ETSI TS 103 097 V2.2.1 (2026-03)

TECHNICAL SPECIFICATION

Intelligent Transport Systems (ITS);
Security;
Security header and certificate formats;
Release 2

Reference
RTS/ITS-005122

Keywords
ITS, privacy, protocol, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Intelligent Transport Systems (ITS).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Security policies require that data structures such as messages used in Intelligent Transport Systems are secured when stored or transferred. For interoperability reasons, a common format for secure data structures featuring security headers and public key certificates needs to be provided.

The present document provides these definitions as a profile of the base standard IEEE Std 1609.2™-2025 [1]. A profile makes use of the definitions in the base standard and defines the use of particular subsets or options available in the base standard. This implies that the present document is to be read and interpreted together with that base standard.

From time to time, new versions of the present document may be published that extend IEEE Std 1609.2™ [1] data types using ASN.1 extension mechanisms to define ETSI originated extensions that are not necessarily endorsed by IEEE.

The present document contains material from IEEE Std 1609.2™-2025 [1], reprinted with permission from IEEE, and Copyright© 2025.

1 Scope

The present document specifies the secure data structure including header and certificate formats for Intelligent Transport Systems. In addition to supporting the use of explicit certificates, the present document includes the amendment for the use of implicit certificates. As such, the present document is backwards compatible for the sender of secure data structures, but implies the additional implementation of implicit certificates on receiver side in order to support interoperability.

NOTE 1: The use of explicit certificates is allowed in previous versions (V1.3.1 and V1.4.1) of the present document.

NOTE 2: The use of implicit certificates is not allowed in previous versions of the present document.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] IEEE P1609.2™: "IEEE Approved Draft Standard for Wireless Access in Vehicular Environments--Security Services for Application and Management Messages". (03/10/2026).

[2] ETSI TS 102 965: "Intelligent Transport Systems (ITS); Application Object Identifier (ITS-AID); Registration; Release 2".

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

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI TS 102 940: "Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management; Release 2".

[i.2] ETSI TS 102 941: "Intelligent Transport Systems (ITS); Security; Trust and Privacy Management; Release 2".

[i.3] ETSI TS 103 601: "Intelligent Transport Systems (ITS); Security; Security management messages communication requirements and distribution protocols; Release 2".

[i.4] CPOC protocol.

[i.5] ETSI TS 103 096-1: "Intelligent Transport Systems (ITS); Testing; Conformance test specifications for ITS Security; Part 1: Protocol Implementation Conformance Statement (PICS); Release 2".

[i.5] ETSI TR 103 902: "Intelligent Transport Systems (ITS); General; Terms and Abbreviations; Release 2".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms and definitions given in IEEE Std 1609.2™ [1] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in IEEE Std 1609.2™ [1], in ETSI TR 103 902 [i.5] and the following apply:

AA	Authorization Authority
AT	Authorization Ticket
CPOC	Central Point Of Contact
EA	Enrolment Authority
EC	Enrolment Credential
KDF	Key Derivation Function
MDM	Misbehaviour Detection Management
PICS	Protocol Implementation Conformance Statement
TLM	Trust List Manager
XOR	Exclusive Or

NOTE: Where an abbreviation defined in ETSI TR 103 902 [i.5] has multiple definitions, the definition appropriate to the context of the present document applies.

4 Basic format elements

4.1 Based on the published version of IEEE Std 1609.2

Data structures in the present document are defined using Abstract Syntax Notation One (ASN.1) and shall be encoded using the Canonical Octet Encoding Rules (COER) as defined in Recommendation ITU-T X.696 [3]. This includes some data structures in the present document for which a "canonical encoding" is used as defined in IEEE Std 1609.2™ [1].

Clauses 5 and 6 specify and describe the data structures with reference to IEEE Std 1609.2™ [1]. The corresponding ASN.1 module is defined in annex A.

The validity of a certificate shall be assessed as defined in IEEE Std 1609.2™ [1] clause 5.1, using the Hash ID-based revocation method for EA and AA certificates, and no revocation method for authorization tickets and enrolment credentials.

NOTE 1: The CRL for EA and AA certificates is defined in ETSI TS 102 941 [i.2].

NOTE 2: The rules for verification of the Root CA certificate against the CTL are defined in ETSI TS 102 941 [i.2].

The validity of signed data shall be assessed as defined in IEEE Std 1609.2™ [1], clause 5.2.

The Protocol Implementation Conformance Statement (PICS) proforma in ETSI TS 103 096-1 [i.5] allows an implementor to state how an implementation complies with the present document, based on the requirements in IEEE Std 1609.2™ [1], with the following exceptions:

- Instead of what stated in IEEE Std 1609.2™ [1] clause 6.4.23, if an implementation claims a conformant implementation of `UnCountryId`, it is not required to recognize any value of `UnCountryId`.
- Instead of what stated in IEEE Std 1609.2™ [1] clause 6.4.24, if an implementation claims a conformant implementation of `CountryAndRegions`, it is not required to recognize any value of `CountryAndRegions`.
- Instead of what stated in IEEE Std 1609.2™ [1] clause 6.4.25, if an implementation claims a conformant implementation of `CountryAndSubregions`, it is not required to recognize any value of `CountryAndSubregions`.
- Instead of what stated in IEEE Std 1609.2™ [1] clause 6.4.26, if an implementation claims a conformant implementation of `RegionAndSubregions`, it is not required to recognize any value of `RegionAndSubregions`.

An implementation can claim compliance if it can parse the above structure(s), i.e. if the presence of those structure(s) in a certificate does not prevent the implementation from understanding the semantics of the rest of the certificate.

NOTE 3: An implementation can also make a conformance claim consisting of a list (or other specification) of the `UnCountryId`, `CountryAndRegions`, `CountryAndSubregions`, `RegionAndSubregions` values that it can recognize.

4.2 Cryptographic algorithms

The digital signature algorithm approved for use in the present document is the Elliptic Curve Digital Signature Algorithm (ECDSA) as specified in IEEE Std 1609.2™ [1].

The hash algorithms approved for use in the present document are SHA-256 and SHA-384 as specified in IEEE Std 1609.2™ [1].

The public key encryption algorithm approved for use in the present document is the Elliptic Curve Integrated Encryption Scheme (ECIES) as specified in IEEE Std 1609.2™ [1].

The symmetric encryption algorithm approved for use in the present document is the Advanced Encryption Standard (AES) with 128-bit keys (AES-128) as specified in IEEE Std 1609.2™ [1].

4.3 Extensions

4.3.1 General process

NOTE 1: This clause and the following clause outline approaches for maintaining and extending the present document and do not directly specify functionality to be implemented.

IEEE Std 1609.2™ [1] structures are extensible using ASN.1 extension mechanisms.

For all extensible IEEE Std 1609.2™ [1] data types other than `HeaderInfo`, extensions will be done by adding new fields after the extension marker in the underlying IEEE Std 1609.2™ [1] data type. To avoid conflicts that might arise if multiple stakeholder groups want to extend the same IEEE Std 1609.2™ [1] data type at the same time, the rapporteur of the present document will need to coordinate with the editor of IEEE Std 1609.2™ [1] and ensure that different extension identifiers are associated with each different extension that is being simultaneously developed.

NOTE 2: In the above paragraph, "extension identifier" refers to the numeric identifier that the ASN.1 encoder automatically associates with an extension field in an ASN.1 structure. The numeric identifier is assigned automatically by the encoder based on the index of the extension field in the list of extension fields in that structure; it is not an identifier that is assigned through a registration process and visible to a human reader of the ASN.1.

4.3.2 HeaderInfo extensions

`HeaderInfo` extensions are included in the component `contributedExtensions`.

The component `contributedExtensions` is of type `ContributedExtensionBlocks` and is a sequence of single extension "blocks" of type `ContributedExtensionBlock`. Each extension block is defined by an identified contributing organization. The ETSI TC ITS WG5 extension block shall be identified by the integer `etsiHeaderInfoContributorId` (2). Within the ETSI TC ITS WG5 extension block, each extension shall be of type `EtsiOriginatingHeaderInfoExtension`.

The type `EtsiOriginatingHeaderInfoExtension` is defined in the module `EtsiTs103097ExtensionModule` specified in clause A.1 and composed of the component `id` and the component `Extn`. The component `id` shall be of type `ExtId` and shall uniquely identify the extension within the set of `EtsiOriginatingHeaderInfoExtension`s. The component `content` shall be associated to the related `id` according to the information object set `EtsiTs103097HeaderInfoExtensions`. The ETSI originated extensions shall be defined as information objects of the class `Extension` and shall be listed in the information object set `EtsiTs103097HeaderInfoExtensions`.

NOTE: This approach allows ETSI to specify new extensions as necessary, using an identifier that is entirely under ETSI's control (the `EtsiTs103097HeaderInfoExtensionId`) to identify those extensions and a separate module called `EtsiTs103097ExtensionModule` that can be updated by ETSI without a need to change the module `IEEE1609dot2.`

The data type `ExtensionModuleVersion` in the module `EtsiTs103097ExtensionModule` shall indicate the version of the module `EtsiTs103097ExtensionModule` and shall be imported into the `EtsiTs103097Module.`

5 Specification of secure data structure

5.1 EtsiTs103097Data

A secure data structure shall be of type `EtsiTs103097Data` as defined in annex A, which corresponds to a `Ieee1609Dot2Data` as defined in IEEE Std 1609.2™ [1], clause 6.3.2, with the constraints defined in this clause, in clause 5.2 and in clause 5.3.

The type `Ieee1609Dot2Data` shall support the following options in the component content:

- The option `unsecuredData` shall be used to encapsulate an unsecured data structure.
- The option `signedData`, corresponding to the type `SignedData` as defined in IEEE Std 1609.2™ [1], clause 6.3.4, shall be used to transfer a data structure with a signature.
- The option `encryptedData`, corresponding to the type `EncryptedData` as defined in IEEE Std 1609.2™ [1], clause 6.3.41, shall be used to transfer an encrypted data structure.

The following corresponding profiles of the type `EtsiTs103097Data` are defined in annex A:

- The parameterized type `EtsiTs103097Data-Unsecured` using the Ieee1609Dot2Data option `unsecuredData`.
- The parameterized type `EtsiTs103097Data-Signed` using the Ieee1609Dot2Data option `signedData` containing the data structure in the component `tbsData.payload.data`.

- The parameterized type `EtsiTs103097Data-SignedExternalPayload` using the `Ieee1609Dot2Data` option `signedData` containing the digest of the data structure in the component `tbsData.payload.extDataHash`.
- The parameterized type `EtsiTs103097Data-Encrypted`, using the `Ieee1609Dot2Data` option `encryptedData` containing the encrypted data structure in the component `ciphertext.aes128ccm.ccmCiphertext`.
- The parameterized type `EtsiTs103097Data- SignedAndEncrypted`, using the parameterized type `EtsiTs103097Data-Encrypted`, containing an encrypted `EtsiTs103097Data-Signed.`
- The parameterized type `EtsiTs103097Data-Encrypted-Unicast` using the parameterized type `EtsiTs103097Data-Encrypted` further constraint to have one entry in the component `recipients`.
- The parameterized type `EtsiTs103097Data-SignedAndEncrypted-Unicast` using the parameterized type `EtsiTs103097Data-Encrypted` containing an encrypted `EtsiTs103097Data-Signed` and further constraint to have one entry in the component `recipients`.

5.2 SignedData

The type `SignedData` shall have the following constraints:

- The component `hashId` of `SignedData` shall indicate the hash algorithm to be used to generate the hash of the message according to IEEE Std 1609.2™ [1], clauses 6.3.5 and 5.3.3.
- The component `tbsData` of `SignedData` shall be of type `ToBeSignedData` as defined in IEEE Std 1609.2™ [1] clause 6.3.6. The type `ToBeSignedData` shall have the component `payload` of type `SignedDataPayload` as defined in IEEE Std 1609.2™ [1], clause 6.3.7, containing either:
  - the component `data`, containing the payload to be signed as an `Ieee1609Dot2Data`; or
  - the component `extDataHash`, containing the hash of data that is not explicitly transported within the structure.

The type `ToBeSignedData` shall have the component `headerInfo` of type `HeaderInfo` as defined in IEEE Std 1609.2™ [1], clause 6.3.9, and constrained to have the following security headers:

- The component `psid` containing the ITS-AID corresponding to the contained message.
- The component `generationTime` as defined in IEEE Std 1609.2™ [1], always present.
- The component `expiryTime`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of message profiles in clause 7.
- The component `generationLocation`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of message profiles in clause 7.
- The component `p2pcdLearningRequest` always absent.
- The component `missingCrlIdentifier` always absent.
- The component `encryptionKey`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of message profiles in clause 7.
- The extension component `inlineP2pcdRequest`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of message profiles in clause 7.
- The extension component `requestedCertificate`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of message profiles in clause 7.

- The extension component `pduFunctionalType`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of message profiles in clause 7.

- In the extension component `contributedExtensions`, any component of type `EtsiOriginatingHeaderInfoExtension` identified in the Information Object Set `EtsiTcItsHeaderInfoExtensions` present or absent according to the specification of the message profiles in clause 7 and according to the specification of the particular extension in the document that specifies it:

  - The extension `EtsiTs102941CrlRequest`, if present, shall indicate that the ITS-Station is requesting a CRL:

    - The component `issuerId` shall indicate the issuer of the CRL.

    - The component `lastKnownUpdate`, if present, shall indicate the value of the `thisUpdate` field of the latest CRL that the ITS-Station has available.

    For a specification of the associated distribution protocol refer to ETSI TS 103 601 [i.3]. For a specification of the CRL format refer to ETSI TS 102 941 [i.2].

  - The extension `EtsiTs102941DeltaCtlRequest`, if present, shall indicate that the ITS-Station is requesting a delta CTL, using the data structure `EtsiTs102941CtlRequest`:

    - The component `issuerId` shall indicate the issuer of the CTL.

    - The component `lastKnownCtlSequence`, if present, shall indicate the value of the `ctlSequence` field of the latest CTL that the ITS-Station has available.

    For a specification of the associated distribution protocol refer to ETSI TS 103 601 [i.3]. For a specification of the delta CTL format refer to ETSI TS 102 941 [i.2].

  - The extension `EtsiTs102941FullCtlRequest`, if present, shall indicate that the ITS-Station is requesting a full CTL:

    - The component `issuerId` shall indicate the issuer of the CTL.

    - The component `lastKnownCtlSequence`, if present, shall indicate the value of the `ctlSequence` field of the latest CTL that the ITS-Station has available.

    - The component `segmentNumber`, if present, shall indicate the value of the sequence number of the missing segment that the ITS-Station is requesting for repetition.

    For a specification of the associated distribution protocol refer to ETSI TS 103 601 [i.3]. For a specification of the full CTL format refer to ETSI TS 102 941 [i.2].

- In the component `contributedExtensions`, any component of type other than `EtsiOriginatingHeaderInfoExtension` always absent.

NOTE: The present document does not specify contributedExtensions fields of type other than `EtsiOriginatingHeaderInfoExtension` and does not specify what an implementation that processes received secure data structures should do, based on such extensions. Anyhow, compliance to the present document requires an implementation to correctly parse received secure data structures that contain those extensions.

The component `signer` of `SignedData` shall be of type `SignerIdentifier` as defined in IEEE Std 1609.2™ [1], clause 6.3.24 and constrained to one of the following choices:

- `digest`, containing the digest of the signing certificate as defined in IEEE Std 1609.2™ [1], clause 6.3.26.

- `certificate`, constrained to only one entry in the `SequenceOfCertificate` list of type `TS103097Certificate`, containing the signing certificate as defined in clause 6 of the present document.

The component `signature` of `SignedData` shall be of type `Signature` as defined in IEEE Std 1609.2™ [1], clause 6.3.37 and shall contain the ECDSA signature as defined in IEEE Std 1609.2™ [1], clauses 6.3.38, 6.3.39 and 5.3.1.

5.3 EncryptedData

The type `EncryptedData` shall have the following constraints:

- The component `recipients` of `EncryptedData` shall be of type `SequenceOfRecipientInfo` as defined in IEEE Std 1609.2™ [1], clause 6.3.42. Every entry shall be either of option `pskRecipInfo` as defined in IEEE Std 1609.2™ [1], clause 6.3.43, of option `certRecipInfo`, or of option `signedDataRecipInfo`, as defined in IEEE Std 1609.2™ [1], clause 6.3.45.
- The encryption scheme used shall be ECIES as defined in IEEE Std 1609.2™ [1], clause 5.3.5. The component `ciphertext` of `EncryptedData` shall be of type `SymmetricCiphertext` as defined in IEEE Std 1609.2™ [1] clause 6.3.49 and contain an `EtsiTs103097Data` encrypted according to IEEE Std 1609.2™ [1], clauses 6.3.50 and 5.3.8.

See Annex B for an informative description of the data encryption/description mechanisms.

6 Specification of certificate format

A certificate contained in a secure data structure shall be of type `EtsiTs103097Certificate` as defined in annex A, which corresponds to a single `Certificate` as defined in IEEE Std 1609.2™ [1], clause 6.4.2, with the constraints defined in this clause. A certificate contained in a secure data structure shall be either of type explicit as defined in IEEE Std 1609.2™ [1] clause 6.4.6, or of type implicit as defined in IEEE Std 1609.2™, clause 6.4.5, at the discretion of the entity that sends the secure data structure.

NOTE 1: This implies that a receiver of secure data structures should be able to validate both types of certificates in order to support interoperability.

The component `toBeSigned` of the type `EtsiTs103097Certificate` shall be of type `ToBeSignedCertificate` as defined in IEEE Std 1609.2™ [1], clause 6.4.8 and constrained as follows:

- The component `id` of type `CertificateId` constrained to choice type `name` or `none`.
- The component `cracaId` set to 000000'H.
- The component `crlSeries` set to 0'D.

NOTE 2: The constraints on `cracaId` and `crlSeries` indicate that certificates defined in the present document are not revoked using mechanisms defined in IEEE Std 1609.2™ [1]. Revocation mechanisms are defined in ETSI TS 102 941 [i.2].

- The component `validityPeriod` with no further constraints. It indicates the validity period of the certificate as the interval [`validityPeriod.start`, `validityPeriod.start` + `validityPeriod duration`), i.e. the interval that goes from `validityPeriod.start` to `validityPeriod.start` + `validityPeriod.duration` with the start inclusive and the end exclusive.
- The component `region` of type `GeographicRegion` as defined in IEEE Std 1609.2 [1], present or absent according to the specification of certificate profiles in clause 7. In addition to what is specified in in IEEE Std 1609.2 [1] clause 6.4.23, the value `65535` shall indicate the European Union "27 countries as of 31 January 2020".

NOTE 3: There are currently no values specified for `CountryAndRegions.regions` defined in IEEE Std 1609.2™ [1] clause 6.4.24 and for `CountryAndSubregions.regionAndsubregions` defined in IEEE Std 1609.2™ [1] clause 6.4.25.

- The component `assuranceLevel` of type `SubjectAssurance`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of certificate profiles in clause 7.
- The component `appPermissions` of type `SequenceOfPsidSsp` as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of certificate profiles in clause 7.
- The component `certIssuePermissions` of type `SequenceOfPsidGroupPermissions`, as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of certificate profiles in clause 7.
- At least one of the components `appPermissions` and `certIssuePermissions` shall be present.
- The component `certRequestPermissions` absent.
- The component `canRequestRollover` absent.
- The component `encryptionKey` of type `PublicEncryptionKey` as defined in IEEE Std 1609.2™ [1], present or absent according to the specification of certificate profiles in clause 7.
- The component `verifyKeyIndicator` of type `VerificationKeyIndicator` as defined in IEEE Std 1609.2™ [1], present and constrained to the choice verificationKey if the certificate type is `explicit`; present and constrained to the choice `reconstructionValue` if the certificate type is `implicit`.
- The extension component `flags` present or absent, see [i.2] for details.
- The extension component `appExtensions` present only if the `AppExtension OperatingOrganizationId` identified by `certExtId-OperatingOrganization,` as defined in `SetCertExtensions,` is contained.
- The component `certIssueExtensions` absent, or optionally present only if the extension component `appExtensions` is present.

NOTE 4: there are no meaningful instances of `CertIssueExtension` specified in `SetCertExtensions` as defined in IEEE Std 1609.2™ [1] clause 6.4.46.

- The component `certRequestExtensions` absent, or optionally present only if the extension component `certIssueExtensions` is present.

NOTE 5: there are no meaningful instances of `CertRequestExtension` specified in `SetCertExtensions` as defined in IEEE Std 1609.2™ [1] clause 6.4.46.

The component `signature` of `EtsiTs103097Certificate` shall be of type `Signature` as defined in IEEE Std 1609.2™ [1], clause 6.3.37 and shall contain the signature, calculated by the signer identified in the issuer component, as defined in IEEE Std 1609.2™ [1], clauses 6.3.38, 6.3.39 and 5.3.1.

7 Security profiles

7.1 Profiles for messages

7.1.1 Security profile for CAMs

The secure data structure containing Cooperative Awareness Messages (CAMs) shall be of type `EtsiTs103097Data-Signed` as defined in clause 5.1 and annex A, containing the CAM as the `ToBeSignedDataContent`, with the additional constraints defined in clause 5.2 and this clause:

- The component `signer` of `SignedData` shall be constrained as follows:
  - As default, the choice `digest` shall be included.

- The choice `certificate` shall be included once, one second after the last inclusion of the choice `certificate` and contain an `EtsiTs103097Certificate` as specified in clause 6.
- If the ITS-S receives a CAM signed by a previously unknown AT, it shall include the choice `certificate` immediately in its next CAM, instead of including the choice `digest`. In this case, the timer for the next inclusion of the choice `certificate` shall be restarted.
- If an ITS-S receives a CAM that includes a `tbsdata.headerInfo` component of type `inlineP2pcdRequest`, then the ITS-S shall evaluate the list of certificate digests included in that component: If the ITS-S finds a certificate digest of the currently used authorization ticket in that list, it shall include the choice `certificate` immediately in its next CAM, instead of including the choice `digest`.

- The component `tbsdata.headerInfo` of `SignedData` shall be further constrained as follows:
  - `psid`: this component shall encode the ITS-AID value for CAMs as assigned in ETSI TS 102 965 [2].
  - The component `inlineP2pcdRequest` shall be included and shall contain the digests of certificates currently unknown to the ITS-Station in the following cases:
    - if the ITS-S received a CAM with the component `signer` of `SignedData` set to the choice `digest`, and this digest points to an unknown authorization ticket;
    - if the ITS-S received a message with the component `signer` of `SignedData` set to the choice `certificate`, and this certificate is signed by an unknown authorization authority certificate, i.e. includes the component `issuer` referencing an unknown certificate.
  - `requestedCertificate`: If an ITS-S receives a CAM with the component `tbsdata.headerInfo` including the component `inlineP2pcdRequest`, then the ITS-S shall evaluate the list of digests included in that component: If the ITS-S finds a digest of a valid certification authority certificate, it shall include the component `requestedCertificate` containing the requested certificate immediately in its next CAM:
    - unless before the generation of the next CAM, the ITS-S received another CAM including the component `requestedCertificate` containing the requested certification authority certificate: in this case the request shall be discarded;
    - unless the component `signer` of `SignedData` is of choice `certificate` according to the rules defined above: in this case the request shall be kept pending and the certificate shall be inserted in the next possible CAM, according to the same conditions.
  - Any component of type `EtsiOriginatingHeaderInfoExtension` appearing in `contributedExtensions` may be present, absent, present under specified conditions, or optional. As different types of `EtsiOriginatingHeaderInfoExtension` are specified in future versions of the present document, those future versions will also state whether and under what circumstances those `EtsiOriginatingHeaderInfoExtension` types are included in CAMs.
  - All other components of the component `tbsdata.headerInfo` allowed to be present according to clause 5 shall not be used and be absent.

7.1.2 Security profile for DENMs

The secure data structure containing Decentralized Environmental Notification Messages (DENMs) shall be of type `EtsiTs103097Data-Signed` as defined in clause 5.1 and annex A, containing the DENM as the `ToBeSignedDataContent`, with the additional constraints defined clause 5.2 and in this clause:

- The component `signer` of `SignedData` shall be of choice `certificate` and contain an `EtsiTs103097Certificate` as specified in clause 6.
- The component `tbsdata.headerInfo` of SignedData shall be further constrained as follows:
  - `generationLocation`: shall be present.

- `psid`: this component shall encode the ITS-AID value for DENMs as assigned in ETSI TS 102 965 [2].
- Any component of type `EtsiOriginatingHeaderInfoExtension` appearing in `contributedExtensions` may be present, absent, present under specified conditions, or optional. As different types of `EtsiOriginatingHeaderInfoExtension` are specified in future versions of the present document, those future versions will also state whether and under what circumstances those `EtsiOriginatingHeaderInfoExtension` types are included in DENMs.
- All other components of the component `tbsdata.headerInfo` allowed to present according to clause 5 shall not be used and be absent.

7.1.3 Generic security profile for other signed messages

The secure data structure containing signed messages other than CAM and DENM shall be of type:

- `EtsiTs103097Data-Signed` as defined in clause 5.1 and annex A, containing the message as the `ToBeSignedDataContent`, or of type;
- `EtsiTs103097Data-SignedExternalPayload` as defined clause 5.1 and in annex A, containing the message digest;

with the additional constraints defined in clause 5.2.

7.1.4 Security profile for encrypted messages

The secure data structure containing encrypted messages shall be of type `EtsiTs103097Data-Encrypted` as defined in clause 5.1 and annex A, containing the message as the `ToBeEncryptedDataContent`, with the additional constraints defined in clause 5.3.

7.1.5 Security profile for signed and encrypted messages

The secure data structure containing signed and then encrypted messages shall be of type `EtsiTs103097Data-SignedAndEncrypted` as defined in clause 5.1 and annex A, containing the message as the `ToBesignedAndEncryptedDataContent`. This corresponds to an `EtsiTs103097Data` of type `EtsiTs103097Data-Encrypted`, containing an EtsiTs103097Data of type `EtsiTs103097Data-Signed`, containing the message as the `ToBeSignedDataContent`.

7.2 Profiles for certificates

7.2.1 Authorization tickets

This clause defines additional aspects of authorization tickets as defined in ETSI TS 102 940 [i.1]. Authorization tickets shall be of type `EtsiTs103097Certificate` as defined in clause 6, with the following constraints:

- The component `issuer` shall be of choice `sha256AndDigest` or `sha384AndDigest` as defined in IEEE Std 1609.2™ [1], clause 6.4.7.
- The `toBeSigned` component `appPermissions` shall be used to indicate message signing permissions, i.e. permissions to sign an `EtsiTs103097Data`.
- The `toBeSigned` component `CertificateId` shall be set to the choice `none`.
- The `toBeSigned` component `certIssuePermissions` shall be absent.

7.2.2 Enrolment credential

This clause defines additional aspects of enrolment credentials (i.e. long-term certificates) as defined in ETSI TS 102 940 [i.1]. Enrolment credentials shall be of type `EtsiTs103097Certificate` as defined in clause 6, with the following constraints:

- The certificate shall be of type `explicit` as specified in IEEE Std 1609.2™, clause 6.4.6.
- The component `issuer` shall be of choice `sha256AndDigest` or `sha384AndDigest` as defined in IEEE Std 1609.2™ [1] clause 6.4.7.
- The `toBeSigned` components `appPermissions` shall be used to indicate message signing permissions, i.e. permissions to sign a certificate request message contained in an `EtsiTs103097Data`.

NOTE: An example of certificate request messages is given in ETSI TS 102 941 [i.2].

- The `toBeSigned` component `CertificateId` shall be set to the choice `name` and shall contain a unique name associated to the enrolment credential.
- The `toBeSigned` component `certIssuePermissions` shall be absent.

7.2.3 Root CA certificates

This clause defines additional aspects of Root CA certificates as defined in ETSI TS 102 940 [i.1]. Root CA certificates shall be of type `EtsiTs103097Certificate` as defined in clause 6, with the following constraints:

- The certificate shall be of type `explicit` as specified in IEEE Std 1609.2, clause 6.4.6.
- The component `issuer` shall be set to `self`.
- These `toBeSigned` components shall be included in addition to those specified in clause 6:
  - `certIssuePermissions` shall be used to indicate issuing permissions, i.e. permissions to sign subordinate certification authority certificates with certain permissions.
  - `appPermissions` shall be used to indicate permissions to sign:
    - CRLs and contain the ITS-AID for the CRL service as assigned in ETSI TS 102 965 [2].
    - CTLs and contain the ITS-AID for the CTL service as assigned in ETSI TS 102 965 [2].

The `toBeSigned` component `CertificateId` shall be set to the choice `name` and shall contain a unique name associated to the root certification authority.

Additional requirements to Root CA certificates are defined in CPOC protocol [i.4].

7.2.4 Subordinate certification authority certificates

This clause defines additional aspects of subordinate certification authority certificates, i.e. enrolment and authorization authority certificates as defined in ETSI TS 102 940 [i.1]. Subordinate certification authority certificates shall be of type `EtsiTs103097Certificate` as defined in clause 6, with the following constraints:

- The certificate shall be of type `explicit` as specified in IEEE Std 1609.2™, clause 6.4.6.
- The component `issuer` shall be set to `sha256AndDigest` or `sha384AndDigest` as defined in IEEE Std 1609.2™ [1] clause 6.4.7.
- These `toBeSigned` components shall be included in addition to those specified in clause 6:
  - `encryption_key`: this component shall contain a public encryption key for ITS-Stations to encrypt messages to the enrolment / authorization authority.

- `certIssuePermissions`: this component shall be used to indicate issuing permissions, i.e. permissions to sign an enrolment credential / authorization ticket with certain permissions.
- `appPermissions`: this component shall be used to indicate message signing permissions, i.e. permissions to sign certificate response messages contained in an `EtsiTs103097Data`.

NOTE: An example of certificate response messages is given in ETSI TS 102 941 [i.2].

The `toBeSigned` component `CertificateId` shall be set to the choice `name` contain a unique name associated to the certification authority, or shall be set to the choice `none`.

Additional requirements to subordinate certification authority certificates are defined in CPOC protocol [i.4].

7.2.5 Trust List Manager certificate

This clause defines additional aspects of Trust List Manager certificates as defined in ETSI TS 102 940 [i.1]. Trust List Manager certificates shall be of type `EtsiTs103097Certificate` as defined in clause 6, with the following constraints:

- The certificate shall be of type `explicit` as specified in IEEE Std 1609.2™, clause 6.4.6.
- The component `issuer` shall be set to `self`.
- These `toBeSigned` components shall be included in addition to those specified in clause 6:
  - `appPermissions`: this component shall contain the ITS-AID for the CTL service as assigned in ETSI TS 102 965 [2].
- The `toBeSigned` component `CertificateId` shall be set to the choice `name` and contain the unique name string associated to the TLM.
- These `toBeSigned` components shall be absent:
  - encryptionKey.
  - certIssuePermissions.

Additional requirements to Trust List Manager certificates are defined in CPOC protocol [i.4].

7.2.6 Misbehaviour Authority certificate

This clause defines additional aspects of Misbehaviour Authority certificates as defined in ETSI TS 102 940 [i.1]. Misbehaviour Authority certificates shall be of type `EtsiTs103097Certificate` as defined in clause 6, with the following constraints:

- The certificate shall be of type `explicit` as specified in IEEE Std 1609.2™ [1] clause 6.4.6.
- The component `issuer` shall be set to `sha256AndDigest` or `sha384AndDigest` as defined in IEEE Std 1609.2™ [1] clause 6.4.7.
- The following `toBeSigned` component shall be included in addition to those specified in clause 6 or constrained as follows:
  - `encryption_key`: this component shall contain a public encryption key for ITS-Stations to encrypt misbehaviour reports to the misbehaviour authority;
  - `certIssuePermissions`: this component shall be absent;
  - `appPermissions`: this component shall be present and shall contain at least the ITS-AID value for the "Misbehaviour Detection Management" (MDM) Service as assigned in ETSI TS 102 965 [2].
- The `toBeSigned` component `CertificateId` shall be set to the choice `name` and shall contain a unique name associated to the misbehaviour authority.

Annex A (normative): ASN.1 Modules

A.1 ETSI TS 103 097 ASN.1 Modules

This clause provides the normative ASN.1 modules containing the syntactical definitions of the data types defined in the present document. The ASN.1 modules import data types from the ASN.1 modules defined in IEEE Std 1609.2™ [1].

The EtsiTs103097Module ASN.1 module is identified by the Object Identifier {itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) secHeaders(103097) core(1) major-version3(3) minor-version-2(2)}. The module can be downloaded as a file as indicated in table A.1. The associated SHA-256 cryptographic hash digest of the referenced file offers a means to verify the integrity of that file.

Table A.1: ETSI TS 103 097 ASN.1 module information

Module name	EtsiTs103097Module
OID	{itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) secHeaders(103097) core(1) major-version-3(3) minor-version-2(2)}
Link	https://forge.etsi.org/rep/ITS/asn1/sec_ts103097/-/raw/v2.2.1/EtsiTs103097Module.asn
SHA-256 hash	245a3c10c176497f658c6a4d9ec804d3226dda2ac10a4351e382ddb5afe9ff72

The ASN.1 extension module is identified by the Object Identifier {itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) secHeaders(103097) extension(2) major-version-1(1) minor-version-2(2)}. The module can be downloaded as a file as indicated in table A.2. The associated SHA-256 cryptographic hash digest of the referenced file offers a means to verify the integrity of that file.

Table A.2: ETSI TS 103 097 ASN.1 extension module information

Module name	EtsiTs103097ExtensionModule
OID	{itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) secHeaders(103097) extension(2) major-version-1(1) minor-version-2(2)}
Link	https://forge.etsi.org/rep/ITS/asn1/sec_ts103097/-/raw/v2.2.1/EtsiTs103097ExtensionModule.asn
SHA-256 hash	b94c9b373567dd9bfb15d61c8c206d5630f6b0c481ef41ddd574c96784a9eb1a

A.2 IEEE 1609.2 ASN.1 modules

This clause provides the relevant ASN.1 modules from IEEE Std 1609.2™ [1] (and its amendments), reprinted with permission from IEEE, Copyright © 2025.

The IEEE 1609.2 schema ASN.1 module is identified by the Object Identifier {iso(1) identified-organization(3) ieee(111) standards-association-numbered-series-standards(2) wave-stds(1609) dot2(2) base(1) schema(1) major-version-2(2) minor-version-7(7)}. The module can be downloaded as a file as indicated in table A.3. The associated SHA-256 cryptographic hash digest of the referenced file offers a means to verify the integrity of that file.

Table A.3: IEEE Std 1609.2 [1] schema ASN.1 module information

Module Name	Ieee1609Dot2
OID	{iso(1) identified-organization(3) ieee(111) standards-association-numbered-series-standards(2) wave-stds(1609) dot2(2) base(1) schema(1) major-version-2(2) minor-version-7(7)}
Link	https://forge.etsi.org/rep/ITS/asn1/ieee1609.2/-/raw/v2025/Ieee1609Dot2.asn
SHA-256 hash	82b5e35cbaadae1c6b2f087626b10d52afc73779c9ac6700c6445830d8824e0b

The IEEE 1609.2 base types ASN.1 module is identified by the Object Identifier {iso(1) identified-organization(3) ieee(111) standards-association-numbered-series-standards(2) wave-stds(1609) dot2(2) base(1) base-types(2) major-version-2(2) minor-version-5(5)}. The module can be downloaded as a file as indicated in table A.4. The associated SHA-256 cryptographic hash digest of the referenced file offers a means to verify the integrity of that file.

Table A.4: IEEE Std 1609.2 [1] base types ASN.1 module information

Module name	Ieee1609Dot2BaseTypes
OID	{iso(1) identified-organization(3) ieee(111) standards-association-numbered-series-standards(2) wave-stds(1609) dot2(2) base(1) base-types(2) major-version-2(2) minor-version-5(5)}
Link	https://forge.etsi.org/rep/ITS/asn1/ieee1609.2/-/raw/v2025/Ieee1609Dot2BaseTypes.asn
SHA-256 hash	bf2b3d66d394449319323f8a59d8084d963c3ad55d6790e1436ab897221690fe

Annex B (informative): Data encryption and decryption

B.1 General

This annex provides a high-level informative description of the cryptographic operations that are needed to be implemented to encrypt/decrypt data using mechanisms compliant with the present document. Data encryption is used for example to encrypt EC requests or AT requests according to ETSI TS 102 941 [i.2].

The data is encrypted with a freshly generated symmetric key (AES-CCM key). Then, this symmetric key is encrypted with the public key of the receiver (using ECIES) and transmitted alongside the encrypted message.

ECIES Scheme Parameters

Key Derivation Function (KDF): it is based on the function KDF2(S) = SHA256(S || counter || P1), applied twice and truncated to the right to obtain 48 octets and with counter being encoded as a 32 bits bit string:

TRUNCATE ("SHA256 (S || 0x00 00 00 01 || P1) || SHA256 (S || 0x00 00 00 02 || P1)", 48)

Symmetric encryption algorithm for ECIES: the XOR function is used (E(m, k) = m $\oplus$ k , E-1(c, k) = c $\oplus$ k). This should not be confused with AES-CCM which is used for message encryption. Here, k is the leftmost 16 bytes of the output from KDF2.

Message Authentication Code (MAC): MAC (m, km) = HMAC (m, km) see FIPS PUB 198-1, and km is the 32 bytes of the output from KDF2 following the first 16 bytes.

Elliptic curve parameters (p: curve prime, G: base point, q: base point order, O point at infinity).

NOTE: The symbol || indicates concatenation.

B.2 Data encryption

Input

m: Data: Message of the sender.

Kr: Encryption public key of the receiver.

Output

C: Encryption of the message m with AES-CCM concatenated with CCM authentication tag.

V: ECIES ephemeral public key.

c: encryption of the AES-CCM key used to encrypt m.

t: ECIES authentication tag.

Algorithm

The sender encrypts the message with AES-CCM:

a) Sender generates a random AES key A (16 octets).

b) Sender chooses a random nonce n, 12 octets.

c) Sender encrypts the message m with AES-CCM mode using the key A and the nonce n. The output is the value C that consists of the encrypted message followed by the encrypted authentication value.

The sender encrypts the key A with ECIES:

a) Sender generates an ephemeral private key r in [1, q-1], and the associated public key V=r.G, 33 octets if compressed.

b) Sender derives a shared secret S from receiver encryption public key Kr:
S = Px, where Px is the x-coordinate of the point P = r.Kr = (Px, Py). (verify that P != O, if not, back to previous step).

c) Sender then derives a set of keys ke and km with derivation algorithm: (ke || km)=KDF2(S), ke is 16 octets long, km is 32 octets long.

d) Sender encrypts the AES key: c=E(A, ke), c is 16 octets long.

e) Sender produces a tag on the encrypted message: t=MAC(c, km), t is 16 octets long.

Sender transmits to the receiver a message containing:

- The identifier for the recipient's certificate (cert_id), 8 octets.
- The encrypted message C.
- The encryption parameters (algorithm identifier aes_128_ccm, nonce n), 13 octets.
- The ephemeral public key (V).
- The encrypted key (c) with the associated tag (t).

B.3 Data decryption

Input

kr: Private key of the receiver.

n: Nonce used for AES-CCM.

C: Output of the AES-CCM encryption: encrypted message || CCM tag.

V: ECIES ephemeral public key.

c: encryption of the AES-CCM key used to encrypt m.

t: ECIES authentication tag.

Output

m: Data: Message of the sender or *failed* if any of the checks fails.

Algorithm

1) ECIES decryption to get the AES-CCM key A:

a) Receiver derives a shared secret S=Px, with (Px, Py)=kr.V or outputs *failed* if s.V = O.

b) Receiver derives the keys (ke || km)=KDF2(S).

c) Receiver checks that the tag t=MAC(c, km), if not, receiver outputs *failed*.

d) Receiver decrypts the key $A = E^{-1}(c, ke) = c \oplus ke$.

AES-CCM decryption and verification of the message:

a) Receiver decrypts the encrypted message part of C using AES-CCM with the key A.

b) Receiver checks the validity of the authentication tag computed on the plaintext, if it is not valid, receiver outputs *failed*.

NOTE: It is important that the nonce of CCM should be carefully chosen to never be used more than once for a given key. To avoid timing attacks, the output of decryption should be given after performing all the computations even if it outputs failed

Annex C (informative): Change history

Date	Version	Information about changes
2017-10	V1.3.1	Entirely re-worked version as a profile of IEEE 1609.2
2020-07	V1.4.1	The following CRs have been implemented: CR#01: Make IEEE 1609.2 HeaderInfo extensible in a way that reduces coordination burden CR#02: Add a note clarifying the use of CRLs CR#03: Delete region restriction for TLM certificate CR#04: Delete link certificate specification CR#05: Provisionally add data type to IEEE 1609.2 module to support ETSI TS 102 941 update CR#06: Correct typos CR#07: Delete ASN.1 type SingleEtsiTs103097Certificate CR#08: Move newly defined TS103097 data types from ETSI TS 102 941 to ETSI TS 103 097 CR#09: Adopt IEEE 1609.2b as reference CR#10: Allow the use of `pduFunctionalType` All the CRs are available at the following link: https://docbox.etsi.org/ITS/Open/Implemented%20CRs/TS%20103%20097/V1.4.1/
2020-11	V2.1.1	Amended V1.4.1 to support the use of implicit certificates (in addition to explicit certificates)
2024	V2.2.1	Updated reference [1] to IEEE 1609.2:2025 and applied the appropriate changes throughout the document. Added the misbehaviour authority certificate profile (based on ITSWG5(22)000009). Added the Etsi originated extensions to support the peer-to-peer request of a Full CTL (based on ITSWG5(23)069007r1). Added Annex B based on Annex F from ETSI TS 102 941

History

Version	Date	Status
V1.1.1	April 2013	Publication
V1.2.1	June 2015	Publication
V1.3.1	October 2017	Publication
V1.4.1	October 2020	Publication
V2.1.1	October 2021	Publication
V2.2.1	March 2026	Publication