

Security Management Message Formats

Project: Cooperative ITS Corridor | Version: 0.14 | State: Draft

Pilot PKI
Security Management Message Formats

Document Meta Data

Project	Cooperative ITS Corridor
Document Name	SecurityManagementFormats_v121.docm
Document Status	Draft
Document Class	Internal

Version	Author	Date	Change (Why, What)
0.1	ESCRYPT	2016-02-04	Initial version
0.2	ESCRYPT	2016-02-23	Update according to TS 103 097 v1.2.1
0.3	ESCRYPT	2016-04-13	Trust list added
0.4	ESCRYPT	2016-04-27	ITS-S registration added, Trust list removed
0.5	ESCRYPT	2016-05-17	Authorization of AT requests added
0.6	ESCRYPT	2016-06-04	Renaming of the terms PCA, LTCA, PC, and LTC by AA, EA, AT, and EC
0.7	ESCRYPT	2016-06-20	Updated ApprovedAuthorizationTicket, Response expected for received EC, SmmSignerInfo field of AtRequest does not have to be encrypted
0.8	ESCRYPT	2016-07-08	Added SmmType itss_registration_update_request.
0.9	ESCRYPT	2016-07-21	Inner signature for proof of private key possession added to EcRequest and AtRequest. New SmmType ec_update_request specified.
0.10	ESCRYPT	2016-08-24	Support of optional encryption key removed
0.11	ESCRYPT	2016-08-30	AtRequest extended by SubjectInfo field, CAConfigurationRequest and CAConfigurationResponse exchanged by EAConfigurationRequest and EAConfigurationResponse
0.12	ESCRYPT	2016-09-28	protocol_version incremented to 3
0.13	ESCRYPT	2017-01-02	Cr1Request, Cr1 and ToBeSignedCr1 fields adapted according to specification in TS 102 941 v1.1.12
0.14	ESCRYPT	2017-02-02	Removed SubjectInfo from AtRequest

Contents

Message Formats	3
Format Specification	4
2.1 SecurityManagementMessage	5
2.2 SmmType	7
2.3 SmmSignerInfo	8
2.4 SmmSignerInfoType	9
2.5 AtRequest	10
2.6 AtResponse	12
2.7 EcRequest	13
2.8 EcResponse	15
2.9 RequestErrorCode	16
2.10 RequestError	17
2.11 ResponseAcknowledgement	18
2.12 CaCertificateRequest	19
2.13 CaCertificateResponse	20
2.14 CrlRequest	21
2.15 Crl	22
2.16 ToBeSignedCrl	23
2.17 EAConfigurationRequest	24
2.18 EAConfigurationResponse	25
2.19 ItsStationRegistrationRequest	26
2.20 AuthorizationValidationRequest	27
2.21 ApprovedAuthorizationTicket	28
Bibliography	29

Message Formats

This document defines formats of Security Management Messages (SMM) for the Pilot PKI of the Cooperative ITS Corridor. These message formats are used in the communication between an ITS station and the Certificate Authorities. In this document, message formats for the following processes are defined:

- Request and response of CA certificates and certificates that are issued by a CAs as specified in section 2.12 and 2.13.
- Request, response and response acknowledgement of enrollment credentials as specified in section 2.7 and 2.80.
- Request, response and response acknowledgement of authorization tickets as specified in section 2.5 and 2.6.
- Request and response of CRLs as specified in section 2.14 and 2.15.
- Request and response of ITS station registration at the EA as specified in section 2.19 and 2.21.
- Authorization of authorization tickets certificates by the EA for requests at the AA.

End-to-end encryption and integrity protection is out of the scope of this document. For the Pilot PKI, the preferred solution is to use web services over https with server authentication using TLS. However, the format also allows for encapsulation of an SMM into a SecuredMessage as defined in ETSI TS 103 097 [1].

The SMM presentation language is equal to the presentation language specified in ETSI TS 103 097 [1].

Format Specification

2.1 **SecurityManagementMessage**

```
struct {
    uint8                                protocol_version;
    SmmType                              type;
    select(type) {
        case request_error:
            RequestError                error_message;
        case ec_request:
            EcRequest                    ec_request;
        case ec_response:
            EcResponse                   ec_response;
        case at_request:
            AtRequest                    at_request;
        case at_response:
            AtResponse                   at_response;
        case response_ack:
            ResponseAcknowledgement      response_ack;
        case crl_request:
            CrlRequest                   crl_request;
        case crl_response:
            Crl                          crl;
        case ca_cert_request:
            CaCertificateRequest          ca_cert_request;
        case ca_cert_response:
            CaCertificateResponse         ca_cert_response<var>;
        case ea_configuration_request:
            EAConfigurationRequest        config_request;
        case ea_configuration_response:
            EAConfigurationResponse       config_response;
        case itss_registration_request:
            ItsStationRegistrationRequest itss_registration_request;
        case authorization_validation_request:
            AuthorizationValidationRequest authorization_validation_request;
        case approved_authorization_ticket:
            ApprovedAuthorizationTicket   approved_authorization_ticket<var>;
        case itss_registration_update_request:
            ItsStationRegistrationRequest itss_registration_update_request;
        case ec_update_request:
            EcRequest                     ec_update_request;
    }
} SecurityManagementMessage;
```

Security Management Message Formats

Project: Cooperative ITS Corridor | Version: 0.14 | State: Draft

The `protocol_version` specifies this protocol's version and shall be set to 3 for conformance with the present document.

2.2 SmmType

```
enum {  
    request_error(0),  
    ec_request(1),  
    ec_response(2),  
    at_request(3),  
    at_response(4),  
    response_ack(5),  
    crl_request(6),  
    crl_response(7),  
    ca_cert_request(8),  
    ca_cert_response(9),  
    ea_configuration_request(10),  
    ea_configuration_response(11),  
    itss_registration_request(12),  
    authorization_validation_request(13),  
    approved_authorization_ticket(14),  
    itss_registration_update_request(15),  
    ec_update_request(16),  
    reserved(240..250),  
    (2^8-1)  
} SmmType;
```

2.3 SmmSignerInfo

```
struct {
    SmmSignerInfoType type;
    select(type){
        case self:
            ;
        case certificate_digest_with_sha256:
            HashedId8          digest;
        case certificate:
            Certificate          certificate;
        case certificate_chain:
            Certificate          certificates<var>;
        case certificate_digest_with_other_algorithm:
            PublicKeyAlgorithm    algorithm;
            HashedId8            digest;
        case encrypted:
            EncryptionParameters  enc_params;
            RecipientInfo         recipients<var>;
            opaque                enc_data<var>;
        case module_id:
        unknown:
            opaque                info<var>;
    }
} SmmSignerInfo;
```

The following fields are defined in ETSI TS 103 097 [1]:

- HashedId8
- Certificate
- PublicKeyAlgorithm
- RecipientInfo
- EncryptionParameters

In the case of encrypted signer information (type = encrypted), enc_data shall contain a serialized structure.

The remaining structure elements are following the descriptions of the SignerInfo in ETSI TS 103 097 [1].

2.4 **SmmSignerInfoType**

```
enum {  
    self(0),  
    certificate_digest_with_sha256(1),  
    certificate(2),  
    certificate_chain(3),  
    certificate_digest_with_other_algorithm(4),  
    encrypted (6),  
    module_id (7),  
    reserved(240..255),  
    (2^8-1)  
} SmmSignerInfoType;
```

2.5 **AtRequest**

```
struct {  
    uint8                                certificate_protocol_version;  
    SmmSignerInfo                        signerInfo;  
    PublicKey                            verification_keys<var>;  
    SubjectAttribute                      subject_attributes<var>;  
    ValidityRestriction                  validity_restrictions<var>;  
    Signature                            proof_of_possession_of_verification_key<var>;  
    Signature                            signature;  
} AtRequest;
```

The `certificate_protocol_version` shall contain the desired certificate version. The `subject_verification_keys` field may contain multiple verification public key. Encryption keys are currently not used by the infrastructure side.

The `signerInfo` contains information about the request signer. Unless a certificate policy or profile explicitly allows other types of `SmmSignerInfo`, it shall have the `SmmSignerInfoType` “certificate (2)” In the case of the `SmmSignerInfoType` “certificate (2)” the ITS-S’s certificate is contained.

The following fields are defined in TS 103 097 [1]:

- `PublicKey`
- `SubjectAttribute`
- `ValidityRestriction`
- `Signature`

Unless explicitly stated in a certificate policy or profile, the `subject_attribute` list shall neither contain public keys nor a reconstruction value. It must contain an assurance level and any combination of `its_aid_list` and `its_aid_ssp_list` as long as every `its_aid` is contained at most in one of the lists. All given `subject_attributes` shall be in the order as defined in ETSI TS 103 097 [1]. The `ValidityRestrictions` shall be used as in ETSI TS 103 097 [1].

There are two different kind of signatures, one to prove the possession of the verification keys and one overall signature.

The `proof_of_possession_of_verification_key` field must contain an inner signatures for each of the public keys contained in the `verification_keys` field using the corresponding private keys. The signature is generated over the encoding of the `protocol_version` of the `SecurityManagementMessage` and all preceding fields, from the `certificate_protocol_version` to the `validity_restrictions`.

The overall signature shall be calculated over the encoding of the `protocol_version` of the `SecurityManagementMessage` and all preceding fields of the `AtRequest`, including the

Security Management Message Formats

Project: Cooperative ITS Corridor | Version: 0.14 | State: Draft

proof_of_possession_of_verification_key field and all encoded lengths. The private key corresponding to the EC shall be used for this signature.

2.6 **AtResponse**

```
struct {  
    Certificate                issued_certificates<var>;  
    Certificate                certificate_chain<var>;  
} AtResponse;
```

The following fields are defined in ETSI TS 103 097 [1]:

- Certificate

The field `certificate_chain` shall contain the certificate of the issuing AA and optionally the certificate of the RCA that has issued the AA certificate.

The reception of the `AtResponse` must be acknowledged with a `ResponseAcknowledgement`.

2.7 EcRequest

```
struct {  
    uint8                certificate_protocol_version;  
    SmmSignerInfo         signerInfo;  
    SubjectInfo           subject_info;  
    SubjectAttribute       subject_attributes<var>;  
    ValidityRestriction    validity_restrictions<var>;  
    Signature              proof_of_possession_of_verification_key;  
    Signature              signature;  
} EcRequest;
```

The signerInfo contains information about the request signer.

The following fields are defined in ETSI TS 103 097 [1]:

- SubjectInfo
- PublicKey
- SubjectAttribute
- ValidityRestriction

The subject_attributes shall contain one verification public key. Encryption keys are currently not used by the infrastructure side.

The EcRequest shall be embedded in a SecurityManagementMessage. If the ITS-station is not in possession of a valid EC

- the SmmType shall be set to ec_request,
- the request shall be signed using the ITS-station's private device key, and
- the signerInfo must be set to module_id.

In case the ITS-station has a valid EC,

- the SmmType shall be set to ec_update_request,
- the request shall be signed with the private key of the currently valid EC, and
- the signerInfo should be set to certificate_digest_with_sha256.

The subject_name of the SubjectInfo must be set to module_id. The SubjectType must be set to enrollment_credential.

The list of subject attributes may contain an assurance level and any combination of its_aid_list and its_aid_ssp_list as long as every its_aid is contained at most in one of the lists.

To proof possession of the verification key that shall be included in the requested Enrollment Credential, one signature for the requested key has to be provided in the proof_of_possession_of_verification_key field. The proof_of_possession_of_verification_key

Security Management Message Formats

Project: Cooperative ITS Corridor | Version: 0.14 | State: Draft

signature is generated over the encoding of the `protocol_version` of the `SecurityManagementMessage` and all preceding fields of the `EcRequest`, from the `certificate_protocol_version` to the `validity_restrictions`.

The overall signature shall be calculated over the encoding of the `protocol_version` of the `SecurityManagementMessage` and all preceding fields of the `EcRequest`, from the `certificate_protocol_version` to the `proof_of_possession_of_verification_key`, including all encoded lengths.

2.8 **EcResponse**

```
struct {  
    Certificate                issued_certificate;  
    Certificate                certificate_chain<var>;  
} EcResponse;
```

The following fields are defined in ETSI TS 103 097 [1]:

- Certificate

The field `certificate_chain` shall contain the certificate of the issuing EA and optionally the certificate of the RCA that has issued the EA certificate. The last element of the chain is the EA certificate that issued the EC and the first element of the chain is the root certificate that issued the EA certificate. The chain may contain optionally be empty or contain only the EA certificate.

The reception of the `EcResponse` must be acknowledged with a `ResponseAcknowledgement` in order for the EC to be activated. Otherwise the request of authorization tickets with the EC may fail.

2.9 RequestErrorCode

```
enum {  
    verification_failure(0),  
    csr_cert_expired(1),  
    csr_cert_revoked(2),  
    csr_cert_unauthorized(3),  
    request_denied(4),  
    csr_cert_unknown(5),  
    canonical_identity_unknown(6),  
    ca_not_available(7),  
    message_processing_error(8),  
    request_in_process(9),  
    message_parsing_error(10),  
    ca_configuration_request_error(11),  
    reserved(240..255),  
    (2^8-1)  
} RequestErrorCode;
```


2.10 **RequestError**

```
struct {  
    opaque                                request_hash[10];  
    RequestErrorCode                      reason;  
} RequestError;
```

The request_hash is the first 10 bytes of the SHA-256 hash of the EcRequest or AtRequest.

2.11 **ResponseAcknowledgement**

```
struct {  
    opaque request_hash[10];  
} ResponseAcknowledgement;
```

The `request_hash` is the first 10 bytes of the SHA-256 hash of the `EcRequest`, `AtRequest`, `ItsStationRegistrationRequest`.

2.12 **CaCertificateRequest**

```
struct {  
    HashedId8                                requested_certificates<var>;  
} CaCertificateRequest;
```

The following fields are defined in ETSI TS 103 097 [1]:

- HashedId8

This request allows ITS-Ss to obtain the current CA certificates provided by the PKI. A `CaCertificateRequest` can be directed to any CA. If `requested_certificates` is an empty list, the contacted CA shall contain its own default certificate. The definition of the default certificate is out of scope of this document.

2.13 **CaCertificateResponse**

```
struct {  
    Certificate                requested_certificates<var>;  
    Crl                       crls<var>;  
} CaCertificateResponse;
```

The following fields are defined in ETSI TS 103 097 [1]:

- Certificate

If the responding CA is not a Root CA, the `cr1_path` shall be empty and the list `requested_certificates` shall contain the requested certificates in the same order as they appear in the `CaCertificateRequest`. The `cr1_path` shall contain at most one CRL.

If the responding CA is a Root CA, a `CaCertificateResponse` shall contain one `cr1_path` consisting of the most recent CRL which contains the signer's certificate and optionally the certificate of the CA that issued the CRL signer's certificate. The list of certificates shall contain all certificates that could appear on the CRL given in `cr1_path`.

NOTE: A response to a CA certificate request may be a list containing several `CaCertificateResponse.entries`.

2.14 **CrlRequest**

```
struct {  
    HashedId8 issuer;  
} CrlRequest;
```

The following field is defined in ETSI TS 103 097 [1]:

- HashedId8

A CRL is always signed by a CRL signer. It only contains revoked certificates that were issued by the same Root CA that issued the CRL signer. The fields in this structure have the following meaning:

- A CRL is always issued on behalf of a Root CA. Thus, the field `issuer` represents the RCA on whose behalf the requested CRL is being issued.

The CA shall respond to this request by providing the most recent full CRL.

2.15 Crl

```
struct {  
    ToBeSignedCrl          unsigned_crl;  
    Signature              signature;  
} Crl;
```

The following field is defined in ETSI TS 103 097 [1]:

- Signature

The signature shall be calculated over the encoding of all preceding fields including all encoded lengths.

2.16 **ToBeSignedCrl**

```
struct {  
    uint8            version;  
    SignerInfo       signer;  
    Time32           issue_date;  
    Time32           next_crl;  
    HashedId8        entries<var>;  
} ToBeSignedCrl;
```

The following fields are defined in ETSI TS 103 097 [1]:

- SignerInfo
- Time32

HashedId8The fields in this structure have the following meaning:

- version specifies the format version of the data structure. Here, it is equal to the protocol version and shall be set to 3.
- signer identifies the signer of the CRL who provided the signature in the Crl data structure.
- issue_date specify the time when the CRL has been issued. The CRL shall include all certificates that were revoked between the issue date of the previous CRL and issue_date.
- next_crl contains the time when the next CRL is expected to be issued.
- entries contains identifiers for each revoked certificate.

2.17 **EAConfigurationRequest**

```
struct {  
    HashedId8                                ca<var>;  
} EAConfigurationRequest;
```

The `EAConfigurationRequest` allows an ITS-S to request the configuration from an EA in order to prepare subsequent authorization ticket requests. This configuration aims to reduce the number of invalid authorization ticket requests.

The following fields are defined in ETSI TS 103 097 [1]:

- `HashedId8`

The ITS-S must specify the `HashedId8` of the EA certificate. If the given `HashedId8` is not known a `RequestError` is returned.

2.18 **EAConfigurationResponse**

```
struct {  
    SignerInfo          signerInfo;  
    uint16              parallel_authorization_ticket_number;  
    uint16              authorization_ticket_lifetime_period;  
    uint16              minimum_authorization_ticket_lifetime;  
    uint16              authorization_ticket_preliminary_request_time;  
    uint32              certificate_time_slot_starting_time;  
    Signature           signature;  
} EAConfigurationResponse;
```

The response contains the authorization ticket configuration of the EA.

The following fields are defined in TS 103 097 [1]:

- SignerInfo
- Signature

The `parallel_authorization_ticket_number` is the maximum number of authorization tickets that can be issued for an ITS station for one and the same point of time. While the `authorization_ticket_lifetime_period` is the maximal lifetime of an authorization ticket in number of days, the `minimum_authorization_ticket_lifetime` is the minimum lifetime of an authorization ticket. The `authorization_ticket_preliminary_request_time` is the maximum number of days allowed between the time when the AT request is sent and the validity start time of the requested certificate. The PKI is based on time slots having a fix starting time, the `certificate_time_slot_starting_time`, encoded in big-endian format, giving the number of International Atomic Time (TAI) seconds since 00:00:00 UTC, 01 January 2004.

The signature shall be calculated over the encoding of the `protocol_version` of the `SecurityManagementMessage` and all preceding fields of the `EAConfigurationResponse`, including all encoded lengths.

2.19 **ItsStationRegistrationRequest**

```
struct {
    opaque                module_id[16];
    PublicKey             device_public_key;
    SubjectAttribute      subject_attributes<var>;
    ValidityRestriction   validity_restrictions<var>;
} ItsStationRegistrationRequest;
```

An OEM or IRS operator can register new ITS-S at an EA. This is required for an ITS-S to request an EC.

The following fields are defined in TS 103 097 [1]:

- PublicKey
- SubjectAttribute
- ValidityRestriction

A public key for the registered device must be provided which is used to verify the signature of an EcRequest. The subject_attribute list may contain an assurance level and any combination of its_aid_list and its_aid_ssp_list as long as every its_aid is contained at most in one of the lists. All given subject_attributes shall be in the order as defined in ETSI TS 103 097 [1]. The validity_restrictions shall be used as in ETSI TS 103 097 [1]. If the subject_attribute or validity_restrictions field is not set, default values as specified by ETSI are used.

The authorization to set the subject_attribute and the validity_restrictions are determined by the client certificate used for sending the message.

The subject_attribute field and the validity_restrictions may be changed after registration by sending an ItsStationRegistrationRequest with the same moduleId and deviceIdentityEccPublicKey.

A ResponseAcknowledgement confirms the registration. The request_hash is the first 10 bytes of the SHA-256 hash of the ItsStationRegistrationRequest. A may be provided on failure.

An OEM or IRS operator can update existing ITS-S at an EA. In order to do that, the SmmType must be set to itss_registration_update_request. This message is used when a registered ITS-S's subject attributes or validity restrictions change or when the device public key changes.

The fields are the same as for the ItsStationRegistrationRequest. The module ID must have been registered before.

2.20 **AuthorizationValidationRequest**

```
struct {  
    uint8                certificate_protocol_version;  
    SmmSignerInfo        signerInfo;  
    SubjectAttribute      subject_attributes<var>;  
    ValidityRestriction  validity_restrictions<var>;  
    Signature            signature;  
    uint32               ticket_count;  
    opaque               signature_hash<var>;  
    opaque               request_hash[10];  
} AuthorizationValidationRequest;
```

This message is used for inter-CA communication between the EA and the AA.

The `certificate_protocol_version`, `signerInfo`, `subject_attributes`, `validity_restrictions` and `signature` must be the same as the parameters provided with the corresponding AT request.

The `signerInfo` shall contain the certificate of the requesting ITS station. The `signature` can be verified by the EA using the public key corresponding to the EC of the requesting ITS station.

The `ticket_count` specifies how many ATs were requested and is therefore equal to the number of verification keys in the `AtRequest`.

The `signature_hash` of the `AtRequest` is calculated by the AA and then used by the EA to verify the signature. Additionally the `request_hash` is provided to assign the `ResponseAcknowledgement` to the request so the EA can set the issue flag for the AT request.

2.21 **ApprovedAuthorizationTicket**

```
struct {  
    SubjectAttribute          subject_attributes<var>;  
    ValidityRestriction      validity_restrictions<var>;  
    uint32                   ticket_count;  
} ApprovedAuthorizationTicket;
```

This message is used for inter-CA communication between the AA and the EA.

The `subject_attributes` must contain the validated assurance level and list of permission from the request message. The `validity_restrictions` must contain a `start_validity` and `end_validity` which were determined by the EA in addition to the regional restrictions. The `ticket_count` specifies how many authorization tickets shall be issued for this period.

Note: A response to an `AuthorizationValidationRequest` may contain several `ApprovedAuthorizationTicket` entries.

Bibliography

- [1] European Telecommunications Standards Institute (ETSI), „TS 103 097 v1.2.1 Intelligent Transport Systems (ITS); Security; Security header and certificate formats,“ ETSI, 2015.