ETSI TS 102 941 V1.2.1 (2018-05)

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

Reference
RTS/ITS-00524

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

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables 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 (https://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.

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.

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.

1 Scope

The present document specifies the trust and privacy management for Intelligent Transport System (ITS) communications. Based upon the security services defined in ETSI TS 102 731 [1] and the security architecture defined in ETSI 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 ETSI 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

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 at https://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.

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 103 097: "Intelligent Transport Systems (ITS); Security; Security header and certificate formats".

[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 8824-1:2015: "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[7] Recommendation ITU-T X.696 (08/2014): "Information Technology-Specification of Octet Encoding Rules (OER)".

[8] Void.

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

[10] ETSI EN 302 637-2: "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service".

[11] ETSI EN 302 637-3: "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service".

[12] ETSI TS 103 301: "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Facilities layer protocols and communication requirements for infrastructure services".

[13] NIST FIPS PUB 198-1: "The Keyed-Hash Message Authentication Code (HMAC)".

[14] Void.

[15] IETF RFC 4862: "IPv6 Stateless Address Autoconfiguration".

[16] ETSI EN 302 636-6-1: "Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 6: Internet Integration; Sub-part 1: Transmission of IPv6 Packets over GeoNetworking Protocols".

[17] Void.

[18] ETSI EN 302 636-4-1: "Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality".

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

[20] IEEE 802.11™: "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications".

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 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".

[i.11] IETF RFC 4306: "Internet Key Exchange (IKEv2) Protocol", December 2005.

[i.12] IETF RFC 4877: "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture".

[i.13] ETSI TS 102 723-8: "Intelligent Transport Systems (ITS); OSI cross-layer topics; Part 8: Interface between security entity and network and transport layer".

[i.14] CVRIA: "Connected Vehicle Reference Implementation Architecture".

NOTE: Available at http://www.iteris.com/cvria/.

[i.15] ISO 21210-2010: "Intelligent Transport Systems (ITS) - Communications access for land mobiles (CALM) - Ipv6 networking".

3 Definitions, abbreviations and notation

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI TS 102 731 [1], ETSI TS 102 940 [5], ISO/IEC 15408-2 [i.1] and the following apply:

delta CTL: partial CTL that only contains CTL entries that have been updated since the issuance of the prior, base CTL

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI EN 302 636-4-1 [18] and the following apply:

AA	Authorization Authority
ASN	Abstract Syntax Notation
AT	Authorization Ticket
CA	Certification Authority
CCH	Control CHannel
CCMS	Cooperative-ITS Certificate Management System
COER	Canonical Octet Encoding Rule
CPOC	C-ITS Point Of Contact
CRL	Certificate Revocation List
CTL	Certificate Trust List
CVRIA	Connected Vehicle Reference Implementation Architecture
DC	Distribution Centre
DENM	Decentralized Environmental Notification Message
EA	Enrolment Authority
EC	Enrolment Credential
ECC	Elliptic Curve Cryptography
ECTL	European Certificate Trust List
EV	Electric Vehicle
GET	command HTTP GET
GN/BTP	GeoNetworking/Basic Transport Protocol
GN6	GeoNetworking-IPv6
HMAC	keyed-Hash Message Authentication Code
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
ITS-AID	ITS Application ID
LTE	Long Term Evolution (4G)
MSEC	Multicast SECurity
OBD	On-Board Diagnosis
PA	Policy Authority
PDU	Protocol Data Unit
PII	Personally Identifiable Information
POP	Proof Of Possession
RCA	Root Certification Authority
SCH	Service CHannel
SLAAC	StateLess Address Auto Configuration
SM	Security Management
SSP	Service Specific Permissions
TCP	Transmission Control Protocol

TCP/IP	Transmission Control Protocol / Internet Protocol
TLM	Trust List Manager
TLS	Transport Layer Security
UPER	Unaligned Packed Encoding Rules
V2I	Vehicle-to-Infrastructure
WLAN	Wireless Local Area Network

3.3 Notation

The requirements identified in the present document include:

a) mandatory requirements strictly to be followed in order to conform to the present document. Such requirements are indicated by clauses without any additional marking;

b) requirements strictly to be followed if applicable to the type of ITS Station concerned.

Such requirements are indicated by clauses marked by "[CONDITIONAL]"; and where relevant is marked by an identifier of the type of ITS-S for which the clauses are applicable as follows:

- [Itss_WithPrivacy] is used to denote requirements applicable to ITS-S for which pseudonymity has to be assured and re-identification by the AA is not allowed. This includes for instance personal user vehicle ITS-S or personal ITS-S Portable.

- [Itss_NoPrivacy] is used to denote requirements applicable to ITS-S for which pseudonymity does not have to be assured and are allowed to be re-identified by the AA. This may be for instance fixed or mobile RSUs or special vehicles.

4 ITS authority hierarchy

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 enrolment credentials that provide 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-S and authorities.

ETSI TS 102 731 [1] specifies requirements on security management services and security management roles such as EAs and AAs. The ITS security architecture is defined in ETSI TS 102 940 [5] and covers both the secured Communication Architecture, the architecture of the ITS-S Communication security system and the Security Management System architecture.

The present document assumes the definition of the security management entities specified in ETSI TS 102 940 [5] and the top-level entities for the management of multiple Root CAs collaborating within a single Trust Model. For ease of reading and for further specification of trust and privacy management the relevant tables from ETSI TS 102 940 [5] are copied here.

Table 1: Functional element roles of the PKI

Functional element	Description
Root Certification Authority	The Root CA is the highest level CA in the certification hierarchy. It provides EA and AA with proof that it may issue enrolment credentials, respectively authorization tickets
Enrolment Authority	Security management entity responsible for the life cycle management of enrolment credentials. Authenticates an ITS-S and grants it access to ITS communications
Authorization Authority	Security management entity responsible for issuing, monitoring the use of authorization tickets. Provides an ITS-S with authoritative proof that it may use specific ITS services
Distribution Centre (optional)	Provides to ITS-S the updated trust information necessary for performing the validation process to control that received information is coming from a legitimate and authorized ITS-S or a PKI certification authority by publishing the CTL and CRL
Sending ITS-S	Acquires rights to access ITS communications from Enrolment Authority Negotiates rights to invoke ITS services from Authorization Authority Sends single-hop and relayed broadcast messages
Relaying ITS-S	Receives broadcast message from the sending ITS-S and forwards them to the receiving ITS-S if required
Receiving ITS-S	Receives broadcast messages from the sending or relaying ITS-S
Manufacturer	Installs necessary information for security management in ITS-S at production
Operator	Installs and updates necessary information for security management in ITS-S during operation

Table 2: Functional element roles of the top-level trust management

Functional element	Description
Policy Authority	Policy authority is a role composed by the representatives of public and private stakeholders (e.g. Authorities, Road Operators, Vehicle Manufacturers, etc.) participating to the C-ITS trust model. It designates and authorizes the TLM and the CPOC to operate in the C-ITS Trust system. It decides if root CAs are trustable and approves/removes the Root CAs operation in C-ITS trust domain by notifying the TLM about approved/revoked Root CAs certificates
Central Point of Contact (optional)	The CPOC is a unique entity appointed by the Policy Authority. It has responsibility to establish and contribute to ensure communication exchange between the Root CAs, to collect the Root CA certificates and provide them to the Trust List Manager (TLM). The CPOC is also responsible for distributing the ECTL to any interested entities in the trust model
Trust List Manager	Trust List Manager is responsible for creating the list of root CA certificates and TLM certificates and signing it. The signed list issued by the TLM is called the ECTL

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 authorization tickets provisioning:

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

ii) privacy of communications between ITS-Ss.

Separation of duties for enrolment (identification and authentication) and for authorization has been shown in ETSI 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.

When the same operational authority manages both the EA and AA, it shall guarantee privacy of requesting ITS-S i.e. providing all the technical and organizational measures to ensure that ITS identity information held by the EA is kept separately to avoid re-identification of pseudonym certificates (ATs).

When dedicated authorities are used only for certificates provisioning to ITS-S which do not have privacy requirements such as Road-Side Units, the EA and AA may not provide technical and operational separation.

6 Trust and privacy management

6.1 ITS-S Security Lifecycle

6.1.1 ITS-S Life-cycle management

The ITS-S Security Lifecycle includes the following stages (see Figure 1):

- initial ITS-S configuration during manufacture;
- enrolment;
- authorization;
- operation and maintenance;
- end of life.

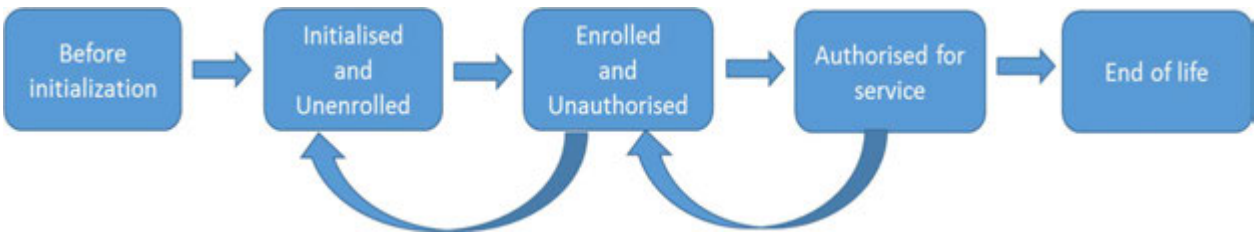


Figure 1: ITS Station Security Life Cycle

6.1.2 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 AA certificates which the ITS-S may use to trust communications from other ITS-S;
  - a public/private key pair for cryptographic purposes (canonical key pair); and
  - the trust anchor (Root CA) public key certificate and the DC network address;
  - in case of a multiple root CAs architecture as specified in [5], the TLM public key certificate and the CPOC network address.

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 three items of information shall be established, all associated with each other (see note 2):
  - the permanent canonical identifier of the ITS-S;
  - the profile information for the ITS-S that may contain an initial list of maximum appPermissions (ITS-AIDs with SSPs), region restrictions and assurance level which may be modified over time;
  - the public key from the key pair belonging to the ITS-S (canonical public key).

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

6.1.3 Enrolment

The ITS-S requests its enrolment certificate from the EA (see clause 6.2.3.2).

The state transitions for enrolment are shown in Figure 2.

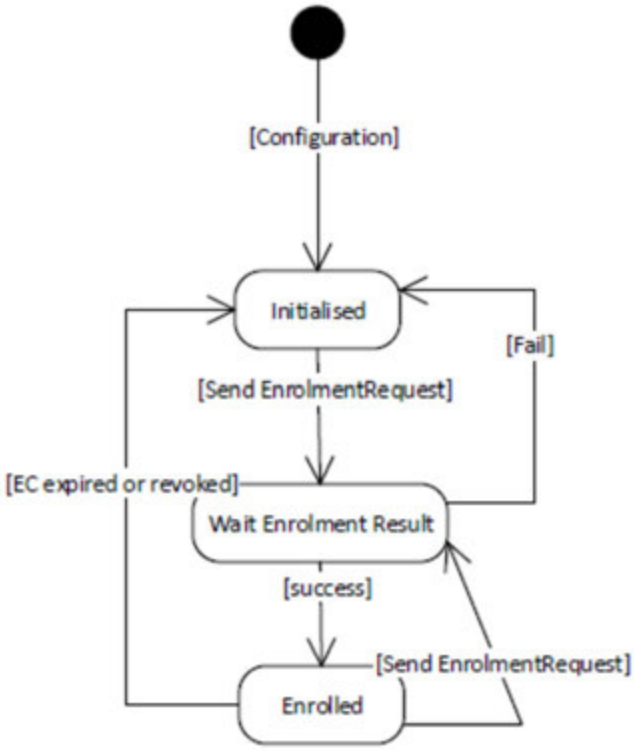


Figure 2: Simplified state machine for the enrolment process

After a successful enrolment process, the ITS-S shall possess an enrolment credential that shall be used in subsequent authorization requests.

For renewing the Enrolment Certificate at the EA, the ITS-S shall send an EnrolmentRequest signed by the previous valid enrolment credential issued by this EA.

6.1.4 Authorization

Having received the enrolment credentials (i.e. in state "Enrolled" as shown in Figure 1), the ITS-S is able to request its authorization ticket(s) from the AA (see clause 6.2.3.3).

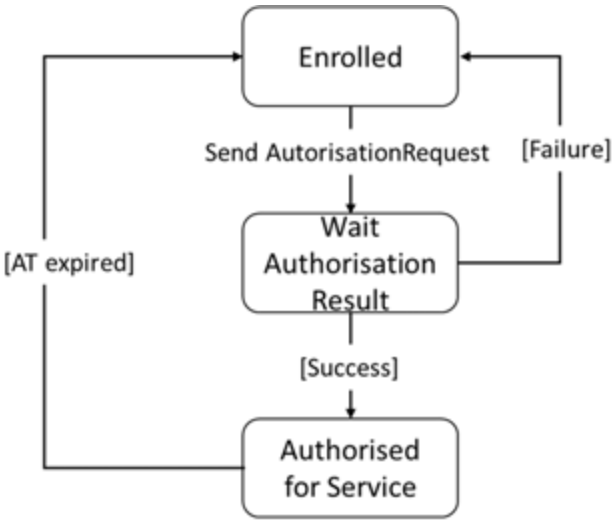


Figure 3: Simplified state machine for the authorization process

NOTE 1: This state machine only considers one instance of Authorization Request for one service (or class of services). Eventually, an ITS-S may perform several Authorization Requests to fill a pool of ATs stored in ITS-S memory (depending of the policy). The management of the pool of ATs within an ITS-S is out of the scope of the present document.

When in the state "Authorized for service" the ITS-S has a set of authorization tickets to allow signed transmission of messages to any other ITS-S that do not reveal the canonical identity nor the enrolment credential of the transmitting ITS-S.

NOTE 2: It is assumed that the ITS service requesting the use of a secure message transfer service only reveals PII in the payload if the release of that PII has been consented by the sending party.

NOTE 3: The ITS-S in state Authorized for Service may still have the possibility to obtain or update its pool of Authorization Tickets by sending Authorization requests (not representing on the Figure 3).

When the complete set of authorization tickets is exhausted, the ITS-S cannot sign transmission of messages to others ITS-S and is back to the state Enrolled.

NOTE 4: In this state diagram, it is assumed that revocation of authorization tickets is not possible as passive revocation is preferred (the ITS-S receives ATs of limited validity period and of limited allowed preloading period, renewing them frequently so that compromised ITS-S can be evicted from the ITS network by not reissuing them new ATs).

6.1.5 Maintenance

If an EA or AA is added to or removed from the system, the Root CA shall inform enrolled ITS-Ss of this change.

When multiple Root CAs are used in the same Trust Domain (as specified in ETSI TS 102 940 [5]), the trust relationship between the different PKIs may be managed by a Trusted Third Party (Trust List Manager). If a Root CA is added or removed from the system, the TLM should inform enrolled ITS-Ss of this change.

The process for updating the trust information lists such as the CTL and the CRL and for publishing these lists by the associated trust authority is specified in clause 6.3 and include different possible methods for updating the enrolled ITS-Ss:

- requesting CRL and CTL from the distribution centre associated to the root CA;
- sending a trust information list (CRL or CTL) across a wireless interface e.g. using a RSU able to transmit the CRL/CTL on ITS G5; or
- providing information to a trusted maintenance entity to enable it to update an individual ITS-S in a controlled environment.

6.1.6 End of life

In case of the device's end of life or following a compromise (revocation decided by its issuing EA), the ITS Station should be revoked and removed from operation in the ITS G5 communication network by means of a passive revocation mechanism, i.e. the EA shall reject the authorization of all the next Authorization Ticket requests for this specific ITS-S.

6.2 Public Key Infrastructure

6.2.0 General

6.2.0.1 Messages format

All the security management messages (SM_PDU) specified in the present document shall follow the message format shown in Figure 5.

Each SM-PDU message shall be encoded into an `EtsiTs103097Data-Encrypted`, `EtsiTs103097Data-Signed` or an `EtsiTs103097Data-SignedAndEncrypted` structure, depending on the specific message. This outermost structure shall contain an `EtsiTS102941Data` which in turn contains the content of the specific message.

Figure 4 shows an example of the use of above structures to provide enrolment/authorization requests and responses between an ITS-S and a Certification Authority. The same diagram applies for authorization validation requests and responses exchanged between Certification Authorities (EA and AA) as specified in clause 6.2.3.4.

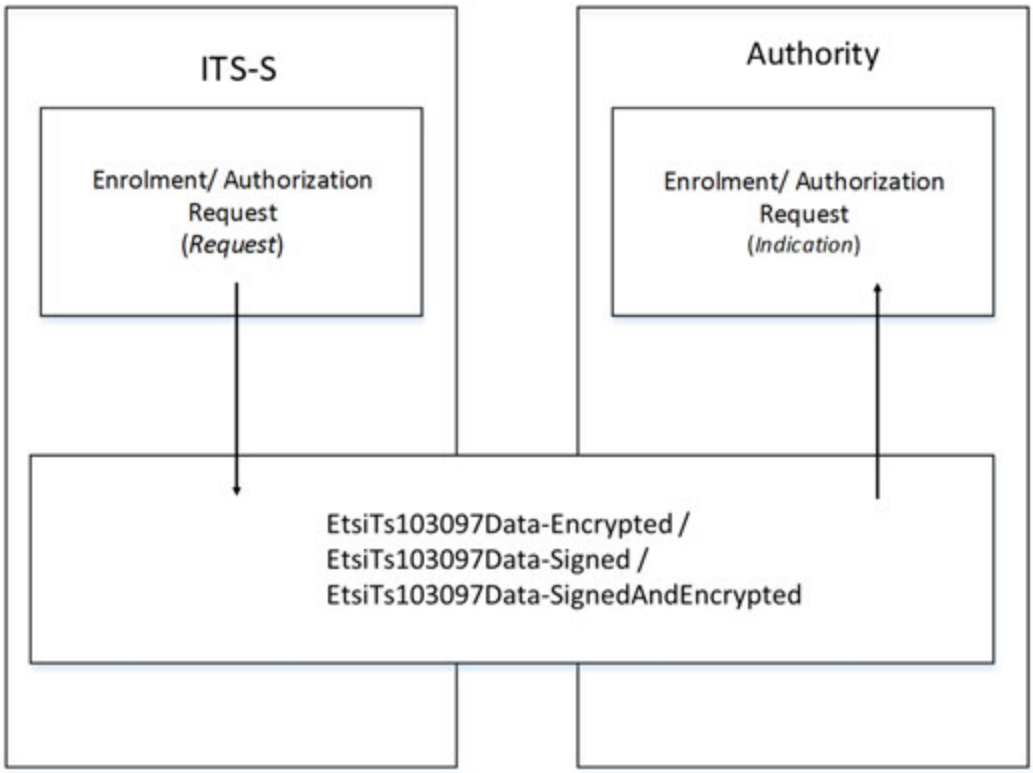


Figure 4: Illustration of using Security Management PDU structure

This outermost data structure `EtsiTs103097Data-Encrypted`, `EtsiTs103097Data-Signed` or `EtsiTs103097Data-SignedAndEncrypted` structure shall encapsulate an `EtsiTS102941Data` as shown in Figure 5. Specification of all containers and enclosed data structures are given in annex A.

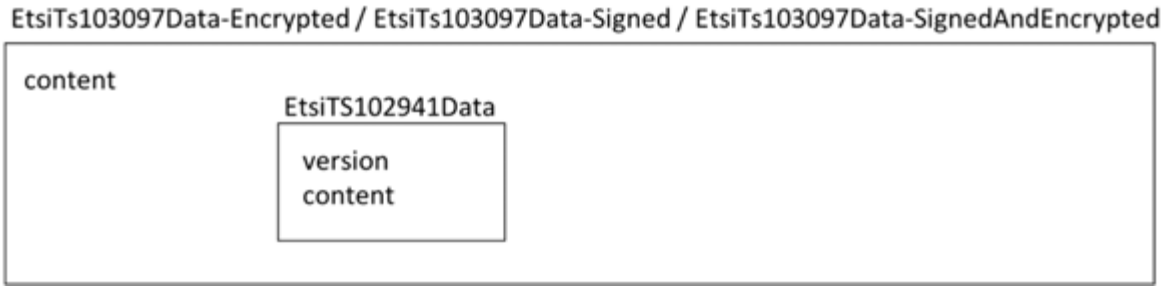


Figure 5: Generic Security Management PDU structure

Each of the SM_PDU is containing an EtsiTS102941Data as specified in clause A.2.1:

```
EtsiTS102941Data::= SEQUENCE {
  version Version (v1),
  content EtsiTS102941DataContent
  }
EtsiTS102941DataContent ::= CHOICE {
…}
```

In the present document, the version of EtsiTS102941Data structure shall be set to version v1 (integer value 1).

Data structures in the present document are defined using Abstract Syntax Notation 1 (ASN.1) and shall be encoded using the Canonical Octet Encoding Rules (COER) as defined in Recommendation ITU-T X.696 [7].

The ASN.1 representation of Security Management PDUs shall be as specified in the annex A of the present document.

6.2.0.2 Signed and encrypted data structures

The present document assumes the definition of the data structures specified in ETSI TS 103 097 [3].

Especially the following TS103097 data structures shall be used for encrypted/signed messages specified in the present document:

- EtsiTs103097Data-Signed,
- EtsiTs103097Data-Encrypted,
- EtsiTs103097Data-SignedAndEncrypted,
- EtsiTs103097Data-SignedExternalPayload.

All signed data structures contain a headerinfo which itself includes a PSID/ ITS-AID. This PSID/ITS-AID is mandatory and shall be used for certificate management with value as assigned in ETSI TS 102 965 [19].

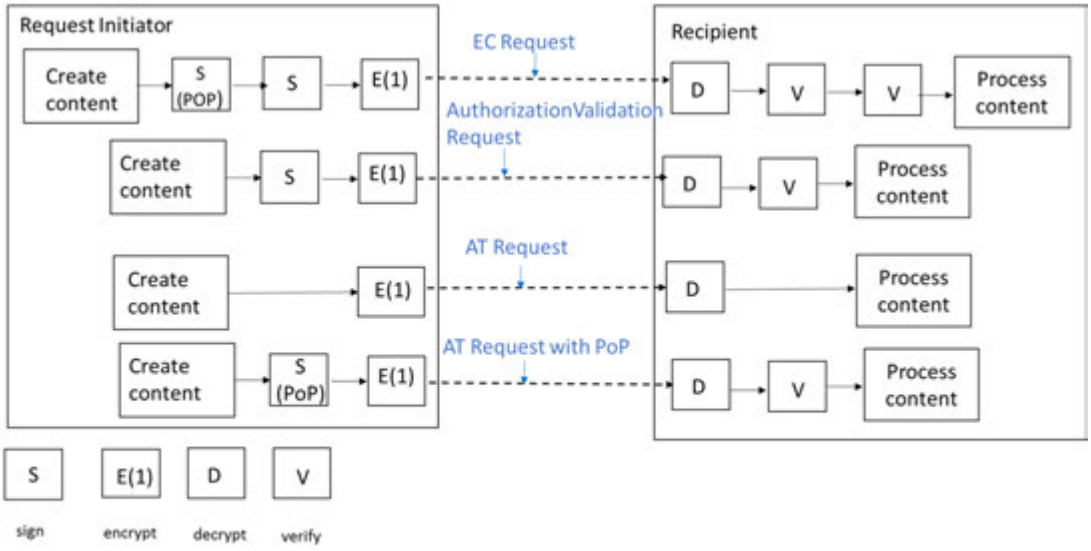
For all messages that are using the EtsiTs103097Data-SignedAndEncrypted structure as outermost message format, the EtsiTs1030971Data-SignedAndEncrypted structure shall be built using the security profile for signed and encrypted messages defined in [3] and shall be composed of an EtsiTs103097Data-Encrypted structure containing an encrypted EtsiTs103097Data-Signed structure, containing a EtsiTs102941 Data structure, containing a version and an EtsiTs102941DataContent structure corresponding to the specific message (see clauses 6.2.3.2 to 6.2.3.4).

For all request and responses messages used for the provisioning of EC or ATs to an ITS-S, the following generic request and responses messages are used as depicted in Figure 6 and Figure 7.

For each generation of a REQUEST message, the request initiator shall generate a new AES symmetric key k which is used to encrypt the inner message as shown in Figure 6.

The EtsiTs103097Data-Encrypted structure shall be used to encapsulate all this message data using encryption option 1: encryption is done using the recipient public encryption key corresponding to the recipient's certificate. The field recipients shall contain **one single** RecipientInfo of type certRecipInfo.

The request initiator shall store the encryption symmetric key (k) in memory until it receives and processes the corresponding RESPONSE message as this key is to be used to decrypt the response (session encryption key). See clause 7.3.

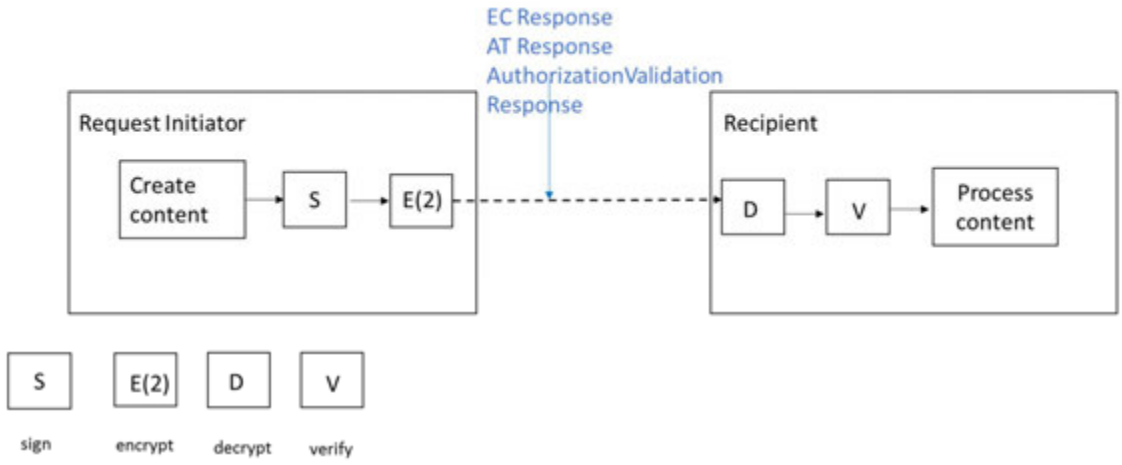


E(1): use EtsiTs103097Data-Encrypted with option 1 (using the recipient public encryption key corresponding to the recipient's certificate): the field recipients shall contain one single RecipientInfo of type certRecipInfo

Figure 6: Generic REQUEST message

NOTE: The AT Request depicted in Figure 6 uses different authentication mechanisms. See clause 6.2.3.3.1 for details.

For generating the RESPONSE message, the Recipient of the request shall use an outermost EtsiTs103097Data-Encrypted structure that encapsulates the response message as shown in Figure 7 using encryption option 2: the AES symmetric key used to encrypt the corresponding request is reused to encrypt the response, with the pskRecipInfo option in the RecipientInfo that indicates the use of a pre-shared symmetric key.



E(2): use EtsiTs103097Data-Encrypted with option 2. The key used to encrypt the corresponding request is reused to encrypt the response using one single RecipientInfo of type pskRecipInfo.

Figure 7: Generic RESPONSE message

An algorithm for encryption of a message from a sender to a receiver (Recipient) is given in annex E.

6.2.1 CA certificate request

The CACertificateRequest message defines a standalone certificate request, which shall be used to transport SubCA (i.e. EA or AA) certificate request to the RCA (across the interface at reference point S_9 or S_{10} as specified in ETSI TS 102 940 [5]).

For the initial application of a SubCA to the RCA, the SubCA needs to submit an application form containing organization identity and other registration information specified in the respective policy. The details of this application form and process are out of the scope of the present document.

The SubCA shall transfer the signed certificate request (`CACertificateRequest`) by an off-band mechanism to the RCA entity: for initial application of the subCA, the subCA shall use a separate communication channel with the RCA (electronic communication means) other than the channel used for transmitting the application form and documents to the RCA. The application form should include the digital fingerprint of the `CACertificateRequest` in printable format. For rekeying, an application form is not needed because the RCA has a trust relation after the initial request.

NOTE 1: The digital fingerprint of the CACertificateRequest is computed using an ETSI TS 103 097 [3] approved hash algorithm.

For the initial application of a Sub CA to the RCA for issuance of its certificate, the SubCA shall generate its key pairs and the `CACertificateRequest` shall be signed with the private key corresponding to the verification public key provided by the Sub-CA for certification, to prove possession of the private key.

After approval of the application form and associated documents supplied by the SubCA and validation of the integrity and authenticity of the certificate request transmitted by the SubCA, the RCA shall generate the certificate of the SubCA, which is provided as a response to the `CACertificateRequest`: the certificate issued by the RCA to the SubCA shall follow the requirements of CA certificate as defined in ETSI TS 103 097 [3].

For the subsequent applications of the SubCA for requesting new certificate to the RCA, the SubCA shall generate its new key pairs. Then the CA certificate request shall be signed with the new private key corresponding to the verification public key (inner signature) and the whole request shall be signed with the current valid private key (outer signature) to ensure the integrity and authenticity of the request. If the current private key has reached its end of validity period or is revoked, the SubCA shall restart the initial certificate application process.

The content of the `CACertificateRequest` message shall be as described in Figure 8.

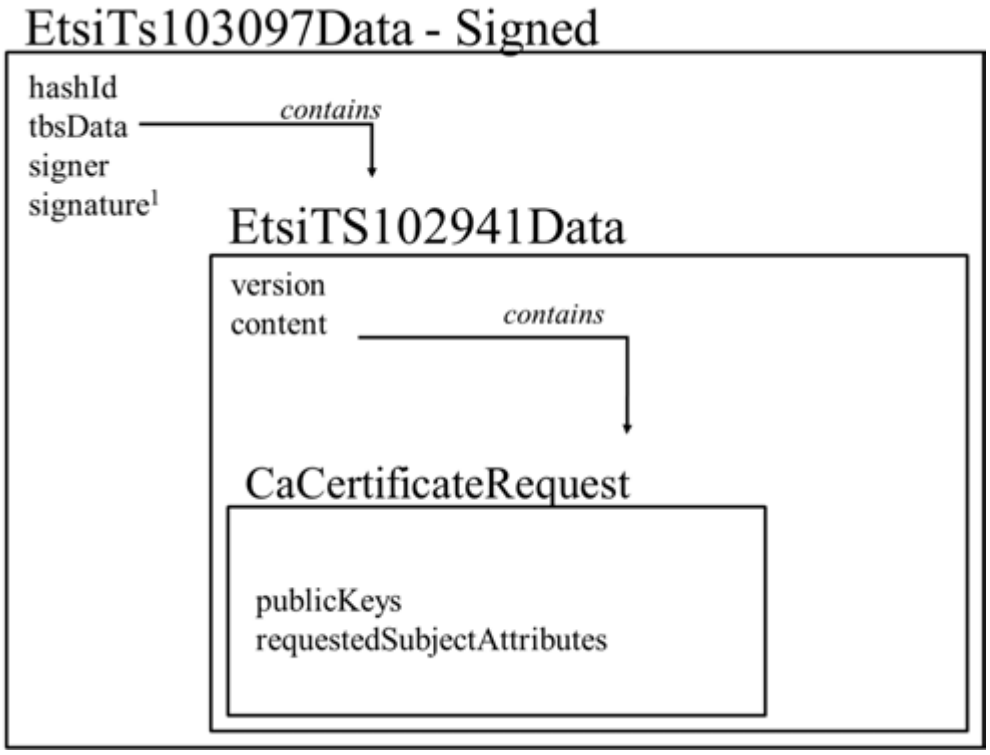
The specification of the `CACertificateRequest` message using ASN.1 [6], [7] shall be as specified in clause A.2.6.

For the initial application to the RCA, an EA or AA shall follow this process to create a CaCertificateRequestMessage:

- An ECC private key is randomly generated, the corresponding public key (`verificationKey`) is provided to be included in the CaCertificateRequest.
- An ECC encryption private key is randomly generated, the corresponding public key (`encryptionKey`) is provided to be included in the CACertificateRequest.
- An `EtsiTs102941Data` structure is built, containing:
  - `version` is set to v1 (integer value set to 1);
  - a `CaCertificateRequest` is built with:
    - publicKeys shall contain verification_key and encryption_key;
    - requestedSubjectAttributes shall contain the requested certificates attributes as specified in ETSI TS 103 097 [3], clause 7.2.4.
- An EtsiTs103097Data-Signed structure is built, containing: hashId, tbsData, signer and signature:
  - the `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3].
  - in `tbsData`:
    - the `payload` shall contain the previous `EtsiTs102941Data` structure;
    - in the `headerInfo`:
      - the `psid` shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];

- the `generationTime` shall be present;
- all other components of the component `tbsdata.headerInfo` not used and absent;

- the `signer` is set to 'self'.
- the `signature` over the `tbsData` computed using the private key corresponding to the new `verificationKey` to be certified (i.e. the request is self-signed).

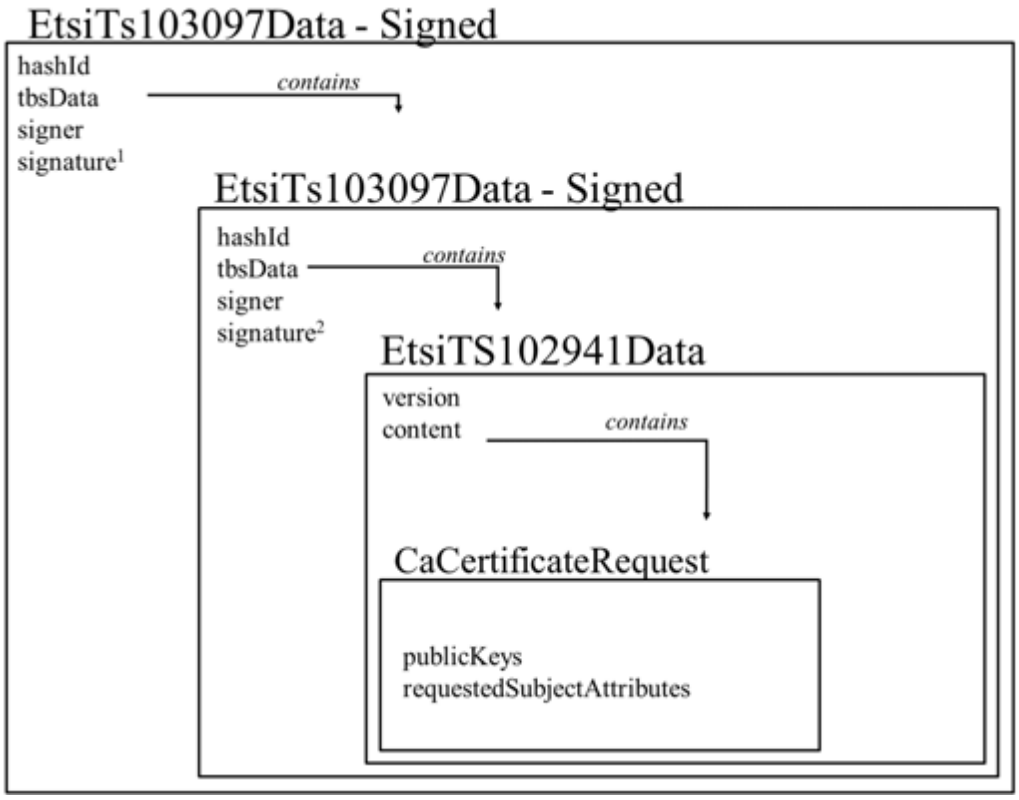


NOTE: [1] Signature computed using the private key corresponding to the verification public key to be certified.

Figure 8: CA Certificate Request message for initial creation

For the re-keying application to the RCA, an EA or AA shall follow this process to create a `CaCertificateRekeyingMessage`:

- An `EtsiTs103097Data-Signed` structure is built, containing: `hashId`, `tbsData`, `signer` and `signature`:
  - the `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3].
  - in the `tbsData`:
    - the `payload` shall contain the previous `CaCertificateRequestMessage` structure;
    - in the `headerInfo`:
      - the `psid` shall be set "secured certificate request" as assigned in ETSI TS 102 965 [19];
      - the `generationTime` shall be present;
      - all other components of the component `tbsdata.headerInfo` not used and absent;
  - the `signer` declared as a `digest`, containing the hashedId8 of the EA or AA certificate;
  - the `signature` over `tbsData` computed using the currently valid private key corresponding to the AA or EA certificate (outer signature).



NOTE: [1] Signature computed using the currently valid private key corresponding to the AA or EA certificate.
[2] Signature computed using the private key corresponding to the verification public key to be certified.

Figure 9: CA certificate Request message for certificate re-keying

NOTE 2: If the SubCA certificate contains an encryption public key, a proof of possession for this private key may be provided by the requesting SubCA. The method for establishing this proof of possession between the SubCA and the RCA is beyond the scope of the present document.

6.2.2 Enrolment/Authorization assumption and requirements

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

For obtaining security credentials from the PKI, the ITS station has to set up a communication with the Certificate Authority (EA or AA). The ITS station may use different communication types for the provisioning of certificates, such as for instance:

- An ITS G5 communication via a roadside station.
- A WLAN consumer network using IEEE 802.11 [20] protocol (via a public or private hotspot or a home network).
- A Cellular network link (3G, 4G or LTE).
- A wired or wireless connection at EV Charging station.
- Using the Vehicle On-Board Diagnostic (OBD) port and a diagnostic system at the Service Garage or inspection workshop.

Figure 10 and Figure 11 present two possible options for ITS-S communication with the PKI, in order to support the security management services as specified in ETSI TS 102 940 [5] and shown in clause 4 above.

This functional model extends the ITS Communication security model specified in ETSI TS 102 940 [5] with the different communication paths. It introduces functional entities as defined in Table 3 and specifies the communication paths needed to support the ITS-S communication via a cellular network (Figure 10) or via an ITS G5 access network to the ITS infrastructure (Figure 11).

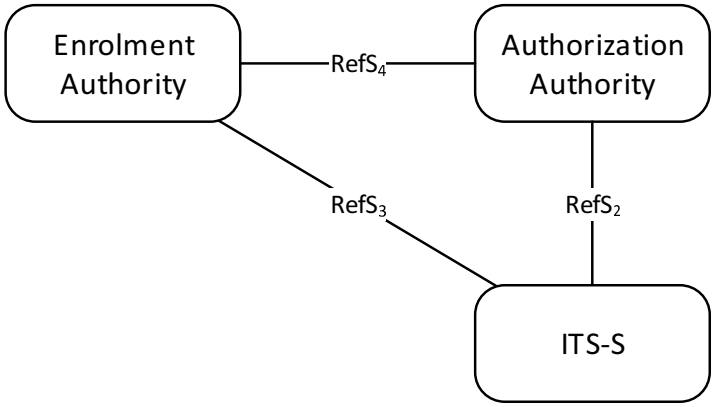


Figure 10: Communication with PKI using a cellular network connection

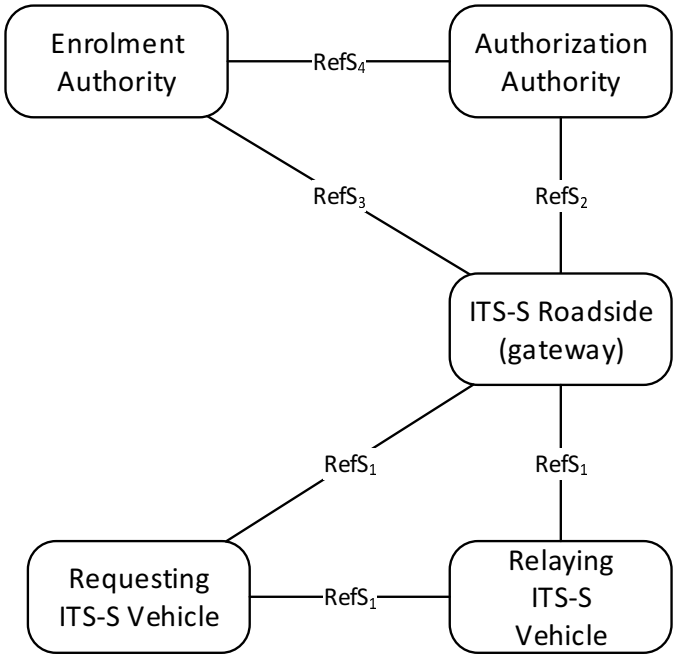


Figure 11: Communication with PKI using a V2I connection

In Figure 11, the model specified in ETSI TS 102 940 [5] is refined by considering one ITS-S sending the certificate request message, one roadside ITS-S for relaying the message via the infrastructure network to the recipient Certificate Authority and the receiving Certificate Authority (EA or AA). The certificate request message shall be sent in one-hop or multi-hops using the GeoNetworking protocol specified in ETSI EN 302 636-4-1 [18].

Table 3: Functional element roles

Functional element	Role
Enrolment Authority	See definition of functional elements, as specified in ETSI TS 102 940 [5]
Authorization Authority	
Requesting ITS-S	
Relaying ITS-S	
ITS-S Roadside Gateway	Receives broadcast messages from mobile ITS-S and relays certificate requests and responses to/from the Receiving Certificate Authority within the PKI

The model depicted in Figure 11 assumes that a mobile ITS-S has the capabilities to transmit security management messages such as defined in clauses 6.2.3.2, 6.2.3.3 and 6.2.3.4 through the ITS G5 communication interface (Reference Point S_1). For that scenario, a communication bandwidth should be made available to the requesting ITS-S on an allocated G5 channel for security management purpose (e.g. CCH, SCH1 or SCH2).

The present document specifies messages format for certificate provisioning that is agnostic to the underlying communication path and communication media.

The term message refers to the Security Management PDUs which are processed by the Security Management Functional entity specified in ETSI TS 102 940 [5], clause 5. There is no application payload in these messages provided by the ITS Applications layer. The generation and processing of Security Management Messages in the ITS-S should be executed in a trusted software platform environment.

Several communication profiles are possible as described in annex C.

For all the communication profiles which are using IP protocol, HTTP/1.1 over TCP/IP shall be used in end-to-end communication paths, over the reference points S_2 and S_3. (directly or after forwarding via reference points S_1). No supplementary cryptographic layer such as TLS is required.

The term message refers to the Security Management PDUs which are processed by the Security Management Functional entity specified in ETSI TS 102 940 [5] clause 5. There is no application payload in these messages provided by the ITS Applications layer. The generation and processing of Security Management Messages in the ITS-S shall be executed in a trusted software platform environment.

NOTE 1: The present document does not specify the interface between Security Entity and the ITS-S communication layers and does not specify the interface between the Security Entity and the Facilities layer, i.e. mechanism for transferring Security Management PDUs via the SF-SAP. It does not specify the way the communication profile parameters are provided to the ITS-S communication stacks.

If an ITS station needs to provide pseudonym change management for privacy protection (e.g. vehicle ITS-S or personal ITS-S Portable), the following requirements shall be satisfied:

- [Itss_WithPrivacy] if the requesting ITS-S uses an Ipv6 communication link to request a CCMS service, the ITS-S shall support temporary pseudonyms instead of permanent identifiers such as defined in ETSI TS 102 731 [1] and further specified in [5] and shall use automatically configured Ipv6 address using Stateless Address Autoconfiguration (SLAAC) as specified in IETF RFC 4862 [15].

- [Itss_WithPrivacy] Upon notification of an ID Change, the ITS-S Ipv6 layer shall modify the MAC address of each virtual interface which implies a change in the Ipv6 addresses associated to the virtual interfaces.

- [Itss_WithPrivacy] In case the ITS-S communication profile used is based on Ipv6 over GeoNetworking (GN6), the ITS-S shall provide support for pseudonym change as specified in ETSI EN 302 636-6-1 [16], clause 11. This function enables change of the GN_Addr and the change of MAC address implies creation of a new temporary Ipv6 address for the ITS-S originating the connection with the CCMS service. For security protection of Ipv6 over GeoNetworking traffic, the ITS-S should protect integrity & authenticity of packets by signature as presented in ETSI EN 302 636-6-1 [16], clause G.1.

- [Itss_WithPrivacy] the ITS-S security entity shall perform an ID change (as specified in ETSI TS 102 940 [5]) each time before sending an AT request to the Authorization Authority (i.e. change the used Ipv6 address, MAC address…) if no previous Authorization request is waiting for the response. Otherwise the ITS-S shall block ID Change until it receives the corresponding response from the AA or after a maximum time limit.

NOTE 2: The security entity uses the ID Change service in order to force ITS-S' IPv6 address change. To this end, it is not mandatory to change the AT prior to use the ID Change service. When no AT change occurs, the HashedId8 passed with the ID_Change should contain a random value (SN-IDCHANGE-EVENT as specified in ETSI TS 102 723-8 [i.13]).

6.2.3 Message Sequences

6.2.3.1 Introduction

The overall ITS-S initialization sequence to achieve ITS secured message transfer on ITS G5 is given in Figure 12. The messages are specified in details in clauses 6.2.3.2, 6.2.3.3 and 6.2.3.4.

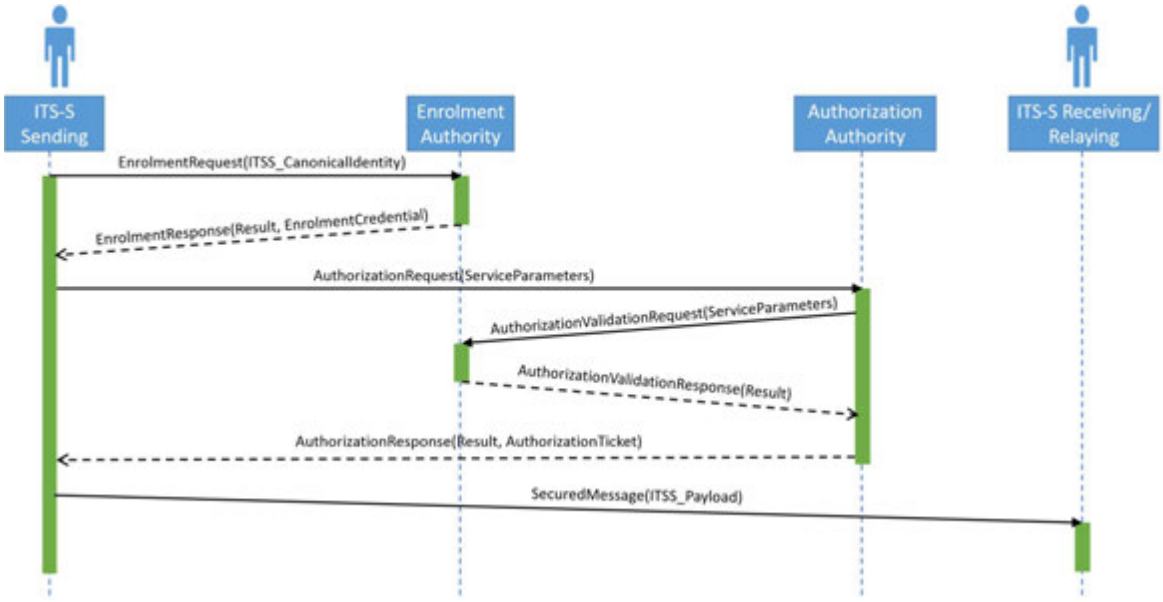


Figure 12: Sequence to achieve signed message transfer between ITS-Ss

6.2.3.2 Enrolment Management

6.2.3.2.0 Overview

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 8 in ETSI TS 102 940 [5]) to request an enrolment certificate to be used in a subsequent authorization request. Figure 13 shows an example of a message sequence for a successful or unsuccessful enrolment request.

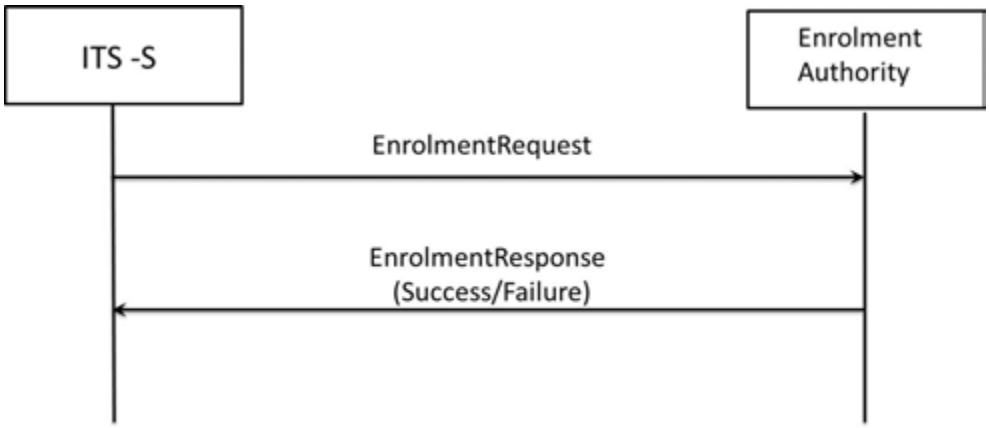


Figure 13: Message sequence for enrolment request and response

6.2.3.2.1 Enrolment request

The following functional requirements are defined on the state machine of Figure 2 (sender ITS-S for the enrolment process):

- The EnrolmentRequest message shall be encrypted using a ETSI TS 103 097 [3] approved encryption algorithm and the public key provided by the enrolment authority.
- For each enrolment request, the ITS-S shall generate a new verification key pair corresponding to an approved signature algorithm as specified in ETSI TS 103 097 [3].
- The contents of the EnrolmentRequest message shall be as described in Figure 14. The specification of the ITS-S EnrolmentRequest message using ASN.1 [6], [7] shall be as specified in clause A.2.

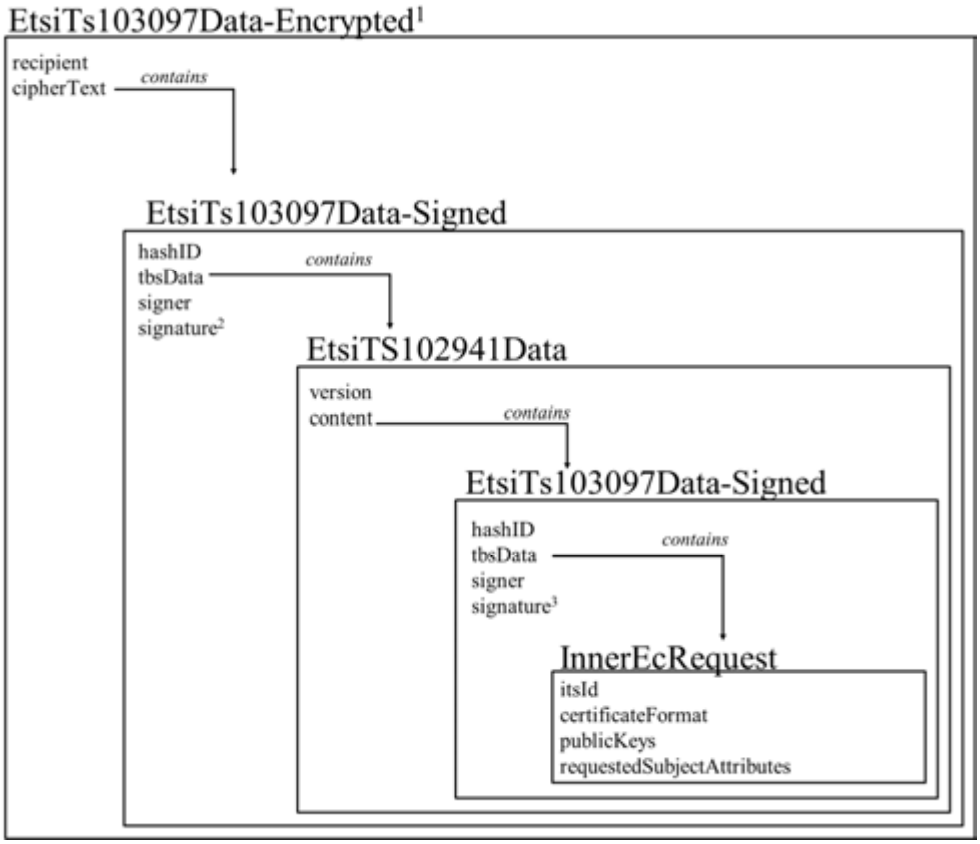
To create an enrolment request, an ITS-S shall follow this process:

- An ECC private key is randomly generated, the corresponding public key (verificationKey) is provided to be included in the EC.
- An `InnerECRequest` structure is built, containing:
  - the identifier of the requesting ITS-S (`itsId`): this identifier shall be set to the canonical identifier of the ITS-S for the initial enrolment with the recipient EA. For a re-enrolment of the ITS-S, the requesting ITS-S shall use the identifier of its current valid Enrolment Credential (EC identifier) which is computed as the HashedID8 of the Enrolment Credential (as specified in ETSI TS 103 097 [3]);
  - the `certificateFormat` which specifies the version used for the certificate format specification. In the present document, the certificate format shall be set to ts103097v131 (integer value 1);
  - the `verificationKey` for the EC;
  - the desired attributes (`requestedSubjectAttributes`);
  - the `requestedSubjectAttributes` shall not contain a `certIssuePermissions` field and the fields `validityPeriod` and `region` are optional because the EA already knows the ITS-S and can set duration and region restrictions on its own.
- For the proof of possession of the verification key pair, an `EtsiTs103097Data-Signed` structure (InnerECRequestSignedForPOP) is built containing: `hashId`, `tbsData`, `signer` and `signature`:
  - the `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3];
  - in the `tbsData`:
    - the `payload` shall contain the previous `InnerEcRequest` structure;
    - in the `headerInfo`:
      - the `psid` shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];
      - the `generationTime` shall be present;
      - all other component of the component `tbsdata.headerInfo` not used and absent;
  - the `signer` is set to 'self';
  - the `signature` over the `tbsData` computed using the private key corresponding to the new `verificationKey` to be certified (i.e. the request is self-signed) to prove possession of the generated verification key pair.
- An `EtsiTs102941Data` structure is built, with:
  - `version` is set to v1 (integer value set to 1);
  - the `content` is set to the previous signed data structure (InnerECRequestSignedForPOP).
- An `EtsiTs1030971Data-Signed` structure is built containing: `hashId`, `tbsData`, `signer` and `signature`:
  - the `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3];
  - in the `tbsData`:
    - the `payload` field shall contain the previous `EtsiTs102941Data` structure;
    - in the `headerInfo`:
      - the `psid` shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];

- the `generationTime` shall be present;
- all other components of the component `tbsData.headerInfo` not used and absent;

- the `signer` declared as `self` when the itsId is set to canonical identifier in the initial request or declared as `digest`, containing the HashedId8 of the EC when the itsId is set to the EC identifier in a successive re-keying request. In the latter case, the digest of the `SignerIdentifier` shall match with the digest in the itsId;
- the `signature` computed using the canonical private key if the `itsId` is set to canonical identifier or the current valid EC private key corresponding to the verification public key.

- An `EtsiTs103097Data-Encrypted` structure is built, with:
  - the component `recipients` containing one instance of `RecipientInfo` of choice `certRecipInfo`, containing:
    - the hashedId8 of the EA certificate in `recipientId`; and
    - the encrypted data encryption key in `encKey`; the public key to use for encryption is the `encryptionKey` found in the EAcertificate referenced in `recipientId`;
  - the component `ciphertext` containing the encrypted representation of the `EtsiTs1030971Data-Signed` structure.



NOTE: [1] Encryption is done with ECIES using the public encryption key of the EA.
[2] In the outer signed data structure, the signature is computed using either the canonical private key if the itsId is set to the canonical identifier or the current valid private key corresponding to the EC's verification public key if the itsId is set to the EC identifier.
[3] In the inner signed data structure (`InnerECRequestSignedForPOP`), the signature is computed on InnerECRequest with the private key corresponding to the new `verificationKey` to prove possession of the generated verification key pair.

Figure 14: EnrolmentRequest message

6.2.3.2.2 Enrolment response

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 `EnrolmentResponse` message shall be encrypted using an ETSI TS 103 097 [3] approved algorithm and the encryption shall be done with the same AES key as the one used by the ITS-S requestor for the encryption of the `EnrolmentRequest` message.

The contents of the `EnrolmentResponse` message shall be as described in Figure 15. The specification of the ITS-S `EnrolmentResponse` message using ASN.1 [6], [7] shall be as specified in clause A.2.

To read an `EnrolmentResponse` message, the ITS-S shall receive an `EtsiTs103097Data-Encrypted` structure, containing an `EtsiTs103097Data-Signed` structure, containing an `EtsiTs102941Data`, containing an `InnerECResponse` structure.

The outermost structure is an `EtsiTs103097Data-Encrypted` structure containing:

- the component `recipients` containing one instance of `RecipientInfo` of choice `pskRecipInfo`, which contains the HashedId8 of the symmetric key used by the ITS-S to encrypt the `EnrolmentRequest` message to which the response is built;
- the component `ciphertext`, once decrypted, contains an `EtsiTs103097Data-Signed` structure.

If the ITS-S has been able to decrypt the content, this expected EtsiTs103097Data-Signed structure shall contain `hashId`, `tbsData`, `signer` and `signature`:

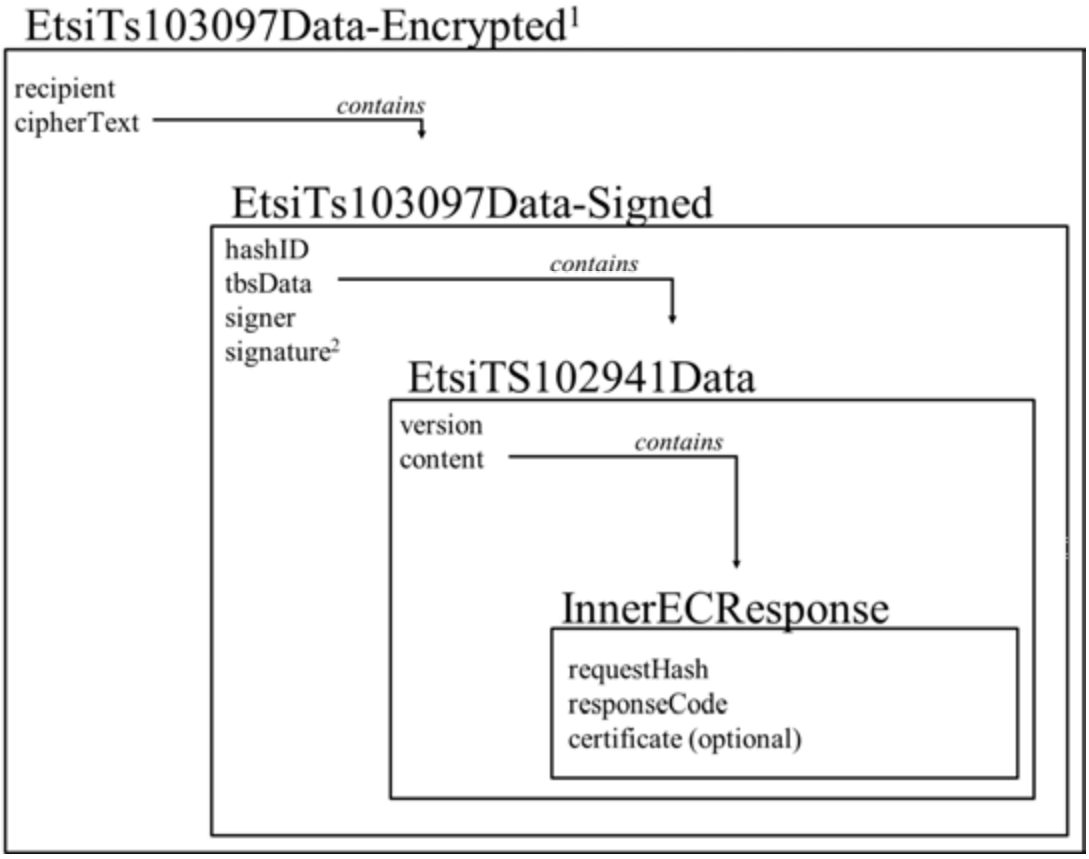
- the `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3].
- in the `tbsData`:
  - the `payload` shall contain the `EtsiTS102941Data` structure;
  - in the `headerInfo`:
    - the `psid` shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];
    - the `generationTime` shall be present;
    - all other components of the component `tbsData.headerInfo` not used and absent;
- the `signer` declared as a `digest`, containing the HashedId8 of the EA certificate;
- the `signature` over `tbsData` computed using the EA private key corresponding to its publicVerificationKey found in the referenced EA certificate.

The `EtsiTS102941Data` shall contain:

- the `version` set to v1 (integer value set to 1);
- the `content` set to `InnerECResponse`.

The `InnerECResponse` shall contain:

- the `requestHash` is the left-most 16 octets of the SHA256 digest of the EtsiTs103097Data - Signed structure received in the request;
- a `responseCode` indicating the result of the request;
- if `responseCode` is 0, indicating a positive response, then a certificate is returned;
- if `responseCode` is different than 0, indicating a negative response, then no certificate will be returned.



NOTE: [1] Encryption is done with the AES key used in ECIES for the encryption of the ECrequest.
[2] Signature computed using the private key corresponding to the EA certificate's verification public key.

Figure 15: EnrolmentResponse message

6.2.3.3 Authorization Management

6.2.3.3.0 Overview

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 clause 6.2.2) to request an authorization ticket to be used in subsequent ITS communications. Figure 16 shows an example of a message sequence for a successful or unsuccessful authorization request.

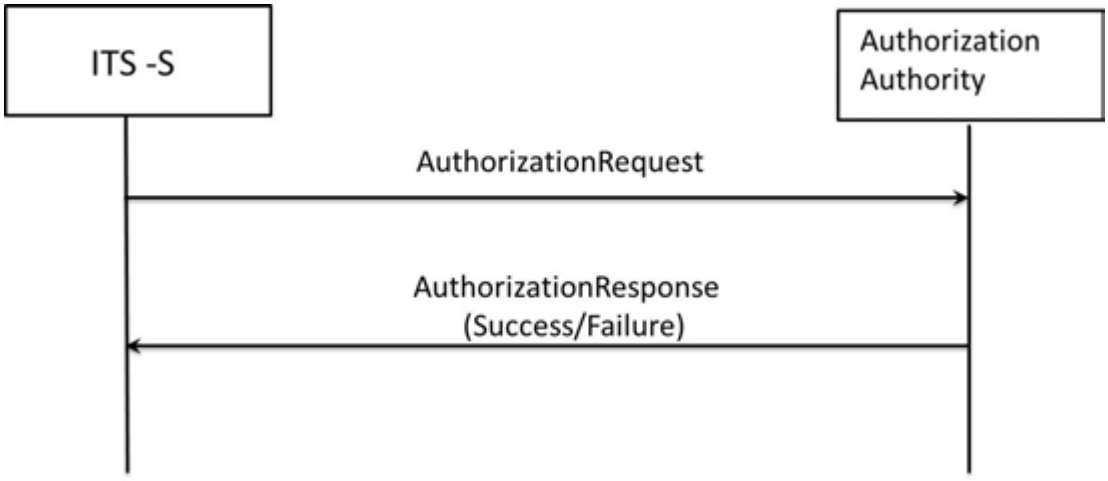


Figure 16: Message sequence for authorization request and response

All messages supporting the authorization process will satisfy the following security and privacy requirements:

- integrity, data origin authenticity, and confidentiality shall be ensured;
- authorization and access control: only registered and authenticated ITS stations shall get Authorization Tickets that enable them to access to cooperative ITS services.

For ITS-S that need privacy protection such as ITS-S vehicles or personal devices, the following requirements apply:

- pseudonymity of the ITS-S requester towards external attackers and against the Authorization Authority should be ensured;
- unlinkability of the ITS-S requesting Authorization Tickets: several successive requests should not be linked to the same ITS-S requester or linked between them.

6.2.3.3.1 Authorization request

The following functional requirements are defined on the state machine of Figure 3 (sender ITS-S for the authorization process):

- The AuthorizationRequest message shall be encrypted using an ETSI TS 103 097 [3] approved encryption algorithm and the public key provided by the authorization authority.
- For each authorization request, the ITS-S shall generate a new verification key pair corresponding to an approved signature algorithm as specified in ETSI TS 103 097 [3].
- The contents of the Authorization Request message shall be as described in Figure 17.
- The complete nested data structure of the Authorization Request message is specified in Figure 17. The specification of the content of the ITS-S Authorization Request message using ASN.1 [6], [7] shall be as specified in clause A.2.

To create an authorization request, the ITS-S shall follow this process:

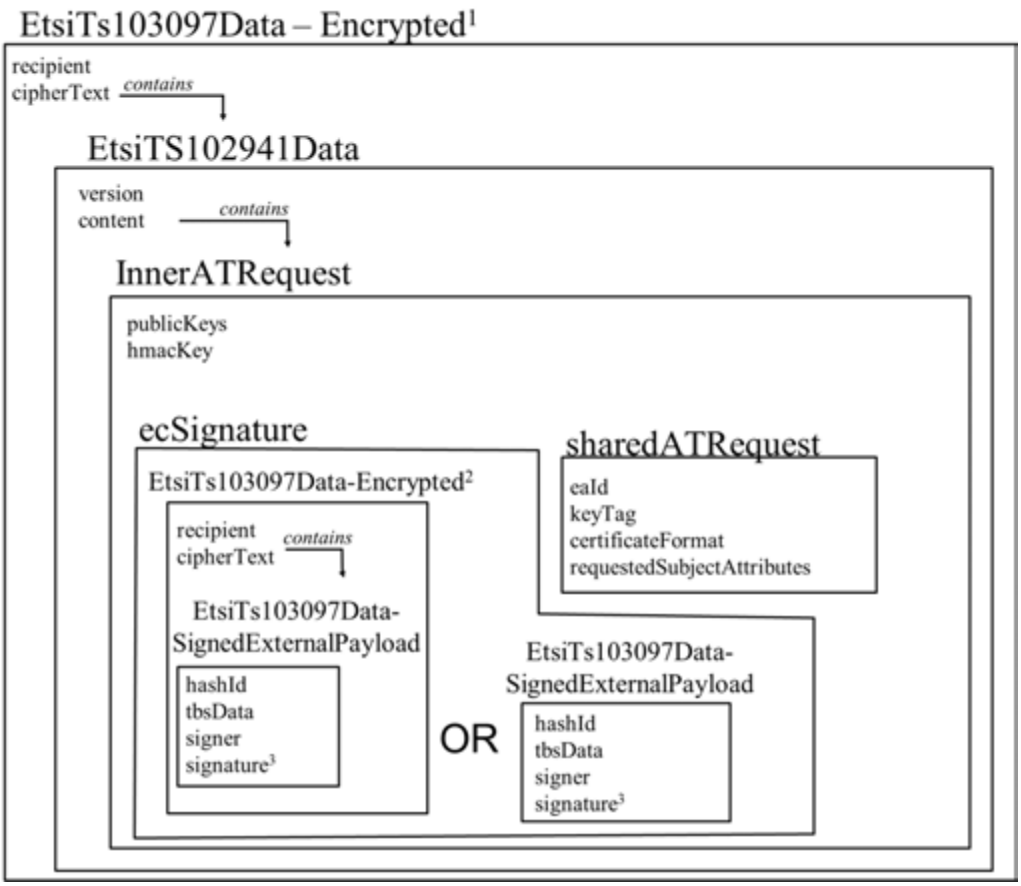
- An ECC private key is randomly generated, the corresponding public key (`verificationKey`) is provided to be included in the AT.
- Optionally, an ECC encryption private key is randomly generated, the corresponding public key (`encryptionKey`) is provided to be included in the AT.
- A random 32 octets long secret key (hmac-key) is generated.
- A tag using the HMAC-SHA256 function is computed using the previously generated hmac-key, on the concatenation of the serialization of `verificationKey` and `encryptionKey` elements (`encryptionKey` is optional); this tag is truncated to the leftmost 128 bits and named `keyTag` (see FIPS 198-1 [13]). By including the tag in the `SharedATRequest` structure the integrity as well as the non-repudiation of the `verificationKey` and `encryptionKey` elements is ensured. The use of an HMAC function ensures that the tag can only be verified by the instances that are in possession of the hmac-key. By this means the AA can verify that the HMAC value of the public keys match the given `keyTag` and the EA is not able to draw conclusions on the keys requested by the ITS-S.
- A `SharedATRequest` structure is built, with:
  - the `eaId` identifying the EA certificate of the EA that can be contacted for validation;
  - the calculated `keyTag`;
  - the `certificateFormat` which specifies the version used for the certificate format specification. In the present document, the certificate format shall be set to ts103097v131 (integer value 1);
  - the desired attributes (`requestedSubjecAttributes`).

- An EtsiTs103097Data-SignedExternalPayload structure is built containing: hashId, tbsData, signer and signature:
  - the hashId shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3];
  - in the tbsData:
    - the payload shall contain an extDataHash with the hash of SharedATRequest;
    - in the headerInfo:
      - the psid shall be se to "secured certificate request" as assigned in ETSI TS 102 965 [19];
      - the generationTime shall be present;
    - all other components of the component tbsdata.headerInfo not used and absent;
  - the signer declared as a digest referencing the hashedId8 of the EC certificate;
  - the signature over tbsData computed using the private key corresponding to the EC's verification public key.
- [Itss_WithPrivacy] An EtsiTs103097Data-Encrypted structure (encryptedEcSignature) is built with:
  - the component recipients with one instance of RecipientInfo of choice certRecipInfo, containing:
    - the HashedId8 of the EA certificate in recipientId; and
    - the encrypted data encryption key in encKey; the public key to use for encryption is the encryptionKey found in the EA certificate referenced in recipientId;
  - the component ciphertext containing the encrypted representation of the EtsiTs103097Data-SignedExternalPayload structure;
  - [Itss_NoPrivacy] For special purpose ITS-Ss which do not require privacy and are allowed to be re-identified by the AA, the message structure shall omit the encryption of the signer information and signature by omitting the pervious step.
- An InnerATRequest structure is built, containing:
  - the publicKeys: a verificationKey requested for certification and an optional encryptionKey to be placed in the same certificate;
  - the generated hmac-key;
  - the SharedATRequest structure;
  - [Itss_WithPrivacy] the encrypted detached signature containing the EtsiTs103097Data-Encrypted structure (encryptedEcSignature);
  - [Itss_NoPrivacy] the not encrypted detached signature containing the EtsiTs103097Data-SignedExternalPayload.
- An EtsiTs102941Data structure is built, with:
  - the version set to v1 (integer value set to 1);
  - the content set to the previous InnerATRequest structure.

The POP signature is optional. That is why there is two types of messages for the authorization request provided in the ASN.1 module present in clause A.2: AuthorizationRequestMessage and AuthorizationRequestMessageWithPop. The use of this signature is recommended since it is an important security mechanism to ensure the trustworthiness of the ITS-S. It ensures that the requesting ITS-S is in possession of the corresponding private key. If present, this `EtsiTs103097Data-Signed` contains a signature providing a proof of possession of the ITS-S verification key that is provided to the AA for certification. This component is an `EtsiTs103097Data-Signed` structure containing (see Figure 18) the `hashId`, the `tbsData`, the `signer` and `signature`:

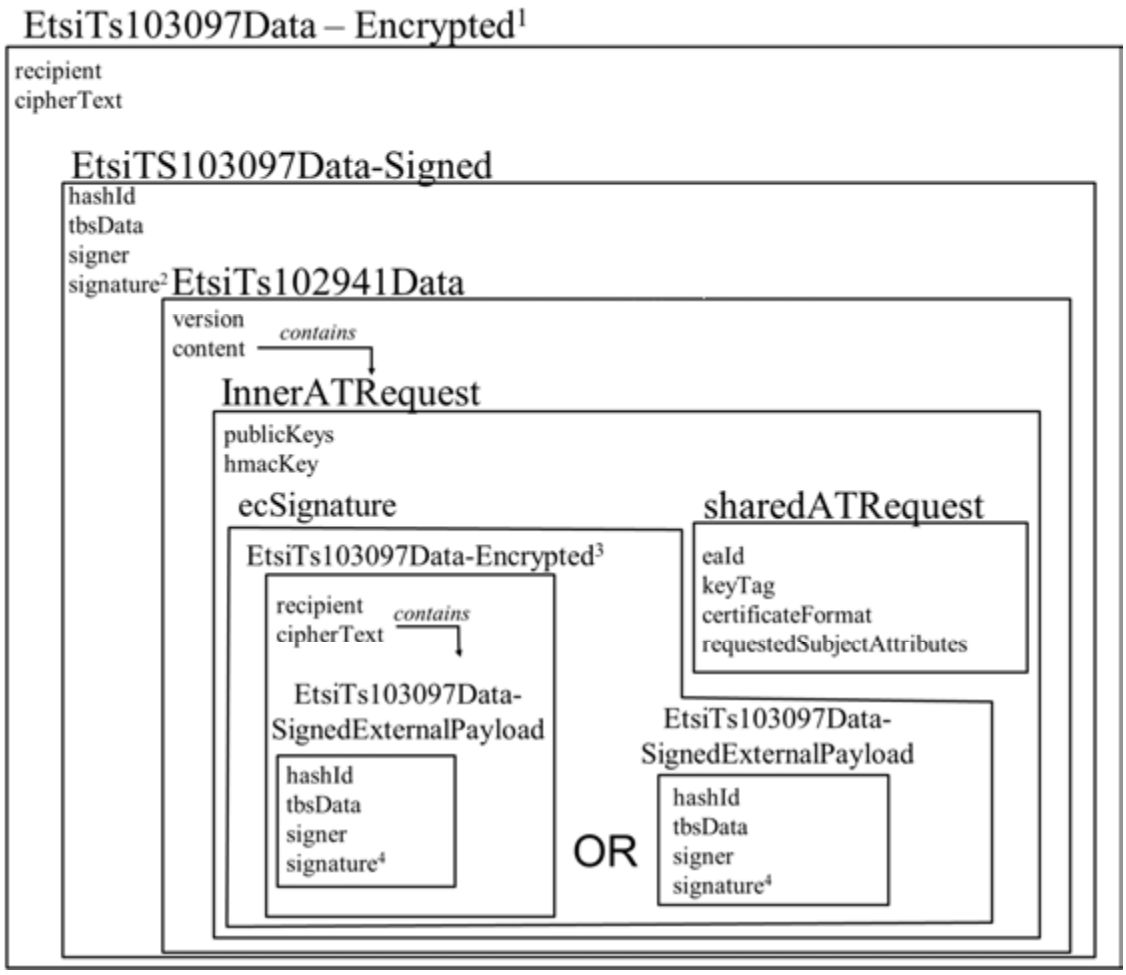
- The `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3].
- In the `tbsData`:
  - the `payload` shall contain the previous `EtsiTs102941Data` structure;
  - in the `headerinfo`:
    - the `psid` shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];
    - the `generationTime` in the `headerInfo` shall be present;
    - all other components of the component `tbsdata.headerInfo` not used and absent.
- The `signer` declared as self.
- The `signature` over `tbsData` computed using the private key corresponding to the verification public key `verificationKey` to be certified and provided in the ATRequest.
- An `EtsiTs103097Data-Encrypted` structure is built, with:
  - the component `recipients` containing one instance of `RecipientInfo` of choice `certRecipInfo`, containing:
    - the hashedId8 of the AA certificate in `recipientId`; and
    - the encrypted data encryption key in `encKey`, the public key to use for encryption is the `encryptionKey` found in the AA certificate referenced in `recipientId`;
  - the component `ciphertext` containing the encrypted representation of the previous `EtsiTs103097Data-Signed` structure if the POP signature is present otherwise the `EtsiTs102941Data structure`.

The `requestedSubjecAttributes` shall not contain a `certIssuePermissions` attribute, but shall contain an `appPermissions` attribute.



NOTE: [1] Encryption is done with ECIES using the public encryption key of the AA.
[2] Encryption is done with ECIES using the public encryption key of the EA.
[3] Signature computed using currently valid private key corresponding to the EC's verification public key.

Figure 17: AuthorizationRequest message



NOTE: [1] Encryption is done with ECIES using the public encryption key of the AA.
[2] Signature computed using the private key corresponding to the verification public key to be certified.
[3] Encryption is done with ECIES using the public encryption key of the EA.
[4] Signature computed using currently valid EC private key corresponding to the verification public key.

Figure 18: AuthorizationRequestMessageWithPop

When receiving an `AuthorizationRequest` or an `AuthorizationRequestMessageWithPop` message, the AA shall make the following verifications before preparing and sending the corresponding `AuthorizationResponse` (see Figure 12):

- AA shall decrypt the `AuthorizationRequest` using the encryption private key corresponding to the recipient certificate (`certRecipInfo` specified in the message `Recipients` field).
- AA shall verify the inner signature (if the POP is present).
- AA shall verify the HMAC-SHA256 value on the concatenation of the serialization of `publicKeys` using the received `hmacKey` to check the request authenticity.
- If the result is equal to the `keyTag` in the received request, the `AA` shall request to the EA identified by its certificate digest (`eaId`) the authorization validation for the requested AT. If result is not equal, the AA shall return an `AuthorizationResponse` with negative response code (different than 0).
- After receiving a positive `AuthorizationValidationResponse` from the EA, the AA shall be authorized to issue the requested AT. Otherwise it shall return a negative response code in the `AuthorizationResponse`.

6.2.3.3.2 Authorization response

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 AuthorizationResponse message shall be encrypted using an ETSI TS 103 097 [3] approved algorithm and the encryption shall be done with the same AES key as the one used by the ITS-S requestor for the encryption of the AuthorizationRequest message.

The complete nested data structure of the AuthorizationResponse message is specified in Figure 19. The specification of the ITS-S AuthorizationResponse message using ASN.1 [6], [7] shall be as specified in clause A.2.

To read an authorization response, the ITS-S shall receive an EtsiTs103097Data-Encrypted structure, containing an EtsiTs103097Data-Signed structure, containing an EtsiTs102941Data structure, containing an authorizationResponse structure:

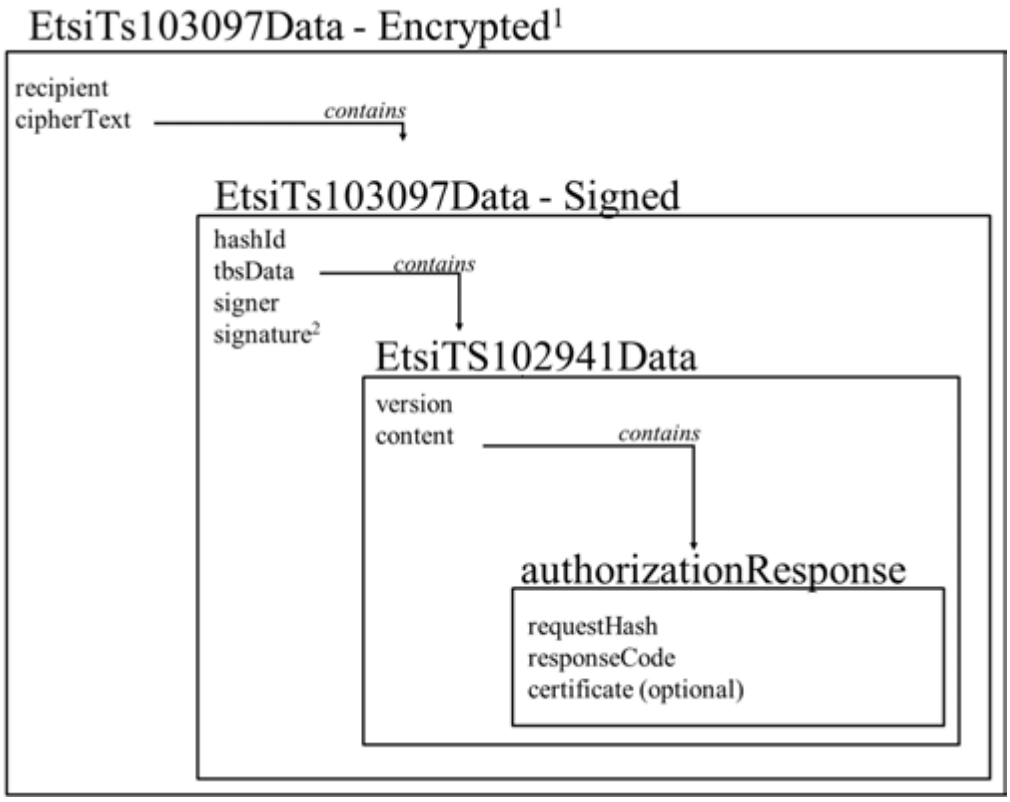
- The outermost structure is an EtsiTs103097Data-Encrypted structure with:
  - the component recipients containing one instance of RecipientInfo of choice pskRecipInfo, which contains the HashedId8 of the symmetric key used by the ITS-S to encrypt the AuthorizationRequest message to which the response is built;
  - the component ciphertext, once decrypted, contains an EtsiTs103097Data-Signed structure.

If the ITS-S has been able to decrypt the content, this expected EtsiTs103097Data-Signed structure shall contain hashId, tbsData, signer and signature:

- The hashId shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3]:
  - in the tbsData:
    - the payload shall contain an EtsiTs102941Data structure;
    - in the headerInfo:
      - the psid shall be se to "secured certificate request" as assigned in ETSI TS 102 965 [19];
      - the generationTime shall be present;
      - all other components of the component tbsdata.headerInfo not used and absent;
  - the signer as a digest referencing the hashedId8 of AA certificate;
  - the signature over tbsData computed using the AA private key corresponding to its public verification key found in the AA certificate.

The authorizationResponse shall contain:

- The requestHash is the left-most 16 octets of the SHA256 digest of the following structure received in the request:
  - EtsiTs102941Data structure received in the request if the POP signature is absent (see Figure 17);
  - EtsiTs103097Data - Signed if POP signature is present (see Figure 18);
- A responseCode indicating the result of the request.
- If responseCode is 0, indicating a positive response, then a certificate is returned.
- If responseCode is different than 0, indicating a negative response, then no certificate will be returned.



NOTE: [1] Encryption is done with the AES key used for the encryption of the ATRequest.
[2] Signature computed using the verification private key associated with the AA certificate.

Figure 19: AuthorizationResponse message

6.2.3.4 Authorization Validation protocol

6.2.3.4.0 Overview

The `AuthorizationValidation` Request message shall be sent by the Authorization Authority (AA) to the Enrolment Authority (EA) across the interface at reference point S_4 (see Figure 8 in ETSI TS 102 940 [5]) to request the validation of the Authorization Ticket (AT) request receiving from an ITS-S.

Figure 20 shows an example of a message sequence for a successful or unsuccessful AT validation request.

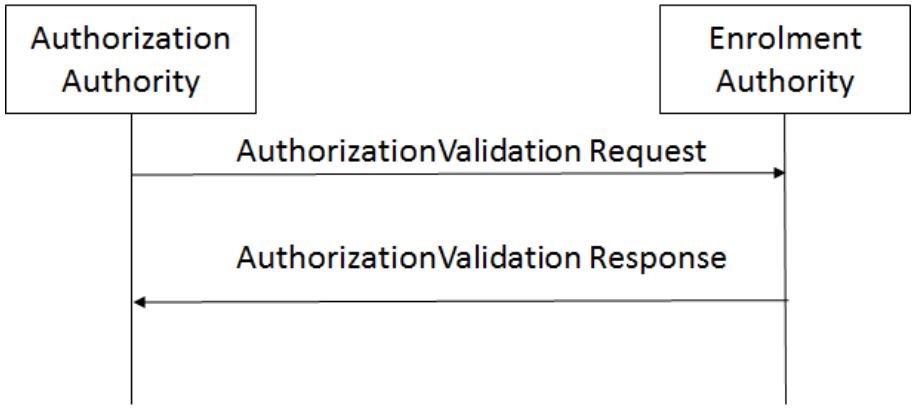


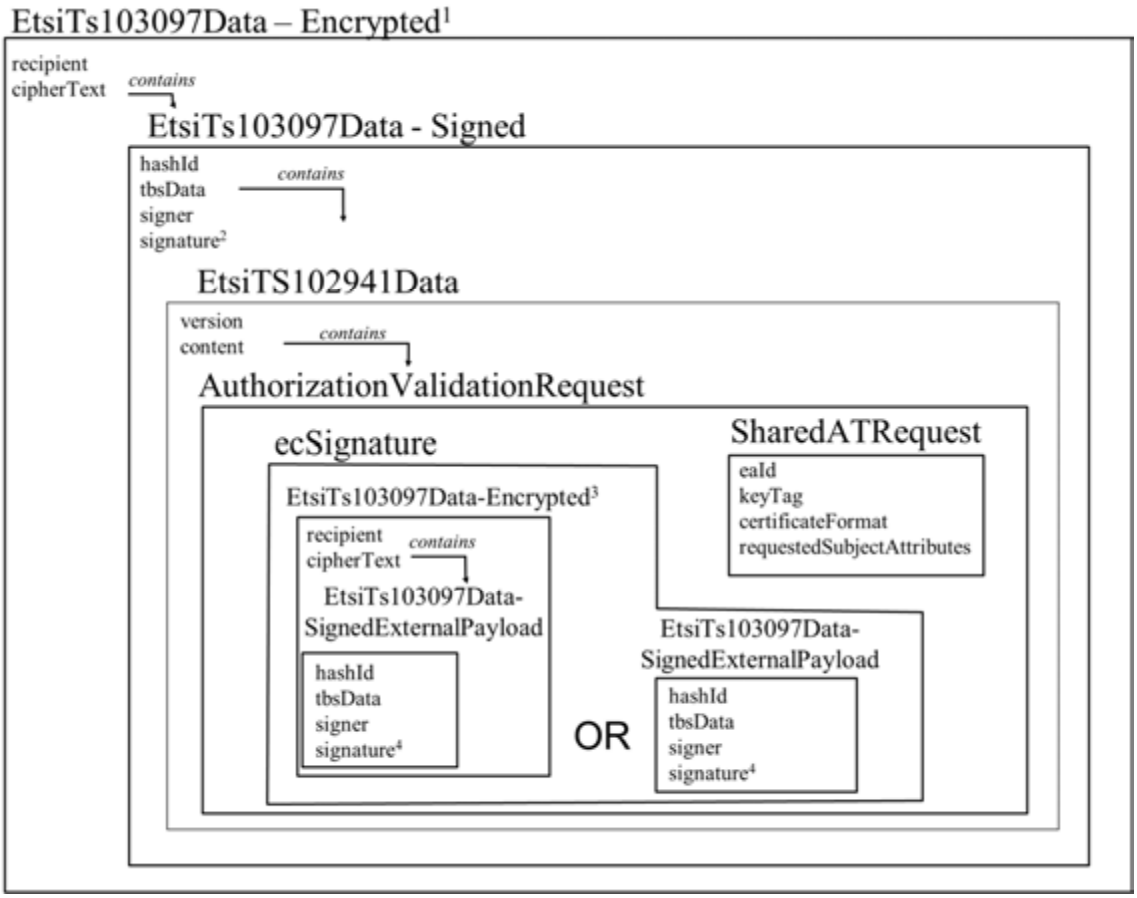
Figure 20: Message sequence for an authorization validation request

6.2.3.4.1 Authorization validation request

The following functional requirements are defined on the communication flow of Figure 20.

The AuthorizationValidationRequest message shall be encrypted using a ETSI TS 103 097 [3] approved encryption algorithm and the public key provided by the enrolment authority.

The contents of the AuthorizationValidationRequest message shall be as described in Figure 21.



NOTE: [1] Encryption is done with ECIES using the public encryption key of the EA.
[2] Signature computed using the private key corresponding to the AA certificate's verification public key.
[3] Encryption is done with ECIES using the public encryption key of the EA.
[4] Signature computed using current valid EC verification private key.

Figure 21: AuthorizationValidationRequest message

The specification of the ITS-S AuthorizationValidationRequest message using ASN.1 [6], [7] shall be as specified in clause A.2.

To create an AuthorizationValidationRequest , the AA shall follow this process:

- An AuthorizationValidationRequest structure is built, with:
  - in the component sharedAtRequest, the sharedAtRequest component from the InnerAtRequest received in the AuthorizationRequestMessage or AuthorizationRequestMessageWithPop;
  - in the component ecSignature, the ecSignature component from the InnerAtRequest received in the AuthorizationRequestMessage or AuthorizationRequestMessageWithPop.
- An EtsiTs102941Data structure is built, with:
  - the version set to v1 (integer value set to 1);
  - the content set to the previous data structure (AuthorizationValidationRequest).

- An EtsiTs103097Data-Encrypted with:
  - the component recipients with one instance of RecipientInfo of choice certRecipInfo, containing:
    - the hashedId8 of the EA certificate in recipientId;
    - and the encrypted data encryption key in encKey; the public key to use for encryption is the encryptionKey found in the EA certificate referenced in recipientId;
  - the component ciphertext containing the encrypted representation of the EtsiTs103097Data-Signed structure.
- An EtsiTs103097Data-Signed structure is built containing hashId, tbsData, signer and signature:
  - the hashId shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3];
  - in the tbsData:
    - the payload shall contain the previous EtsiTs102941Data structure;
    - in the headerInfo:
      - the psid shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];
      - the generationTime shall be present;
      - all other components of the component tbsdata.headerInfo not used and absent;
    - the signer declared as a digest referencing the hashedId8 of the AA certificate;
    - the signature over tbsData computed using the AA private key corresponding to its public verification key found in the AA certificate.

6.2.3.4.2 Authorization validation response

The AuthorizationValidationResponse message shall be sent by the EA to the AA across the interface at reference point S_4 in response to a received AuthorizationValidationRequest message.

The following functional requirements are defined on the communication flow of Figure 20.

The AuthorizationValidationResponse message shall be encrypted using an ETSI TS 103 097 [3] approved algorithm and the response encryption key provided in the AuthorizationValidationRequest message.

The complete nested data structure of the AuthorizationValidationResponse message is specified in Figure 22. The specification of the ITS-S AuthorizationValidationRequest message using ASN.1 [6], [7] shall be as specified in clause A.2.

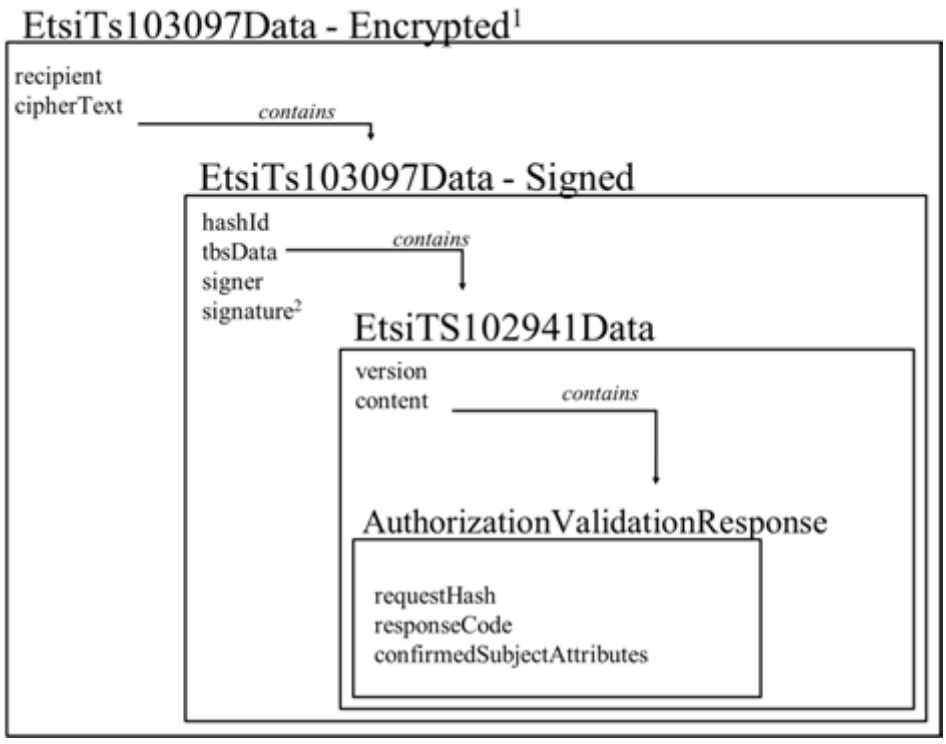
To read an authorization validation response, the AA shall receive an EtsiTs103097Data-Encrypted structure, containing an EtsiTs103097Data-Signed structure, containing an EtsiTs102941Data structure, containing an AuthorizationValidationResponse structure.

To create an AuthorizationValidationResponse message, an EA shall follow this process:

- An AuthorizationValidationResponse structure is built, containing:
  - the requestHash is the left-most 16 octets of the SHA256 digest of the EtsiTs103097Data-Signed structure received in the AuthorizationValidationRequest;
  - a responseCode the response code applying to the request, based on EA internal verification results;
  - if responseCode is 0, in the field confirmedSubjectAttributes, the attributes the EA wishes to confirm, except for certIssuePermissions which is not allowed to be present;

- if `responseCode` is different than 0, no component `confirmedSubjectAttributes`.

- An `EtsiTs102941Data` structure is built, with:
  - the version set to v1 (integer value set to 1);
  - the content set to the previous data structure (`AuthorizationValidationResponse`).
- An `EtsiTs103097Data-Encrypted` with:
  - the component `recipients` containing one instance of `RecipientInfo` of choice `pskRecipInfo`, which contains the HashedId8 of the symmetric key used by the ITS-S to encrypt the `AuthorizationRequest` message to which the response is built;
  - the component `ciphertext` containing the encrypted representation of the `EtsiTs103097Data-Signed`.
- An `EtsiTs103097Data-Signed` structure is built containing `hashId, tbsData, signer` and `signature`:
  - the `hashId` shall indicate the hash algorithm to be used as specified in ETSI TS 103 097 [3];
  - in the `tbsData`:
    - the `payload` shall contain the previous `EtsiTs102941Data` structure;
    - in the `headerInfo`:
      - the `psid` shall be set to "secured certificate request" as assigned in ETSI TS 102 965 [19];
      - the `generationTime` shall be present;
      - all other components of the component `tbsdata.headerInfo` not used and absent;
  - the `signer` declared as a `digest` referencing the hashedId8 of the EA certificate;
  - the `signature` over `tbsData` computed using the EA private key corresponding to its public verification key found in the referenced EA certificate.



NOTE: [1] Encryption is done with the AES key used for the encryption of the AuthorizationValidationRequest.
[2] Signature computed using the private key corresponding to the AA certificate's verification public key.

Figure 22: AuthorizationValidationResponse

A successful response shall contain a subject assurance, a start date and an end date, which shall be in the produced AT certificate. The subject assurance is defined in ETSI TS 103 097 [3].

6.3 Generation, distribution and use of Trust information lists

6.3.1 Generation and distribution of CTL by TLM

The Trust List Manager (TLM) is a superior entity in the C-ITS trust model. It is responsible for approving or rejecting the insertion of a Root CA certificate in the Certificate Trust List (CTL) in accordance with the Policy Authority (PA) as specified in ETSI TS 102 940 [5].

NOTE: In the planned deployment of C-ITS in Europe, the CTL issued by the TLM is called ECTL (European Certificate Trust List).

The TLM shall update and publish the Certificate Trust List (CTL) using the Full CTL format specified in clause 6.3.4. Especially, the update and distribution of CTL information to all C-ITS entities is needed to support the following use cases:

- add a new Root CA;
- update trust information of a Root CA, i.e. renewing the Root CA certificate, removing expired Root CA certificate;
- delete a Root CA certificate (revoked);
- update the TLM certificate after a renewal process (creation of new key and generation of TLM certificates);
- update the CPOC access point (URL) to enable secure distribution of TLM contact details.

The CTL issued by the TLM shall be signed using the TLM verification key and time-stamped using a time-stamp counter (ctlSequence) which is specified as a monotonically increasing counter, with increased value of 1 and maximum value 255. The list is valid when received and shall contain a validity end (nextUpdate).

The CTL issued by the TLM shall contain only the following information:

- TLM certificate and link certificate (optional);
- Root CA certificates and link certificates (optional);
- CPOC access point.

For each update of the CTL, the TLM shall also generate the Delta CTL list using the formats specified in clause 6.3.4. The sequence number of the Full CTL (ctlSequence) shall be identical to the sequence number of the Delta CTL.

The full and delta CTL should be published via the C-ITS Point Of Contact (CPOC).

6.3.2 Generation and distribution of CTL by RCA

The Root CA is the root of trust for the PKI hierarchy. The RCA is responsible of managing and providing trust information about all its subordinate authorities (EAs, AAs), e.g. CA certificates and access points (URL) as specified in ETSI TS 102 940 [5].

The RCA shall update and publish the Certificate Trust List (CTL) using the Full CTL format specified in clause 6.3.4. Especially, the update and distribution of CTL information to all C-ITS entities is needed to support the following use cases:

- add a new trusted EA or AA;
- update EA certificates (renewing the EA certificate, removing expired EA certificate) and access points (URL);
- update AA certificates (renewing the AA certificate, removing expired AA certificate) and access points (URL);
- update DC access point to enable secure distribution of RCA contact details.

The CTL issued by the RCA shall be signed using the private key corresponding to the RCA certificate's verification public key and time-stamped using a time-stamp counter (ctlSequence) which is specified as a monotonically increasing counter, with increased value of 1 and maximum value 255. The list is valid when received and shall contain a validity end (nextUpdate).

The CTL issued by a RCA may contain the following information:

- EA certificates, link certificates (optional) and access points;
- AA certificates and access point;
- DC access point.

The CTL issued by a RCA shall not contain the following information: the TLM certificate and associated linked certificate (optional) and the Root CAs certificates.

For each update of the CTL, the RCA shall also generate the Delta CTL list using the formats specified in clause 6.3.4. The sequence number of the Full CTL (ctlSequence) shall be identical to the sequence number of the Delta CTL.

The full and delta CTL should be published via the Distribution Centre (DC) associated to the RCA.

6.3.3 Generation and distribution of CRL by RCA

A Certificate Revocation List (CRL) is issued and signed by the RCA verification key. It contains the CA certificates identifiers (HashedID8) that are no longer worthy of being trusted. It is accessible through a defined DC associated to the RCA.

NOTE: The CRL may contain revoked Root CA certificates.

The CRL issued by a Root CA shall not contain RCA certificates identifiers of other RCAs.

The RCA shall update the CRL by adding the identifiers of the revoked CA certificates and removing the identifiers of the expired ones.

The generated CRL CA shall be signed and time-stamped (thisUpdate) and shall use the CRL data structure as specified in clause A.2.7.

The RCA shall publish and distribute the updated CRL CA via a Distribution Centre (DC).

6.3.4 Specification of Full CTL and Delta CTL

The CTL messages are split in two different message types depending of the issuer category: the TlmCertificateTrustListMessage shall be issued by a TLM entity and an RcaCertificateTrustListMessage shall be issued by a Root CA.

The `signer` in the `SignedData` shall contain the certificate of the CTL issuer.

The CTL data structure as specified in clause A.2.7 takes into account the need for transmitting the Full CTL list over any kind of medium (providing that message fragmentation is supported) and the possible transmission of a small size format for transmission over ITS G5 (Delta CTL).

A `ToBeSignedTlmCtl` or `ToBeSignedRcaCtl` shall be of type `FullCtl` or `DeltaCtl`. Both types use the same common format `CtlFormat`.

The type `CtlFormat` shall have the following components:

- version indicates the version of the CTL Format. For this version of the Technical Specification the version is set to 1;
- nextUpdate indicates the time when a next update of the CTL is expected, and consequently the time after which the CTL is to be considered expired;
- isFullCtl is a flag indicating if the list is a full or delta list;
- ctlSequence indicated the sequence number of the list and is monotonically increased with steps of one unit, and clipped around 255;
- ctlCommands contains the CTL commands add or delete, add is of type CtlEntry, and indicates that the entry shall be trusted; delete is of type CtlDelete, and indicates explicitly that an entry that was previously trusted is not trustable anymore and shall be removed.

A `ToBeSignedCtl` of type `FullCtl` shall have `isFullCtl` set to `TRUE` and shall contain only `ctlCommands` of type `CtlEntry`, generating the full list of trustable entries.

A `ToBeSignedCtl` of type `DeltaCtl` shall have `isFullCtl` set to `FALSE`, and shall contain `ctlCommands` of type `CtlEntry` to indicate trust in new or updated entries, and `ctlCommands` of type `CtlDelete` to indicate that trust in previous entries has been revoked, and that such entries shall be considered deleted from the full List. A `ToBeSignedCtl` of type `DeltaCtl` shall always be generated with respect to the previous full list, i.e. the full list with ctlSequence with one unit less than the current.

Note that a full list and a delta list may exist in parallel, e.g. a full list of `ctlSequence` 5 may coexist with a delta list of `CtlSequence` 5, generated with respect to the Full List of `CtlSequence` of 4.

A `CtlEntry` shall provide either a `rca`, `ea`, `aa`, `dc` or `tlm` entry of type respectively `RootCaEntry`, `EaEntry`, `AaEntry`, `DcEntry`, `TlmEntry`.

A `CtlDelete` shall provide either the `cert` to revoke a `CtlEntry` entry corresponding to a certificate (i.e. `rca`, `ea`, `aa`, `tlm`) using its HashedId8 or the `dc` to delete a `CtlEntry` entry corresponding to a distribution centre using its URL.

`RootCAEntry` shall contain a Root CA certificate and optionally a link certificate.

`EaEntry` shall contain an EA certificate, the URL for the connection by the AA, and optionally the URL for the connection by the ITS-Station.

`AaEntry` shall contain an AA certificate, and optionally the URL for the connection by the ITS-Station.

`TlmEntry` shall contain the TLM Certificate which is a self-signed certificate, optionally the TLM link certificate and the CPOC URL for the connection by the all the entities of C-ITS trust domain.

`DcEntry` shall contain the URL of the Distribution Centre and a list of certificate digests (HashedId8).

The CTL issued by a RCA shall contain one or multiple DCEntry with elements as follows:

- URL of Distribution Centre concerned;
- the RCA(s) certificate digests that publish via the Distribution Centre. This list of certificate digests may not be exhaustive.

A CTL issued by a RCA shall have at least one DC entry where its certificate is contained in the list of certificate digests.

6.3.5 Transmission of CTL and CRL

Different methods for updating the enrolled ITS-Ss may be used as specified in clause 6.1.5. Communication profiles for transmitting the CTL or CRL CA to enrolled ITSs are specified in annex D.

For broadcasting a CTL issued by TLM over ITS-G5, an ITS-S shall re-transmit the received TlmCertificateTrustListMessage defined in clause A.2 without modifications. The CTL transmitted issued by the TLM shall contain a DeltaCtl (the value of CtlFormat shall be set to DeltaCtl). The communication profile for CTL broadcast communication is specified in clause D.3.

For broadcasting a CTL issued by RCA over ITS-G5, an ITS-S shall re-transmit the received RcaCertificateTrustListMessage defined in clause A.2 without modifications. The CTL transmitted issued by a RCA shall contain a DeltaCtl (the value of CtlFormat shall be set to DeltaCtl). The communication profile for CTL broadcast communication is specified in clause D.3.

For broadcasting a CRL over ITS-G5, an ITS-S shall re-transmit the received CertificateRevocationListMessage defined in Annex A without modifications.

The communication profile for CRL broadcast communication is specified in clause D.3.

6.3.6 CTL and CRL use by ITS-Ss

Upon the reception of the CTL or the CRL, the ITS-S verifies that it is signed by the associated RCA.

The ITS-S uses the CTL to update its trust information list in terms of trust anchors. This information allows to build a candidate certificate chain only based on declared `issuer` component as specified in [3].

The ITS-S extracts the list of revoked CA certificates from the CRL and utilizes this information while performing the validation of the certification path.

7 Security association and key management between ITS Stations

7.0 Introduction

A detailed set of use case examples for ITS applications is presented in ETSI TR 102 638 [i.2]. In addition, ETSI 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 ETSI TS 102 942 [4] and ETSI TS 102 943 [9].

7.1 Broadcast SAs

Broadcast applications such as CA basic service, DEN basic service or Infrastructure messages service specified in [10], [11] and [12] require authentication, authorization and integrity but not confidentiality. Senders of CAM and DENM shall obtain this service by signing with an authorization ticket using the mechanisms specified in ETSI TS 103 097 [3] clause 7.1 (Security profile for CAMs), 7.2 (Security profile for DENMs) or 7.3 (Generic profile for other broadcast messages) respectively.

To enable broadcast message signature verification, a receiving ITS-S or a relaying ITS-S shall have the capabilities to construct the certificate chain from the certificate of the Sending ITS-S up to a trust anchor. A trust anchor is either a trusted root certificate or any other CA certificate that is known to be trustworthwy from the receiving or relaying ITS-S.

A receiving or relaying ITS-S shall have the capabilities to cryptographically verify the constructed certificate chain.

A receiving or relaying ITS-S shall have the capabilities to cryptographically verify the signature of the broadcasted message.

7.2 Multicast SAs

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

The following specifications assume that the ITS-S is using an IP-based multicast communication.

An ITS-S may join a multicast group using an authorization ticket (see clause 6.2.3.3) 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].

For other use cases using communications that are not IP-based, ETSI TS 103 097 [3] provides data structures that may be used to establish SA in a multicast group.

For key establishment, the multicast group leader should generate a fresh symmetric key k, use it to encrypt the first multicast message (with AES-CCM), then encrypt k for each recipient with the corresponding recipient key.

The structure `EtsiTs103097Data-Encrypted` can be used to encapsulate all this data in the following way:

- The field `recipients` shall contain one or more `RecipientInfo`s, one for each key used to encrypt the symmetric key k.
- Each `RecipientInfo` shall be of type `certRecipInfo` or `signedDataRecipInfo`:
  - `certRecipInfo` shall be used if the symmetric key k is encrypted with a public encryption key present in a certificate.
  - `signedDataRecipInfo` shall be used if the symmetric key k is encrypted with an encryption key present in a previously received `SignedData` structure.
- The field `ciphertext` shall contain the encrypted message.

After each recipient in the multicast group receives the encrypted data structure, it shall decrypt the key k and the encrypted multicast message. This key may be reused to encrypt further confidential multicast messages, with the `pskRecipInfo` option in the `RecipientInfo` that indicates the use of a pre-shared symmetric key. However, the AES-CCM nonce shall be chosen randomly and shall never be reused with the same key.

If the multicast group already has a pre-shared key obtained through another key establishment protocol, the group can use this key with the structure `EtsiTs103097Data-Encrypted` and the `pskRecipInfo` option in the `RecipientInfo`.

The structure EtsiTs103097Data-SignedAndEncrypted can be used if authentication is also required in the key establishment step and/or the multicast group communication.

NOTE: The detailed requirements and the protocols for secure group communications with message authentication, authorization and encryption will be later specified in ETSI TS 102 943 [9].

7.3 Unicast SAs

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

The following specifications assume that the ITS-S is using an IP-based unicast communication.

An ITS-S may join such services using its authorization ticket 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 RFC 4301 [i.4], IETF RFC 4877 [i.12], IETF RFC 4306 [i.11], IETF RFC 4302 [i.5] and IETF RFC 4303 [i.6]. Also, security in the transport layer can be provided using methods such as the IETF Transport Layer Security (TLS) [i.7].

For other use cases such as the Trust and Privacy management services specified in the present document, other unicast solutions using the ETSI ITS security standards for ITS G5 communications shall be considered (ETSI TS 103 097 [3]).

For non-IP unicast SA, it is possible to proceed in the same way as multicast SA to establish a shared key. The SA initiator should generate a fresh symmetric key k, use it to encrypt the first confidential message, then encrypt k for the recipient with the corresponding recipient key. The structure `EtsiTs103097Data-Encrypted` shall be used to encapsulate all this data in the following way:

- The field `recipients` shall contain one `RecipientInfo` of type `certRecipInfo` or `signedDataRecipInfo` depending on which recipient key is used.
- The field `ciphertext` shall contain the encrypted message.

After the recipient receives the encrypted data structure, it shall decrypt the key k and the encrypted message. This key may be reused to encrypt further confidential messages, with the `pskRecipInfo` option in the `RecipientInfo` that indicates the use of a pre-shared symmetric key. However, the AES-CCM nonce shall be chosen randomly and shall never be reused with the same key.

If the units involved in a unicast communication already have a pre-shared key obtained through another key establishment protocol, the units can use this key with the structure `EtsiTs103097Data-Encrypted` and the `pskRecipInfo` option in the `RecipientInfo`.

The structure EtsiTs103097Data-SignedAndEncrypted can be used if authentication is also required in the key establishment step and/or the unicast communication.

NOTE: The detailed requirements and the protocols for Secure unicast communications with message authentication, authorization and encryption will be later specified in ETSI TS 102 943 [9].

Annex A (normative): ITS security management 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. Security Management Messages specified in the present document shall comply with the structures specified here but the definitive encoding of messages in an implementation of the present document is specified in clause 6.2.0.

A.2 Security management messages structures

A.2.1 Security data structures

```
EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
   version1(1) }

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS
  HashedId8, Time32, PublicEncryptionKey, PublicVerificationKey, Signature
FROM
  IEEE1609dot2BaseTypes {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)}

  CertificateId, SubjectAssurance, SequenceOfPsidSsp, SequenceOfPsidGroupPermissions,
  ValidityPeriod, GeographicRegion
FROM
  IEEE1609dot2 {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)}

  EtsiTs103097Data-Encrypted, EtsiTs103097Data-Signed,
   EtsiTs103097Data-SignedExternalPayload
FROM
  EtsiTs103097Module { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5)
   ts(103097) securedMessageV1(0)}
;

CertificateFormat::= INTEGER {
  ts103097v131 (1)
  }(1..255)

CertificateSubjectAttributes ::= SEQUENCE {
  id       CertificateId OPTIONAL,
  validityPeriod        ValidityPeriod OPTIONAL,
  region                GeographicRegion OPTIONAL,
  assuranceLevel        SubjectAssurance OPTIONAL,
  appPermissions        SequenceOfPsidSsp OPTIONAL,
  certIssuePermissions  SequenceOfPsidGroupPermissions OPTIONAL,
  ...
  }(WITH COMPONENTS { ..., appPermissions PRESENT} |
   WITH COMPONENTS { ..., certIssuePermissions PRESENT})

EcSignature::= CHOICE {
  encryptedEcSignature  EtsiTs103097Data-Encrypted{EtsiTs103097Data-
   SignedExternalPayload},
```

```
  ecSignature           EtsiTs103097Data-SignedExternalPayload
  }

PublicKeys ::= SEQUENCE {
  verificationKey       PublicVerificationKey,
  encryptionKey         PublicEncryptionKey OPTIONAL
  }

Version ::= INTEGER {v1(1)}

EtsiTs103097Data-Encrypted-Unicast {ToBeEncryptedDataContent} ::= EtsiTs103097Data-
   Encrypted {ToBeEncryptedDataContent}
(WITH COMPONENTS {...,
  content (WITH COMPONENTS {
    encryptedData (WITH COMPONENTS {...,
       recipients (SIZE(1))
    })
  })
})

EtsiTs103097Data-SignedAndEncrypted-Unicast {ToBesignedAndEncryptedDataContent} ::=
   EtsiTs103097Data-Encrypted-Unicast {EtsiTs103097Data-Signed
   {ToBesignedAndEncryptedDataContent}}

END
```

A.2.2 Security Management messages for CA

```
/**************************************************************************************
   This file contains the EtsiTs102941Messages module containing all possible PKI
messages.
   It should be used when all PKI messages needs to be implemented (for example, for CA
development)
**************************************************************************************/
EtsiTs102941MessagesCa
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
messagesCa(0) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Data-Signed,
--EtsiTs103097Data-Encrypted,
EtsiTs103097Data-SignedExternalPayload
--EtsiTs103097Data-SignedAndEncrypted
FROM EtsiTs103097Module
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0)}

Version,
EtsiTs103097Data-Encrypted-Unicast,
EtsiTs103097Data-SignedAndEncrypted-Unicast
FROM EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1) }

InnerEcRequestSignedForPop, InnerEcResponse
FROM EtsiTs102941TypesEnrolment
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) enrolment(4)
version1(1) }

InnerAtRequest, InnerAtResponse
FROM EtsiTs102941TypesAuthorization
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authorization(5) version1(1) }

ToBeSignedCrl, ToBeSignedTlmCtl, ToBeSignedRcaCtl
```

```
FROM EtsiTs102941TrustLists
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
trustLists(6) version1(1) }

AuthorizationValidationRequest, AuthorizationValidationResponse
FROM EtsiTs102941TypesAuthorizationValidation
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authValidation(7) version1(1) }

CaCertificateRequest
FROM EtsiTs102941TypesCaManagement
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
caManagement(8) version1(1) }

;

/*************
-- Messages
*************/
EnrolmentRequestMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast {EtsiTs102941Data
(WITH COMPONENTS{..., content (WITH COMPONENTS{enrolmentRequest PRESENT})})}
EnrolmentResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{enrolmentResponse
PRESENT})})}
AuthorizationRequestMessage ::= EtsiTs103097Data-Encrypted-Unicast {EtsiTs102941Data
(WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationRequest PRESENT})})}
AuthorizationRequestMessageWithPop ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationRequest
PRESENT})})}
AuthorizationResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationResponse
PRESENT})})}
CertificateRevocationListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateRevocationList PRESENT})})}
TlmCertificateTrustListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateTrustListTlm PRESENT})})}
RcaCertificateTrustListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateTrustListRca PRESENT})})}
AuthorizationValidationRequestMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH
COMPONENTS{authorizationValidationRequest PRESENT})})}
AuthorizationValidationResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH
COMPONENTS{authorizationValidationResponse PRESENT})})}
CaCertificateRequestMessage ::= EtsiTs103097Data-Signed {EtsiTs102941Data(WITH
COMPONENTS{..., content (WITH COMPONENTS{caCertificateRequest PRESENT})})}
CaCertificateRekeyingMessage ::= EtsiTs103097Data-Signed {EtsiTs103097Data-Signed
{EtsiTs102941Data(WITH COMPONENTS{..., content (WITH COMPONENTS{caCertificateRequest
PRESENT})})}}

/*************
-- EtsiTs102941Data
*************/

EtsiTs102941Data::= SEQUENCE {
  version Version (v1),
  content EtsiTs102941DataContent
  }

EtsiTs102941DataContent ::= CHOICE {
  enrolmentRequest                    InnerEcRequestSignedForPop,
  enrolmentResponse                   InnerEcResponse,
  authorizationRequest                InnerAtRequest,
  authorizationResponse               InnerAtResponse,
  certificateRevocationList           ToBeSignedCrl,
  certificateTrustListTlm             ToBeSignedTlmCtl,
  certificateTrustListRca             ToBeSignedRcaCtl,
  authorizationValidationRequest      AuthorizationValidationRequest,
  authorizationValidationResponse     AuthorizationValidationResponse,
  caCertificateRequest                CaCertificateRequest,
```

```
  ...
  }

END
```

A.2.3 Security Management messages for ITS-S_WithPrivacy

```
********************************************************************************
   This file contains the EtsiTs102941MessagesItss module providing the ITS-S subset
   of messages defined in the module EtsiTs102941MessagesCA
   It should never be imported together with the module EtsiTs102941MessagesCA.
   Use the EtsiTs102941MessagesCA if all possible PKI message types are needed.

   This module blocks the usage of unencrypted EC signature for AA requests.
********************************************************************************/
EtsiTs102941MessagesItss
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
messagesItss(1) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Data-Signed
--EtsiTs103097Data-Encrypted,
--EtsiTs103097Data-SignedAndEncrypted
FROM EtsiTs103097Module
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0) }

EtsiTs103097Data-Encrypted-Unicast,
EtsiTs103097Data-SignedAndEncrypted-Unicast,
Version
FROM EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1) }

InnerEcRequestSignedForPop, InnerEcResponse
FROM EtsiTs102941TypesEnrolment
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) enrolment(4)
version1(1) }

InnerAtRequest, InnerAtResponse
FROM EtsiTs102941TypesAuthorization
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authorization(5) version1(1) }

ToBeSignedCrl, ToBeSignedTlmCtl, ToBeSignedRcaCtl
FROM EtsiTs102941TrustLists
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
trustLists(6) version1(1) }

;

/************
-- Messages
************/

EnrolmentRequestMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast {EtsiTs102941Data
(WITH COMPONENTS{..., content (WITH COMPONENTS{enrolmentRequest PRESENT})})}
EnrolmentResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{enrolmentResponse
PRESENT})})}
AuthorizationRequestMessage ::= EtsiTs103097Data-Encrypted-Unicast {EtsiTs102941Data
(WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationRequest PRESENT})})}
AuthorizationRequestMessageWithPop ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationRequest
PRESENT})})}
```

```
AuthorizationResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationResponse
PRESENT})})}
CertificateRevocationListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateRevocationList PRESENT})})}
TlmCertificateTrustListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateTrustListTlm PRESENT})})}
RcaCertificateTrustListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateTrustListRca PRESENT})})}

/************
-- EtsiTs102941Data
************/

EtsiTs102941Data::= SEQUENCE {
  version Version (v1),
  content EtsiTs102941DataContent
  }

EtsiTs102941DataContent ::= CHOICE {
  enrolmentRequest                        InnerEcRequestSignedForPop,
  enrolmentResponse                       InnerEcResponse,
  authorizationRequest                    InnerAtRequest,
  authorizationResponse                   InnerAtResponse,
  certificateRevocationList               ToBeSignedCrl,
  certificateTrustListTlm                 ToBeSignedTlmCtl,
  certificateTrustListRca                 ToBeSignedRcaCtl,
  ...
  } (WITH COMPONENTS{...,
    authorizationRequest (WITH COMPONENTS{...,
      ecSignature (WITH COMPONENTS{...,
        encryptedEcSignature PRESENT
      })
    })
  })

END
```

A.2.4 Security Management messages for ITSS_NoPrivacy

```
/*****************************************************************************
   This file contains the EtsiTs102941MessagesItss-OptionalPrivacy module providing the
   same subset of messages as the EtsiTs102941MessagesItss module.
   It should never be used together with the EtsiTs102941MessagesCA and
EtsiTs102941MessagesItss

   This module allows the usage of unencrypted EC signature for AA requests.
*****************************************************************************/
EtsiTs102941MessagesItss-OptionalPrivacy
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
messagesItssOp(2) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Data-Signed
--EtsiTs103097Data-Encrypted,
--EtsiTs103097Data-SignedAndEncrypted
FROM EtsiTs103097Module
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0) }

EtsiTs103097Data-Encrypted-Unicast,
EtsiTs103097Data-SignedAndEncrypted-Unicast,
Version
FROM EtsiTs102941BaseTypes
```

```
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1) }

InnerEcRequestSignedForPop, InnerEcResponse
FROM EtsiTs102941TypesEnrolment
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) enrolment(4)
version1(1) }

InnerAtRequest, InnerAtResponse
FROM EtsiTs102941TypesAuthorization
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authorization(5) version1(1) }

ToBeSignedCrl, ToBeSignedTlmCtl, ToBeSignedRcaCtl
FROM EtsiTs102941TrustLists
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
trustLists(6) version1(1) }

;

/*************
-- Messages
*************/

EnrolmentRequestMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast {EtsiTs102941Data
(WITH COMPONENTS{..., content (WITH COMPONENTS{enrolmentRequest PRESENT})})}
EnrolmentResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{enrolmentResponse
PRESENT})})}
AuthorizationRequestMessage ::= EtsiTs103097Data-Encrypted-Unicast {EtsiTs102941Data
(WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationRequest PRESENT})})}
AuthorizationRequestMessageWithPop ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationRequest
PRESENT})})}
AuthorizationResponseMessage ::= EtsiTs103097Data-SignedAndEncrypted-Unicast
{EtsiTs102941Data (WITH COMPONENTS{..., content (WITH COMPONENTS{authorizationResponse
PRESENT})})}
CertificateRevocationListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateRevocationList PRESENT})})}
TlmCertificateTrustListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateTrustListTlm PRESENT})})}
RcaCertificateTrustListMessage ::= EtsiTs103097Data-Signed{EtsiTs102941Data (WITH
COMPONENTS{..., content (WITH COMPONENTS{certificateTrustListRca PRESENT})})}

/*************
-- EtsiTs102941Data
*************/

EtsiTs102941Data::= SEQUENCE {
  version Version (v1),
  content EtsiTs102941DataContent
  }

EtsiTs102941DataContent ::= CHOICE {
  enrolmentRequest                    InnerEcRequestSignedForPop,
  enrolmentResponse                   InnerEcResponse,
  authorizationRequest                InnerAtRequest,
  authorizationResponse               InnerAtResponse,
  certificateRevocationList           ToBeSignedCrl,
  certificateTrustListTlm             ToBeSignedTlmCtl,
  certificateTrustListRca             ToBeSignedRcaCtl,
  ...
  }

END
```

A.2.5 Enrolment and authorization data types

A.2.5.1 Enrolment

```
EtsiTs102941TypesEnrolment
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
enrolment(4) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Certificate,
EtsiTs103097Data-Signed
FROM EtsiTs103097Module
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0)}

CertificateFormat, CertificateSubjectAttributes, EcSignature, HashedId8, PublicKeys,
Version
FROM EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1) }

;

/************
-- EnrolmentRequest/Response
************/

EnrolmentResponseCode ::= ENUMERATED {
  ok(0),
  cantparse, -- valid for any structure
  badcontenttype, -- not encrypted, not signed, not enrolmentrequest
  imnottherecipient, -- the "recipients" doesn't include me
  unknownencryptionalgorithm, -- either kexalg or contentencryptionalgorithm
  decryptionfailed, -- works for ECIES-HMAC and AES-CCM
  unknownits, -- can't retrieve the ITS from the itsId
  invalidsignature, -- signature verification of the request fails
  invalidencryptionkey, -- signature is good, but the responseEncryptionKey is bad
  baditsstatus, -- revoked, not yet active
  incompleterequest, -- some elements are missing
  deniedpermissions, -- requested permissions are not granted
  invalidkeys, -- either the verification_key of the encryption_key is bad
  deniedrequest, -- any other reason?
  ... }

InnerEcRequestSignedForPop::= EtsiTs103097Data-Signed{InnerEcRequest}

InnerEcRequest ::= SEQUENCE {
  itsId                                IA5String,
  certificateFormat                    CertificateFormat,
  publicKeys                           PublicKeys,
  requestedSubjectAttributes           CertificateSubjectAttributes (WITH
COMPONENTS{certIssuePermissions ABSENT}),
  ...
 }

InnerEcResponse ::= SEQUENCE {
  requestHash                          OCTET STRING (SIZE(16)),
  responseCode                         EnrolmentResponseCode,
  certificate                          EtsiTs103097Certificate OPTIONAL,
  ...
 }
  (WITH COMPONENTS { responseCode (ok), certificate PRESENT }
     | WITH COMPONENTS { responseCode (ALL EXCEPT ok), certificate ABSENT }
  )
```

```
END
```

A.2.5.2 Authorization

```
EtsiTs102941TypesAuthorization
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authorization(5) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Certificate,
EtsiTs103097Data-Signed
FROM EtsiTs103097Module
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0)}

CertificateFormat, CertificateSubjectAttributes, EcSignature, HashedId8, PublicKeys,
Version
FROM EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1) }

;

/************
-- AuthorizationRequest/Response
************/

AuthorizationResponseCode ::= ENUMERATED {
  ok(0),
  -- ITS->AA
  its-aa-cantparse, -- valid for any structure
  its-aa-badcontenttype, -- not encrypted, not signed, not authorizationrequest
  its-aa-imnottherecipient, -- the "recipients" of the outermost encrypted data doesn't
include me
  its-aa-unknownencryptionalgorithm, -- either kexalg or contentencryptionalgorithm
  its-aa-decryptionfailed, -- works for ECIES-HMAC and AES-CCM
  its-aa-keysdontmatch, -- HMAC keyTag verification fails
  its-aa-incompleterequest, -- some elements are missing
  its-aa-invalidencryptionkey, -- the responseEncryptionKey is bad
  its-aa-outofsyncrequest, -- signingTime is outside acceptable limits
  its-aa-unknownea, -- the EA identified by eaId is unknown to me
  its-aa-invalidea, -- the EA certificate is revoked
  its-aa-deniedpermissions, -- I, the AA, deny the requested permissions
  -- AA->EA
  aa-ea-cantreachea, -- the EA is unreachable (network error?)
  -- EA->AA
  ea-aa-cantparse, -- valid for any structure
  ea-aa-badcontenttype, -- not encrypted, not signed, not authorizationrequest
  ea-aa-imnottherecipient, -- the "recipients" of the outermost encrypted data doesn't
include me
  ea-aa-unknownencryptionalgorithm, -- either kexalg or contentencryptionalgorithm
  ea-aa-decryptionfailed, -- works for ECIES-HMAC and AES-CCM
  -- TODO: to be continued...
  invalidaa, -- the AA certificate presented is invalid/revoked/whatever
  invalidaasignature, -- the AA certificate presented can't validate the request
signature
  wrongea, -- the encrypted signature doesn't designate me as the EA
  unknownits, -- can't retrieve the EC/ITS in my DB
  invalidsignature, -- signature verification of the request by the EC fails
  invalidencryptionkey, -- signature is good, but the key is bad
  deniedpermissions, -- permissions not granted
  deniedtoomanycerts, -- parallel limit
  ... }
```

```
InnerAtRequest ::= SEQUENCE {
  publicKeys                    PublicKeys,
  hmacKey                       OCTET STRING (SIZE(32)),
  sharedAtRequest               SharedAtRequest,
  ecSignature                   EcSignature,
  ...
  }

SharedAtRequest ::= SEQUENCE {
  eaId                          HashedId8,
  keyTag                        OCTET STRING (SIZE(16)),
  certificateFormat             CertificateFormat,
  requestedSubjectAttributes    CertificateSubjectAttributes (WITH
COMPONENTS{certIssuePermissions ABSENT}),
  ...
  }

InnerAtResponse ::= SEQUENCE {
  requestHash                   OCTET STRING (SIZE(16)),
  responseCode                  AuthorizationResponseCode,
  certificate                   EtsiTs103097Certificate OPTIONAL,
  ...
 }
  (WITH COMPONENTS { responseCode (ok), certificate PRESENT }
  | WITH COMPONENTS { responseCode (ALL EXCEPT ok), certificate ABSENT }
  )

END
```

A.2.5.3 AuthorizationValidation

```
EtsiTs102941TypesAuthorizationValidation
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authValidation(7) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Certificate
FROM EtsiTs103097Module
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0)}

CertificateFormat, CertificateSubjectAttributes,EcSignature, HashedId8, PublicKeys,
Version
FROM EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1) }

SharedAtRequest
FROM EtsiTs102941TypesAuthorization
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
authorization(5) version1(1)}

;

/************
-- AuthorizationValidationRequest/Response
************/

AuthorizationValidationResponseCode ::= ENUMERATED {
  ok(0),
  cantparse, -- valid for any structure
  badcontenttype, -- not encrypted, not signed, not permissionsverificationrequest
  imnottherecipient, -- the "recipients" of the outermost encrypted data doesn't include
me
  unknownencryptionalgorithm, -- either kexalg or contentencryptionalgorithm
  decryptionfailed, -- works for ECIES-HMAC and AES-CCM
```

```
  invalidaa, -- the AA certificate presented is invalid/revoked/whatever
  invalidaasignature, -- the AA certificate presented can't validate the request
signature
  wrongea, -- the encrypted signature doesn't designate me as the EA
  unknownits, -- can't retrieve the EC/ITS in my DB
  invalidsignature, -- signature verification of the request by the EC fails
  invalidencryptionkey, -- signature is good, but the responseEncryptionKey is bad
  deniedpermissions, -- requested permissions not granted
  deniedtoomanycerts, -- parallel limit
  deniedrequest, -- any other reason?
  ... }

AuthorizationValidationRequest ::= SEQUENCE {
  sharedAtRequest               SharedAtRequest,
  ecSignature                   EcSignature,
  ...
  }

AuthorizationValidationResponse ::= SEQUENCE {
  requestHash                   OCTET STRING (SIZE(16)),
  responseCode                  AuthorizationValidationResponseCode,
  confirmedSubjectAttributes    CertificateSubjectAttributes (WITH
COMPONENTS{certIssuePermissions ABSENT}) OPTIONAL,
  ...
  }

END
```

A.2.6 Offline message structures

```
EtsiTs102941TypesCaManagement
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
caManagement(8) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Certificate, EtsiTs103097Data-Signed
FROM
EtsiTs103097Module
{itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0)}

PublicKeys, CertificateSubjectAttributes
FROM EtsiTs102941BaseTypes
{ itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1)}

;

/************
-- CA certificate request
************/

CaCertificateRequest ::= SEQUENCE {
  publicKeys                  PublicKeys,
  requestedSubjectAttributes  CertificateSubjectAttributes,
  ...
}

END
```

A.2.7 Trust lists data types

```
EtsiTs102941TrustLists
```

```
  { itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941)
trustLists(6) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS

EtsiTs103097Certificate, EtsiTs103097Data-SignedAndEncrypted, EtsiTs103097Data-Signed
FROM
EtsiTs103097Module
{itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(103097)
securedMessageV1(0)}

HashedId8, Time32, Version --, CertificateAuthorityConstraints
FROM EtsiTs102941BaseTypes
{itu-t(0) identified-organization(4) etsi(0) itsDomain(5) wg5(5) ts(102941) baseTypes(3)
version1(1)}

;

/************
-- CRL
************/
ToBeSignedCrl ::= SEQUENCE {
  version    Version,
  thisUpdate Time32,
  nextUpdate Time32,
  entries SEQUENCE OF CrlEntry,
  ...
}

CrlEntry ::= HashedId8

/************
-- TLM CTL
************/

ToBeSignedTlmCtl ::= CtlFormat (FullCtl | DeltaCtl) (WITH COMPONENTS {...,
  ctlCommands ( WITH COMPONENT(
    ( WITH COMPONENTS {...,
      add ( WITH COMPONENTS {...,
        ea ABSENT,
        aa ABSENT
      })
    })
  ))
})

/************
-- RCA CTL
************/

ToBeSignedRcaCtl ::= CtlFormat (FullCtl | DeltaCtl) ( WITH COMPONENTS {...,
  ctlCommands ( WITH COMPONENT(
    ( WITH COMPONENTS {...,
      add ( WITH COMPONENTS {...,
        rca ABSENT,
        tlm ABSENT
      })
    })
  ))
})

/************
-- CTL
************/

FullCtl::= CtlFormat ( WITH COMPONENTS {...,
  isFullCtl ( TRUE ),
```

```
  ctlCommands ( WITH COMPONENT(
    ( WITH COMPONENTS {...,
      delete ABSENT
    })
  ))
})

DeltaCtl::= CtlFormat (WITH COMPONENTS {...,
  isFullCtl(FALSE)
})


CtlFormat ::= SEQUENCE {
  version     Version,
  nextUpdate  Time32,
  isFullCtl   BOOLEAN,
  ctlSequence INTEGER (0..255),
  ctlCommands SEQUENCE OF CtlCommand,
  ...
}

CtlCommand ::= CHOICE {
  add     CtlEntry,
  delete  CtlDelete,
  ...
}

CtlEntry ::= CHOICE {
  rca   RootCaEntry,
  ea    EaEntry,
  aa    AaEntry,
  dc    DcEntry,
  tlm   TlmEntry,
  ...
}

CtlDelete ::= CHOICE {
  cert  HashedId8,
  dc    DcDelete,
  ...
}

TlmEntry::= SEQUENCE {
  selfSignedTLMCertificate EtsiTs103097Certificate,
  linkTLMCertificate       EtsiTs103097Certificate OPTIONAL,
  accessPoint              Url
}

RootCaEntry ::= SEQUENCE {
  selfsignedRootCa      EtsiTs103097Certificate,
  linkRootCaCertificate EtsiTs103097Certificate OPTIONAL
}

EaEntry ::= SEQUENCE {
  eaCertificate     EtsiTs103097Certificate,
  aaAccessPoint     Url,
  itsAccessPoint    Url OPTIONAL
}

AaEntry ::= SEQUENCE {
  aaCertificate EtsiTs103097Certificate,
  accessPoint Url
}

DcEntry ::= SEQUENCE {
  url   Url,
  cert  SEQUENCE OF HashedId8
}
```

```
DcDelete ::= Url

Url::= IA5String

END
```

Annex B (normative): Service specific parameters (SSPs) definition

B.1 Overview

Service permissions are indicated by a pair of identifiers within a certificate, the ITS-AID and the SSP. The ITS-Application Identifier (ITS-AID) as given in ETSI TS 102 965 [19] indicates the overall type of permissions being granted.

The Service Specific Permissions (SSP) is a field that indicates specific sets of permissions within the overall permissions indicated by the ITS-AID. The originating ITS-S shall provide SSP information in its certificate for all generated signed messages.

The SSP information for Security Management messages is constructed in a common way, containing 1 or more octets depending of the actual service. For each octet, the most significant bit (MSB) shall be the leftmost bit. The transmission order shall always be the MSB first.

The first octet shall control the SSP version and be interpreted in the following way:

0: No version, length 1 octet; the value shall only be used for testing purposes.

1: First version, SSP contains information as defined in the present document.

2 to 255: Reserved for future usage.

B.2 CTL SSPs definition

The interpretation of the CTL SSP octet scheme is defined as depicted in Figure B.1.

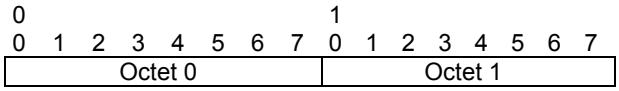


Figure B.1: The CTL SSP schema

The SSP for the CTL service shall be of 2 octets length, corresponding to the scheme defined in Table B.1.

Table B.1: Octet Scheme for CTL SSPs

Octet #	Description
0	SSP version control
1	Service-specific parameters

The service specific parameter bits shall be as defined in Table B.2.

Table B.2: CTL service-specific permissions

Bit position	Permission	Bit Value
0 (80h)	The certificate can be used to sign CTL containing the TLM entries	0: certificate not allowed to sign 1: certificate allowed to sign
1 (40h)	The certificate can be used to sign CTL containing the Root CA entries	0: certificate not allowed to sign 1: certificate allowed to sign
2 (20h)	The certificate can be used to sign CTL containing the EA entries	0: certificate not allowed to sign 1: certificate allowed to sign
3 (10h)	The certificate can be used to sign CTL containing the AA entries	0: certificate not allowed to sign 1: certificate allowed to sign
4 (08h)	The certificate can be used to sign CTL containing the DC entries	0: certificate not allowed to sign 1: certificate allowed to sign
5 to 7	unused	

According to clauses 6.3.1 and 6.3.2 of the present document, only combinations of SSPs defined in the Table B.3 shall be allowed.

Table B.3: Allowed combinations of CTL SSPs

CTL type	Allowed CTL entries	Value
TLM CTL (ECTL)	• TLM certificate entries; • Root CA entries; • DC entry (for CPOC access point).	C8h
RootCA CTL	• EA entries; • AA entries; • DC access point entries.	38h

B.3 CRL SSPs definition

The interpretation of the CRL SSP octet scheme is defined as depicted in Figure B.2.

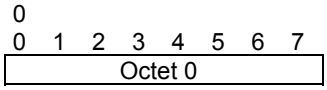


Figure B.2: The CRL SSP schema

The SSP for the CTL service shall be of 1 octet length, containing the SSP version control only.

B.4 Certificate request messages SSPs definition

The interpretation of the Certificate Request SSP octet scheme is defined as depicted in Figure B.3.

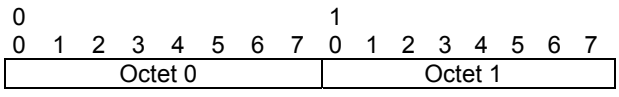


Figure B.3: The Security Management SSP schema

The SSP for the Security Management service shall be of 2 octets length, corresponding to the scheme defined in Table B.4.

Table B.4: Octet Scheme for Security Management SSPs

Octet #	Description
0	SSP version control
1	Service-specific parameters

The service specific parameter shall be as defined in Table B.5.

Table B.5: Security Management service-specific permissions

Bit position	Permission	Bit Value
0 (80h)	The certificate can be used to sign the Enrolment Request messages	0: certificate not allowed to sign 1: certificate allowed to sign
1 (40h)	The certificate can be used to sign the Authorization Request messages	0: certificate not allowed to sign 1: certificate allowed to sign
2 (20h)	The certificate can be used to sign the AuthorizationValidation Request messages	0: certificate not allowed to sign 1: certificate allowed to sign
3 (10h)	The certificate can be used to sign the Authorization Response messages	0: certificate not allowed to sign 1: certificate allowed to sign
4 (08h)	The certificate can be used to sign the Authorization Validation Response messages	0: certificate not allowed to sign 1: certificate allowed to sign
5 (04h)	The certificate can be used to sign the Enrolment Response messages	0: certificate not allowed to sign 1: certificate allowed to sign
6 (02h)	The certificate can be used to sign the CA Certificate Request messages	0: certificate not allowed to sign 1: certificate allowed to sign
7 (01h)	Unused	

B.5 Security Management certificate permissions

The overall allowance of certificate permissions for Security Management purpose is defined in Table B.6.

Table B.6: SM_PDU certificate permissions

ITS Entity	Certificate	CTL SSP					CRL	Security Management SSP						
		TLM entry (bit 0)	RootCA entry (bit 1)	EA entry (bit 2)	AA entry (bit 3)	DC entry (bit 4)		Enr Req (bit 0)	Auth Req (bit 1)	Auth Valid Req (bit 2)	Auth Resp (bit 3)	Auth Valid Resp (bit 4)	Enr Resp (bit 5)	CA Cert Req (bit 6)
TLM	TLM	A	A	-	-	A	-	-	-	-	-	-	-	-
RootCA	Root	-	-	A	A	A	A	I	I	I	I	I	I	I
EA	EA	-	-	-	-	-	-	I	I	-	-	A	A	A
AA	AA	-	-	-	-	-	-	-	-	A	A	-	-	A
ITS-S	EC	-	-	-	-	-	-	A	A	-	-	-	-	-
	AT	-	-	-	-	-	-	-	-	-	-	-	-	-
A Certificate may contain correspondent application permission. I Certificate may contain correspondent certificate issuing permission. - Certificate shall not contain correspondent permission.														

Annex C (informative): Communication profiles for security credential provisioning services (EC request, AT request)

This annex follows the CVRIA approach ([i.14]) for the specification of the layered sets of communications protocols that are required to support communications between the ITS-S and other ITS stations and PKI authorities.

Figure C.1, Figure C.2, Figure C.3 and Figure C.4 specify the communication protocols of the requesting ITS-S for different communication media (cellular networks or ITS G5 access network).

Figure C.1 specifies the communication protocols that should be used between the ITS-S and the PKI authorities when a cellular network connection is used. It shows the mapping to ITS-S Communication layers such as specified in ETSI EN 302 665 [2]. This figure specifies the reference points S_2 and S_3 in the model of Figure 10.

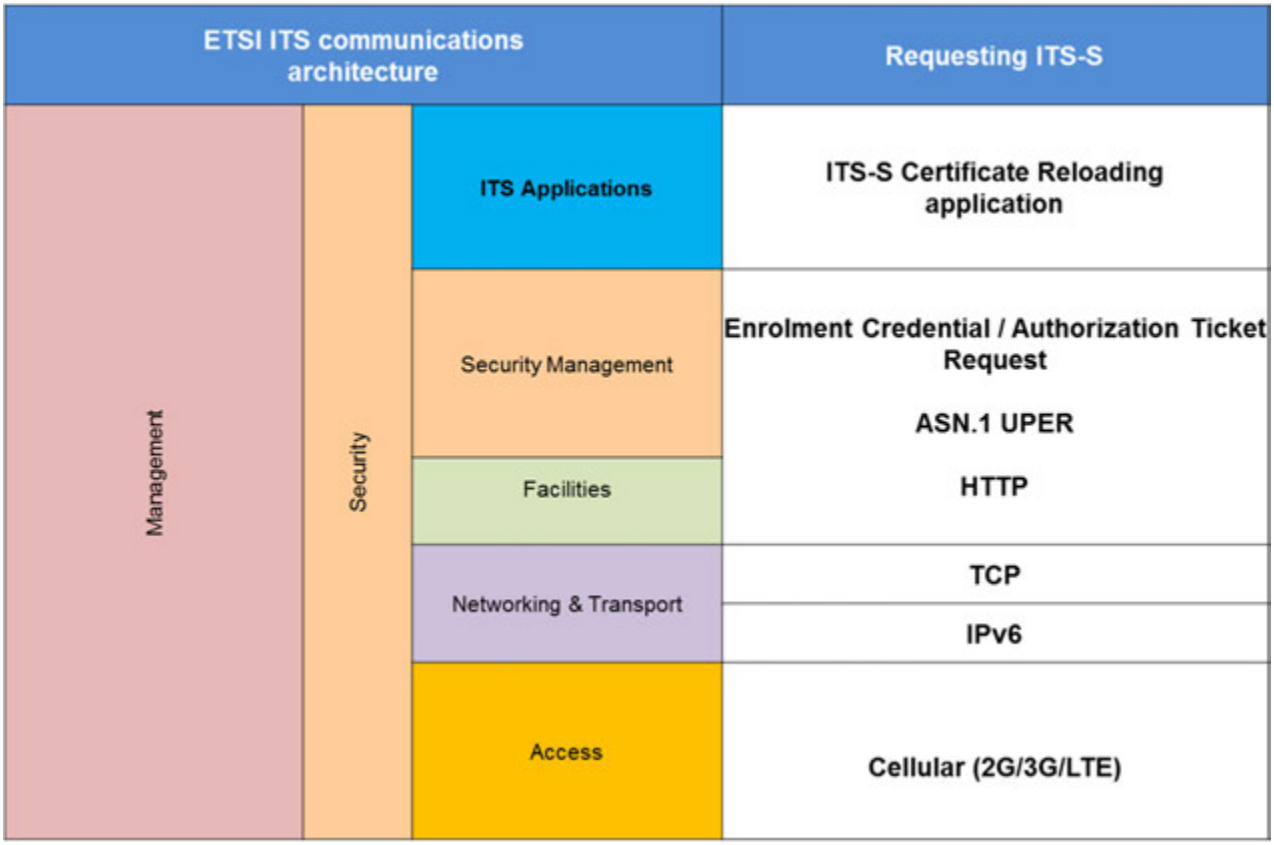


Figure C.1: Communication protocols between Requesting ITS-S and PKI authorities using cellular networks

Figure C.2, Figure C.3 and Figure C.4 depict the possible communication stacks of the requesting ITS-S when communication is based on ITS G5 media. These figures specify alternative communication profiles available on ITS G5 and the required security services that should be applied for message authentication. These figures specify the reference points S_1 between the requesting ITS-S and the Relaying ITS-S or the ITS-S Roadside Gateway shown in Figure 11.

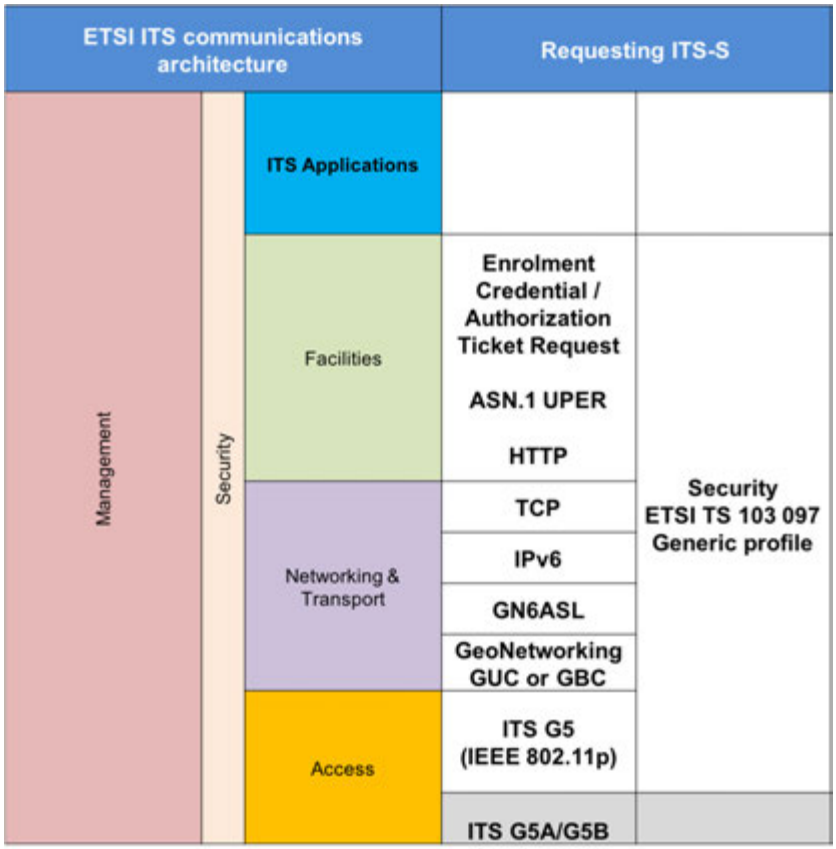


Figure C.2: Communication protocols of the Requesting ITS-S Vehicle and ITS-S Roadside Gateway using Ipv6 over GeoNetworking

NOTE 1: With respect to Figure C.2, the protocol layers include GN6ASL as specified in ETSI EN 302 636-6-1 [16] and the Ipv6 (and mobility extensions as defined in [i.15]).

NOTE 2: Security management services may use any service channel (SCH) available in the ITS G5A, except the CCH channel reserved for safety messages or other supplementary ITS channels when available (G5B, G5D).

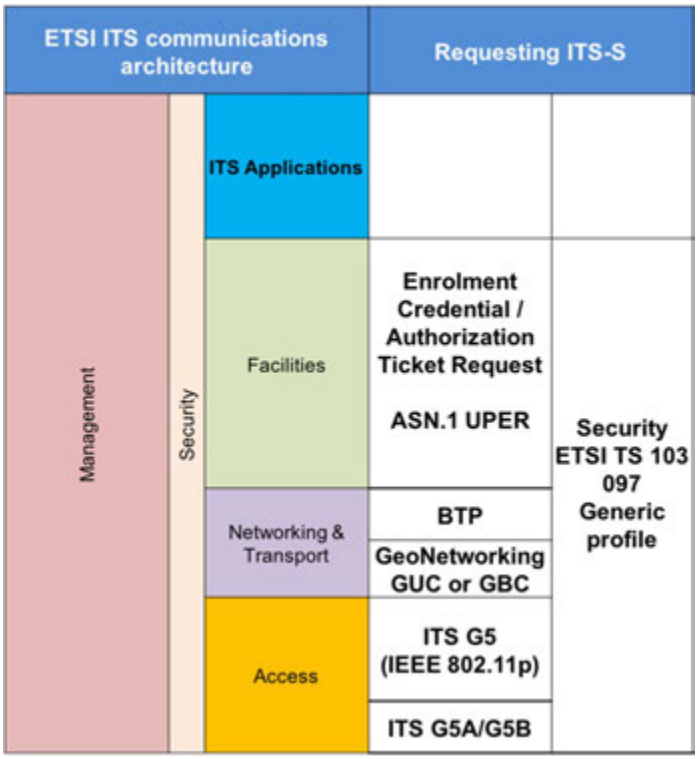


Figure C.3: Communication protocols between Requesting ITS-S Vehicle and ITS-S Roadside Gateway using GN/BTP protocol

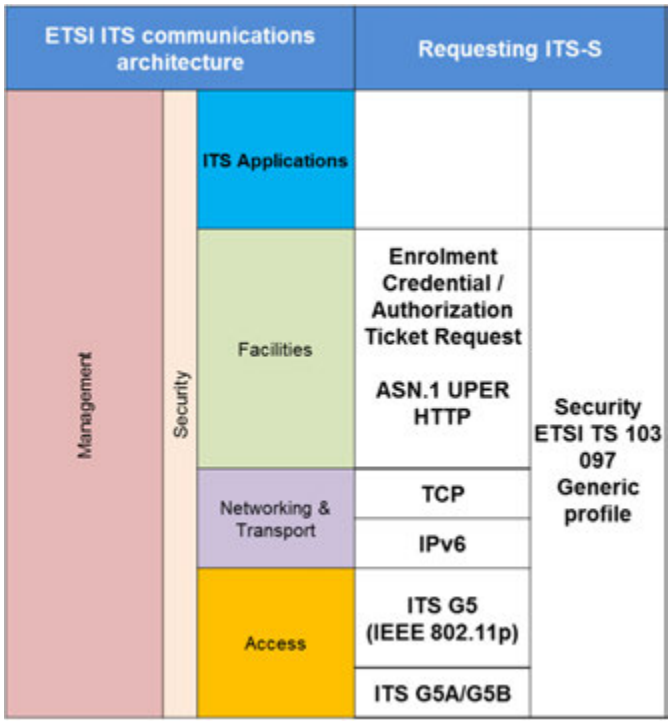


Figure C.4: Communication protocols of the Requesting ITS-S Vehicle and ITS-S Roadside Gateway using Ipv6 over G5

Figure C.5 specifies the communication path and the communication protocols used between the requesting ITS-S and the PKI when V2I communication is based on Ipv6 over GeoNetworking, for sending a service request to the CCMS, for instance requesting a EC (or AT) certificate to the EA (or AA).

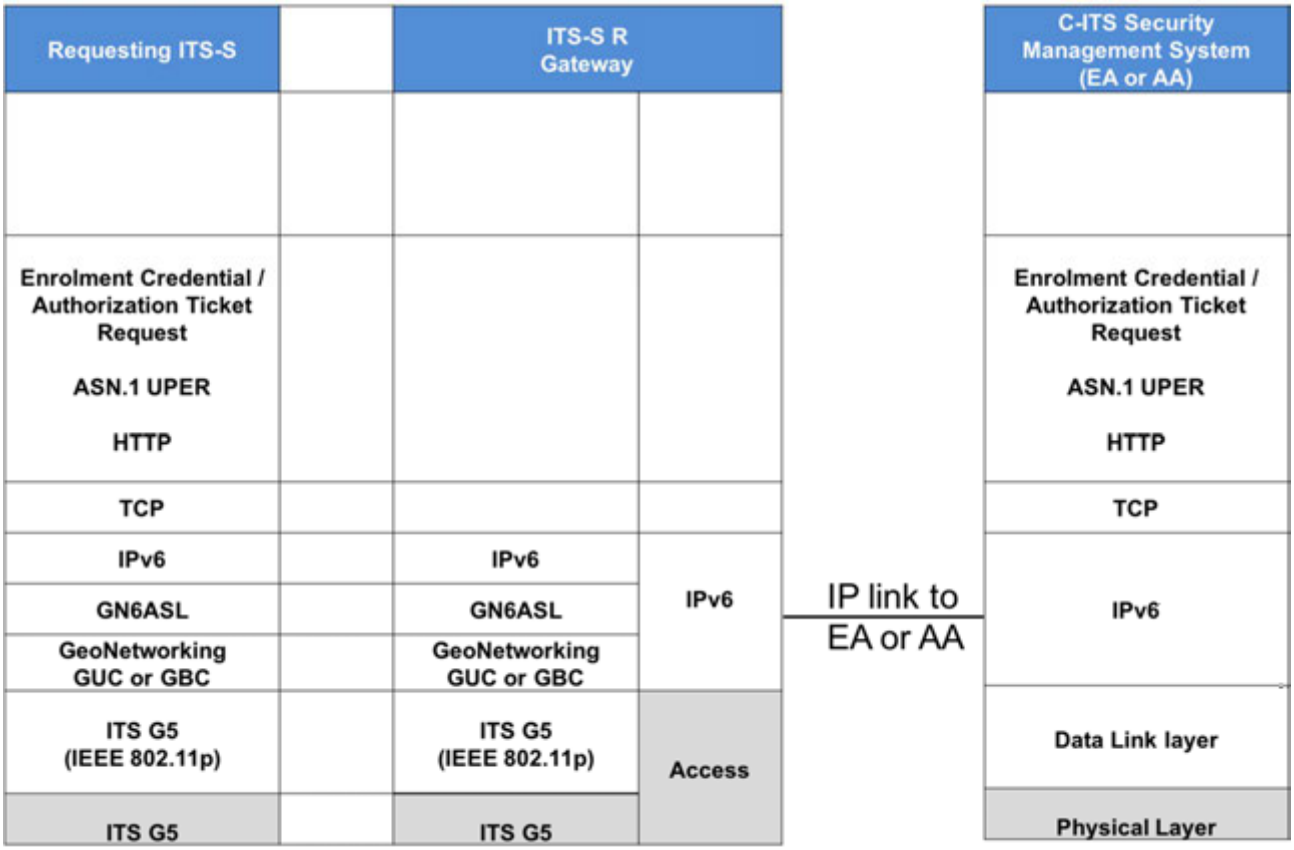


Figure C.5: Communication path between requesting ITS-S and PKI authorities

Annex D (normative): Communication profiles for CTL and CRL

D.1 CTL request and response protocol

In order to obtain the CTL from a Distribution Centre, an ITS-S may contact the DC using an HTTP GET as specified in Table D.1 using the Certificate Identifier of its Root CA (HashedID8) or the Certificate Identifier of a trusted Root CA included in the ECTL.

Table D.1: Get CTL

GET http://dc_access_point/getctl/HashedId8 The abs_path part of the HTTP request is built by taking the DC access point (from the CTL or from an ad-hoc configuration), appending "/getctl/", and the uppercase hexadecimal representation of HashedId8. Inputs: • No inputs Outputs: • Content-type: application/x-its-ctl • Content: binary encoded CTL object issued by the entity identified by HashedId8

The format of CTL is described in clause A.2.7.

D.2 CRL request and response protocol

In order to obtain the CRL from a Distribution Centre, an ITS-S may contact the DC using an HTTP GET as specified in the Table D.2 using the Certificate Identifier of its Root CA (HashedID8) or the Certificate Identifier of a trusted Root CA included in the ECTL.

Table D.2: Get CRL

GET http://dc_access_point/getcrl/HashedId8 The abs_path part of the HTTP request is built by taking the DC access point (from the CTL or from an ad-hoc configuration), appending "/getcrl/", and the uppercase hexadecimal representation of HashedId8. Inputs: • No inputs Outputs: • Content-type: application/x-its-crl • Content: binary encoded CRL object issued by the entity identified byHashedId8

The format of CRL is described in clause A.2.7.

D.3 Broadcast communication of CTL/CRL

This clause specifies the communication profile which may be used for transmitting the trust information list such as the CRL or CTL on ITS G5.

To provide a CTL/ CRL broadcasting service over G5, the RSU shall provide a proxy application which periodically transmits updated trust list information issued by the Distribution Centre. The Trust List message shall be sent in one-hop (SHB packets) using the GeoNetworking protocol specified in ETSI EN 302 636-4-1 [18].

The size of CRL/CTL messages to be transmitted should be small, using the CRL/ DeltaCTL format structures as specified in clause A.2.3. No data segmentation/reassembling is provided in the lower or upper protocol layers depicted in Figure D.1.

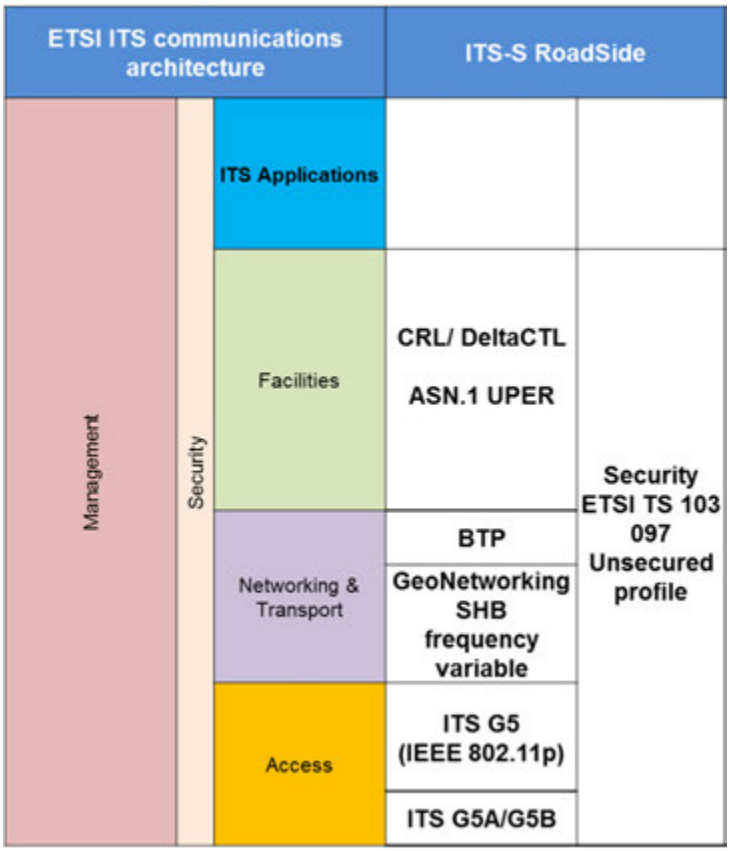


Figure D.1: Communication protocols for CRL/CTL transmission using GN/BTP protocol over G5

Annex E (informative): Encryption of a message from a sender to a receiver

This annex describes cryptographic operations to be implemented to encrypt a message using mechanisms compliant with the present document. Message encryption is used for example to encrypt EC requests or AT requests.

The message 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.

Detailed description:

Parameters

- Key Derivation Function (KDF). Here: KDF(S) = SHA256(S || counter)
- Symmetric encryption algorithm for ECIES. Here a 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.
- Message Authentication Code (MAC). Here: MAC (m, km) = HMAC (m, km)
- Elliptic curve parameters (p: curve prime, G: base point, q: base point order, O point at infinity).
- ||: concatenation.

Encryption

Input

m: 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

1) 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.
2) 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)=KDF(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.

3) 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)

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: 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)=KDF(S).

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

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

2) 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 F (informative): Bibliography

- 26th International Conference on Computer Communication and Networks (ICCCN), July 2017: "Securing PKI Requests for C-ITS Systems", JP. Monteuuis, Badis Hammi, Eduardo Sallés, Houda Labiod, Rémi Blancher, Erwan Abalea, Brigitte Lonc.
- ISE Project (SystemX): "PKI system requirements specification, v1.1", (PUBLIC); March 2015.
- ISE Project (SystemX): "PKI architecture and technical specifications, v1.1", (PUBLIC), July 2015.
- SCOOP@F technical specifications Release 2: Deliverable 2.4.4.6: "PKI architecture and technical specifications", May 2016, Available on, http://www.scoop.developpement-durable.gouv.fr/en/technical-specifications-a22.html.

Annex G (informative): Change history

Date	Version	Information about changes
June 2012	1.1.1	First version of the TS
April 2018	1.2.1	Revised to add new roles in Security architecture and align to ETSI TS 103 097 ASN.1 format

History

Document history		
V1.1.1	June 2012	Publication
V1.2.1	May 2018	Publication