EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Communication Networks

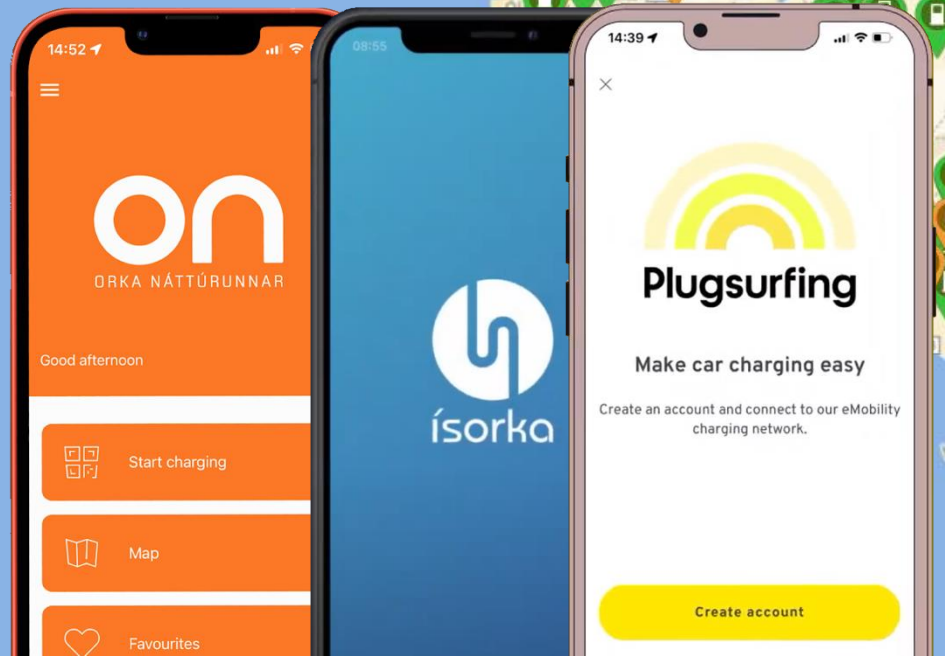
OAuth Cross-Device Flow for Enhanced Authorization in Electric Vehicle Charging

by Jonas Primbs, University of Tübingen, Germany

https://kn.cs.uni-tuebingen.de

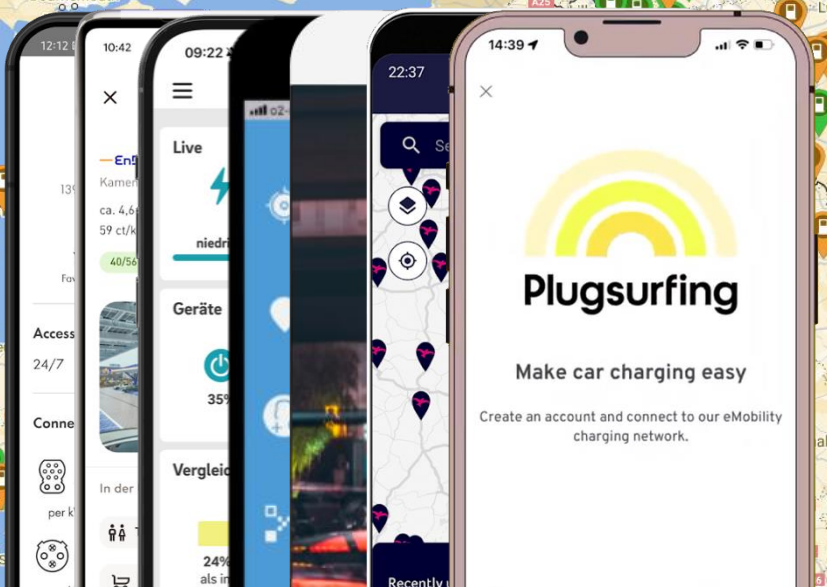
EV Charging in Iceland …

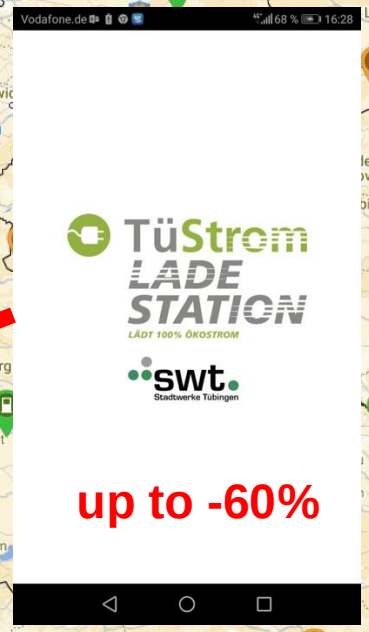
- 2 large energy providers



EV Charging in Germany …

- 7 large energy providers
- 2 large charge point operators
- 1.985 local energy providers





Alternative: AutoCharge

- **Used by**
  - Tesla + Supercharger
  - Shell Recharge
- **Idea**
  - Establish Powerline connection between **electric vehicle (EV)** and **charge point (CP)**
  - C**harge point operator (CPO)** identifies the EV by its MAC address for automatic billing
- **Downsides**
  - MAC address spoofing is no rocket science…

Shell Recharge

Secure Solution: Plug & Charge (PnC)

Goal

- Simplicity of AutoCharge with security of asymmetric cryptography
  - Based on X.509 certificates
- Standardized in **ISO 15118**

Charging Process

1. Driver plugs EV into CP
2. CP authenticates EV by X.509 **Contract Certificate (CC)** using challenge-response
3. CP charges EV
4. CP sends costs to CPO
5. CPO sends bill to eMSP
6. eMSP adds bill to end-of-year invoice of driver

Where does the CC come from?

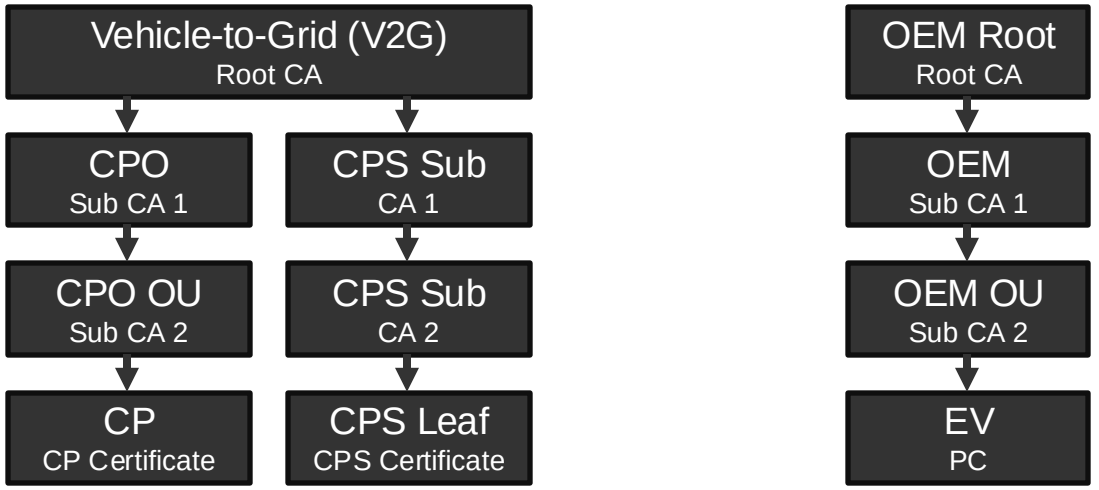
Secure Solution: Plug & Charge (PnC)

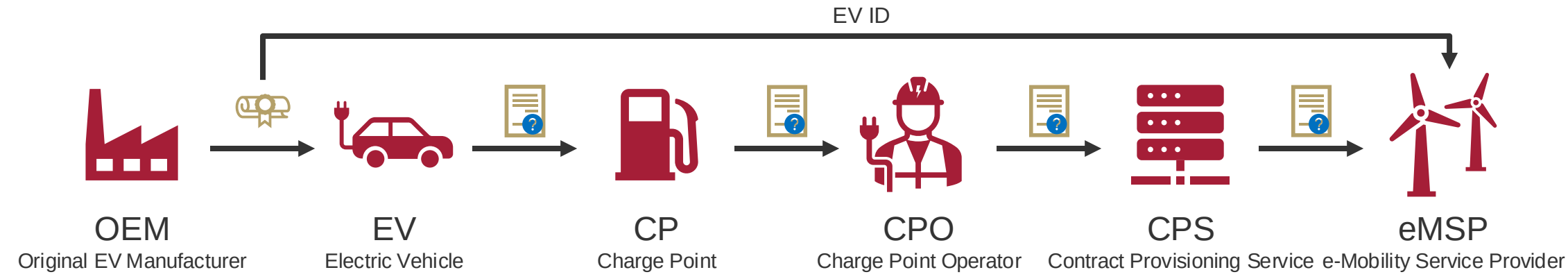
Provisioning

1. OEM provides EV with **Provisioning Certificate (PC)**
2. Driver somehow registers EV's unique ID (EV ID) at eMSP
   - EV ID = Common Name (CN) X.509 field of PC

   Standard says "out of scope"
3. Driver plugs EV into CP for the first time
   - EV and CP establish mutual TLS (mTLS) connection
4. EV sends contract provisioning request via CP, CPO, and CPS to eMSP

Public Key Infrastructure (PKI)



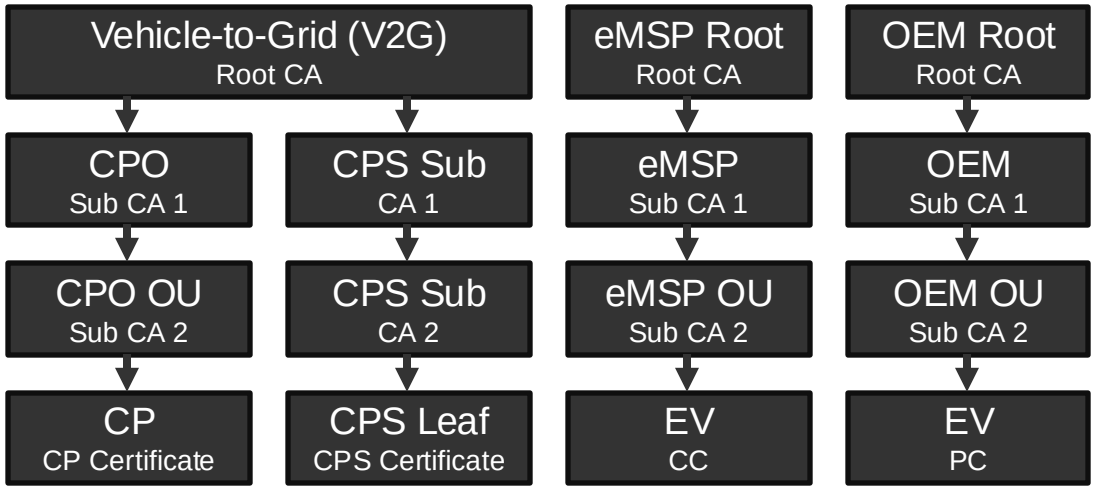


Secure Solution: Plug & Charge (PnC)

Provisioning

5. eMSP generates an asymmetric key pair for the EV
6. eMSP signs the public key → **Contract Certificate (CC)**
7. eMSP encrypts private key with EV's PC public key
8. eMSP sends encrypted private key + CC to EV
9. EV decrypts private key with PC private key and uses CC for charging

Public Key Infrastructure (PKI)



EV
Electric Vehicle

CP
Charge Point

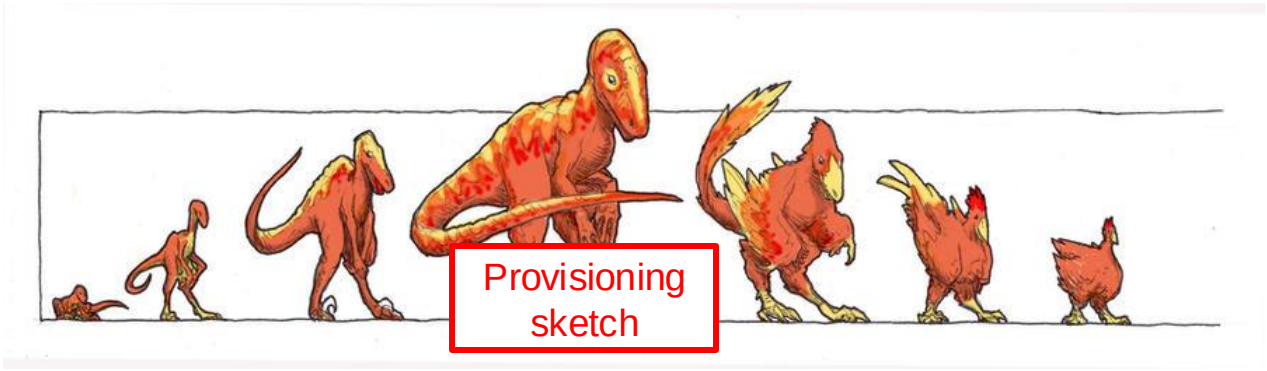
CPO
Charge Point Operator

CPS
Contract Provisioning Service

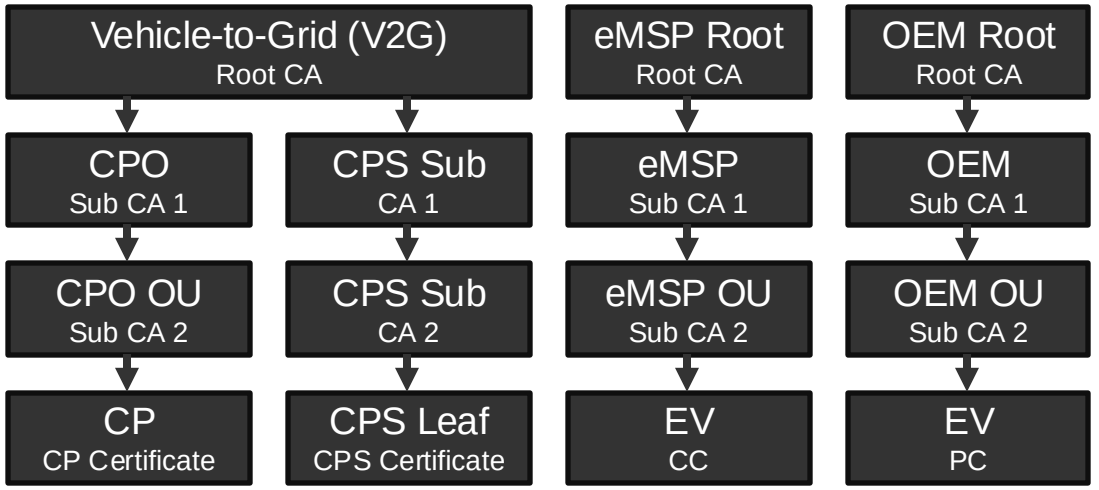
eMSP
e-Mobility Service Provider

Secure Solution: Plug & Charge (PnC)

ISO 15118

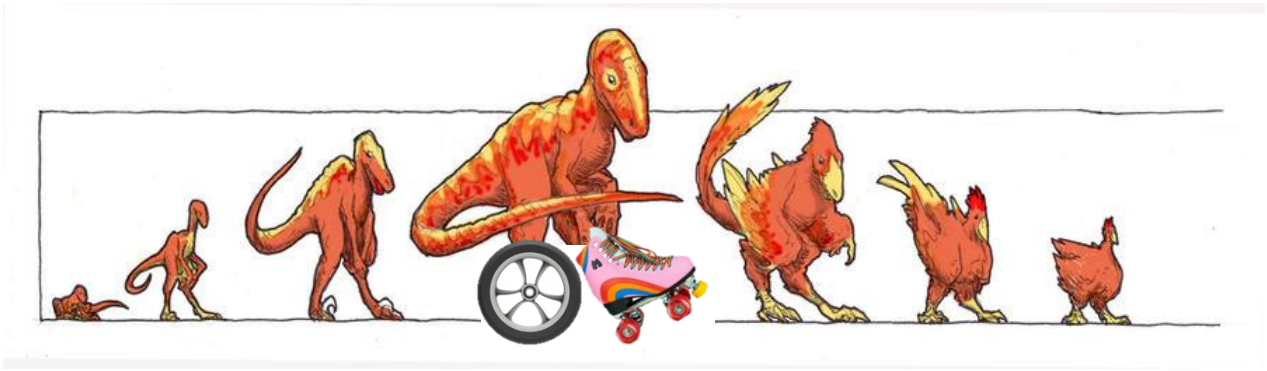


Public Key Infrastructure (PKI)

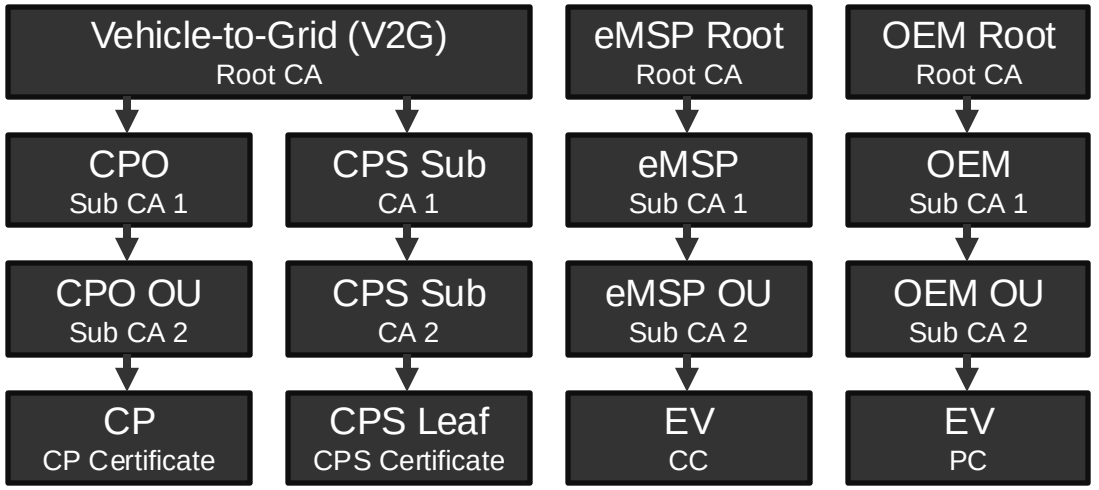


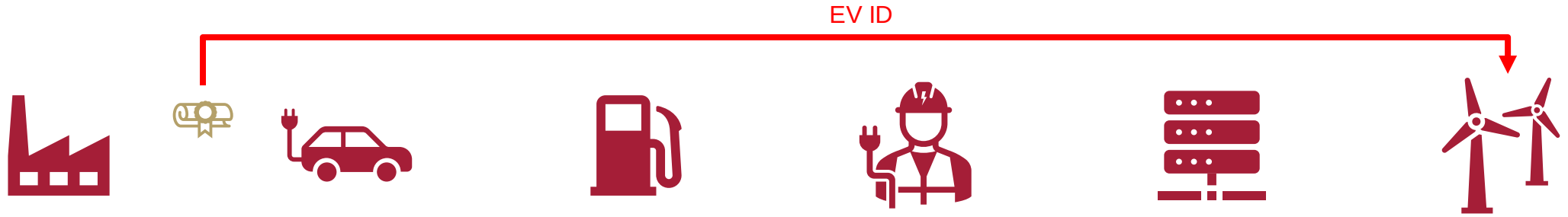
Secure Solution: Plug & Charge (PnC)

ISO 15118



Public Key Infrastructure (PKI)

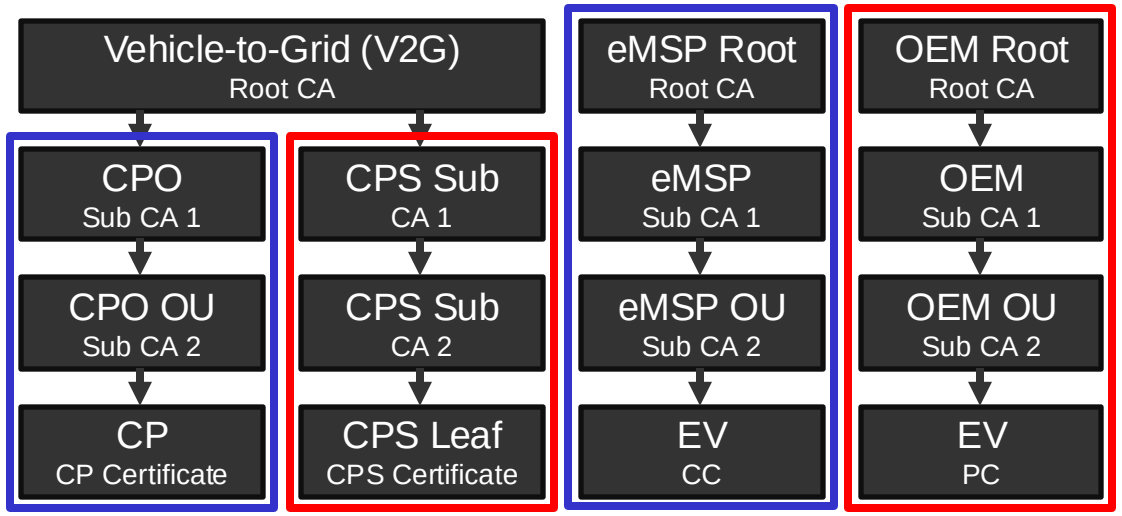


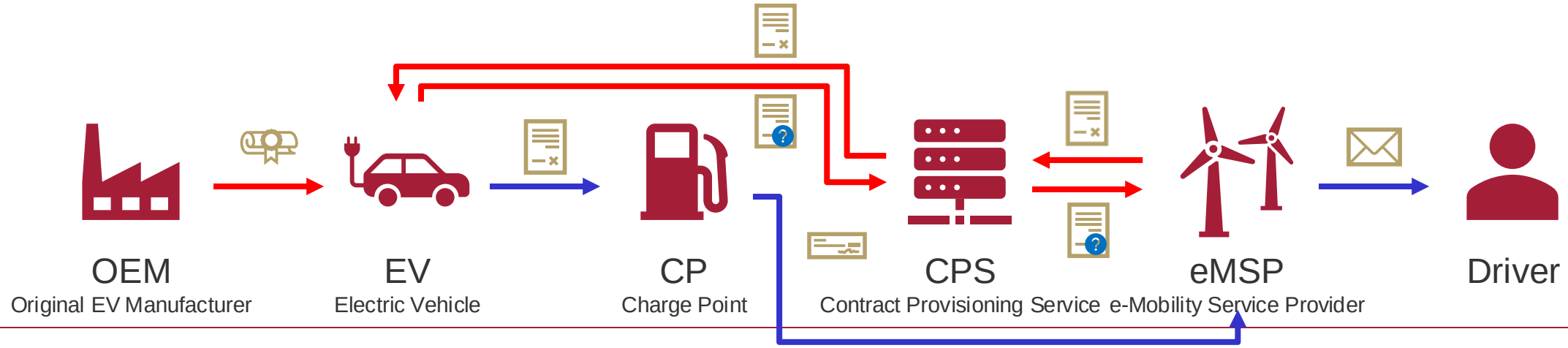


Summary: Plug & Charge (PnC)

- **PnC requires 4 PKI chains**
  - 2 exclusively for provisioning
    - OEM chain for EV authentication with **Provisioning Certificate (PC)**
    - CPS chain for CPS authentication
  - 2 for charging
    - CPO chain for CP authentication
    - eMSP chain for EV authentication with CC
- **Security Issues**
  - Private key for CC is known by eMSP
  - Private key is transferred via CPS, CPO, and CP

Public Key Infrastructure (PKI)

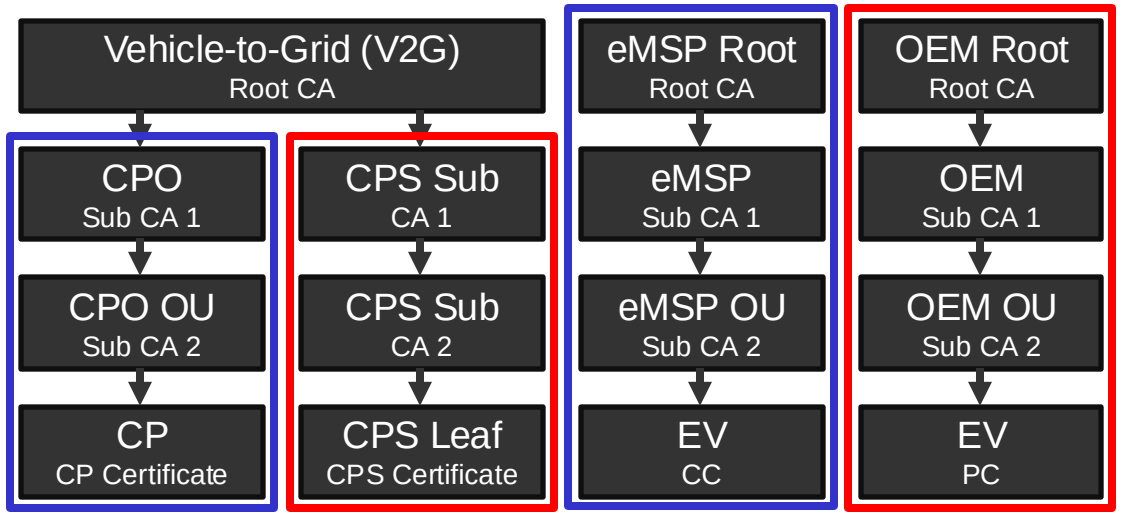


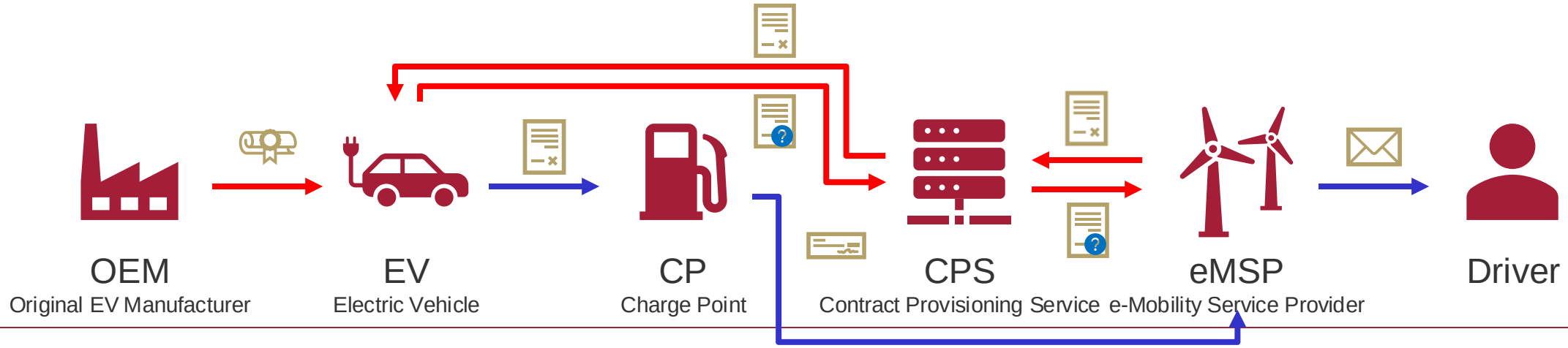


Summary: Plug & Charge (PnC)

- **Terrible Usability**
  - No standard for lookup of EV ID
  - No standard for registration of EV at eMSP
  - → Depends on combination of EV and eMSP
- **No fine-grained authorization**
  - **Contract Certificate (CC)** expires with contract period (approx. 1 year)
  - Uses revocation infrastructure of X.509
  - → Bad for rental cars

Public Key Infrastructure (PKI)



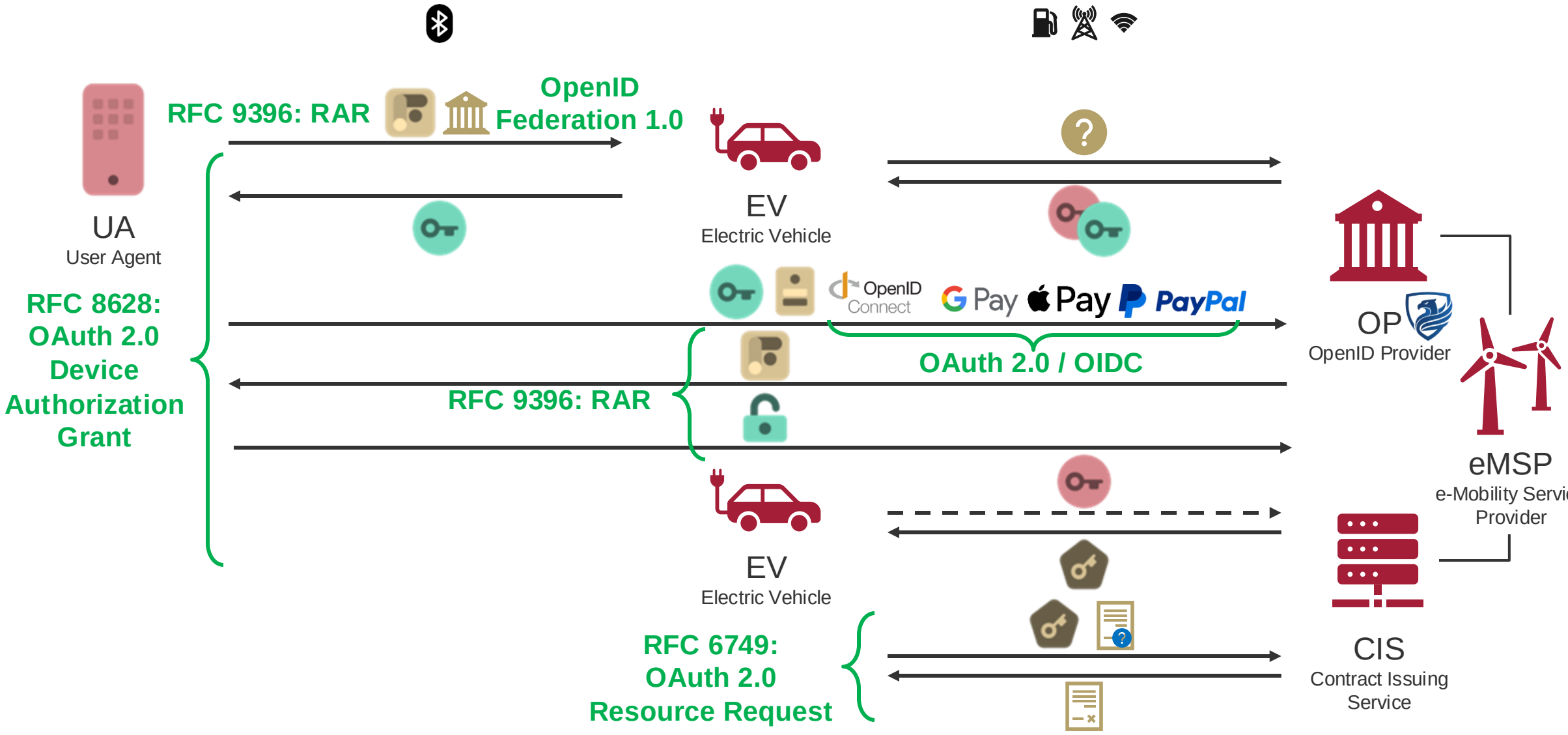


Proposed Solution: OAuth 2 + Extensions

1. Driver connects **User Agent (UA)** to EV via Bluetooth
2. Driver configures authorization
   - Preferred eMSP from list in OpenID Federation
   - Fine-grained preferences, e.g., validity period
3. UA sends configuration to EV
4. EV sends authorization request to preferred eMSP's **Identity Provider (IdP)**
   - Contains **Rich Authorization Request (RAR)**
   - Uses Device Authorization Flow
5. IdP responds with Device Code, User Code, and Verification URI to EV
   - EV starts polling for Access Token with Device Code
   - EV displays User Code to Driver
   - EV forwards Verification URI to UA
6. UA opens Verification URI for Driver
   - Driver logs in to IdP
   - Driver enters User Code
   - Driver authorizes Authorization Details from RAR
7. EV polls for an Access Token at IdP
   - Proves authorization with Device Code
8. IdP responds with Access Token
   - Rejected by IdP, if Device Code is not authorized yet
9. EV prepares **Contract Certificate Request (CCR)**
   - Generates asymmetric key pair in TPM
     - Private key cannot be exported
   - Creates **Certificate Signing Request (CSR)**
10. EV requests CC from CIS
    - Contains CSR
    - Proves authorization with Access Token
11. CIS issues CC to EV
    - Validity period of CC depends on Authorization Details from Access Token
    - Other authorization details may be applied too

visualization on next slide →

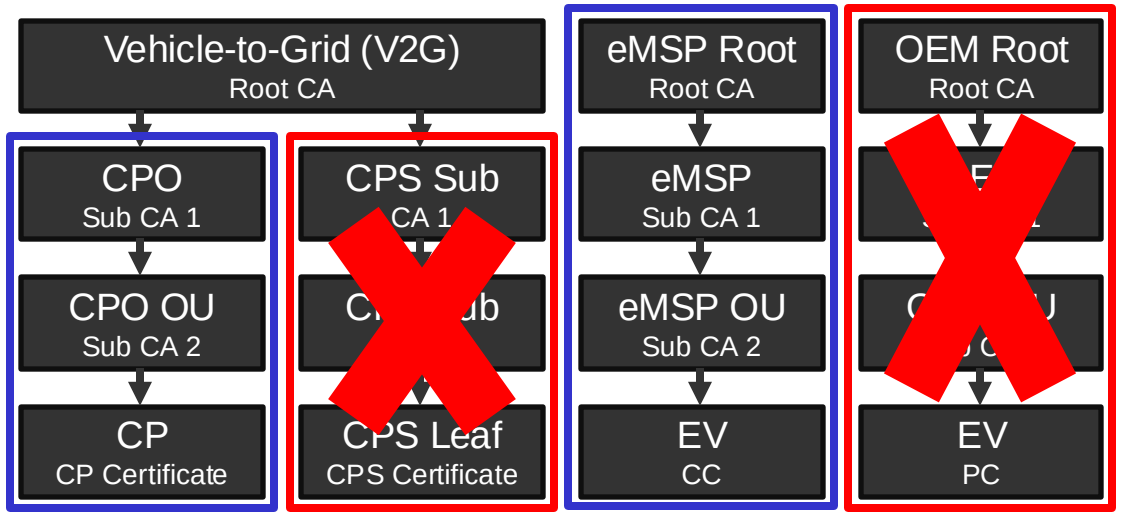
Proposed Solution: OAuth 2 + Extensions

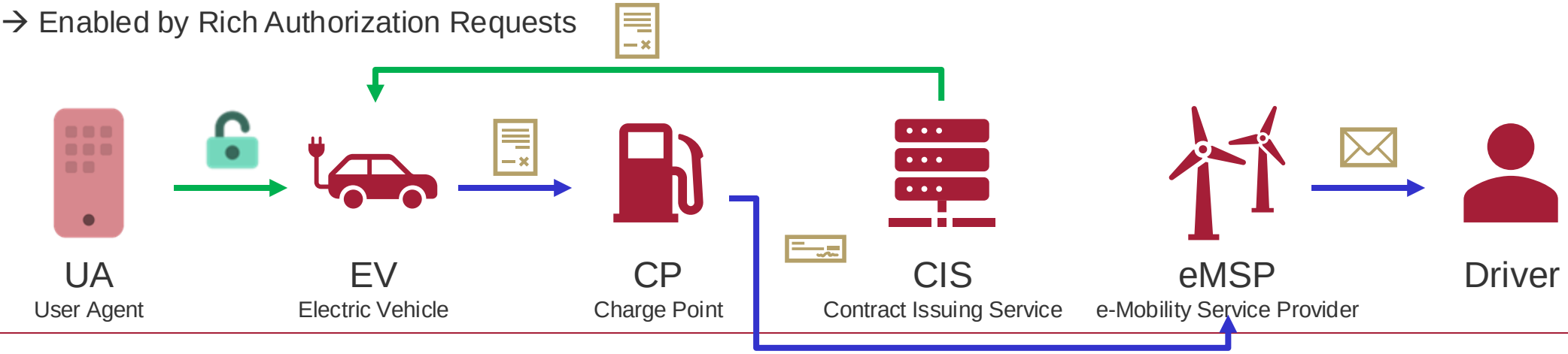


Conclusion

- **Reduced Complexity**
  - PKI chains for provisioning not needed anymore
  - → Replaced by OpenID Federation as trust anker
- **Better Security**
  - EV generates key pair in Trusted Platform Module
- **Better Usability**
  - Standardized & guided process
  - → Enabled by OAuth 2.0 Device Authorization Flow
- **Fine-grained Authorization**
  - CC validity can be specified by Driver
  - → Enabled by Rich Authorization Requests

Public Key Infrastructure (PKI)





Thank You!

- Thanks to all the standard authors for writing the standards!
- Thanks to Authlete for providing free access to their services!
- Thank you for listening!

- **Questions & Feedback?**

- Paper preprint: https://arxiv.org/abs/2501.14397