IEEE certificate contains same data strucutres as the one defined by ETSI except "Type" field, which specifies the type of certificate if it is implicit or explicit.

The main differences are listed below:

- The structure of US Security Credentials Management System (SCMS) is different from EU PKI

- Revocation distribution is not supported yet in ETSI TS 103 097

- SCMS provides high privacy for pseudonym resolution with 2 Linkage Authorities (LA1 and LA2): LA1/2 generate 2 linkage values that are added in the certificate. This allow to connect all short-term certificates from a specific device for ease of revocation in the event of misbehavior.

- Certificate Encoding

  * As described in the IEEE 1609.2 and ETSI standards, the internal representation of the certificate structure is encoded into a flat octet string in network byte order (i.e. big-endian).

  * IEEE 1609.2 is developing for future an ASN.1 version of the standard using X.696 (OER) [X696].

A.2. ETSI vs X.509

- Distinguished Name: There is no Distinguished Name based on X.500 in C-ITS certificates. Instead, Subject Names are defined as a string of maximum 32 bytes. There are no naming convention for Subject Names and the unicity of Subject Names for CAs and end-entities is not required. Pseudonym certificates does not use Distinguished Names for pseudonymous authentication (Subject Name is empty): the digest of the certificate is used as unique identifier. It is defined as a HashedID8 attribute which represents the 8 least significant bytes of the SHA-256 hash computation of the certificate.

- Geographical attributes: The C-ITS certificates of CAs and end-entities may contain geographical validity attributes (location) which doesn't exist in x509.

- Trust Assurance Level (TAL): The C-ITS certificates of CAs and end-entities must contain a TAL:

  * For the security of a V2X communication system, assurance about the in-vehicle security of participants is vital: the receiver of a message has to be able to rely on the fact that the sender has generated the message correctly (i.e. the car sensors information is accurate and of integrity). Hence, a security breach on the sender would have an impact on all the receivers of a message.

  * Only vehicles with a reasonable "level of security" should be able to obtain certificates from the PKI. The Car-to-Car Communication Consortium (C2C-CC) introduced different levels of trust, defined as Trust Assurance Levels and the authorization tickets (i.e. pseudonym certificates) of the vehicle must include the value of the vehicle TAL.

In order to authenticate end-entities in TLS with ETSI certificates the following issues still have to be addressed:

- No certificate profile for ITS-S Centre (i.e. Internet Server) is defined yet in ETSI TS 103 097 specification.

- A field is required in certificates for ITS-S Centre to provide the server's FQDN (Fully Qualified Domain Name). To this end, either the use of the Subject Name field is possible (although the size is limited to 32 bytes) or a new SubjectAttribute may be defined for this purpose.

Appendix B. ETSI Encoding Example

The hex sequence shown in Figure 2 presents an encoded secured message with signed payload as a generic encoded octet string.

```
     00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
01 | 02 80 ba 80 02 02 01 53 88 de c6 40 c6 e1 9e 01
02 | 00 52 00 00 04 d4 81 34 8a cd d1 d9 9c 1f fb a4
03 | c7 0e 6d 2a 5d 13 ca b0 a1 e6 cf 63 22 9f 69 79
04 | b4 53 c0 15 c7 da 3a 12 7c 8f 39 44 59 b1 2f 94
05 | d4 cb 9a 12 ce e1 1d 87 40 8d 91 ac 95 6c 90 c8
06 | b3 b2 9f 4c 22 02 e0 21 0b 24 03 01 00 00 25 04
07 | 01 00 00 00 0b 01 15 04 39 83 15 4c bc 02 03 00
```

```
08 | 00 00 71 ff 9a 0d 80 16 ca cb cd d8 1c d1 4f 81
09 | 94 3c dd c7 74 51 1e 2b f7 15 7b 33 e5 4f 7b 6b
10 | 6e 5b 5d 07 94 70 be 40 a6 46 e0 55 9c 19 89 28
11 | b5 b8 ed cf bd c2 29 70 53 95 1d bc 51 cb d6 a3
12 | e1 d0 00 00 01 41 ae 0f 26 64 c0 05 24 01 55 20
13 | 50 02 80 00 31 01 00 14 00 30 14 4a d9 f8 7e 59
14 | 9e 09 2b 00 00 00 00 00 00 00 00 80 00 00 00 00
15 | 00 00 00 07 d1 00 00 01 02 00 00 00 02 09 2b 40
16 | 56 b4 9d 20 0d 69 3a 40 1f ff ff fc 22 30 d4 1e
17 | 40 00 0f c0 00 7e 02 76 ea 87 33 a9 d7 4f ff d0
18 | 84 14 00 00 43 01 00 00 61 6d 42 37 dd 2c ea b7
19 | 27 31 c2 3b cb 5d 61 8f 88 17 df 0d a8 7b d2 b8
20 | d3 54 8f 71 09 8a f1 88 d2 43 04 a8 61 6a 95 bf
21 | 5e 07 45 a1 06 e9 33 9f 9e 69 ba b3 3c bc 68 28
22 | 93 5a 66 ea 11 a0 37 69
```

Figure 2: Example of encoded ETSI secured message with signed payload

In the parsed data structure, the contents are presented in the form:

```
struct SecuredMessage {
  uint8 protocol_version: 2
  HeaderField<186> header_fields {
    struct HeaderField {
      HeaderFieldType type: signer_info (128)
      struct SignerInfo signer {
        SignerInfoType type: certificate (2)
        struct Certificate certificate {
          uint8 version: 2
          struct SignerInfo signer_info {
            SignerInfoType type: certificate_digest_with_sha256 (1)
            HashedId8 digest: 5388DEC640C6E19E
          }
          struct SubjectInfo subject_info {
            SubjectType subject_type: authorization_ticket (1)
            opaque<0> subject_name:
          }
          SubjectAttribute<82> subject_attributes {
            struct SubjectAttribute {
              SubjectAttributeType type: verification_key (0)
              struct PublicKey key {
                PublicKeyAlgorithm algorithm: ecdsa_nistp256_with_
                                              sha256 (0)
                struct EccPoint public_key {
                  EccPointType type: uncompressed (4)
                  opaque[32] x: D481348ACDD1D99C1FFBA4C70E6D2A5D
                                13CAB0A1E6CF63229F6979B453C015C7
                  opaque[32] y: DA3A127C8F394459B12F94D4CB9A12CE
```

```
                              E11D87408D91AC956C90C8B3B29F4C22
        }
      }
    }
    struct SubjectAttribute {
      SubjectAttributeType type: assurance_level (2)
      SubjectAssurance assurance_level: assurance level = 7,
                                        confidence = 0
                                        (bitmask = 11100000)
    }
    struct SubjectAttribute {
      SubjectAttributeType type: its_aid_ssp_list (33)
      ItsAidSsp<11> its_aid_ssp_list {
        struct ItsAidSsp {
          IntX its_aid: 36
          opaque<3> service_specific_permissions: 010000
        }
        struct ItsAidSsp {
          IntX its_aid: 37
          opaque<4> service_specific_permissions: 01000000
        }
      }
    }
  }
  ValidityRestriction<11> validity_restrictions {
    struct ValidityRestriction {
      ValidityRestrictionType type: time_start_and_end (1)
      Time32 start_validity: 2015-03-05 00:00:00 UTC
      Time32 end_validity: 2015-04-28 23:59:59 UTC
    }
    struct ValidityRestriction {
      ValidityRestrictionType type: region (3)
      struct GeographicRegion region {
        RegionType region_type: none (0)
      }
    }
  }
  struct Signature {
    PublicKeyAlgorithm algorithm: ecdsa_nistp256_with_sha256 (0)
    struct EcdsaSignature ecdsa_signature {
      struct EccPoint R {
        EccPointType type: x_coordinate_only (0)
        opaque[32] x: 71FF9A0D8016CACBCDD81CD14F81943C
                      DDC774511E2BF7157B33E54F7B6B6E5B
      }
      opaque[32] s: 5D079470BE40A646E0559C198928B5B8
                    EDCFBDC2297053951DBC51CBD6A3E1D0
    }
```

```
            }
          }
        }
      }
      struct HeaderField {
        HeaderFieldType type: generation_time (0)
        Time64 generation_time: 2015-03-17 15:26:48.000 UTC
      }
      struct HeaderField {
        HeaderFieldType type: its_aid (5)
        IntX its_aid: 36
      }
    }
    struct Payload payload_field {
      PayloadType type: signed (1)
      opaque<85> data: 2050028000310100140030144AD9F87E
                       599E092B0000000000000000080000000
                       0000000007D1000001020000002092B
                       4056B49D200D693A401FFFFFFC2230D4
                       1E40000FC0007E0276EA8733A9D74FFF
                       D084140000
    }
    TrailerField<67> trailer_fields {
      struct TrailerField {
        TrailerFieldType type: signature (1)
        struct Signature signature {
          PublicKeyAlgorithm algorithm: ecdsa_nistp256_with_sha256 (0)
          struct EcdsaSignature ecdsa_signature {
            struct EccPoint R {
              EccPointType type: x_coordinate_only (0)
              opaque[32] x: 616D4237DD2CEAB72731C23BCB5D618F
                            8817DF0DA87BD2B8D3548F71098AF188
            }
            opaque[32] s: D24304A8616A95BF5E0745A106E9339F
                          9E69BAB33CBC6828935A66EA11A03769
          }
        }
      }
    }
  }
```

Figure 3: Example of parsed ETSI secured message with signed payload

Appendix C. IEEE Encoding Example

The hex sequence shown in Figure 4 presents an encoded signed data structure as a flat encoded octet string.

```
     00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--
01 | 02 01 03 02 02 04 f3 db 4f 6f ca b6 49 65 01 09
02 | 63 65 72 74 4e 61 6d 65 31 01 05 e0 00 00 01 00
03 | 04 00 00 00 00 00 00 00 01 00 02 d4 a8 61 1d ce
04 | d8 8c a7 a2 e9 6a 8d 7e 49 0f 3c 9a 46 27 c0 72
05 | 26 ed 67 8d 04 74 41 02 00 03 9c b6 6f 87 4a 40
06 | 7c 21 83 40 22 db 6d 0a 80 d0 14 cb df 24 fc a0
07 | 83 f8 e2 00 81 b0 7c 14 b8 e7 02 19 90 d0 57 4b
08 | 14 d2 80 29 1f c4 e6 a6 73 12 68 74 96 77 c2 52
09 | 34 ae bb e4 29 da 16 60 61 19 74 c6 b3 53 98 0e
10 | 70 e3 3d 4f b9 03 99 76 05 44 e9 74 70 d9 92 bb
11 | 3c 37 92 c3 51 d4 7d 8e ea b1 03 0a e0 00 00 01
12 | 0c 73 6f 6d 65 20 63 6f 6e 74 65 6e 74 00 00 e7
13 | 2a dc 3e dc 09 00 00 00 00 00 00 00 00 00 00 00
14 | 02 ca bf a2 0d 82 ae 3e 25 a3 8c 9c dd 2e cf 94
15 | 9f cc 7c 7f d9 d8 83 89 f5 08 f7 aa bb 5b ef 21
16 | bd 7a 2e 79 6c c7 de 01 af b1 93 35 5b e2 f5 88
17 | 19 76 70 e4 ae 09 cf 3b ee
```

Figure 4: Example of encoded IEEE 1609.2 v2 signed data structure

In the parsed data structure, the contents are presented in the form:

```
protocol_version (0, 1): 02
type (1, 1): 01 (signed)
signed_data (2, 263):
  signer (2, 169):
    type (2, 1): 03 (certificate)
    certificates (3, 168):
      version_and_type (3, 1): 02 (explicit)
      unsigned_certificate (4, 102):
        holder_type (4, 1): 02 (identified localized)
        cf (5, 1): 04 (encryption_key)
        signer_id (6, 8): f3 db 4f 6f ca b6 49 65
        signature_alg (14, 1): 01 (ECDSA NIST P256)
        scope (15, 18):
          id_scope (15, 18):
            name_len (15, 1): 09
            name (16, 9): 63 65 72 74 4e 61 6d 65 31
            permissions (25, 7):
              type (25, 1): 01 (specified)
```

```
                permissions_list_len (26, 1): 05
                permissions_list (27, 5):
                  psid (27, 4): e0 00 00 01
                  service_specific_permissions_len (31, 1): 00
              region (32, 1):
                region_type (32, 1): 04 (none)
            expiration (33, 4): 00 00 00 00 (00:00:34 01 Jan 2004 UTC)
            crl_series (37, 4): 00 00 00 01
            verification_key (41, 30):
              algorithm (41, 1): 00 (ECDSA NIST P224)
              public_key (42, 29):
                type (42, 1): 02 (compressed, lsb of y is 0)
                x (43, 28):
                  d4 a8 61 1d ce d8 8c a7 a2 e9 6a 8d 7e 49 0f 3c
                  9a 46 27 c0 72 26 ed 67 8d 04 74 41
            encryption_key (71, 35):
              algorithm (71, 1): 02 (ECIES NIST P256)
              supported_symm_alg (72, 1): 00 (AES 128 CCM)
              public_key (73, 33):
                type (73, 1): 03 (compressed, lsb of y is 1)
                x (74, 32):
                  9c b6 6f 87 4a 40 7c 21 83 40 22 db 6d 0a 80 d0
                  14 cb df 24 fc a0 83 f8 e2 00 81 b0 7c 14 b8 e7
          signature (106, 65):
            ecdsa_signature (106, 65):
              R (106, 33):
                type (106, 1): 02 (compressed, lsb of y is 0)
                x (107, 32):
                  19 90 d0 57 4b 14 d2 80 29 1f c4 e6 a6 73 12 68
                  74 96 77 c2 52 34 ae bb e4 29 da 16 60 61 19 74
                s (139, 32):
                  c6 b3 53 98 0e 70 e3 3d 4f b9 03 99 76 05 44 e9
                  74 70 d9 92 bb 3c 37 92 c3 51 d4 7d 8e ea b1 03
      unsigned_data (171, 37):
        tf (171, 1): 0a (use_generation_time, use_location)
        psid (172, 4): e0 00 00 01
        data_len (176, 1): 0c
        data (177, 12): 73 6f 6d 65 20 63 6f 6e 74 65 6e 74
        generation_time (189, 9):
          time (189, 8): 00 00 e7 2a dc 3e dc 09
               (19:08:23 20 Jan 2012 UTC)
          log_std_dev (197, 1): 00 (1.134666 ns or less)
        generation_location (198, 10):
          latitude (198, 4): 00 00 00 00
          longitude (202, 4): 00 00 00 00
          elevation (206, 2): 00 00
      signature (208, 57):
        ecdsa_signature (208, 57):
```

```
          R (208, 29):
            type (208, 1): 02 (compressed, lsb of y is 0)
            x (209, 28):
              ca bf a2 0d 82 ae 3e 25 a3 8c 9c dd 2e cf 94 9f
              cc 7c 7f d9 d8 83 89 f5 08 f7 aa bb
            s (237, 28):
              5b ef 21 bd 7a 2e 79 6c c7 de 01 af b1 93 35 5b
              e2 f5 88 19 76 70 e4 ae 09 cf 3b ee
```

Figure 5: Example of parsed IEEE 1609.2 v2 signed data structure

Authors' Addresses

Arnaud Kaiser
IRT SystemX
8 avenue de la Vauve
91120 Palaiseau
France

EMail: arnaud.kaiser@irt-systemx.fr

Houda Labiod
Telecom Paristech
46 rue Barrault
75634 Paris cedex 13
France

EMail: houda.labiod@telecom-paristech.fr

Brigitte Lonc
Renault
France

EMail: brigitte.lonc@renault.com

Mounira Msahli
Telecom Paristech
46 rue Barrault
Paris 75634
France

EMail: mounira.msahli@telecom-paristech.fr

Ahmed Serhrouchni
Telecom ParisTech
46 rue Barrault
Paris 75634
France

EMail: ahmed.serhrouchni@telecom-paristech.fr