V2X standards research report

Difference in Secured protocol data units (SPDUs)

SPDUs are defined in subclause 6.3 in *IEEE std 1609.2* and subclause 6.4 in *YD/T 3957*
All data structures listed in this section are come from *YD/T 3957* page 21 section 6.4 and explanation text are translated. Unlisted data structures are identical in both standards.

HashAlgorithm

HashAlgorithm defined in *YD/T 3957* :

```
HashAlgorithm ::= ENUMERATED {
  sha256,
  ...,
  sha384,
  sm3
}
```

Compare to IEEE other Hash algorithms are added. In which SM3 is the hash algorithm defined in *GM/T 0004-2012*.

In IEEE it is marked clearly that only sha256 is used in the standard.

> "The only hash algorithm approved for use in this standard is SHA-256 as specified in the Federal
> Information Processing Standard (FIPS) 180-4. In this standard, the phrase "the SHA-256 hash of [an octet
> string]" is used to mean "the hash of [that octet string] obtained using SHA-256 as specified in FIPS
> 180-4".

HashedData

```
HashedData::= CHOICE {
  sha256HashedData      OCTET STRING (SIZE(32)),
  ...,
  sha384HashedData      OCTET STRING (SIZE(48)),
  sm3HashedData         OCTET STRING (SIZE(32))
  }
```

Again, multiple hash algorithms are supported. Note that SM3 digest has the same length as SHA256.

HeaderInfo

```
HeaderInfo ::= SEQUENCE{
  psid                    Psid
  generationTime          Time64 OPTIONAL,
  expiryTime Time64       OPTIONAL,
  generationLocation      ThreeDLocation OPTIONAL,
```

```
  p2pcdLearningRequest    HashedId3 OPTIONAL,
  missingCrlIdentifier    MissingCrlIdentifier OPTIONAL,
  encryptionKey           EncryptionKey OPTIONAL,
  ...
```

psid is called **aid** in *YD/T 3957*. It is defined as follow.

```
Aid ::= INTEGER (0..MAX)
```

And it is encoded as decimal number.

Three more data field is added in *YD/T 3957*

```
inlineP2pcdRequest        SequenceOfHashedId3 OPTIONAL,
requestedCertificate      Certificate OPTIONAL,
pduFunctionalType         PduFunctionalType OPTIONAL
```

- `inlineP2pcdRequest`, if present, is used to request an unknown certificate. This field only exist when `p2pcdLearningRequest` is not present.
- `requestedCertificate`, if present, is used to provide certificate according to P2PCD mechanism.
- `-pduFunctionalType`, if present, indicate that SPDU will be used by processes outside the application process. This is defined bellow

SymmetricEncryptionKey

It is defined as follow in *YD/T 3957*

```
SymmetricEncryptionKey ::= CHOICE {
  aes128Ccm     OCTET STRING(SIZE(16)),
  ...,
  sm4Ccm        OCTET STRING(SIZE(16))
}
```

In addition to aes128 specified in *IEEE std 1609.2*, SM4 in CCM mode is also supported.

SymmAlgorithm

```
SymmAlgorithm ::= ENUMERATED {
  aes128Ccm,
  ...,
  sm4Ccm
}
```

SM4 is added.

BasePublicEncryptionKey

```
BasePublicEncryptionKey ::= CHOICE {
  eciesNistP256             EccP256CurvePoint,
  eciesBrainpoolP256r1      EccP256CurvePoint,
  ...,
  ecencSm2                  EccP256CurvePoint
}
```

Sm2 is also used here. Which is another elliptic curve cryptography algorithm.

PduFunctionalType

```
PduFunctionalType ::= INTEGER (0..255)
tlsHandshake              PduFunctionalType ::= 1
iso21177ExtendedAuth      PduFunctionalType ::= 2
```

This data structure identifies the functional entity intended to use the SPDU. For SPDUs that contain a PduFunctionalType, it shall conform to the the security attribute corresponding to the PduFunctionalType value, not the applied SPDU security attribute of the AID. The data structure contains :

- `tlsHandshake` indicates that the signed SPDU cannot be used directly as an application PDU, but is used to provide information about the holder's access to the Transport Layer Security (TLS) handshake process, which is used to protect communication with the application process;
- `iso21177ExtendedAuth` indicates that signed SPDU cannot be used directly as an application PDU, but is used to provide additional information about holder's access to the ISO 21177 security subsystem.

SingerIdentifier

```
SignerIdentifier ::= CHOICE {
  digest          HashedId8,
  certificate     SequenceOfCertificate,
  self            NULL,
  ...,
  x509            OCTET STRING
}
```

x509 certificate is added here in addition to those specified in *IEEE std 1609.2* (The first three here). But no explanation is given in *YD/T 3957*

Signature

```
Signature ::= CHOICE {
  ecdsaNistP256Signature           EcdsaP256Signature,
  ecdsaBrainpoolP256r1Signature    EcdsaP256Signature,
  ...,
  ecdsaBrainpoolP384r1Signature    EcdsaP384Signature,
  ecdsaNistP384Signature           EcdsaP384Signature,
  sm2Signature                     EcsigP256Signature
}
```

Signatures generated by ecdsaBrainpoolP384r1, ecdsaNistP384 and sm2 is added here.

EcdsaP384Signature

```
EcdsaP384Signature ::= SEQUENCE {
  rSig      EccP384CurvePoint,
  sSig      OCTET STRING (SIZE (48))
}
```

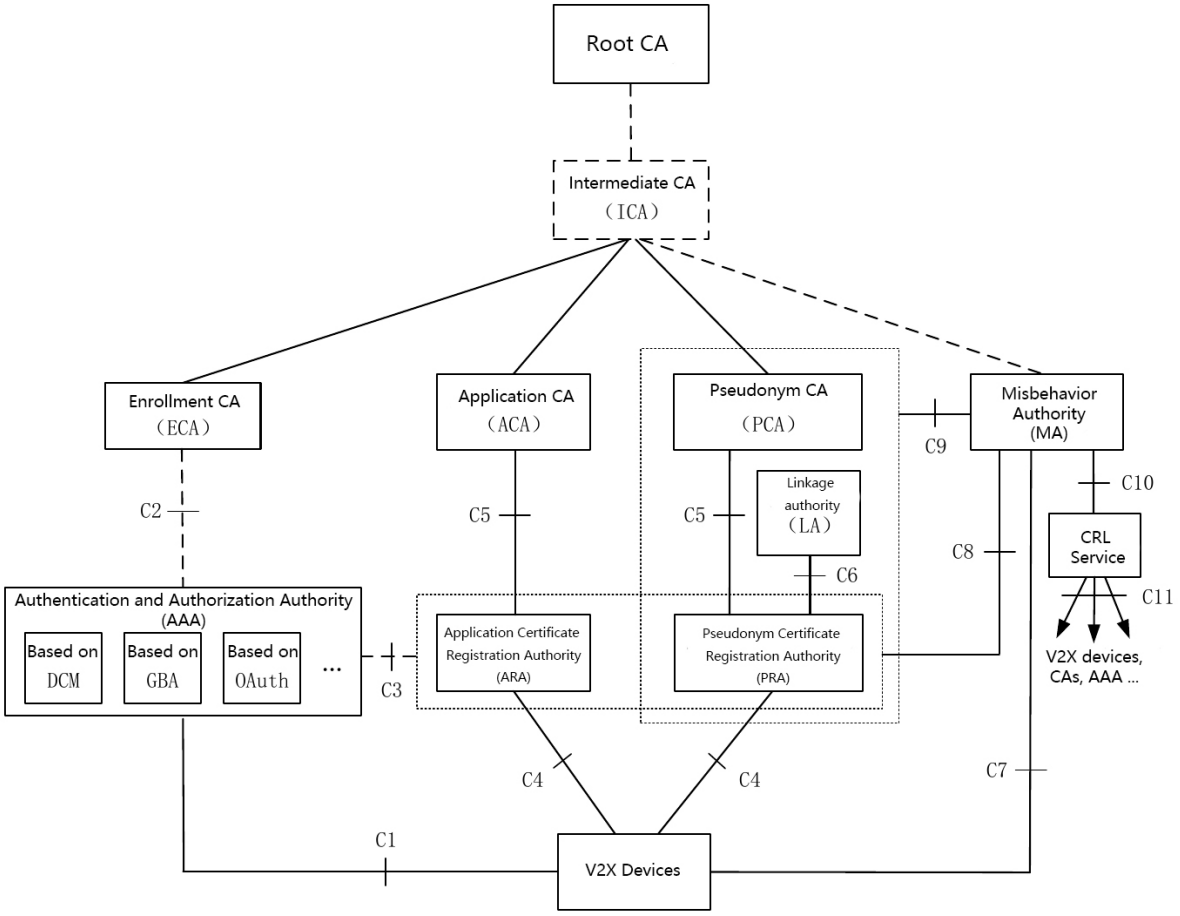
SymmetricCiphertext

```
SymmetricCiphertext ::= CHOICE {
  aes128ccm        One28BitCcmCiphertext,
  ...,
  sm4Ccm           One28BitCcmCiphertext
}
```

This data structure encapsulates a ciphertext generated with an approved symmetric algorithm. Note that sm4Ccm uses the same data structure as aes128ccm.

PKI architecture

PKI architecture specified in *YD/T 3957* is similar to the architecture in *IEEE std 1609.2*. However there are some differences in enrollment, pseudonym CA and Linkage Authority.



Note: Dotted line may or may not be present in the actual implementation.

Enrollment of V2X EE

> "An enrollment certificate is a certificate used to authenticate interactions with the SCMS. The most
> important use of enrollment certificates in this document is to authenticate a request for
> authorization certificates, which are the certificates used in application interactions."
> --- IEEE std 1609.2.1 section 4.1.3

IEEE standard specified that an EE should request a enrollment certificate from Enrollment Certificate Authority(ECA). In YD/T standard 3 different PKI architecture are specified with three ways of obtaining an enrollment certificate. They are based on DCM, GBA and OAuth.

PKI architecture based on Device Configuration Manager (DCM)

The V2X device interacts with the registration certificate authority through the DCM service system to obtain the registration certificate, and the DCM is a device trusted by the V2X device and the CA system. Therefore, DCM, V2X EE and CA system should be pre-configured with the security credentials required to establish a security connection with each other. When a V2X device applies for a registration certificate, it shall obtain from DCM the relevant information of the subsequent application for registration certificate, such as the certificates of each certificate issuing authority in PKI. When applying for V2X device registration certificate through DCM, the security of the connection between V2X device and DCM service system should be ensured. For example, it can be based on physical environment security, TLS security protocol or dedicated end-to-end secure connection to achieve the relevant operation.

PKI architecture based on Generic Bootstrapping Architecture (GBA)

Architecture based on GBA can be implemented with or without ECA. The GBA mechanism is implemented in accordance with 3GPP TS 33.220, and the GBA authentication authorization system includes the Bootstrapping Server Function (BSF) network element and Network Application Function (NAF) network element defined in the 3GPP standard, which are responsible for the authentication of OBU, RSU and other certificate application subjects, the authorization of service applications and the provision of GBA shared session keys. It is responsible for the authentication of OBU, RSU and other certificate application subjects, authorization of business applications and provision of GBA shared session keys. In order to ensure the security of sensitive parameter information such as random numbers and keys during operation and processing at the terminal side, the system prioritizes the implementation of GBA_U method.

PKI architectures based on GBA can be furtherly separated into 2 types:

1. With ECA:
   Under the system architecture with ECA, the subject of certificate application first applies for EC registration certificate through GBA certification authority system, and then applies for other application certificates (such as pseudonym certificate, application certificate). Under this architecture, the GBA authentication system needs to establish a secure communication channel with the EC registration certificate authority in advance to ensure the security of data interaction between them.
   When applying for an EC registration certificate, the GBA authentication system carries out two-way identity authentication with OBU and RSU terminals based on the user identification and root key in USIM, and provides the GBA shared session key for ECA to establish a secure connection with the terminal. With the GBA shared session key, OBU and RSU terminals can interact with ECA and proceed with other process such as EC certificate application and renewal.
   When applying for PC, AC and other application certificates, PCA, ACA and other certificate authorities access the GBA authentication system to authenticate OBU and RSU terminals and obtain authorization for the terminal's service requests. After successful authentication and authorization, the GBA certification authority system provides the GBA shared session key to the certificate authority. With the GBA shared session key, OBU and RSU terminals securely access the certificate authority and use the EC registration certificate to enable the application and update of PC, AC and other application certificates.

2. without ECA:
   Under the system architecture without ECA, the certificate applicant first accesses the GBA authentication system to obtain a service token for the service request, and then applies for

other application certificates such as PC and AC based on the service token.
When applying for a service token, the GBA authentication system carries out two-way identity authentication with OBU and RSU terminals based on the user identification and root key in USIM, and authorizes the service request of the terminal according to the policy. After successful authorization, the GBA authentication system issues a service token to the terminal and provides the GBA shared session key to the application certificate authority. With the GBA shared session key, OBU and RSU terminals securely access the certificate authority and use the service token to enable application and update of PC, AC and other application certificates.

PKI architecture based on OAuth 2.0

The PKI architecture based on OAuth 2.0 authorization server is a ECA-less system architecture. When the authentication server is an OAuth authorization server, the OBU is authorized to access the PRA service through the OAuth mechanism. This process is based on the OAuth 2.0 mechanism defined in the IETF standard "RFC 6749: OAuth 2.0 Authorization Architecture." The OBU requests authorized access to the PRA service from the OAuth authorization server using the client credentials defined in RFC 6749. The OAuth authorization server accepts or rejects the request based on strategy. If accepted, the Access Token is issued to the OBU, and the OBU requests a pseudonymous certificate from the service provider PRA using the Access Token, establishes a TLS secure channel between the OBU and PRA, and the PRA verifies the Access Token and provides access to the pseudonymous certificate after successful verification. See Appendix B for OAuth-based Token authorization mechanism.

Architecture differences on pseudonym certificate

> “To protect the privacy of system participants, the ACA may provide an end entity with several simultaneously valid authorization certificates so that the end entity may use different certificates at different times for the same activity. These certificates are known as 'pseudonym certificates.'

> “Pseudonym certificates, identification certificates, and application certificates are the only subtypes of authorization certificate identified in this document.
> --- IEEE std 1609.2.1 section 4.1.1

In *IEEE std 1609.2.1* is specified that pseudonym certificate, identification certificates and application certificates are all issued to EEs by ACA. However in *YD/T 3957* pseudonym certificate are issued by a separate Pseudonym CA and both identity certificate (for OBU) and application certificate (for RSU and other V2X devices) are issued by Application CA.

Architecture differences on linkage authority

> The Linkage Authority (LA) implements the linkage value provisioning function, including individual linkage values and group linkage values. It receives linkage value requests from PRA, verifies the validity of the requests, and generates linkage values; it accepts link seed queries from MA, verifies the validity of the requests, and returns the link seeds required by MA. Before the certificate revocation, the linkage value will not disclose the information that multiple pseudonymous certificates belong to the same certificate applicant entity. **This standard uses a single LA architecture.** linkage value supports efficient revocation, simplifies the revocation process, optimizes revocation performance, and provides trusted anti-tracking capability for OBU. To ensure the independence of the linkage value, the trustworthiness of the LA can be guaranteed based on the mechanism of multi-agency common management.
> --- YD/T 3957 section 6.1.7.8

In *YD/T 3957* it is clearly stated that only single linkage authority is implemented compare to multiple LA specified in *IEEE std 1609.2.1*.