ETSI TS 102 941 V1.1.1 (2012-06)

Technical Specification

Intelligent Transport Systems (ITS);
Security;
Trust and Privacy Management

Reference
DTS/ITS-0050015

Keywords
interoperability, ITS, management, 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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2012.
All rights reserved.

DECT™, **PLUGTESTS**™, **UMTS**™ and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE**™ are Trade Marks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, 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.

Foreword

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

1 Scope

The present document specifies the trust and privacy management for Intelligent Transport System (ITS) communications. Based upon the security services defined in TS 102 731 [1] and the security architecture define in TS 102 940 [5], it identifies the trust establishment and privacy management required to support security in an ITS environment and the relationships that exist between the entities themselves and the elements of the ITS reference architecture defined in EN 302 665 [2].

The present document identifies and specifies security services for the establishment and maintenance of identities and cryptographic keys in an Intelligent Transport System (ITS). Its purpose is to provide the functions upon which systems of trust and privacy can be built within an ITS.

2 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 at http://docbox.etsi.org/Reference.

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

2.1 Normative references

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

[1] ETSI TS 102 731: "Intelligent Transport Systems (ITS); Security; Security Services and Architecture".

[2] ETSI EN 302 665: "Intelligent Transport Systems (ITS); Communications Architecture".

[3] ETSI TS 102 867: "Intelligent Transport Systems (ITS); Security; Stage 3 mapping for IEEE 1609.2".

[4] ETSI TS 102 942: "Intelligent Transport Systems (ITS); Security; Access control".

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

[6] ISO/IEC 8824-1:2008: "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[7] ISO/IEC 8825-2:2008: "Information technology -- ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)".

[8] IEEE P1609.2/D12 (January 2012): "IEEE Draft Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages".

NOTE: Available from http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?reload=true&punumber=6140528.

[9] ETSI TS 102 943: "Intelligent Transport Systems (ITS); Security; Confidentiality services".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ISO/IEC 15408-2: "Information technology - Security techniques - Evaluation criteria for IT security; Part 2: Security functional components".

[i.2] ETSI TR 102 638: "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions".

[i.3] IETF RFC 4046: "Multicast Security (MSEC) Group Key Management Architecture".

[i.4] IETF RFC 4301: "Security Architecture for the Internet Protocol".

[i.5] IETF RFC 4302: "IP Authentication Header".

[i.6] IETF RFC 4303: "IP Encapsulating Security Payload (ESP)".

[i.7] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[i.8] IETF RFC 3547: "The Group Domain of Interpretation".

[i.9] IETF RFC 3830: "MIKEY: Multimedia Internet KEYing".

[i.10] IETF RFC 4535: "GSAKMP: Group Secure Association Key Management Protocol".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

anonymity: ability of a user to use a resource or service without disclosing the user's identity

authorization authority: authority that provides an ITS-S with permission to invoke ITS applications and services

canonical identifier: structured identifier that is globally unique

enrolment authority: authority that validates that an ITS-S can be trusted to function correctly

pseudonymity: ability of a user to use a resource or service without disclosing its user identity while still being accountable for that use

unlinkability: ability of a user to make multiple uses of resources or services without others being able to link these uses together

unobservability: ability of a user to use a resource or service without others, especially third parties, being able to observe that the resource or service is being used

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AA	Authorization Authority
CA	Certification Authority
CAM	Cooperative Awareness Message
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DENM	Decentralized Environmental Notification Message
EA	Enrolment Authority

ITS	Intelligent Transport System
ITS-AID	ITS Application ID
ITS-S	ITS Station
MSEC	Multicast Security
PKI	Public Key Infrastructure
PSID	Provider Service Identifier
SA	Security Association
SSP	Service Specific Permissions
TLS	Transport Layer Security

4 ITS authority hierarchy

4.1 Overview

Trust and privacy management requires secure distribution and maintenance (including revocation when applicable) of trust relationships, which may be enabled by specific security parameters that include 3rd party certificates of proof of identity or other attributes such as pseudonym certificates. Public key certificates and Public Key Infrastructure (PKI) are used to establish and maintain trust between the ITS-S and other ITS stations and authorities.

TS 102 731 [1] defines the security management roles taken by:

- manufacturers:
  - insert an ITS authoritative identity (canonical identifier) into each ITS-S;
- Enrolment Authorities (EA):
  - verify an ITS Station (ITS-S) as a whole; and
- Authorization Authorities (AA):
  - authorize an ITS-S to use a particular application, service, or privilege.

Separation of enrolment (identification and authentication) and authorization has been shown in TS 102 731 [1] as an essential component of privacy management and provides protection against attacks on a user's privacy. However, it is possible for the EA role to be delegated to the manufacturer and for the EA and AA roles to be assumed by a single authority.

NOTE: EN 302 665 [2] defines an ITS registration authority role to protect against the distribution of malicious ITS applications. Registration authorities are responsible for registering and managing ITS applications exclusively and are not involved in operational security management.

4.2 ITS authorities

4.2.1 Enrolment Authority

The EA issues a proof of identity authenticating the canonical identifier issued to the ITS-S. The proof of identity does not reveal the canonical identifier to a 3rd party and may be used by the ITS-S to request authorization of services from an AA.

The functions provided by the EA are as follows:

- the authentication of the canonical identifier of an ITS-S;
- the provision of proof of authentication of the ITS-S.

4.2.2 Authorization Authority

An ITS-S that has enrolled with, and been authenticated by, an EA may apply to an AA for specific permissions within the enrolment authority's domain and the AA's authorization context. These privileges are denoted by means of authorization credentials in the form of IEEE 1609.2 [8] authorisation certificates. Each authorization certificate specifies a particular authorization context which comprises a set of permissions.

EXAMPLE 1: An authorization certificate might grant permission to an ITS-S to broadcast messages from a particular message set. Alternatively, it might grant permission to claim certain privileges.

The authorization context is specified either by explicitly encoding the permissions granted or by including a reference to a known policy that specifies the context.

NOTE: An AA will normally be responsible for a particular set of contexts which may be specified by one or more of the following:

- application (for example, cooperative awareness applications for personal user vehicles, emergency service vehicles or tolling);
- time period;
- geographic region (nation, state, locality); or
- any other criteria that can be encoded.

The authorization system may comprise a hierarchy of authorization authorities with lower-layer authorities authorizing ITS stations and higher-layer authorities authorizing lower-level authorities.

EXAMPLE 2: The following three layer structure might be appropriate for official use vehicles:

a) ITS global (National) authorization authority;

b) ITS regional authorization authority; and

c) ITS local authorization authority.

EXAMPLE 3: For personal user vehicles, it might be appropriate to have a single authorization authority (either national or system-wide) for CAMs and DENMs, because short certificate chains reduce the packet size associated with authorization data.

An AA should be unable to link the proof of authentication to the canonical identifier of an ITS-S without the collusion of the EA that performed the verification of the canonical identifier of the ITS-S.

4.2.3 Root CA

Each CA hierarchy (for EA or AA) has at its summit a Root Certificate, which is the ultimate root of trust for all certificates within that hierarchy. In order to trust an incoming message, an ITS-S must have access at least to the root certificate at the summit of the hierarchy for the authorization certificate attached to the message. The ITS-S may obtain root certificates during the manufacture or maintenance lifecycle stages described in clauses 6.1.1 to 6.1.4 respectively. In principle root certificate information may be distributed over the air through a cross-certification process, but the present document does not specify messages to support this use case.

5 Privacy in ITS

ISO/IEC 15408-2 [i.1] identifies 4 key attributes that relate to privacy:

- anonymity;
- pseudonymity;
- unlinkability; and
- unobservability.

Anonymity alone is insufficient for protection of an ITS user's privacy and unsuitable as a solution for ITS, as one of the main requirements of ITS is that the ITS-S should be observable in order to provide improved safety. Consequently, pseudonymity and unlinkability offer the appropriate protection of the privacy of a sender of basic ITS safety messages (CAM and DENM). Pseudonymity ensures that an ITS-S may use a resource or service without disclosing its identity but can still be accountable for that use [i.1]. Unlinkability ensures that an ITS-S may make multiple uses of resources or services without others being able to link them together [i.1].

Pseudonymity shall be provided by using temporary identifiers in ITS safety messages, and never transmitting the station's canonical identifier in communications between ITS stations. Unlinkability can be achieved by limiting the amount of detailed immutable (or slowly changing) information carried in the ITS safety message, thus preventing the possible association of transmissions from the same vehicle over a long time period (such as two similar transmissions broadcast on different days).

ITS Privacy is provided in two dimensions:

(i) privacy of ITS registration and authorisation signalling:

- ensured by permitting knowledge of the canonical identifier of an ITS-S to only a limited number of authorities;
- provided by the separation of the duties and roles of ITS authorities into an entity verifying the canonical identifier known as the Enrolment Authority (EA) and an entity responsible for authorising and managing services known as the Authorization Authority (AA);

(ii) privacy of communications between ITS-Ss.

6 Trust and privacy management

6.1 ITS-S Security Lifecycle

The ITS-S Security Lifecycle includes the following stages:

- manufacture;
- enrolment;
- authorization;
- maintenance.

6.1.1 Manufacture

As part of the ITS-S manufacturing process, the following information elements associated with the identity of the station shall be established within the ITS-S itself and within the Enrolment Authority (EA).

- in the ITS-S, the following information elements shall be established using a physically secure process. The specification of this physically secure process is out of scope for the present document.
    - a canonical identifier which is globally unique (see note 1);
    - contact information for the EA and AA which will issue certificates for the ITS-S:
        - network address;
        - public key certificate;
    - the set of current known trusted EA certificates which the ITS-S may use to initiate the enrolment process;
    - the set of current known trusted AA certificates which the ITS-S may use to trust communications from other ITS-S;

- a public/private key pair for cryptographic purposes; and
- optionally, a canonical certificate which associates the canonical identifier with the public key of the ITS-S and the certificate chain back the root authority.

NOTE 1: the management of the canonical identifier and the means to guarantee uniqueness are not addressed in the present document.

- in the EA, the following four items of information, all associated with each other (see note 2):
  - the permanent canonical identifier of the ITS-S;
  - the enrolment identifier issued in the enrolment certificate;
  - the location of profile information for the ITS-S; and
  - the public key from the key pair belonging to the ITS-S.

NOTE 2: The process for establishing this information within the ITS-S and the EA is beyond the scope of the present document.

6.1.2 Enrolment

The ITS-S requests its enrolment certificate from the EA (see clause 6.2.2.2).

6.1.3 Authorization

Having received the enrolment credentials, the ITS-S requests its authorization certificate(s) from the AA (see clause 6.2.2.3).

6.1.4 Maintenance

If an EA or AA is added to or removed from the system, the associated authority (not defined by the present document) should inform enrolled ITS-Ss of this change. The process for achieving this is beyond the scope of the present document but possible methods include:

- sending a certificate revocation list as specified in IEEE 1609.2 [8] across a wireless interface; or
- providing information to a trusted maintenance entity to enable it to update an individual ITS-S in a controlled environment.

6.2 Public Key Infrastructure

6.2.1 Assumption and requirements

The present document assumes the ITS security reference model that is described in TS 102 940 [5].

6.2.2 Message Sequences

6.2.2.1 Introduction

The message sequences specified in clauses 6.2.2.2 and 6.2.2.3 for ITS-S enrolment and authorization are based on the protocol messages defined in TS 102 867 [3] and IEEE 1609.2 [8]. Each of the messages shall be encoded into a 1609Dot2Data (see clause 6 in IEEE 1609.2 [8]) with the appropriate enumeration set into its "type" field to indicate whether it is encrypted, signed or unsecured. Figure 1 shows an example of the use of 1609Dot2Data structure to provide enrolment/authorization requests and responses.

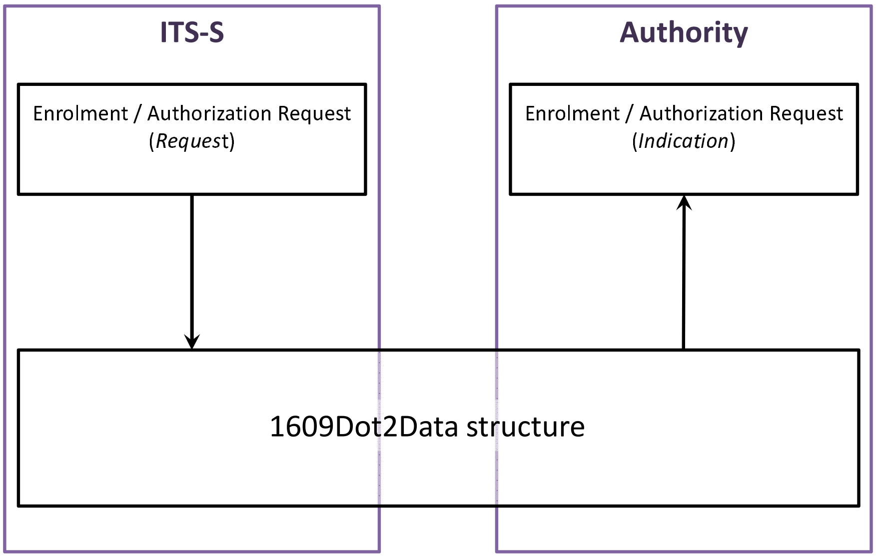

Figure 1: illustration of using 1609Dot2Data structure

6.2.2.2 Enrolment Request

The Enrolment Request message shall be sent by an ITS-S to the Enrolment Authority (EA) across the interface at reference point S_3 (see Figure 7 in TS 102 940 [5]) to request an enrolment certificate to be used in a subsequent authorization request. Figure 2 shows an example of a message sequence for a successful or unsuccessful enrolment request.

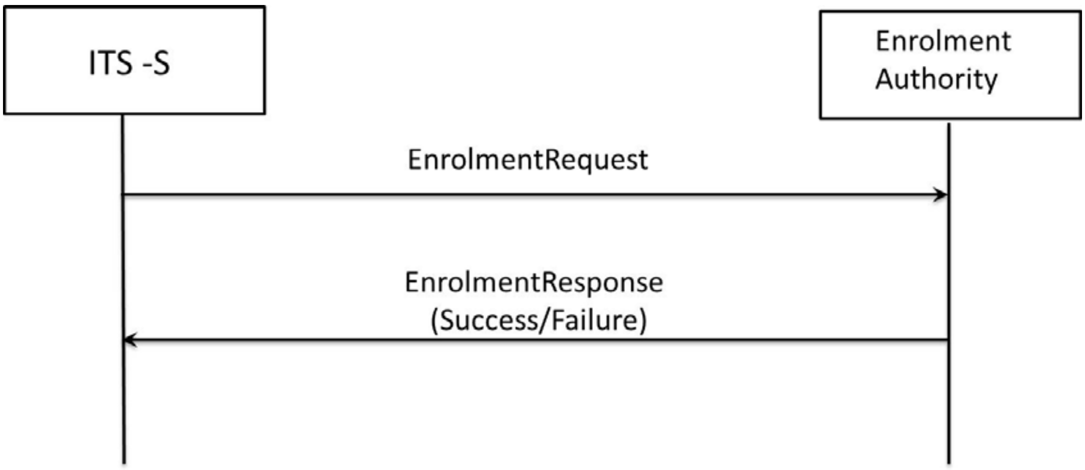

Figure 2: Message sequence for enrolment request and response

The contents of the ITS-S Enrolment Request message shall be as described in Table 1 and shall be implemented as an IEEE 1609.2 ToBeEncrypted message of type certificate_request [8]. For information only, the content of the ITS-S Enrolment request message is described using ASN.1 [6], [7] in clause A.2.

Table 1: Contents of ITS-S EnrolmentRequest message

Field name	Description	Contents (see note)	IEEE 1609.2 [8] mapping
signerEnrolRequest	The canonical certificate or the public/private key pair that uniquely identifies the ITS-S (initially provided during the bootstrap process)	A certificate or certificate chain that allows the EA to determine which keying material to use to verify the request	Field: *info* Type: *SignerIdentifier* Constraints: • *type* shall be set to a value of either *certificate* or *certificate_chain*
enrolCertRequest	The certificate request	Start time End time ITS-S' public key Certificate specific data	Field: *unsigned_csr* Type: *ToBeSignedCertificateRequest* Constraints: • *subject_type* shall be set to the value *sec_data_exch_csr* • *cf* shall not be set to include the *encryption_key* flag • *CertSpecificData . SecDataExchCaScope .permitted_subject_types* shall be set to either *sec_data_exch_anonymous* or *sec_data_exch_identified_localized*. • *CertSpecificData . SecDataExchCaScope .name* shall be structured as the country code plus ITS service provider code plus ITS-S identifier • *CertSpecificData .SecDataExchCaScope .region* shall be an identifier for the requested area of validity of the enrolment credentials • *CertSpecificData .SecDataExchCaScope .PsidArray.type* shall be set to *specified*. • *CertSpecificData . SecDataExchCaScope .PsidArray .permissions_list* shall contain a list of the ETSI ITS-AIDs to be supported.
signature	Signature of the enrolment request	The cryptographic signature over all fields of the enrolment request created using the private key belonging to the ITS-S	Field: *signature* Type: *Signature* Constraints: • shall not be *set* to the value *unknown*
NOTE: The whole EnrolmentRequest message shall be encrypted using an IEEE 1609.2 [8] approved algorithm and the public key provided by the enrolment authority.			

The EnrolmentResponse message shall be sent by the EA to the ITS-S across the interface at reference point S_3 in response to a received EnrolmentRequest message.

The contents of the successful ITS-S Enrolment Response message shall be as described in Table 2 and shall be implemented as an IEEE 1609.2 ToBeEncrypted message of type certificate_response [8]. For information only, the content of the ITS-S Enrolment Response message is described using ASN.1 in clause A.2.

Table 2: Contents of a successful ITS-S EnrolmentResponse message

Field name	Description	Contents (see note 1)	IEEE 1609.2 [8] mapping
ackRequest	An indication of whether an acknowledgement is requested by the Enrolment Authority	(see note 2)	Field: *f* Type: *flags* Constraints: • Shall be set to the value *Not requested*.
signedCertChain	The enrolment certificate chain	The enrolment certificate containing the pseudonymous identifier to be used by the ITS-S; and the chain of certificates back to the originating enrolment CA	Field: *certificate_chain* Type: *Certificate* Constraints: • None
crlPath	The CRLs required to validate a certificate	Empty as public certificates are not listed in CRLs. (see note 2)	Field: *crl_path* Type: *Crl* Constraints: • Shall be set to the value *empty*
NOTE 1: The whole EnrolmentResponse message shall be encrypted using an IEEE 1609.2 [8] approved algorithm. NOTE 2: This element is included only for compatibility with IEEE 1609.2 [8].			

The contents of the unsuccessful ITS-S Enrolment Response message shall be as described in Table 3 and shall be implemented as an IEEE 1609.2 ToBeEncrypted message of type certificate_request_error [8]. For information only, the content of the ITS-S Enrolment Request message is described using ASN.1 in clause A.2.

Table 3: Contents of an unsuccessful ITS-S EnrolmentResponse message

Field name	Description	Contents (see note)	IEEE 1609.2 [8] mapping
signerEnrolResp	The enrolment authority identified as signer of this error message	A certificate or certificate chain that allows the ITS-S to determine which keying material to use to verify the response	Field: *signer* Type: *SignerIdentifier* Contraints: • *type* shall *be* set to a value of either *certificate* or *certificate_chain*
requestHash	Allows the requester to link this response to the request	The first 10 bytes of the SHA-256 hash calculated over the plaintext EnrolmentRequest before the request is encrypted	Field: *request_hash* Type: *opaque* Constraints: • Shall be of length 10 octets
enrolResult	The error code of the unsuccessful enrolment response		Field: *reason* Type: *CertificateRequestErrorCode* Constraints: • None
signature	The enrolment authority's signature over the response	The cryptographic signature of the unsuccessful EnrolmentResponse created using the private key belonging to the enrolment authority	Field: *signature* Type: *Signature* Constraints: • shall not be set to the value *unknown*
NOTE: The whole EnrolmentResponse message shall be encrypted using an IEEE 1609.2 [8] approved algorithm.			

6.2.2.3 Authorization Request

The Authorization Request message shall be sent by an ITS-S to the Authorization Authority (AA) across the interface at reference point S_2 (see Figure 7 in [5]) to request an authorization certificate to be used in a subsequent ITS communications. Figure 3 shows an example of a message sequence for a successful or unsuccessful authorization request.

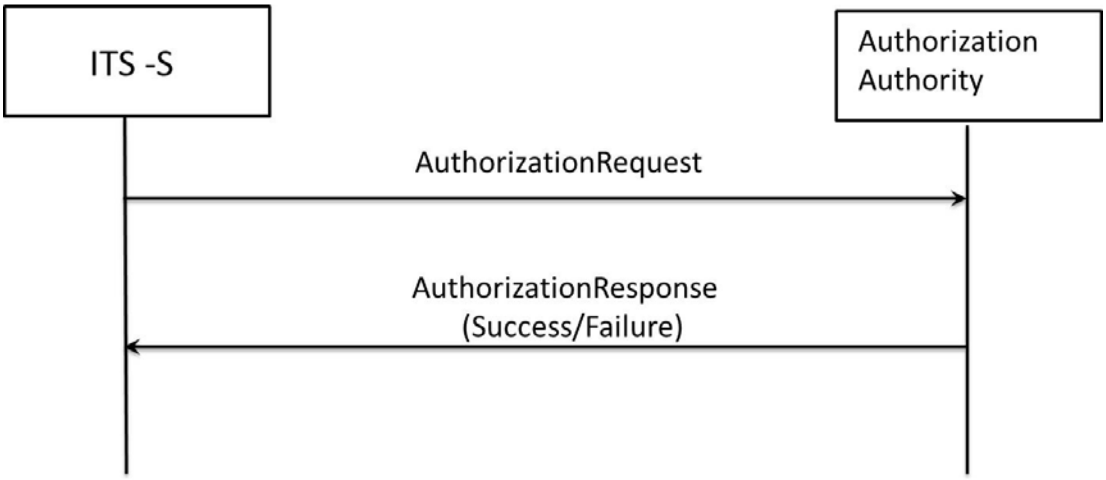

Figure 3: Message sequence for authorization request and response

The contents of the ITS-S Authorization Request message shall be as described in Table 4 and shall be implemented as an IEEE 1609.2 ToBeEncrypted message of type certificate_request [8]. For information only, the content of the ITS-S Authorization Request message is described using ASN.1 [6], [7] in clause A.2.

Table 4: Contents of ITS-S AuthorizationRequest message

Field name	Description	Contents (note 1)	IEEE 1609.2 [8] mapping
signerAuthRequest	The enrolment certificate chain	The enrolment certificate containing the pseudonymous identifier to be used by the ITS-S; and the chain of certificates back to the originating enrolment CA	Field: *info* Type: *SignerIdentifier* Constraints: • *type* shall be set to a value of either *certificate* or *certificate_chain*
authCertRequest	The certificate request	• Start time • End time • ITS-S' authorization certificate public key • Subject name - Optional (note 2) • Additional data - Optional (note 3) • Permissions • Region of validity - Optional (note 4)	Field: *unsigned_csr* Type: *ToBeSignedCertificateRequest* Constraints: • subject_type shall be set to the value sec_data_exch_anonymous, sec_data_exch_identified_not_localized, or sec_data_exch_identified_localized • cf shall not be set to include the encryption_key flag • PsidSspArray.type shall be set to specified
signature	Signature of the authorization request	The cryptographic signature over all fields of the enrolment request created using the private enrolment certificate key belonging to the ITS-S	*Field*: signature *Type*: Signature Constraints: • shall not be set to the value *unknown*
NOTE 1: The whole AuthorizationRequest message shall be encrypted using an IEEE 1609.2 [8] approved algorithm and the public key provided by the authorization authority. NOTE 2: Shall be included if *subject_type* in the IEEE 1609.2 *unsigned_csr* field is set to either *sec_data_exch_identified_not_localized* or *sec_data_exch_identified_localized.* NOTE 3: Shall be included if *subject_type* in the IEEE 1609.2 *unsigned_csr* field is set to *sec_data_exch_anonymous.* NOTE 4: Shall be included if *subject_type* in the IEEE 1609.2 *unsigned_csr* field is set to either *sec_data_exch_anonymous* or *sec_data_exch_identified_localized.*			

The AuthorizationResponse message shall be sent by the AA to the ITS-S across the interface at reference point S_2 in response to a received AuthorizationRequest message.

The contents of the successful ITS-S Authorisation Response message shall be as described in Table 5 and shall be implemented as an IEEE 1609.2 ToBeEncrypted message of type certificate_response [8]. For information only, the content of the ITS-S Authorisation Response message is described using ASN.1 in clause A.2.

Table 5: Contents of a successful ITS-S AuthorizationResponse message

Field name	Description	Contents (see note 1)	IEEE 1609.2 [8] mapping
ackRequest	An indication of whether an acknowledgement is requested by Authorization Authority	(see note 2)	Field: *f* Type: *flags* Constraints: • Shall be set to the value *Not requested*.
signedCertChain	The authorization certificate chain	The authorization certificate; and the chain of certificates back to the top authorization CA	Field: *certificate_chain* Type: *Certificate* Constraints: • Shall be set to comply with AuthorizationRequest's authCertRequest.
reconPrivateValue	The reconstruction private value to derive the private key	Optional field	Field: *recon_priv* Type: *opaque* Constraints: • Only available if *version_and_type* equals *implicit certificate* (3)
crlPath	The CRLs required to validate a certificate	CRL	Field: *crl_path* Type: *Crl* Constraints: • None
NOTE 1: The whole AuthorizationResponse message shall be encrypted using an IEEE 1609.2 approved algorithm. NOTE 2: This element is included only for compatibility with IEEE 1609.2 [8].			

The contents of the unsuccessful ITS-S Authorization Response message shall be described in Table 6 and shall be implemented as an IEEE 1609.2 ToBeEncrypted message of type certificate_request_error [8]. For information only, the content of the ITS-S Authorization Request message is described using ASN.1 in clause A.2.

Table 6: Contents of an unsuccessful ITS-S AuthorizationResponse message

Field name	Description	Contents (see note)	IEEE 1609.2 [8] mapping
signerAuthResp	The authorization authority identified as signer of this error message	A certificate or certificate chain that allows the ITS-S to determine which keying material to use to verify the response	Field: *signer* Type: *SignerIdentifier* Contraints: • *type* shall *be* set to a value of either *certificate* or *certificate_chain*
requestHash	Allows the requester to link this response to the request	The first 10 bytes of the SHA-256 hash calculated over the plaintext AuthorizationRequest before the request is encrypted	Field: *request_hash* Type: *opaque* Constraints: • Shall be of length 10 octets
authResult	The error code of the unsuccessful enrolment response		Field: *reason* Type: *CertificateRequestErrorCode* Constraints: • None
signature	The authorization authority's signature over the response	The cryptographic signature of the unsuccessful AuthorizationResponse created using the private key belonging to the authorization authority	Field: *signature* Type: *Signature* Constraints: • shall not be set to the value *unknown*
NOTE: The whole AuthorizationResponse message shall be encrypted using an IEEE 1609.2 approved algorithm.			

7 Security association and key management between ITS Stations

A detailed set of use case examples for ITS applications is presented in TR 102 638 [i.2]. In addition, TS 102 940 [5] categorizes the application communication (addressing) patterns used as:

- Broadcast;
- Multicast;
- Unicast.

In contrast to the strictly safety-related broadcast applications (CAM and DENM), multicast and unicast applications are assumed to be offered by several providers and, possibly, to be commercially sensitive. Therefore, the requirements depend heavily on the specific application and the respective business model.

With the exception of broadcast applications, all other multicast and unicast communications can use either asymmetric or symmetric key systems to provide for Security Association (SA) lifecycle and the related key management (registration, key establishment, updates and removal).

Unicast and multicast applications shall use link layer encryption and regular changes of the ITS MAC addresses to protect the privacy of the ITS-S (and its user) as well as all higher layer information from radio channel eavesdropping. Further details can be found in TS 102 942 [4] and TS 102 943 [9].

7.1 Broadcast SAs

Broadcast applications such as CAM and DENM require authentication, authorisation and integrity but not confidentiality. Senders of CAM and DENM shall obtain this service by signing with an authorization certificate using the mechanisms of IEEE 1609.2 [8] (see clause 6.2.2.3 and Table 5, as well as TS 102 867 [3]). Figure 4 illustrates the use of the authorization certificate to sign a CAM or DENM between ITS stations. The "SignerInfo" field in Figure 4 is a 1609.2 field that contains either the certificate or a reference to it.

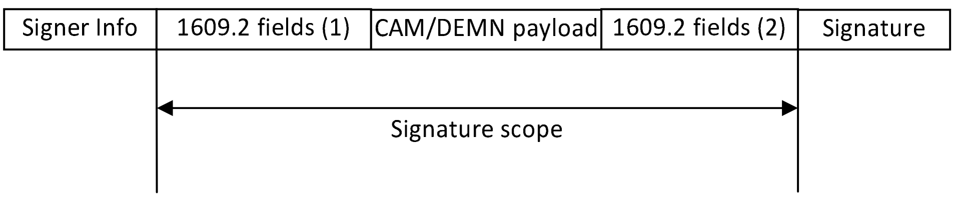

Figure 4: CAM and DENM signed using authorization certificates

7.2 Multicast SAs

Multicast applications such as public transport information and Point of Interest notification services require secure group communications with message authentication, authorisation and encryption depending on that group's particular security policy.

An ITS-S may join a multicast group using an authorisation certificate (see clause 6.2.2.3 and Table 5) followed, possibly, by further registration steps.

The key management for multicast applications can be controlled by the multicast service provider or a separate security manager. Such key management may be application-specific or it may use a standard multicast key management system such as the IETF Multicast Security (MSEC) Group Key Management Architecture [i.3], [i.8], [i.9] and [i.10].

7.3 Unicast SAs

Unicast applications such as automatic access control, parking management and media downloading services require secure unicast communications with message authentication, authorisation and encryption.

An ITS-S may join such services using its authorisation certificate (see clause 6.2.2.3 and Table 5) followed, possibly, by further registration protocol steps.

Unicast key management may be application-specific or it may use a standard key management systems such as network layer security using IPsec as defined by the IETF [i.4], [i.5] and [i.6]. Also, security in the transport layer can be provided using methods such as the IETF Transport Layer Security (TLS) [i.7].

Annex A (informative): ITS security messages specified in ASN.1

A.1 ITS trust and privacy messages specified in ASN.1

The ASN.1 [6] modules in this annex specify data types for ITS trust and privacy services together with useful ASN.1 value notations. The ASN.1 is included here only for guidance. Messages associated with ITS security services should comply with the structures specified here but the definitive encoding of messages in an implementation of the present document is specified in clause 5 of IEEE 1609.2 [8].

A.2 Enrolment and authorization message structures

```
ITStandp0v0 { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) itstandp(2941)
operation(0) version0(0)}

DEFINITIONS AUTOMATIC TAGS  ::= BEGIN

  EnrolmentRequest
        ::= SEQUENCE  {  -- corresponds to the CertificateRequest in 1609.2
                        signerEnrolRequest   SignerIdentifier,
                        enrolCertRequest     ToBeSignedEnrolmentCertificateRequest,
                        signature            Signature
                      }
           -- The Enrolment Request message shall be encrypted using an IEEE 1609.2
           -- approved algorithm and the public key provided by the EAS

  EnrolmentResponse
        ::= CHOICE  {
                      successfulEnrolment SuccesfulEnrolment,
                      failedEnrolment     FailedEnrolment
                    }

  AuthorizationRequest
        ::= SEQUENCE  {
                       signerAuthRequest SignerIdentifier,
                       authCertRequest   CHOICE {
                                              anonRequest       ToBeSignedAuthCertReq-anon,
                                              idNonLocRequest   ToBeSignedAuthCertReq-idNonLoc,
                                              idLocRequest      ToBeSignedAuthCertReq-idLoc
                                             },
                        signature         Signature
                      }
           -- The Authorization Request message shall be encrypted using an IEEE 1609.2
           -- approved algorithm and the public key provided by the ITS-S

  AuthorizationResponse
        ::= CHOICE  {
                      successfulExplicitAuthorization  SuccessfulExplicitAuthorization,
                      successfulImplicitAuthorization  SuccessfulImplicitAuthorization,
                      failedAuthorization              FailedAuthorization
                    }

  ToBeSignedEnrolmentCertificateRequest
        ::= SEQUENCE  {
                        versionAndType          ImplicitOrExplicit,
                        requestTime             ItsTime,
                        subjectType             SecDataExchCsr,
                        cf                      UseStartVal-AndOr-Lifetime,
                        enrolCertSpecificData   SecDataExchCaCertSpecificData,
                        expiration              ItsTime,
                        startValidity           ItsTime OPTIONAL,
                        lifetime                CertificateDuration OPTIONAL,
                        verificationKey         PublicKey,
                        responseEncryptionKey   PublicKey
                      }

  ToBeSignedAuthCertReq-anon
```

```
      ::= SEQUENCE  {
                      versionAndType         ImplicitOrExplicit,
                      requestTime            ItsTime,
                      subjectType            SecDataExchAnon,
                      cf                     UseStartVal-AndOr-Lifetime,
                      authCertSpecificData   SecDataExchAnonymousCertSpecificData,
                      startValidity          ItsTime OPTIONAL,
                      lifetime               CertificateDuration OPTIONAL,
                      responseEncryptionKey  PublicKey
                    }

ToBeSignedAuthCertReq-idNonLoc
      ::= SEQUENCE  {
                      versionAndType         ImplicitOrExplicit,
                      requestTime            ItsTime,
                      subjectType            SecDataExchIdNonLoc,
                      cf                     UseStartVal-AndOr-Lifetime,
                      authCertSpecificData   SecDataExchIdentifiedNotLocalizedCertSpecificData,
                      startValidity          ItsTime OPTIONAL,
                      lifetime               CertificateDuration OPTIONAL,
                      responseEncryptionKey  PublicKey
                    }

ToBeSignedAuthCertReq-idLoc
      ::= SEQUENCE  {
                      versionAndType         ImplicitOrExplicit,
                      requestTime            ItsTime,
                      subjectType            SecDataExchIdLoc,
                      cf                     UseStartVal-AndOr-Lifetime,
                      authCertSpecificData   SecDataExchIdentifiedLocalizedCertSpecificData,
                      startValidity          ItsTime OPTIONAL,
                      lifetime               CertificateDuration OPTIONAL,
                      responseEncryptionKey  PublicKey
                    }

SuccesfulEnrolment
      ::= SEQUENCE {
                      ackRequest        NotRequested,
                      signedCertChain   CertificateChain,
                      crlPath           NullCrl
                    }
FailedEnrolment ::=  FailedCertResponse

SuccessfulExplicitAuthorization
      ::= SEQUENCE  {
                      ackRequest        NotRequested,
                      signedCertChain   CertificateChain,
                      crlPath           Crl
                    }

SuccessfulImplicitAuthorization
      ::= SEQUENCE  {
                      ackRequest        NotRequested,
                      signedCertChain   CertificateChain,
                      reconPrivateValue OCTET STRING,
                      crlPath           Crl
                    }

FailedAuthorization ::= FailedCertResponse

FailedCertResponse
      ::= SEQUENCE  {
                      signerEnrolResp   SignerIdentifier,
                      requestHash       OCTET STRING (SIZE (10)),
                      enrolResult       CertificateRequestErrorCode,
                      signature         Signature
                    }

PublicKey
      ::= SEQUENCE  {
                      algorithm             EcdsaNistWithShaAlgorithms,
                      public-key            EccPublicKey
                    }

PKAlgorithm ::= INTEGER
    ecdsaNistp224WithSha224 PKAlgorithm ::= 0
    ecdsaNistp256WithSha256 PKAlgorithm ::= 1
    eciesNistp256           PKAlgorithm ::= 2
```

```
    unknownAlgorithm        PKAlgorithm ::= 3
  EcdsaNistWithShaAlgorithms ::= PKAlgorithm ( ecdsaNistp224WithSha224 | ecdsaNistp256WithSha256 )


AcknowledgeRequest  ::= BOOLEAN
    requested     AcknowledgeRequest  ::= TRUE
    notRequested  AcknowledgeRequest  ::= FALSE
  Requested     ::= AcknowledgeRequest  (requested)
  NotRequested  ::= AcknowledgeRequest  (notRequested)

SignerIdentifier
      ::= SEQUENCE  {
                      type    SignerIdType,
                      digest  CertId8,
                      id      OCTET STRING
                    }

SignerIdentifierType  ::= Integer8
    self                            SignerIdentifierType  ::= 0
    certificateDigestWithEcdsap224  SignerIdentifierType  ::= 1
    certificateDigestWithEcdsap256  SignerIdentifierType  ::= 2
    certificate                     SignerIdentifierType  ::= 3
    certificateChain                SignerIdentifierType  ::= 4
    unknownSigner                   SignerIdentifierType  ::= 5

  Self                            ::= SignerIdentifierType  (self)
  CertificateDigestWithEcdsap224  ::= SignerIdentifierType  (certificateDigestWithEcdsap224)
  CertificateDigestWithEcdsap256  ::= SignerIdentifierType  (certificateDigestWithEcdsap256)
  Cert                            ::= SignerIdentifierType  (certificate)
  CertChain                       ::= SignerIdentifierType  (certificateChain)
  UnknownSignerIdType             ::= SignerIdentifierType  (unknownSigner)
  SignerIdType                    ::= SignerIdentifierType  (certificate | certificateChain)

CertId8  ::= OCTET STRING (SIZE (8))

Time32  ::= INTEGER (1..4294967295)

CertificateDuration
      ::= SEQUENCE  {
                      timeUnit  TimeUnit,
                      timeValue INTEGER (0..8191)
                    }

TimeUnit  ::= Integer3
  seconds       TimeUnit  ::= 0
  minutes       TimeUnit  ::= 1
  hours         TimeUnit  ::= 2
  sixtyHours    TimeUnit  ::= 3
  years         TimeUnit  ::= 4

ExplicitCertificate
      ::= SEQUENCE  {
                      versionAndType      ExplicitCert,
                      unsignedCertificate CHOICE
                                                { rootCert          UnsignedRootCertificate,
                                                  intermediateCert UnsignedIntermediateCertificate
                                                },
                      signature             Signature
                    }
ImplicitCertificate
      ::= SEQUENCE  {
                      versionAndType        ImplicitCert,
                      unsignedCertificate   UnsignedIntermediateCertificate,
                      reconstructionValue   EccPublicKey
                    }

RootCertificate ::= ExplicitCertificate

IntermediateCertificate
      ::= CHOICE  {
                    explicitCertificate ExplicitCertificate,
                    implicitCertificate ImplicitCertificate
                  }

CertificateChain
      ::= SEQUENCE  {
                      intermediateCerts   SEQUENCE OF IntermediateCertificate,
                      rootCertificate     RootCertificate
```

```
                    }

Certificate
      ::= CHOICE {
                    rootCertificate         RootCertificate,
                    intermediateCertificate IntermediateCertificate
                 }
ToBeSignedCertificate
      ::= CHOICE  { unsignedIntermediateCert  IntermediateCertificate,
                    unsignedRootCert          RootCertificate
                  }

UnsignedIntermediateCertificate
      ::= SEQUENCE {
                      subjectType IntermediateCert,
                      cf          UseStartVal-AndOr-Lifetime,
                      scope       CHOICE {
                                          secDataExchCaScope          SecDataExchCaScope,
                                          anonymousScope              AnonymousScope,
                                          identifiedNotLocalizedScope IdentifiedNotLocalizedScope,
                                          identifiedScope             IdentifiedScope
                                         },
                      expiration        ItsTime,
                      lifetime          CertificateDuration OPTIONAL,
                      start-validity    ItsTime OPTIONAL,
                      crl-series        CrlSeries,
                      verification-key  PublicKey OPTIONAL
                    }

UnsignedRootCertificate
      ::= SEQUENCE  {
                      subjectType       RootCa,
                      cf                UseStartVal-AndOr-Lifetime,
                      scope             RootCaScope,
                      expiration        ItsTime,
                      lifetime          CertificateDuration OPTIONAL,
                      start-validity    ItsTime OPTIONAL,
                      crl-series        CrlSeries,
                      verification-key  PublicKey
                    }

RootCaScope
      ::= SEQUENCE {
                      name                    IA5String (SIZE (0..31)),
                      permittedSubjectTypes   SubjectTypeFlags,
                      secureDataPermissions   PsidArray,
                      region                  GeographicRegion
                    }

SecDataExchCaScope
      ::= SEQUENCE  {
                      eaId                  IA5String (SIZE (0..32)),       -- name of EA
                      permittedSubjectTypes SecDataExchCaTypes,
                      permissions           PsidArray,
                      region                GeographicRegion
                    }

IdentifiedScope
      ::= SEQUENCE  {
                      subject-name      OCTET STRING,
                      permissions       PsidSspArray,
                      region            GeographicRegion
                    }

IdentifiedNotLocalizedScope
      ::= SEQUENCE  {
                      subject-name      OCTET STRING,
                      permissions       PsidSspArray
                    }

AnonymousScope
      ::= SEQUENCE  {
                      additional-data   OCTET STRING,
                      permissions       PsidSspArray,
                      region            GeographicRegion
                    }

CertificateRequestErrorCode
```

```
      ::= ENUMERATED  {
                        verification-failure(0),
                        csr-cert-expired(1),
                        csr-cert-revoked(2),
                        csr-cert-unauthorized(3),
                        request-denied(4),
                        csr-cert-unknown (5),
                        canonical-identity-unknown (6)
                      }

PsidArray
      ::= SEQUENCE {
                      type                SpecifiedArray,
                      permissions-list    PsidList
                    }

Psid
      ::= CHOICE  {
                    its-aid       ITS-AID,
                    port          Port
                  }

PsidList  ::= SEQUENCE OF Psid

ITS-AID ::= OCTET STRING (SIZE (1..4))

Port
      ::= SEQUENCE  {
                      portIndicator       PortIndicator,
                      portNumber          PortNumber
                    }
PortIndicatorType ::= OCTET STRING (SIZE (1))
portIndicator PortIndicatorType ::= 'DF'H
PortIndicator ::= PortIndicatorType (portIndicator)
PortNumber ::= OCTET STRING (SIZE (2))

PsidSspArray
      ::= SEQUENCE  {
                      type                SpecifiedArray,
                      permissions-list    PsidSspList
                    }

PsidSsp
      ::= SEQUENCE  {
                      its-aid     ITS-AID,
                      ssp         SSP
                    }

PsidSspList ::= SEQUENCE OF PsidSsp

SSP ::= OCTET STRING

ArrayType  ::= Integer8
    fromIssuer  ArrayType  ::= 0
    specified   ArrayType  ::= 1
    unknownType ArrayType  ::= 2

  SpecifiedArray ::= ArrayType (specified)

Signature  ::= EcdsaSignature

EcdsaSignature
      ::= SEQUENCE {
                      r     EccPublicKey,
                      s     CHOICE  {
                                     ecdsa-nistp224-with-sha224-s  Integer28,
                                     ecdsa-nistp256-with-sha256-s  Integer32
                                    }
                    }

EccPublicKey
      ::= SEQUENCE  {
                      type        EccPublicKeyType,
                      x           CHOICE  {
                                            ecdsa-nistp224-with-sha224-X  Integer28,
                                            ecdsa-nistp256-with-sha256-X  Integer32
                                          },
                      y           CHOICE  {
```

```
                                              ecdsa-nistp224-with-sha224-Y  Integer28,
                                              ecdsa-nistp256-with-sha256-Y  Integer32
                                            } OPTIONAL
                    }

  EccPublicKeyType
        ::= ENUMERATED  {
                          xCoordinateOnly (0),
                          compressedLsbY0 (2),
                          compressedLsbY1 (3),
                          uncompressed    (4)
                        }
    XCoordinateOnly ::= EccPublicKeyType  (xCoordinateOnly)
    CompressedLsbY0 ::= EccPublicKeyType  (compressedLsbY0)
    CompressedLsbY1 ::= EccPublicKeyType  (compressedLsbY1)
    Uncompressed    ::= EccPublicKeyType  (uncompressed)

  SecDataExchCaCertSpecificData                     ::= SecDataExchCaScope
  SecDataExchAnonymousCertSpecificData              ::= AnonymousScope
  SecDataExchIdentifiedNotLocalizedCertSpecificData ::= IdentifiedNotLocalizedScope
  SecDataExchIdentifiedLocalizedCertSpecificData    ::= IdentifiedScope

  VersionAndType  ::= Integer8
      explicitCert  VersionAndType  ::= 2
      implicitCert  VersionAndType  ::= 3

    ExplicitCert  ::= VersionAndType  (explicitCert)
    ImplicitCert  ::= VersionAndType  (implicitCert)
    ImplicitOrExplicit  ::= VersionAndType  ( explicitCert  | implicitCert  )

  SubjectType ::= Integer8
      secDataExchAnonymousSubj              SubjectType ::= 0
      secDataExchIdentifiedNotLocalizedSubj SubjectType ::= 1
      secDataExchidentifiedLocalizedSubj    SubjectType ::= 2
      secDataExchCsrSubj                    SubjectType ::= 3
      wsaSubj                               SubjectType ::= 4
      wsaCsrSubj                            SubjectType ::= 5
      secDataExchCaSubj                     SubjectType ::= 6
      rootCaSubj                            SubjectType ::= 255

    SecDataExchCa       ::= SubjectType (secDataExchCaSubj)
    RootCa              ::= SubjectType (rootCaSubj)
    SecDataExchCsr      ::= SubjectType (secDataExchCsrSubj)
    SecDataExchAnon     ::= SubjectType (secDataExchAnonymousSubj)
    SecDataExchIdNonLoc ::= SubjectType (secDataExchIdentifiedNotLocalizedSubj)
    SecDataExchIdLoc    ::= SubjectType (secDataExchidentifiedLocalizedSubj)
    SecDataExchCaTypes  ::= SubjectType (secDataExchAnonymousSubj |
secDataExchidentifiedLocalizedSubj)
    IntermediateCert    ::= SubjectType (ALL EXCEPT (rootCaSubj))

  SubjectTypeFlags
        ::= BIT STRING  {
                          messageAnonymous              (0),
                          messageIdentifiedNotLocalized (1),
                          messageIdentifiedLocalized    (2),
                          messageCsr                    (3),
                          wsa                           (4),
                          wsaCsr                        (5),
                          messageCa                     (6),
                          wsaCa                         (7),
                          crlSigner                     (8)
                        }

  CertificateContentFlags
        ::= BIT STRING  {
                          useStartValidity   (0),
                          lifetimeIsDuration (1),
                          encryptionKey      (2)
                        }
    UseStartValidity            ::= CertificateContentFlags ({useStartValidity})
    LifetimeIsDuration          ::= CertificateContentFlags ({lifetimeIsDuration})
    UseStartVal-AndOr-Lifetime
        ::= CertificateContentFlags ({useStartValidity} | (ALL EXCEPT {encryptionKey}) )

  GeographicRegion
        ::= SEQUENCE  {
                        region-type           RegionType,
                        circlular-region      CircularRegion OPTIONAL,
```

```
                      rectangular-region    RectangularRegion OPTIONAL,
                      polygonial-region     PolygonialRegion OPTIONAL,
                      other-region          OCTET STRING OPTIONAL
                    }

RegionType  ::= Integer8
  from-issuer   RegionType  ::= 0
  circle        RegionType  ::= 1
  rectangle     RegionType  ::= 2
  polygon       RegionType  ::= 3
  none          RegionType  ::= 4

  CircularRegion
        ::= SEQUENCE  {
                        center          TwoDLocation,
                        radius          Integer16
                      }

  RectangularRegion
        ::= SEQUENCE  {
                        upper-left        TwoDLocation,
                        lower-right       TwoDLocation
                      }

  PolygonialRegion  ::= SEQUENCE OF TwoDLocation

  TwoDLocation
        ::= SEQUENCE  {
                        latitude        Sint32,
                        longitude       Sint32
                      }

Crl
      ::= CHOICE  {
                    validCrl  ValidCrl,
                    nullCrl   NullCrl
                  }

ValidCrl
      ::= SEQUENCE  {
                      version           Integer8,
                      signerCrl         SignerIdentifier,
                      unsignedCrl       ToBeSignedCrl,
                      signature         Signature
                    }

ToBeSignedCrl
      ::= CHOICE  {
                    idOnlyCrl       IdOnlyCrl,
                    idAndExpiryCrl  IdAndExpiryCrl
                  }
IdOnlyCrl
      ::= SEQUENCE  {
                      type            IdOnly,
                      crlSeries       CrlSeries,
                      caId            OCTET STRING (SIZE (8)),
                      crlSerial       Integer8,
                      startPeriod     ItsTime,
                      issueDate       ItsTime,
                      nextCrl         ItsTime,
                      entries         IdList
                    }
IdAndExpiryCrl
      ::= SEQUENCE  {
                      type            IdAndExpiry,
                      crlSeries       CrlSeries,
                      caId            OCTET STRING (SIZE (8)),
                      crlSerial       Integer8,
                      startPeriod     ItsTime,
                      issueDate       ItsTime,
                      nextCrl         ItsTime,
                      entries         IdAndExpiryList
                    }

CrlType
      ::= ENUMERATED  {
                        idOnly        (0),
                        idAndExpiry   (1)
```

```
                        }
    IdOnly      ::= CrlType (idOnly)
    IdAndExpiry ::= CrlType (idAndExpiry)

  CrlSeries ::= Integer32

  IdList  ::= SEQUENCE OF CrlEntryId

  IdAndExpiryList
        ::= SEQUENCE  {
                        crlId         CrlEntryId,
                        expiry        ItsTime
                      }

  CrlEntryId  ::= OCTET STRING (SIZE (10))

  NullCrl ::= NULL

  Integer3    ::= INTEGER (0..7)
  Integer8    ::= INTEGER (0..255)
  Integer16   ::= INTEGER (0..65535)
  Integer28   ::= INTEGER (0..268435456)
  Integer32   ::= INTEGER (0..4294967295)
  Sint32      ::= INTEGER (-65535..65535)

  ItsTime   ::= Integer32  --number of seconds since 00:00:00 UTC 1st January 2004

END
```

Annex B (informative): Secret-key use cases and application categories

Clause 4.1.1 in TS 102 940 [5] categorizes application communications patterns as:

- Broadcast;
- Groupcast;
- Unicast with local participants;
- Unicast with remote infrastructure entity;
- Unicast with remote infrastructure entity, communications session needed to persist across multiple contacts with infrastructure entity.

With the exception of broadcast applications, all other controlled multicast and unicast communication may use symmetric key systems to provide trust management and enrolment and authorisation services similar to those of clause 5.2.

In addition, detailed use case examples are presented annex C in TR 102 638 [i.2]. The symmetric key systems can be used in all the use cases in clause C.3 and electronic payment use cases such as Electronic toll collect (clause C.2.9).

Annex C (informative): Extensions to IEEE 1609.2 to support additional security functions

C.1 Rationale

In order to be able to offer ITS security standards which are truly global, the present document and its related specifications (TS 102 940 [5], TS 102 942 [4] and TS 102 943 [9]) have been developed as profiles of IEEE. 1609.2 [8]. However, there are some capabilities that are not included in IEEE. 1609.2 [8] but could usefully be included in a future edition of 1609.2.

C.2 Use of a cryptographic digest of the signer identifier

If the requester of an enrolment certificate is already known to the certificate authority then the authority will be able to correctly interpret a signer identifier with a digest type. Consequently, it would be beneficial for IEEE. 1609.2 [8] to allow the *signer* field to take the value *certificate_digest_with_ecdsap224* or *certificate_digest_with_ecdsap256* in a *ToBeEncrypted* message of type *certificate_request*.

C.3 Encryption of the signer identifier in an authorization certificate request

In order to support the presence of an encrypted signer identifier in an authorization certificate request, make the following changes:

Table C.1: Encryption of Signer Identifier in IEEE. 1609.2 [8]

SignerIdentifierType	Add new enumerated value, *encrypted*
ContentType	Add new enumerated value, *certificate_request_signer*

SignerIdentifier	Add a new *SignerIdentifierType* case, thus:
	`case encrypted`
	`ToBeEncrypted    encryptedSigner`

ToBeEncrypted
(see note)
	Add a new *ContentType* case, thus:
	`case certificate_request_signer`
	`SignerIdentifier    signer`

NOTE: The *SignerIdentifier* within the *ToBeEncrypted* data type should not be of the type *encrypted.*	

C.4 Request and transmission of multiple authorization certificates

In order to save processing and communications bandwidth, it would be useful to be able to request and receive multiple authorization certificates using a single request and a single response. This can be achieved with the following changes to the 1609.2 data types:

- Modify *ToBeSignedCertificate* such that more than one *PublicKey* can be included, as follows:

```
....
    PublicKey               verification_key<var>;
....
```

- Move all elements except the *Crl* element from *ToBeEncryptedCertificateResponse* to a new type, *CertificateResponse*
- Add a new element to *ToBeEncryptedCertificateReponse*, as follows:

```
ToBeEncryptedCertificateResponse
    {
        CertificateResponse         certificate_info<var>;
        Crl                             crl_path<var>;
    }   ToBeEncryptedCertificateResponse
```

Annex D (informative): Bibliography

ISO/IEC 15031-3: "Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 3: Diagnostic connector and related electrical circuits, specification and use".

History

Document history		
V1.1.1	June 2012	Publication