ETSI TS 103 097 V1.2.1 (2015-06)

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

Reference

RTS/ITS-00531

Keywords

ITS, privacy, protocol, security

ETSI

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

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

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

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

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

Foreword

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

Modal verbs terminology

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

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

Introduction

Security mechanisms for ITS consist of a number of parts. An important part for interoperability is a common format for data elements being transferred between ITS stations for security purposes.

The present document intends to provide such a format definition. A special focus is to include as much as possible from existing standards. At the same time, the major goal is simplicity and extensibility of data structures.

1 Scope

The present document specifies security header and certificate formats for Intelligent Transport Systems. These formats are defined specifically for securing G5 communication.

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 reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

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

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

[1] IEEE™ 1363-2000: "IEEE Standard Specifications For Public Key Cryptography".

[2] NIMA Technical Report TR8350.2: "Department of Defense World Geodetic System 1984. Its Definition and Relationships with Local Geodetic Systems".

[3] ISO 3166-1: "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes".

[4] NIST SP 800-38C: "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality".

[5] IETF RFC 2246: "The TLS Protocol Version 1.0".

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

[7] ETSI TS 102 965 (V1.2.1): "Intelligent Transport Systems (ITS); Application Object Identifier (ITS-AID); Registration".

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 reference 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] IEEE™ 1363a-2004: "Standard Specifications For Public Key Cryptography - Amendment 1: Additional Techniques".

[i.2] IEEE™ 1609.2-2012 (draft D12): "Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages".

[i.3] IEEE™ 1609.2-2012 (draft D17): "Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages".

[i.4] IEEE™ 1609.3-2010: "Wireless Access in Vehicular Environments (WAVE) - Networking Services".

[i.5] Standards for Efficient Cryptography 4 (SEC 4): "Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV)".

[i.6] Antipa A., R. Gallant, and S. Vanstone: "Accelerated verification of ECDSA signatures", Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005: Springer, 2005, pp. 307-318.

3 Definitions and abbreviations

3.1 Definitions

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

enumeration: set of values with distinct meaning

3.2 Abbreviations

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

AES	Advanced Encryption Standard
CA	Certificate Authority
CAM	Cooperative Awareness Message
CRL	Certificate Revocation List
DENM	Decentralized Environmental Notification Message
DHAES	Diffie-Hellman: An Encryption Scheme
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECQV	Elliptic Curve Qu-Vanstone

NOTE: Implicit Certificate Scheme.

G5	5,9 GHz radio communications
ITS	Intelligent Transport Systems
ITS-AID	ITS Application ID
ITS-S	Intelligent Transport Systems Station
LSB	Least Significant Bit
NIMA	National Imagery and Mapping Agency
NIST SP	National Institute of Standards and Technology, Special Publication
PSID	Provider Service Identifier

NOTE: It is a synonym for ITS-AID.

SSP	Service Specific Permissions
TAI	Temps Atomique International (International Atomic Time)
TLS	Transport Layer Security
UTC	Universal Time Coordinated
WGS	World Geodetic System

4 Basic format elements

4.1 Presentation Language

The presentation language is derived from the Internet Engineering Task Force (IETF) RFC 2246 (TLS) [5] and from IEEE 1609.2-2012 [i.2] (draft D12) and is described in table 1. The encoding of multi-byte elements of the presentation language shall always use network byte order, i.e. big endian byte order, if applicable.

NOTE: The presentation language is not formally defined. Parsing tools based on this notation cannot be guaranteed to be consistent or complete.

Table 1: Presentation language

Element	Description	Example(s)
Variable names	Variable names are given in lower case	variable_name
Basic data types	Basic data types are given in lower case	uint8, uint16, uint32, uint64
Composed data types	Composed data types are given with at least the first letter in upper case	MyDataType
Comments	Comments start with the "//" indicator	// This is a comment
Numbers	Numbers are given as signed or unsigned big-endian octets	uint8, uint16, uint32, uint64, sint32

Fixed-length vectors	Fixed-length vectors have a data type and a fixed octet size given in square brackets	uint8 Coordinates[2];
		// two uint8 values
		uint32 Coordinates[8];
		// two uint32 values

Variable-length vectors with fixed-length length encoding	The number in angle brackets gives the maximum number of octets. Depending on the maximum size, the first 1 byte, 2 bytes, 4 bytes or 8 bytes encode the actual field length	uint8 AsciiChar; AsciiChar Name<2^8-1>; // "abc" encoded as // 0x03, 0x61, 0x62, 0x63 AsciiChar LongName<2^16-1>; // "abc" encoded as // 0x00, 0x03, 0x61, 0x62, 0x63

Variable-length vectors with variable-length length encoding	<var> indicates variable-length encoding. The length itself is encoded with a number of "1" bits according to the additional number of octets used to encode the length, followed by a "0" bit and the actual length value. The maximum length shall be 2^{56} - 1, i.e. at most seven "1" bits followed by a "0" bit shall be used for the variable-length length encoding. The length of variable-length vectors with variable-length length encoding shall be encoded as positive integer using the minimum number of bits necessary	uint8 AsciiChar;
		AsciiChar Name<var>;
		
		// encoding examples: (the bits with
		// grey background represent the
		// length encoding of the vector's
		// length, X the first of the //
		vector's following payload bits)
		
		// Vector length 5:
		// Bits: 00000101 XXXXXXXX XXXXXXXX
		
		// Vector length 123:
		// Bits: 01111011 XXXXXXXX XXXXXXXX
		
		// Vector length 388:
		// Bits: 10000001 10000100 XXXXXXXX

Opaque fields	Opaque fields are blocks of data whose content interpretation is not further specified	opaque fieldname[n]; opaque fieldname<n>; opaque fieldname<var>;

Enumerations	Enumerations are list of labels with a unique value for each label, and optionally a maximum value (which then determines length of encoding)	enum {de(0), fr(1), it(2)} Country;
		enum {de(0), fr(1), it(2), (2^8-1)}
		Country;
		// both variants encoding in one
		// octet
		enum {de(0), fr(1), it(2), (2^16-1)}
		Country;
		// Encoding in two octets

Constructed types	Constructed types contain other types	struct {
		    Name name;
		    Country country;
		} Person;

Case statements	Case statements are used inside constructed types to change the contents of the constructed type depending on the value of the variable given in brackets	struct {
		    Name name;
		    Country country;
		    select(country) {
		    case de:
		        uint8 age;
		    case fr:
		        AsciiChar given_name<2^8-1>;
		    }
		} Person;

External data	This is external data that has impact on a struct, e.g. in a select statement. It shall be described from where the external data is obtained	struct {
		    Name name;
		    extern Country country;
		    select(country) {
		    case de:
		        uint8 age;
		    case fr:
		        AsciiChar given_name<2^8-1>;
		    }
		} Person;

4.2 Specification of basic format elements

4.2.1 IntX

```
int_x    IntX;
```

This data type encodes an integer of variable length. The length of this integer is encoded by a number of 1 bits followed by a 0 bit, where the number of 1 bits is equal to the number of additional octets used to encode the integer besides those used (partially) to encode the length. The encoding of the length shall use at most 7 bits set to 1.

EXAMPLE: 00001010 encodes the integer 10, while 10001000 10001000 encodes the integer 2 184. The bits encoding the length of the element are coloured with a grey background.

NOTE: This definition is similar to the definition of PSID in IEEE 1609.3-2010 [i.4], clause 8.1.3, but allows bigger values of the encoded integer.

4.2.2 PublicKeyAlgorithm

```
enum {
    ecdsa_nistp256_with_sha256(0),
    ecies_nistp256(1),
    reserved(240..255),
    (2^8-1)
} PublicKeyAlgorithm;
```

This enumeration lists supported algorithms based on public key cryptography. Values in the range of 240 to 255 shall not be used as they are reserved for internal testing purposes.

NOTE: This definition is similar to the one in IEEE 1609.2 Draft D12 [i.2], clause 6.2.16, but `ecdsa_nistp224_with_sha224` is not supported by the present document. As a consequence, the numbering of identical elements (e.g. `ecies_nistp256`) differs.

4.2.3 SymmetricAlgorithm

```
enum {
    aes_128_ccm (0),
    reserved (240..255),
    (2^8-1)
} SymmetricAlgorithm;
```

This enumeration lists supported algorithms based on symmetric key cryptography. Values in the range of 240 to 255 shall not be used as they are reserved for internal testing purposes. The algorithm `aes_128_ccm` denotes the symmetric key cryptography algorithm AES-CCM as specified in NIST SP 800-38C [4].

NOTE: Except naming, this definition is identical to the one in IEEE 1609.2 Draft D12 [i.2], clause 6.2.23.

4.2.4 PublicKey

```
struct {
    PublicKeyAlgorithm  algorithm;
    select(algorithm) {
    case ecdsa_nistp256_with_sha256:
        EccPoint        public_key;
    case ecies_nistp256:
        SymmetricAlgorithm  supported_symm_alg;
        EccPoint        public_key;
    unknown:
        opaque          other_key<var>;
    }
} PublicKey;
```

This structure defines a wrapper for public keys by specifying the used algorithm and - depending on the value of `algorithm` - the necessary data fields:

- ecdsa_nistp256_with_sha256: the specific details regarding ECC contained in an EccPoint structure shall be given. The `EccPoint` used in a `PublicKey` shall not have `EccPointType` `x_coordinate_only`.

- ecies_nistp256: the specific details regarding ECC contained in an EccPoint structure and the symmetric key algorithm contained in a SymmetricAlgorithm structure shall be given. The `EccPoint` used in a `PublicKey` shall not have `EccPointType` `x_coordinate_only`.
- unknown: in all other cases, a variable-length vector containing opaque data shall be given.

NOTE: Except naming of included types, this definition is identical to the one in IEEE 1609.2 Draft D12 [i.2], clause 6.3.31.

4.2.5 EccPoint

```
struct {
    extern PublicKeyAlgorithm   algorithm;
    extern uint8                field_size;
    EccPointType                type;
    opaque                      x[field_size];
    select(type) {
    case x_coordinate_only:
    case compressed_lsb_y_0:
    case compressed_lsb_y_1:
                                ;
    case uncompressed:
        opaque                  y[field_size];
    unknown:
        opaque                  data<var>;
    }
} EccPoint;
```

This structure defines a public key based on elliptic curve cryptography according to IEEE 1363-2000 [1], clause 5.5.6. An `EccPoint` encodes a coordinate on a two dimensional elliptic curve. The x coordinate of this point shall be encoded in `x` as an unsigned integer. Depending on the key type, the `y` coordinate shall be encoded case-specific:

- `x_coordinate_only`: only the x coordinate is encoded, no additional data shall be given.
- `compressed_lsb_y_0`: the point is compressed and y's least significant bit is zero, no additional data shall be given.
- `compressed_lsb_y_1`: the point is compressed and y's least significant bit is one, no additional data shall be given.
- `uncompressed`: the y coordinate is encoded in the field `y` as an unsigned integer. The y coordinate contained in a vector of length `field_size` containing `opaque` data shall be given.
- `unknown`: in all other cases, a variable-length vector containing `opaque` data shall be given.

The `uint8` `field_size` defining the lengths of the vectors containing the raw keys shall be derived from the given algorithm and the mapping as defined in table 2. The necessary algorithm shall be given as an external link to the parameter `pk_encryption` specified in the structure `RecipientInfo`.

Table 2: Derivation of field sizes depending on the used algorithm

PublicKeyAlgorithm value	Length in octets
ecdsa_nistp256_with_sha256	32
ecies_nistp256	32

NOTE: Except inclusion of all remaining elements of the enumeration EccPointType that previously matched to case `uncompressed` and inclusion of case `unknown`, this definition is identical to the EccPublicKey in IEEE 1609.2 Draft D12 [i.2], clause 6.2.18.

4.2.6 EccPointType

```
enum {
    x_coordinate_only(0),
    compressed_lsb_y_0(2),
    compressed_lsb_y_1(3),
    uncompressed(4),
    (2^8-1)
} EccPointType;
```

This enumeration lists supported ECC point types.

NOTE: This definition is identical to the one in IEEE 1609.2 Draft D12 [i.2], clause 6.2.19.

4.2.7 EncryptionParameters

```
struct {
    SymmetricAlgorithm  symm_algorithm;
    select(symm_algorithm) {
    case aes_128_ccm:
        opaque          nonce[12];
    unknown:
        opaque          params<var>;
    }
} EncryptionParameters;
```

This structure holds basic parameters and additional data required for encryption and decryption of data using different symmetric encryption algorithms. In case of `aes_128_ccm` a 12 octet nonce shall be given. The parameter `Tlen` according to NIST SP 800-38C [4] shall be set to `Tlen` = 128 (bits) and no associated data shall be given. In other cases the data shall be given as a variable-length vector containing `opaque` data. It is out of scope of this definition how resulting ciphertexts are transported. Typically, a ciphertext should be put into a `Payload` data structure marked as `encrypted` using the `PayloadType`.

NOTE: This structure is not available in IEEE 1609.2 Draft D12 [i.2].

4.2.8 Signature

```
struct {
    PublicKeyAlgorithm  algorithm;
    select(algorithm) {
    case ecdsa_nistp256_with_sha256:
        EcdsaSignature  ecdsa_signature;
    unknown:
        opaque          signature<var>;
    }
} Signature;
```

This structure defines a container that encapsulates signatures based on public key cryptography. Depending on the value of `algorithm`, different data structures define the algorithm-specific details:

- `ecdsa_nistp256_with_sha256`: the signature contained in an `EcdsaSignature` structure shall be given.
- `unknown`: in all other cases, a variable-length vector containing the signature as `opaque` data shall be given.

The data in this structure can be used to verify a data structure's integrity. In conjunction with a matching `SignerInfo` structure, the data structure's authenticity can also be verified.

It is necessary to note the following points:

- Clause 5.6 defines which parts of a `SecuredMessage` data structure are covered by a signature.
- The length of the `security_field<var>` variable length vector in the `SecuredMessage` containing the `Signature` field shall be calculated before creating the signature using the length of the signature.