

# V2X Certificate Management with IEEE 1609.2.1: Status and Deployment

William Whyte

# Overall goals

## Primary use cases

- Authorization certificate request and download by end entities
  - For multiple different applications
  - For pseudonym / non-pseudonym certificates
- Certificate revocation for pseudonym / non-pseudonym certificates
- Root certificate / trust management

## Support use cases

- Obtain / renew enrollment certificates (certificates used for authentication of end entities in SCMS communications)
- Distribution of sets of trusted CA certificates so they don't have to be received within application exchanges
- Misbehavior report upload
- Support multiple certificates per device for pseudonymity
  - Butterfly keys
  - Linkage values

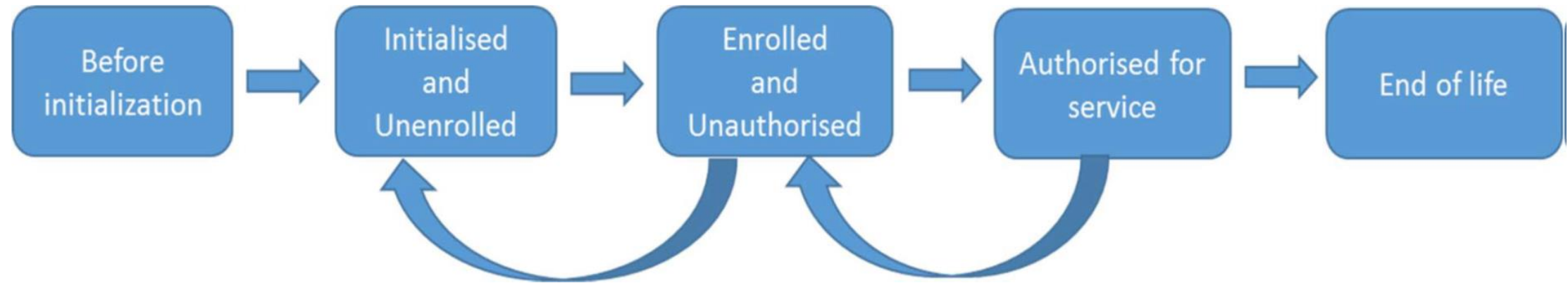
# History

- **Crash Avoidance Metrics Partnership (CAMP)** - OEM organization carrying out pre-competitive research into vehicle safety technologies with funding contributed by USDOT
  - <https://www.campllc.org/projects/past-camp-projects/>
  - Vehicle Safety Communications (2005) - recommended digital certificates
  - Vehicle Safety Communications for Applications (2006-2009) - initial SCMS concepts
  - V2V-Communications Security (2010-2013) - further SCMS concepts
  - Vehicle Safety Communications Security Studies (2012-2014) - initial SCMS design
  - SCMS Proof-of-Concept Implementation (2015-2018) - Final SCMS design used in US Connected Vehicle Pilot Deployments
    - Docs at <https://wiki.campllc.org/display/SCP> -- hosting planned to transfer to USDOT site in 2022
- **IEEE 1609.2.1**
  - Version 1: IEEE Std 1609.2.1 published January 2021
  - Version 2: Currently under revision: ballot started 2021-11-29, publication expected Q2 2022
    - Minor changes / cleanup / bug fixes
  - Adoption within ETSI to support butterfly keys - TS 102 940 v 2.1.1, 2021-10
  - Large-scale deployment will happen after publication of V2, currently deployments are still focused on the CAMP protocols
- **Internationally: ETSI 102 940/102 941, CCSA SCMS system**

# 1609.2.1 enhancements / benefits compared to CAMP

- **Root CA management**
  - Different models of role of Electors, 1609.2.1 is more clear / robust
- **Certificate permissions**
  - 1609.2.1 has less emphasis on enrollment step
  - More ability to adapt permissions in certs issued to enrolled devices
  - Allowing X.509 enrollment certificates allows for greater reuse of existing provisioning infrastructure at OEMs
- **Supplementary authorization**
  - 1609.2.1 supports use of OAuth and (in principle) other authorization mechanisms to manage access to the RA for some operations
  - Use of off-the-shelf web authorization technologies allows easier scalability for volume deployments
- **ACPC based certificate access management**
  - Optional technology enabling better robustness in the case of widespread compromise
  - Not discussed in this presentation

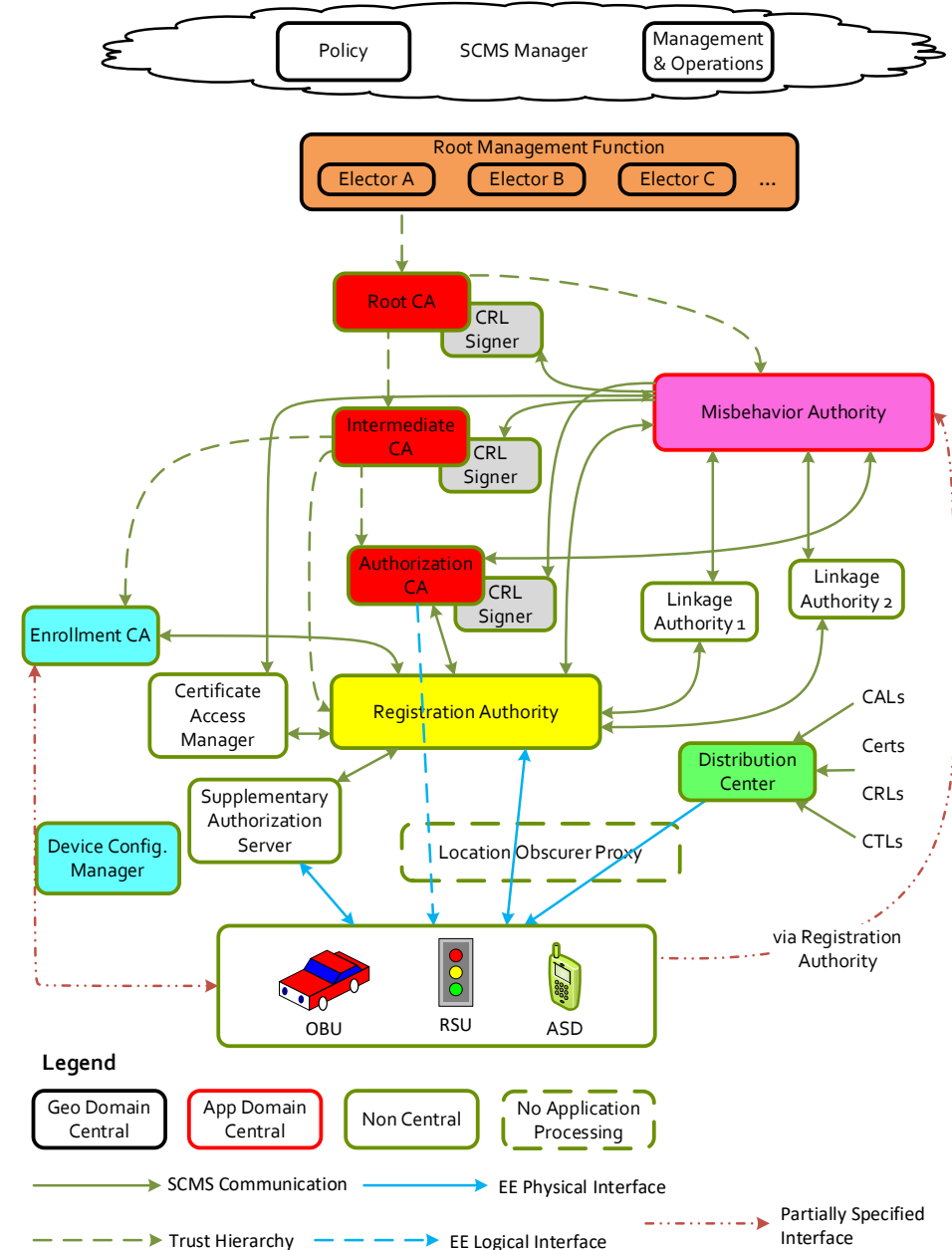
# Basic overview



- End Entity is provisioned to become initialized (non-SCMS activity)
- EE interacts with ECA to become enrolled - obtains enrollment certificate
- EE interacts with RA to become authorized - authorization cert requests are signed by enrollment certificate
- While authorized, EE interacts with RA / DC to
  - Request new certificates (RA only)
  - Update system information (RA/DC)
  - Submit misbehavior reports (RA only)
- At end of life, EE may be revoked by CRL signer

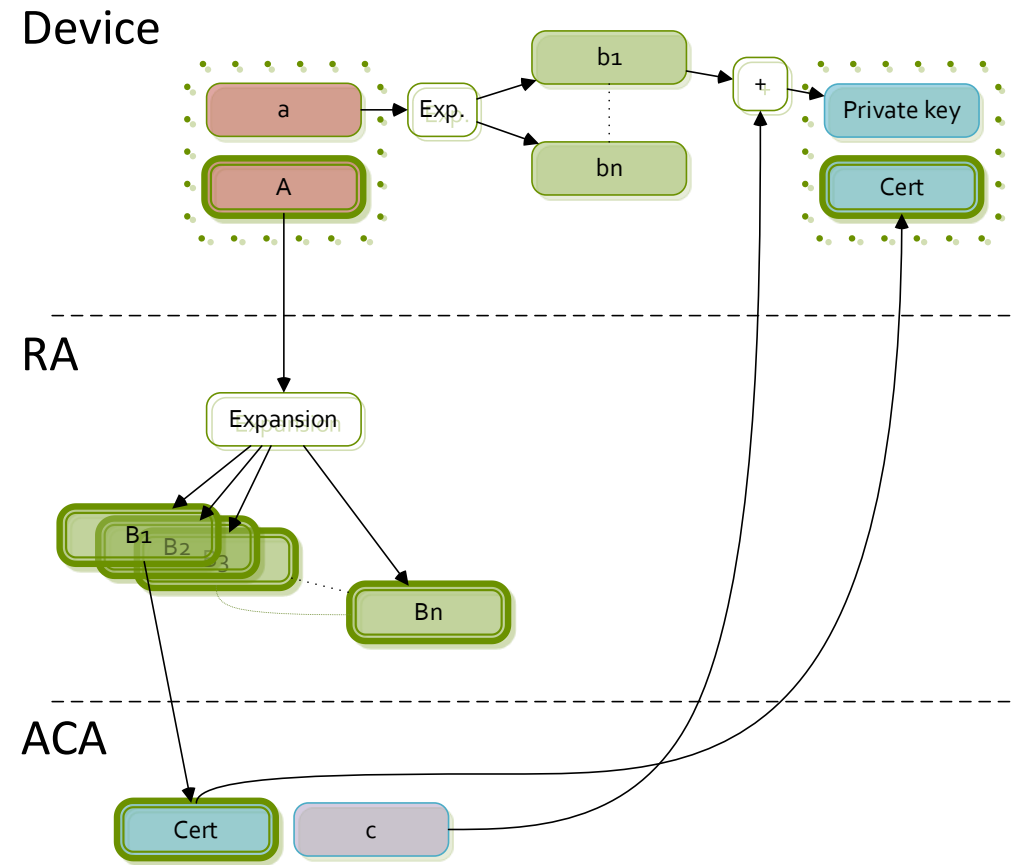
# Architecture

- **Enrollment CA / Device Configuration Manager** - manage initial issuance of enrollment certificates
- **Registration Authority** - gateway to SCMS for device in operational state:
  - Certificate request/response
  - Misbehavior report upload
  - System configuration information
- **Distribution center** - distributes public system configuration information if not available via RA
  - E.g. CRLs
- **MA** manages misbehavior reporting and reaction; **CRL signers** issue CRLs on instructions from MA
  - 1609.2 design couples each certificate to a single CRL signer which is the only one that can revoke it
- **Root / Intermediate CAs** - standard PKI approach
- **Root Management Function** - manages which root CAs are trusted by other system participants, based on SCMS Manager policy



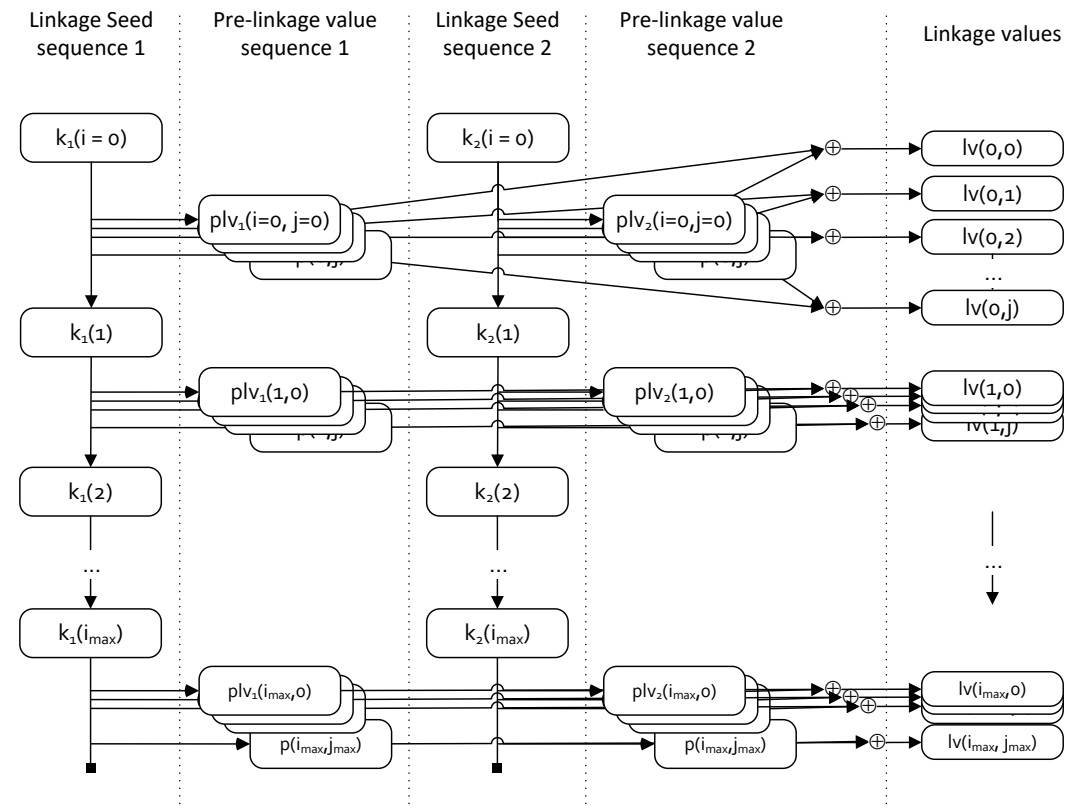
# Feature 1: butterfly keys

- Allows a single request to generate certificates for an arbitrary length of time with keys that cannot be correlated
  - Simplifies management, authentication and storage
- End entity generates “caterpillar” keypair and expansion function
- RA applies expansion function to “diversify” caterpillar public key into a number of “cocoon” keys
- ACA further diversifies with a self-chosen diversification code which it returns to the EE along with the certificate
- Properties:
  - Only the EE knows the private keys
  - Only the EE knows the complete set of certificates that belong to that EE
  - Certificate generation peak / average can be more easily managed



# Feature 2: revocation via linkage values

- With multiple certificates per device, revocation lists could be very long
- To avoid this, pseudonym certificates contain a linkage value generated from a seed via a hash tree
- Publishing the current seed on a CRL allows all future linkage values to be calculated and any cert with a revoked linkage value is considered revoked
  - One CRL entry per revoked device no matter how many certificates it has
- 1609.2.1 additionally supports having two linkage seeds to provide privacy against tracking by an insider at the CA



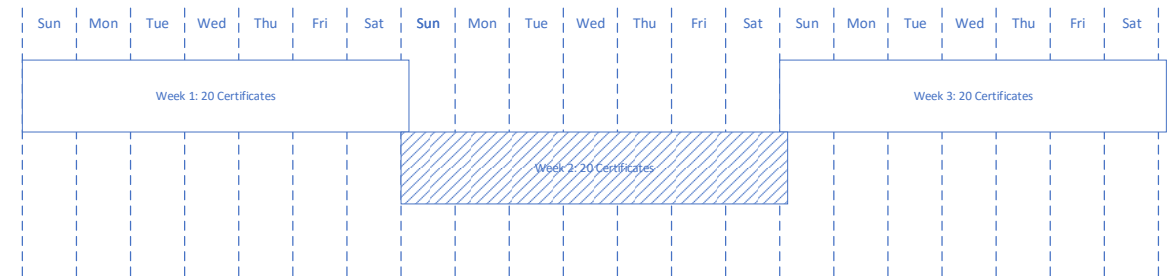


# Feature 3: Implicit certificates

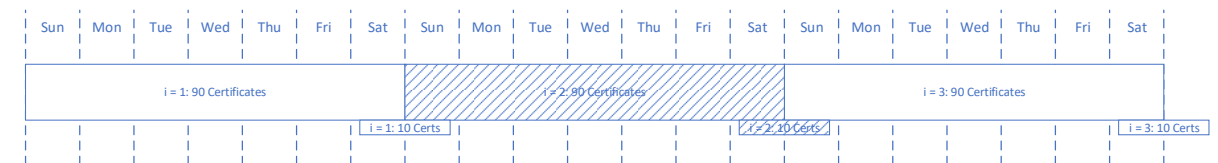
- **End-entity certificates are “implicit” (ECQV)**
  - “Implicit” certs - replace (public key + signature) with public key reconstruction value
    - The receiver uses the CA public key and the EE certificate to “reconstruct” the EE public key (see backup slides)
      - This process only gives the correct public key if the certificate was correctly issued by the indicated CA
    - The receiver then verifies the certificate and message together by checking that the public key verifies the signature
      - Slight loss of security metadata:
        - Failure could mean something about the certificate was wrong or something was wrong with the signature verification
        - There is no way to tell which, but the verifier typically would have no way to make use of this information
  - **Save 64 bytes per certificate**
  - Speed up the first verification of a certificate chain
  - IP associated with them; letter of assurance provided by Blackberry to IEEE 1609
- **CA certificates are explicit**
  - “Standard” / conventional certificates
  - Simplifies trust chain and verification
  - Larger size is not burdensome because CA certificates are rarely sent over the air

# 1609.2.1 pseudonym certificate issuance and use

- Butterfly keys allow an end entity to upload a single request and the system then generates the appropriate number of certificates for each time period
- US model:
  - 20 certificates per week
  - Rollover at 5 am Eastern on Tuesday (least traffic)
  - Duration = 1 week + 1 hour
  - SAE J2945/1 gives certificate change algorithm
    - Every five minutes unless device hasn't moved
    - No significant protection against reuse
- EU model
  - 60-100 certificates per week
    - 90 have duration exactly a week
    - 10 have duration of two hours, overlapping the transition
  - No stated rollover time
  - C-ITS security policy ([https://cpoc.jrc.ec.europa.eu/data/documents/c-its\\_security\\_policy\\_release\\_1.pdf](https://cpoc.jrc.ec.europa.eu/data/documents/c-its_security_policy_release_1.pdf)) recommends a distance-based change strategy that reduces the risk that a certificate is used at the start of two different journeys



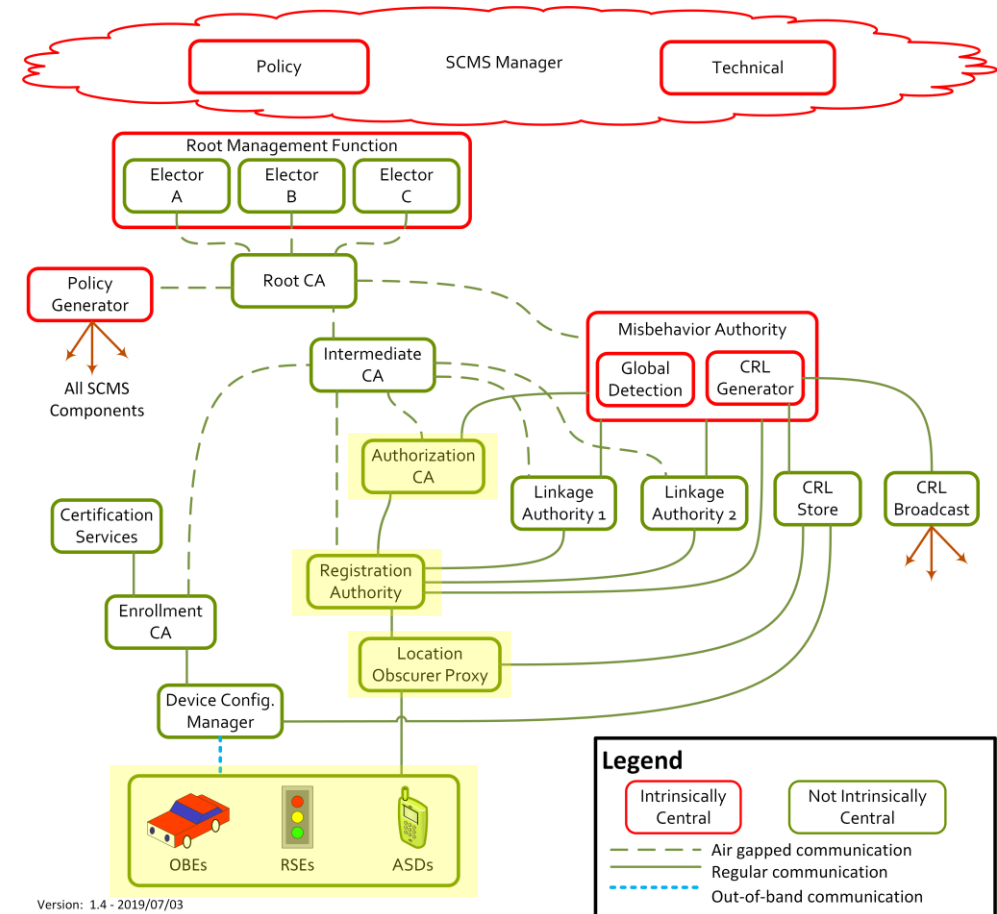
US model



EU model

# 1609.2.1 pseudonym certificate download

- **CAMP model:**
  - Request is sent through RA
  - RA and ACA cooperate to generate certificates up to three years in advance of the current time
  - RA stores the certificates for download by the EE
  - EE downloads up to three years' worth of certificates when connectivity is available
- **ETSI model:**
  - Each certificate is requested individually directly from the ACA and immediately issued
  - Devices can download up to three months of certificates in advance (no end entity revocation)
- **1609.2.1 follows the CAMP model but does not hardwire advance generation time, download time, etc**
  - US SCMS Manager working on policy
    - Individual deployments can set their own requirements
  - Three years is maybe more than necessary with modern connectivity - shorter download periods reduce CRL size

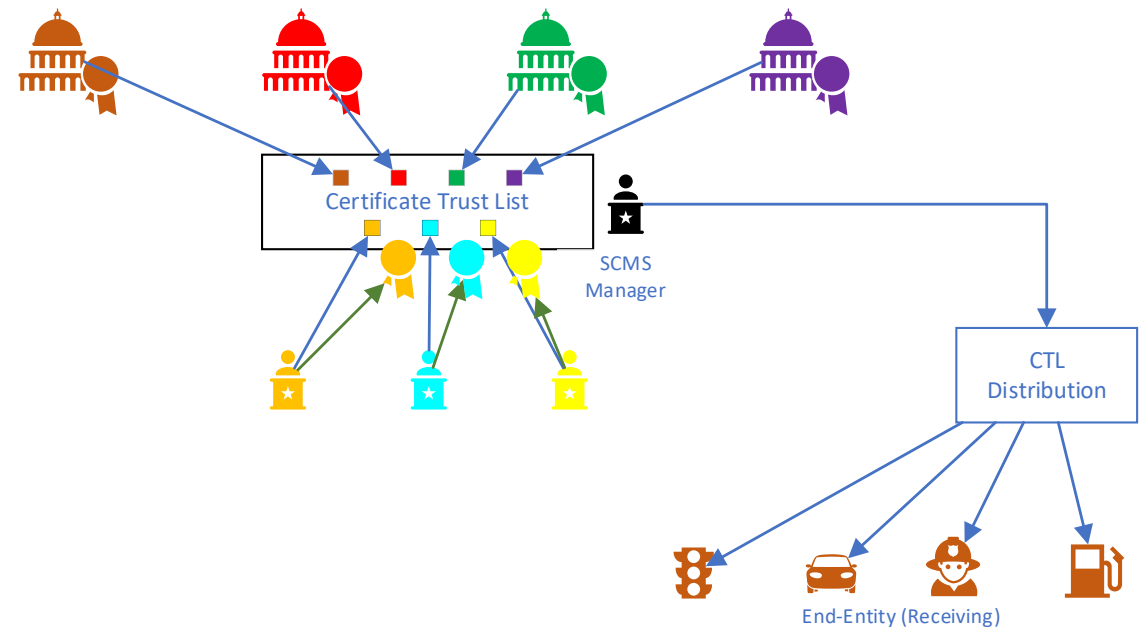


# Root CA management: problem statement

- Most certificates are trusted because they are signed by an already-trusted CA
- Root CA certificates are self signed
- There will probably be multiple root CAs in the system
  - EU model - national root CAs, OEM root CAs
  - Conclusion: design should not assume a single root CA
- How to manage adding new root CAs / expiry and rollover of root CA certificates / (hopefully rare) root CA revocation?
- Assumption: there is some policy authority / SCMS Manager that makes policies that root CAs abide by - intent is for CAs to be trusted if they abide by this policy, not trusted if they don't

# Root CA management: 1609.2.1

- See backup slides for description of ETSI and CAMP approaches
- SCMS Manager creates CTL containing list of Electors and of trusted Root CAs
- Electors are independent entities that “notarize” SCMS Manager decisions
  - Electors are required to validate that the SCMS Manager presented that CTL to them for signing but are not responsible for the decision as to what is on the CTL
- Participants trust CTL if m-of-N signatures verify
- Good robustness and transparency
  - Robust against compromise of up to (N-m) Electors
  - If one Elector cert rolls over every 2 years, a device can be turned off for  $2*(N-m)$  years and still have m trusted Electors -> still become up to date
  - Electors can sign multiple different CTLs which are distinguished by ElectorGroupId (name being changed to CtlSeriesId)



# Future work





- **1609.2: The theme is modularity**
  - Extend to formally support Chinese crypto algorithms
    - 1609.2 is already modular with respect to cryptography
    - 1609 WG would love to support Korean algorithms as well if appropriate and preserve global consistency
  - Add HeaderInfo extension mechanism so that other SDOs can extend 1609.2 functionality without being tied to the IEEE standardization cycle
    - Already being used by ETSI
  - New certificate field, Operating Organization ID
    - Can be used to determine whether to allow an activity based on whether the organization is allowed
      - E.g. in US, out-of-state police cars cannot request signal preemption, in-state police cars can
    - Requires maintenance of an “allow list” or “deny list” of permitted organization IDs
- **1609.2.1**
  - No current plans for future work on the standard though deployment work is necessary
- **Supporting standards:**
  - Misbehavior reporting based on ETSI TS 103 759 (almost complete)

# Deployment

- **SCMS Manager profile work**
  - Publication anticipated 2022
  - Identify preferred options / values for parameters in 1609.2.1 interfaces to simplify testing / deployment
    - There are a lot of options!
- **OmniAir conformance testing anticipated 2022**
  - OmniAir has already hosted interop testing based on v1
- **Deployment as default cert management approach anticipated 2022 onwards**
  - OEMs have the option of carrying out cert management by different interfaces but for smaller suppliers/deployments 1609.2.1 will be preferred and will rapidly become the most widespread approach



# Thank you

Follow us on:    

For more information, visit us at:

[www.qualcomm.com](http://www.qualcomm.com) & [www.qualcomm.com/blog](http://www.qualcomm.com/blog)

Nothing in these materials is an offer to sell any of the components or devices referenced herein.

©2018-2021 Qualcomm Technologies, Inc. and/or its affiliated companies. All Rights Reserved.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Other products and brand names may be trademarks or registered trademarks of their respective owners.

References in this presentation to “Qualcomm” may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes our licensing business, QTL, and the vast majority of our patent portfolio. Qualcomm Technologies, Inc., a subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of our engineering, research and development functions, and substantially all of our products and services businesses, including our QCT semiconductor business.

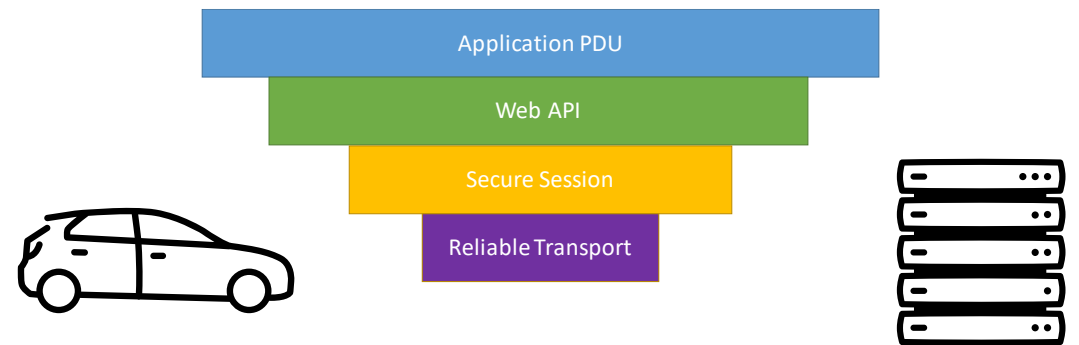


# Backup

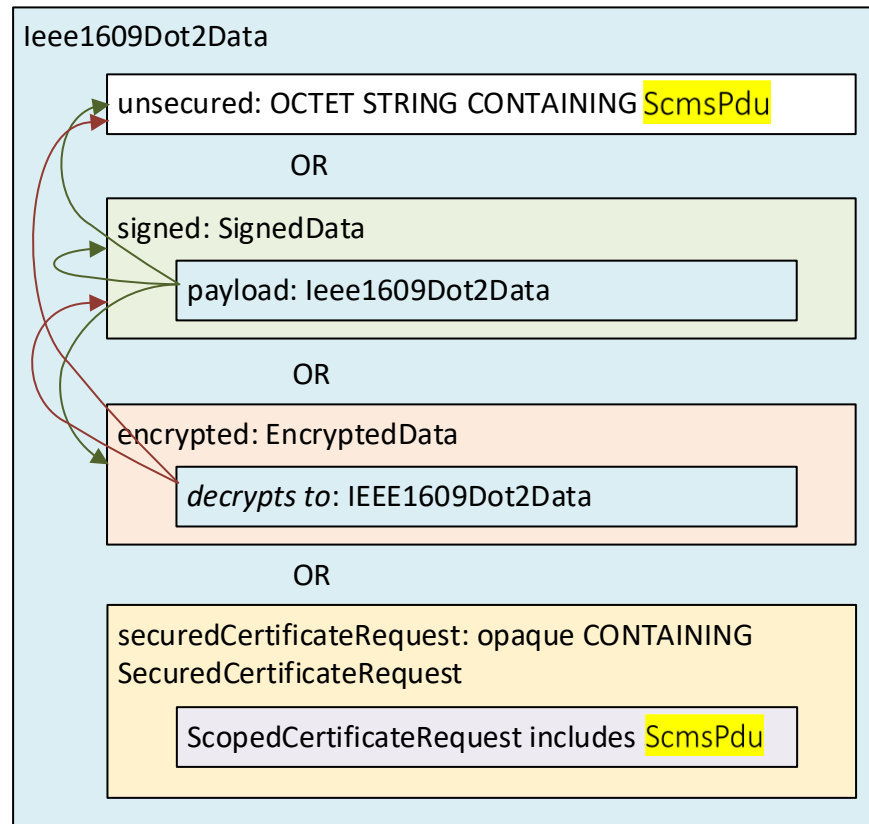
---

# Layered structure (1)

- Standard Web Application architecture for EE-to-SCMS communications
- Underlying reliable transport layer
  - TCP/IP
  - SCMS component network locations identified by URLs, translated to IP addresses by standard DNS processes
- Secure session: TLS (1.3 recommended - 1609.2.1 adds support for ISO 21177, a TLS extensions using 1609.2 certificates)
- Web API: One API supported, a REST API identifying which command is requested and giving high-level parameters
- Application PDU: contains detailed information for the command
  - This also has a layered structure, see next slide



# Layered structure (2)



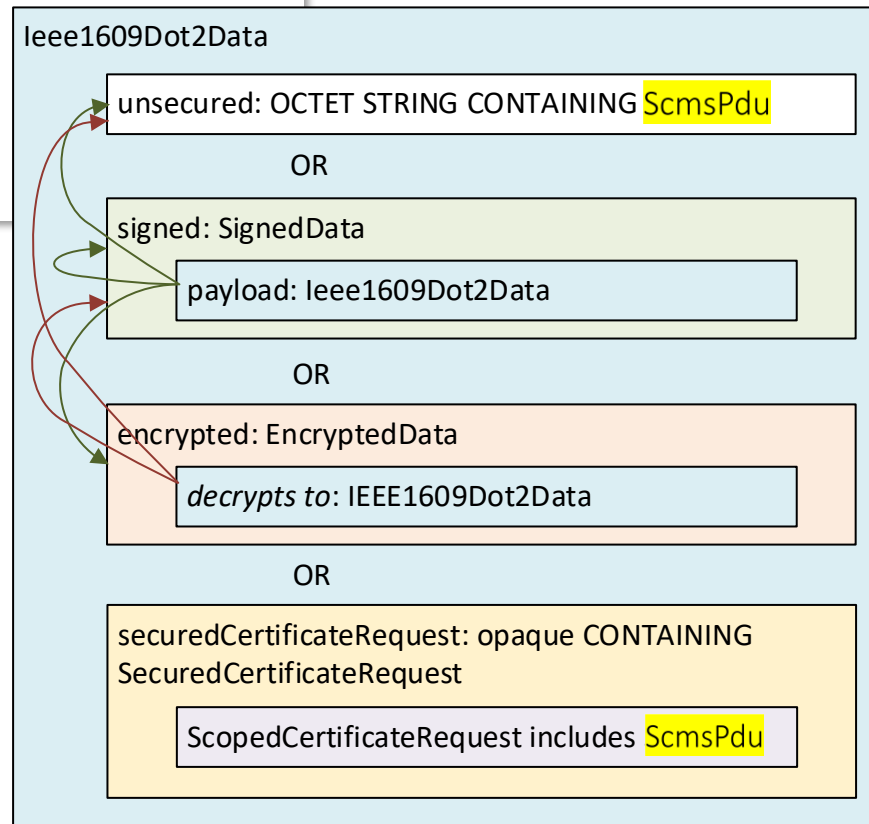
```
ScmsPdu ::= SEQUENCE {  
    version  Uint8 (2),  
    content  CHOICE {  
        aca-ee  AcaEeInterfacePdu,  
        aca-la  AcaLaInterfacePdu,  
        aca-ma  AcaMaInterfacePdu,  
        aca-ra  AcaRaInterfacePdu,  
        cert   CertManagementPdu,  
        eca-ee  EcaEeInterfacePdu,  
        ee-ma  EeMaInterfacePdu,  
        ee-ra  EeRaInterfacePdu,  
        la-ma  LaMaInterfacePdu,  
        la-ra  LaRaInterfacePdu,  
        ma-ra  MaRaInterfacePdu,  
        ...  
    }  
}
```

```
EeRaInterfacePdu ::= CHOICE {  
    eeRaCertRequest      EeRaCertRequest,  
    raEeCertAck          RaEeCertAck,  
    raEeCertInfo         RaEeCertInfo,  
    eeRaDownloadRequest  EeRaDownloadRequest,  
    eeRaSuccessorEnrollmentCertRequest EeEcaCertRequestSpdu,  
    ...  
}
```

# Layered structure (2)

```
AcaEeCertResponsePrivateSpdu ::= Ieee1609Dot2Data-EncryptedSigned {
  ScmsPdu-Scoped {
    AcaEeInterfacePdu (WITH COMPONENTS {
      acaEeCertResponse
    })
  },
  SecurityMgmtPsid
}
```

```
EeRa1609Dot2AuthenticatedCertRequestSpdu ::=
  Ieee1609Dot2Data-SignedEncryptedCertRequest {
    ScmsPdu-Scoped {
      EeRaInterfacePdu (WITH COMPONENTS {
        eeRaCertRequest
      })
    },
    SignerSingleCert
  }
```



```
ScmsPdu ::= SEQUENCE {
  version Uint8 (2),
  content CHOICE {
    aca-ee AcaEeInterfacePdu,
    aca-la AcaLaInterfacePdu,
    aca-ma AcaMaInterfacePdu,
    aca-ra AcaRaInterfacePdu,
    cert CertManagementPdu,
    eca-ee EcaEeInterfacePdu,
    ee-ma EeMaInterfacePdu,
    ee-ra EeRaInterfacePdu,
    la-ma LaMaInterfacePdu,
    la-ra LaRaInterfacePdu,
    ma-ra MaRaInterfacePdu,
    ...
  }
}
```

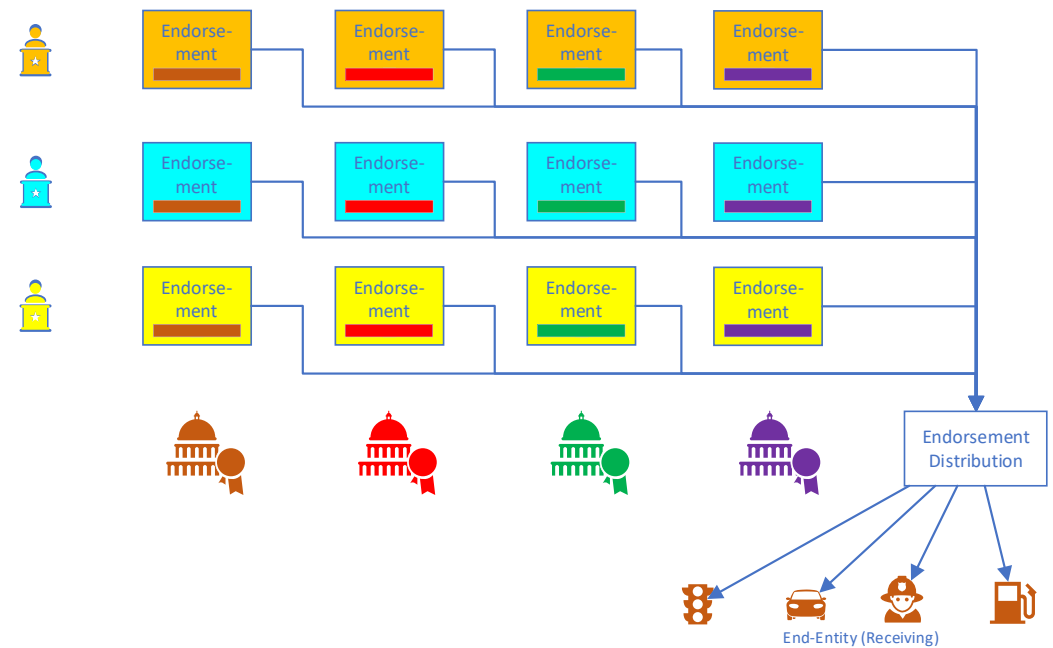
```
EeRaInterfacePdu ::= CHOICE {
  eeRaCertRequest EeRaCertRequest,
  raEeCertAck RaEeCertAck,
  raEeCertInfo RaEeCertInfo,
  eeRaDownloadRequest EeRaDownloadRequest,
  eeRaSuccessorEnrollmentCertRequest EeEcaCertRequestSpdu,
  ...
}
```

# ECQV Implicit certificates

- CA has key  $(c, C)$
- User generates ephemeral key  $(a, A)$ , sends to CA with cert information  $I$ 
  - $I$  is non-cryptographic information, ie does not include public key
- CA generates ephemeral key  $(d, D)$ , and:
  - Calculates
    - $B = A + D$
    - $h = \text{hash}(I || B)$
    - $s = h*d + c$ 
      - Standard Schnorr/ElGamal-like trick:  $h$  is known, but every possible  $d$  has a corresponding possible  $c$
  - Sends  $(B, s)$  to user
- User calculates private key  $e = s + \text{hash}(I||B) * a$ 
  - $h*a + h*d + e$
- Corresponding public key is  $E = \text{hash}(I||B) * B + C$ 
  - $h*A + h*D + C$
- When sending a signed message, user attaches the “implicit” cert  $[I, B]$ ; verifier uses cert to obtain public key and verify

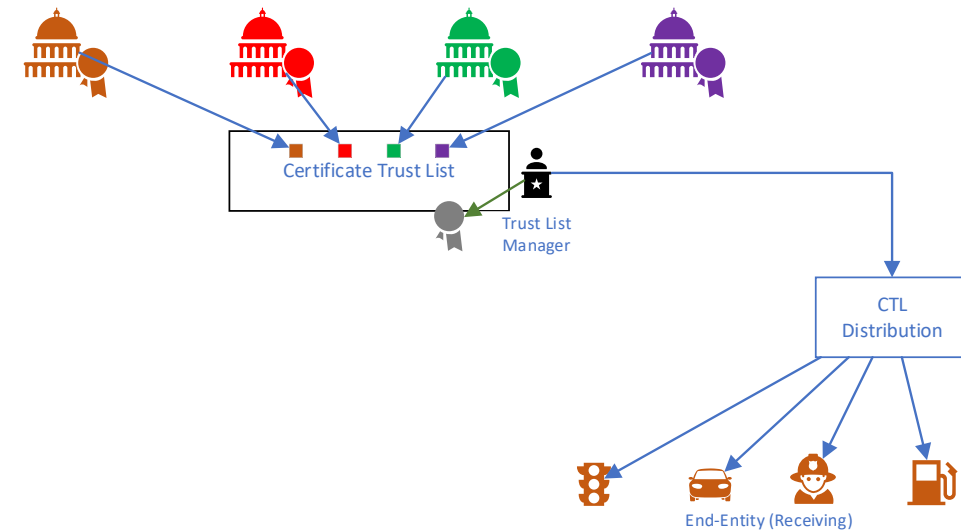
# Root CA management: CAMP specification

- N electors (N = 3 or 5)
  - Each elector individually endorses each root CA by issuing an “endorsement”
  - Electors also endorse other electors
- System participants trust Root CAs (or Electors) if there is a quorum of endorsements for that Root CA
  - Quorum  $m = 2$  or 3
- System is robust against compromise / expiry of electors
  - So long as m electors are trustworthy, new electors can be created and old ones retired
- However, CAMP specification is unclear about
  - Schedule and coordination of updates
  - How to aggregate individual ballots
  - Who is responsible for trust decisions - is it individual electors or someone else?
- Each elector certificate can only sign one set of root CAs - no possibility to reuse elector information
- Design runs risk of fragmentation and unclear policy - no good way to recover
  - What happens if electors A, B, C endorse elector K but electors D, E choose not to?



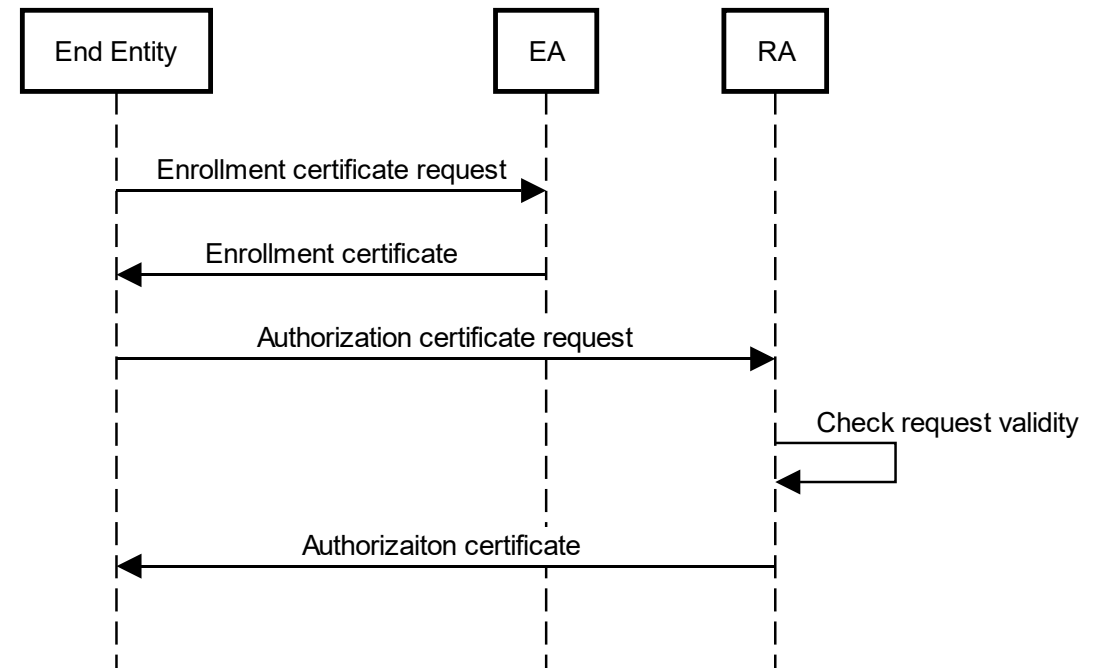
# Root CA management: ETSI 102 940

- Trust List Manager creates and signs Certificate Trust List (CTL) based on which root CAs follow its policy
  - No concerns around aggregation or coordination
  - Responsibility is clear: TLM selects the list of trusted root CAs based on its policy
- However, single signer creates single point of technical failure
  - If signing device is compromised there is no straightforward recovery path
  - When signing certificate expires and new certificate is issued, participants need to obtain the CTL after the creation of the new certificate and before the expiry of the old one
    - Narrow window - typically three months
    - Participants that miss the window are locked out with no straightforward recovery path



# Certificate permissions

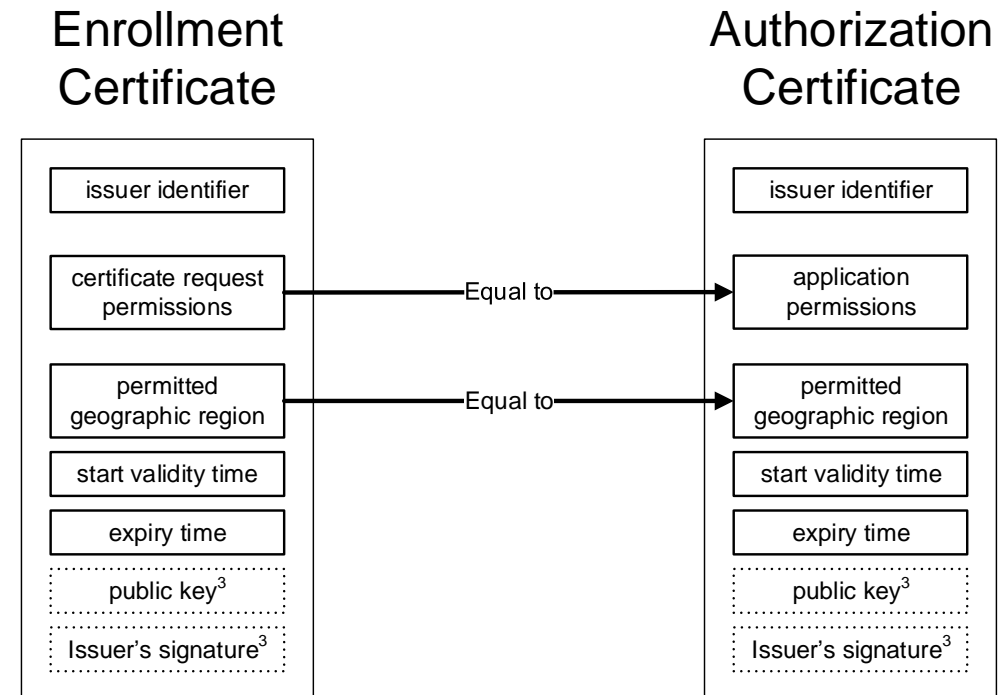
- Both CAMP and 1609.2.1 have the same overall flow for certificate request
  - EE requests and obtains enrollment certificate from EA
  - EE requests authorization certification from RA using enrollment certificate to authenticate request
  - RA checks that EE is entitled to auth cert and issues auth cert if so
- The difference is in how the RA establishes what is permitted in the auth cert





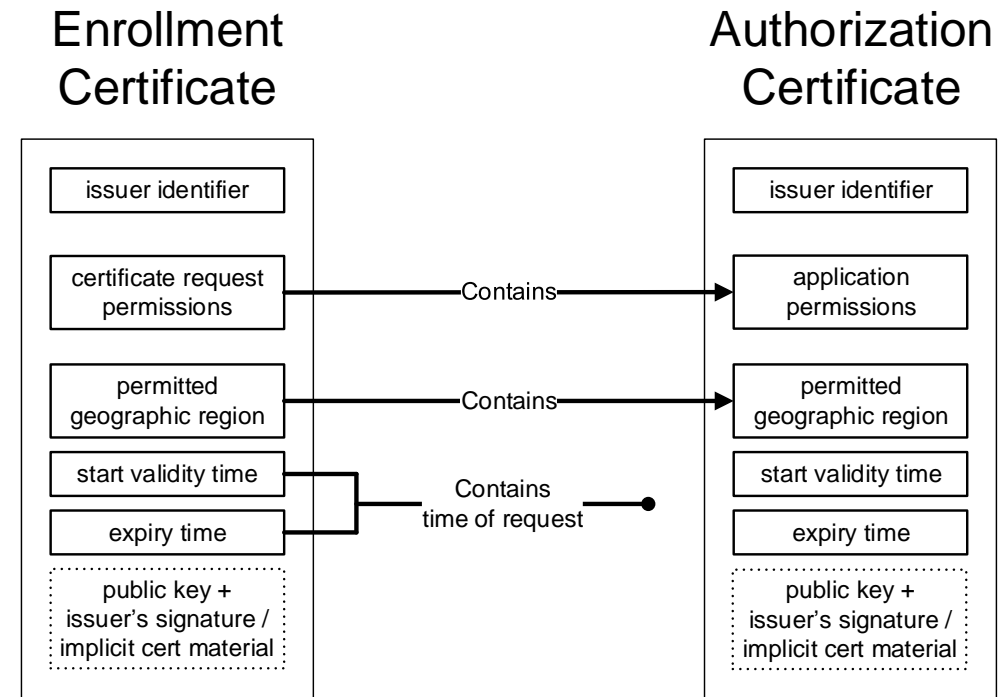
# Certificate permissions: CAMP

- Authorization (application / identification / pseudonym) certificate permissions have to exactly match enrollment certificate permissions
- This matching of permissions makes it difficult to
  - Add applications in the field
  - Move RSUs, such as work zone RSUs, from one location to another
  - And, so on
- Puts pressure on enrollment process to get permissions exactly right
- Hard to correct or change afterwards because enrollment is burdensome



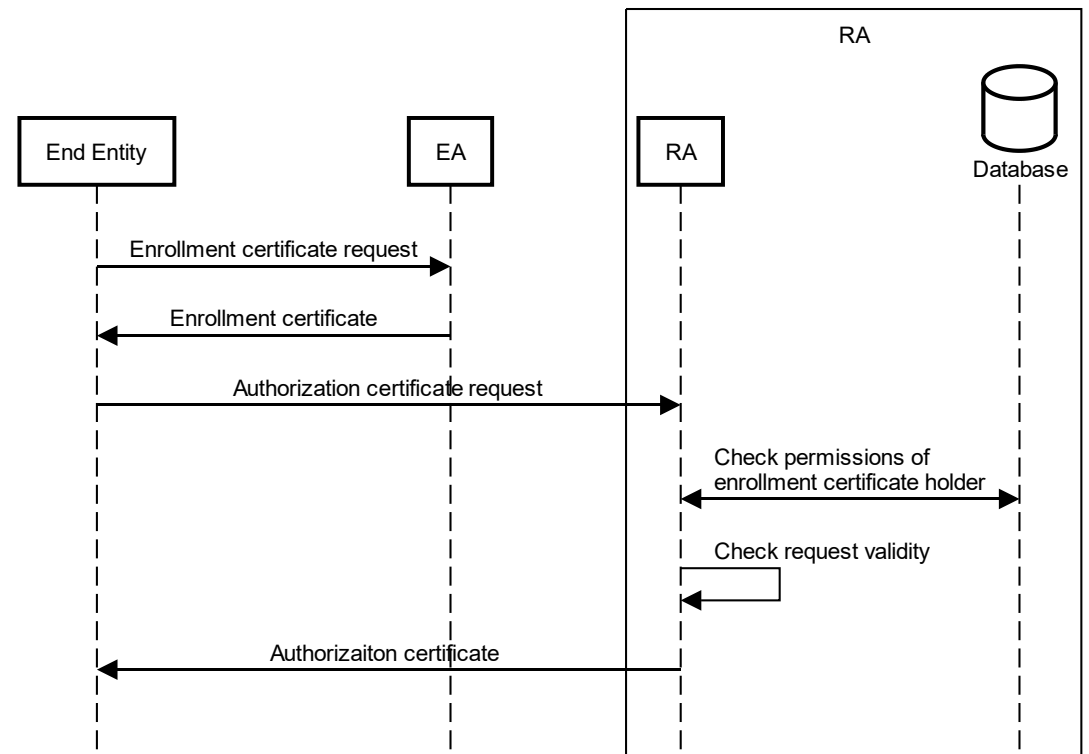
# Certificate permissions: 1609.2.1

- Permissions constraint is relaxed from “Enrollment permissions must be equal to authorization permissions” to “Enrollment permissions can be a superset of authorization permissions”
- RA stores and manages permissions associated with each enrollment cert, using the enrollment cert as a lookup key
- Now:
  - Auth cert request needs to include requested permissions
  - Don't need subtypes of request for different types of auth cert as RA is assumed to know
  - Can support X.509 enrollment certs as permissions are not necessarily encoded in cert



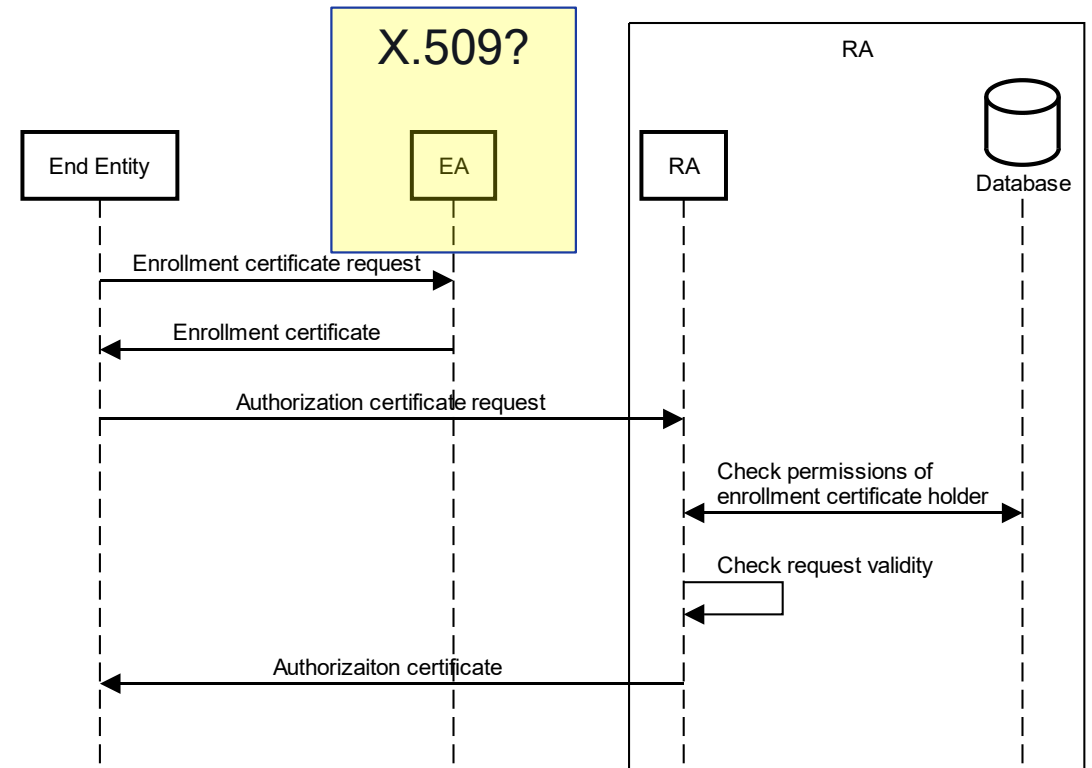
# Certificate permissions: 1609.2.1

- Permissions constraint is relaxed from “Enrollment permissions must be equal to authorization permissions” to “Enrollment permissions can be a superset of authorization permissions”
- RA stores and manages permissions associated with each enrollment cert, using the enrollment cert as a lookup key
- Now:
  - Auth cert request needs to include requested permissions
  - Don't need subtypes of request for different types of auth cert as RA is assumed to know
  - Can support X.509 enrollment certs as permissions are not necessarily encoded in cert



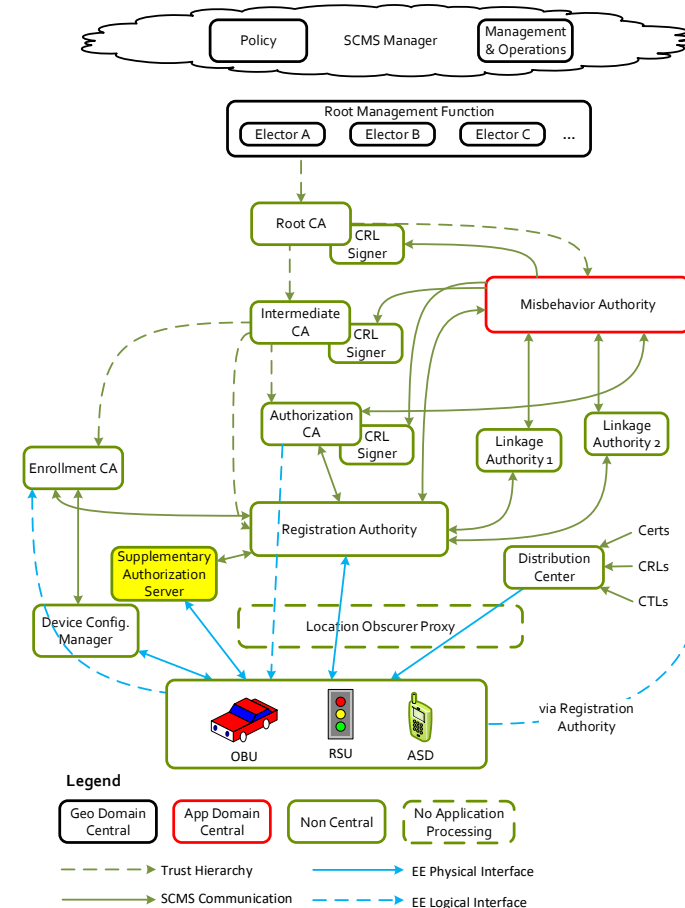
# Certificate permissions: 1609.2.1

- Permissions constraint is relaxed from “Enrollment permissions must be equal to authorization permissions” to “Enrollment permissions can be a superset of authorization permissions”
- RA stores and manages permissions associated with each enrollment cert, using the enrollment cert as a lookup key
- Change cert request format from CAMP as auth cert request needs to include requested permissions
- Enrollment certs can be X.509 as enrollment cert is just a lookup key
  - Useful for OEMs who already have an X.509 PKI



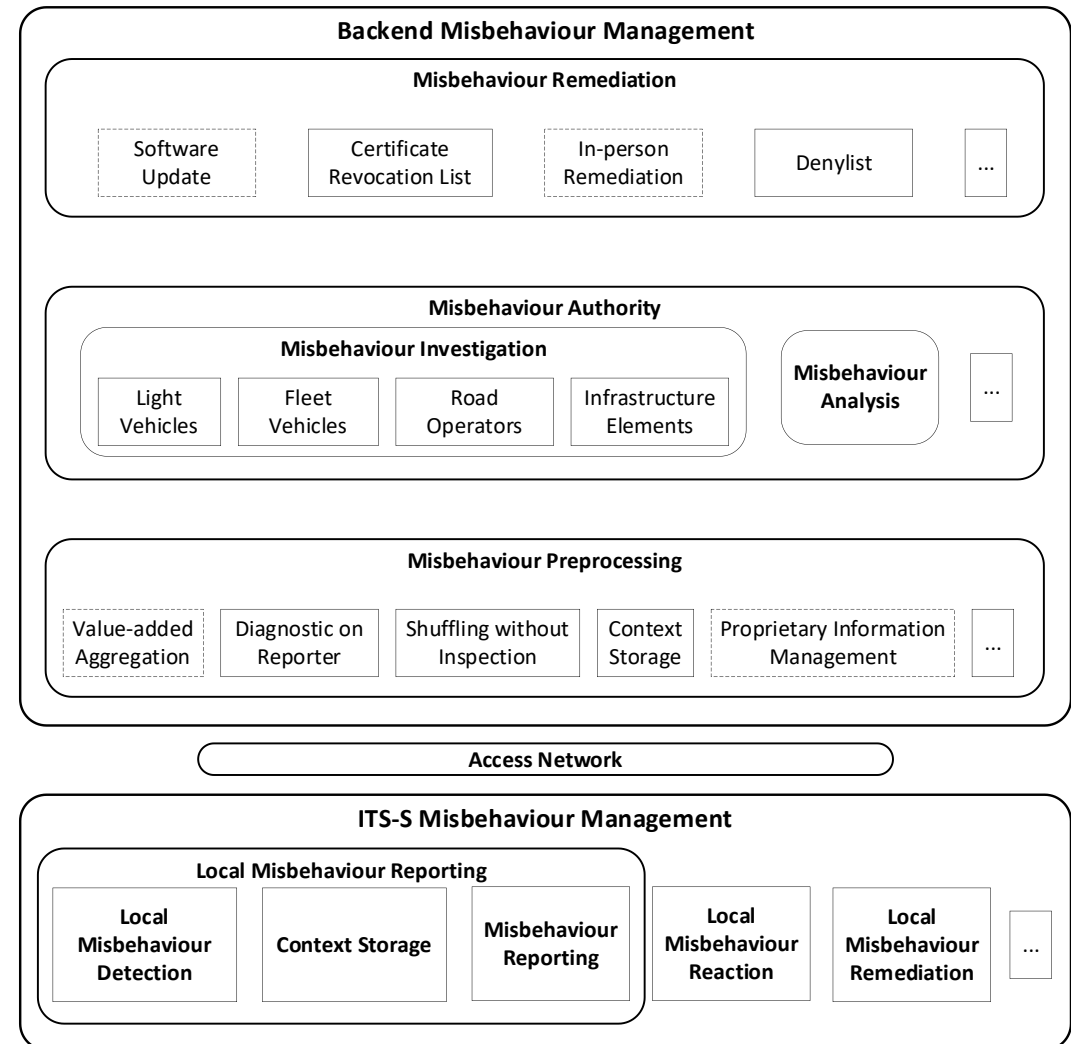
# Supplementary authorization

- Decoupling of certificate permissions enables supplementary authorization in IEEE 1609.2.1
  - Used for access control on download requests
    - Important: auth cert request messages are still directly signed by the enrollment cert
- Currently, only one mechanism is specified: OAuth 2.0 bearer authorization
- OAuth 2.0 is an optional feature of the SCMS
  - The EE authenticates with an OAuth authorization server (OAS) to receive an access token
  - The EE includes that access token in requests to the Registration Authority (RA)
  - The RA uses the access token to authorize the EE request
- Allows use of off-the-shelf web technologies ==> scalable access control management



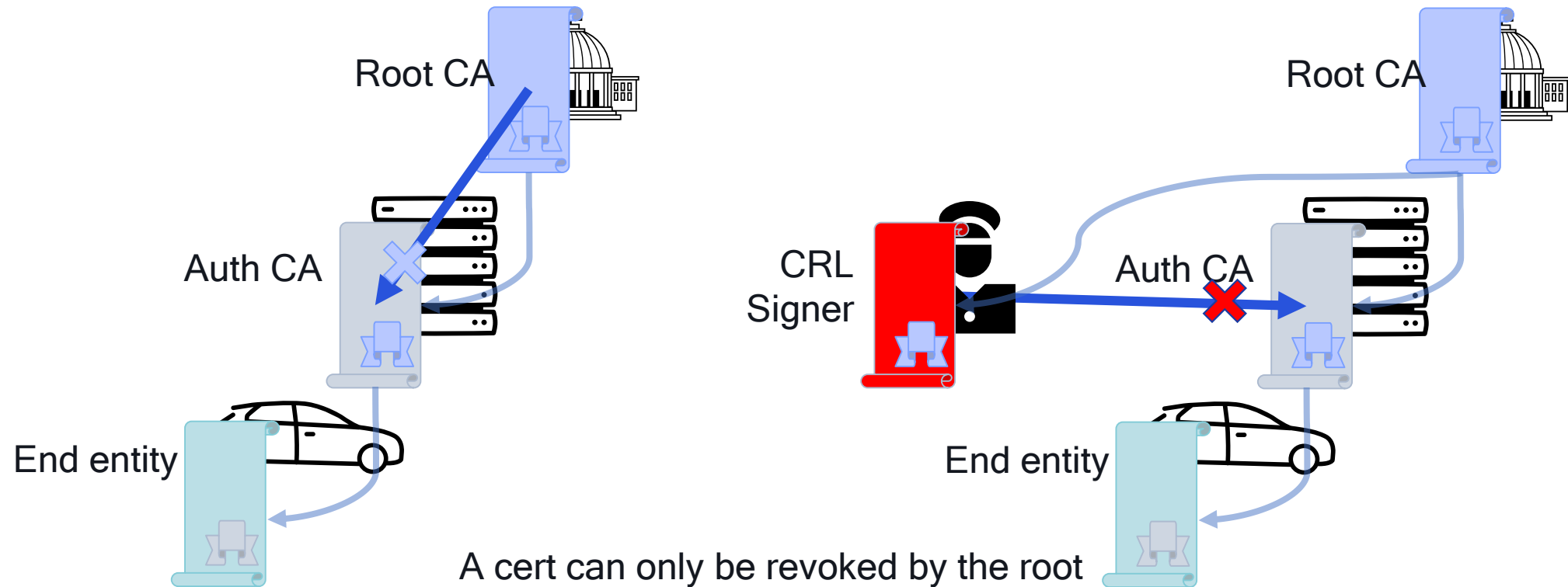
# Misbehavior Management: Architecture

- Architecture diagram being considered for ETSI TS 103 759
  - Not in 1609.2.1, but consistent with its architecture
- ITS-S detects misbehavior and generates report
- Misbehavior preprocessing optionally
  - Improves privacy: shuffles reports from different reporters
  - Improves data quality
    - Runs diagnostics on reporters
    - Aggregates reports and removes duplicates
  - ...
- MA analyses reports and determines if misbehavior occurred and what to do about it
  - There can be different MAs for different types of misbehavior, e.g. channel jamming is different from sending false messages
- Misbehavior Remediation takes the actions determined by the MA



# Cert revocation - current IEEE model

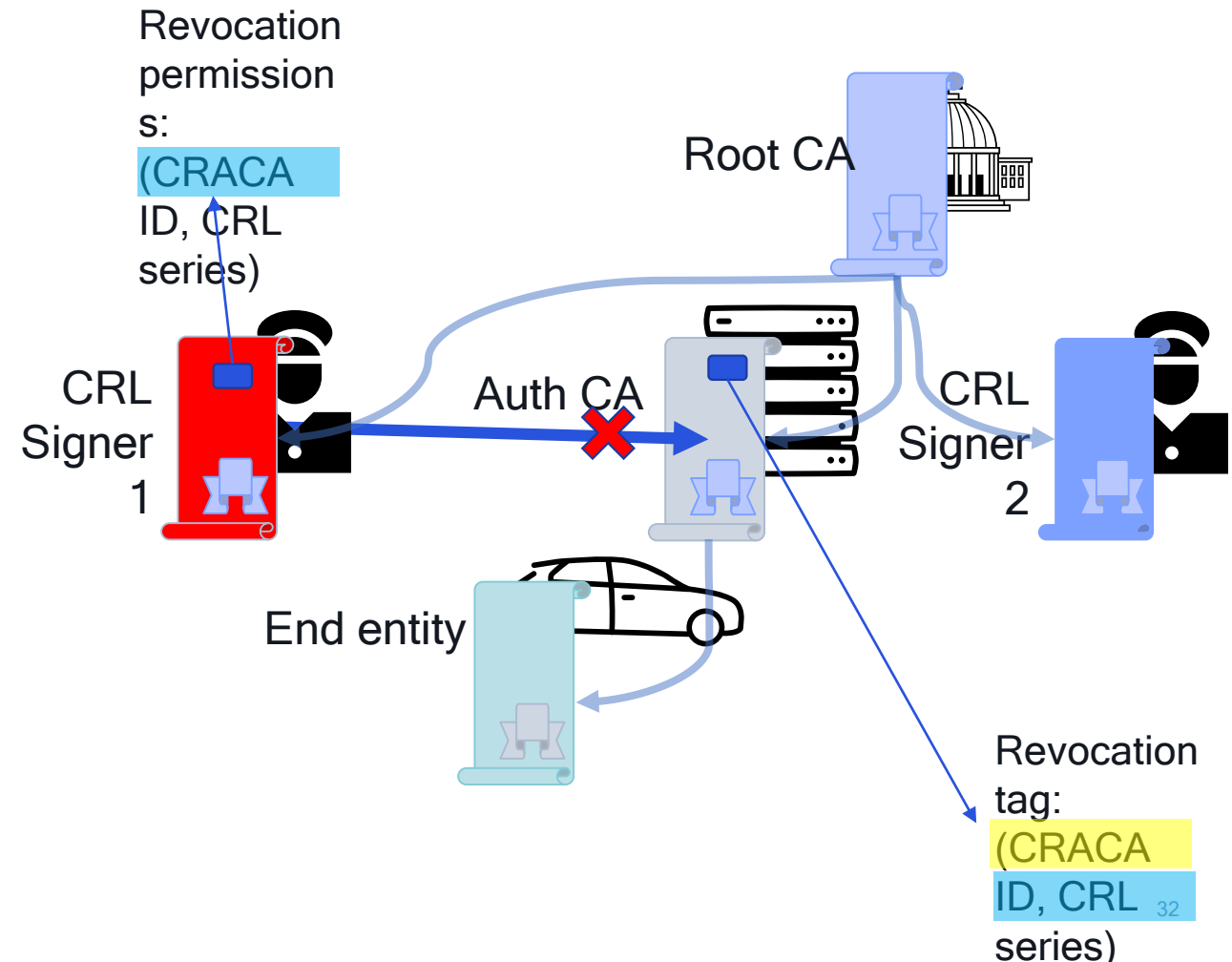
(as applied to current ETSI depth-2, no-EE-revocation architecture)



A cert can only be revoked by the root CA that it chains to or by a specific CRL signer designated by that root

# Cert revocation - current IEEE model

- So that a receiver knows which CRL a certificate might appear on, each “potentially revocable” certificate contains:
  - CRACA ID - identifies the certificate in the chain that EITHER revokes OR creates the relevant CRL signed and delegates revocation power to it
  - CRL Series - identifies the CRL
- Each CRL signer cert contains
  - Identifier for its CRACA (which is either itself or the CA cert that issued it)
  - The CRL series it’s responsible for
- Each “potentially revocable” cert can appear only on one “series” of CRLs, and each series of CRLs can only be issued by one CRL signer





# Three benefits of delegation

- Even if the Root CA key is not active, the CRL signer key can continue to be active, allowing certs issued by that CA to be revoked for the full lifetime of the root CA cert
  - Addresses the transition issue
- CRL issuance will happen at regular intervals (nextUpdate date); use of separate delegated CRL signer allows CRL issuance to be “airgapped” from ability to issue new certs, reducing the number of times when cert-issuing ability could be compromised
  - Reduces operational risk
- “Waking up” the CRL signer might be less expensive than waking up a root CA - policies around signing CRLs are often more relaxed than policies around enabling a root CA to issue certs
  - Reduces OPEX (though separate CRL signer might increase CAPEX).

# Lessons learned from deployment (1)

- Production SCMS is needed for RSU certificate updates
  - RSUs need to be able to access the SCMS at least once a week
- Production SCMS is needed for certificate top-off for the OBUs.
  - OBUs need to be able to access the SCMS at least once a week
  - Test all connections, especially if OBUs are connecting through RSUs and especially if IPv6-to-IPv4 translation is to be carried out.
  - Test correct operation if the first certificate download fails - i.e. repeat download request
- Test that the system behaves correctly when devices transition between certificates, i.e. when one certificate expires and a new one is expected to be used
  - Devices attempt to update certificates in a timely manner
  - Devices stop using the old certificates and start using the new certificates at the appropriate time, i.e. before expiry of the old and after start-validity of the new
  - If there are conditions that inhibit certificate change (e.g. alert state per J2945/1) test that system transitions correctly after condition stops holding

# Lessons learned from deployment (2)

- Integrate the CV needs into the agency's cybersecurity program early on in the project.
  - Network traffic necessary for SCMS operation will require various firewall settings, proxy servers, etc., that will need to meet the agency's overall cybersecurity standards
- Maintain a secure, trusted network environment for the CV pilot deployment system.
  - Try to isolate it from the general traffic management network
- Test the OTA download and upload mechanisms before extensive installation of the ASDs and RSUs.
- For multi-application devices, understand whether there will be one certificate per application, one certificate per device, or something else
  - Some RSU vendors only support a single application certificate covering all PSIDs.
- Ensure that the RSU firmware permits sending pre-signed messages (MAP, TIM).