PNNL-34843

Inventory of Public Key Cryptography in US Electric Vehicle Charging

September 2023

Thomas E Carroll †
Lindsey M Redington
Addy M Moran-Schmoker
Andrew J Murray

† Corresponding author: Thomas.Carroll@pnnl.gov

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

PNNL-34843

Inventory of Public Key Cryptography in US Electric Vehicle Charging

September 2023

Thomas E Carroll †
Lindsey M Redington
Addy M Moran-Schmoker
Andrew J Murray

† Corresponding author: Thomas.Carroll@pnnl.gov

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Summary

Electric vehicles (EVs) and charging infrastructure are networked systems, which employ high-level communications in support of charging and grid service decisions. Public key cryptography (PKC) are algorithms that underlie security and privacy protections of the EV charging information exchanges. We are entering a new epoch where quantum computing threats must be seriously considered. A sufficiently large quantum computer, so named Cryptographically Relevant Quantum Computer (QRQC), will be able to perform the mathematical operations to efficiently attack the underpinnings of traditional PKC, thus jeopardizing the digital foundations for trust, communications security, and data security. Estimates suggest a QRQC can break public key encryption and digital signatures in the manner of tens to hundreds of hours (Grumbling and Horowitz 2019, Table 4.1), compared to traditional computing that would demand more than 10^{18} years in a brute force-style attack. A consensus belief of quantum theorists, quantum experimenters, and cryptographers suggest that the quantum threat will be likely realized in the next twenty years (Mosca and Piana 2022). To address the threat, post-quantum cryptography, which is cryptosystems that are designed to be secure against both traditional and quantum computing threats, must be adopted. Migration from traditional PKC to quantum-resilient cryptography is a global undertaking and likely represents the largest transition in computing history. The nascent state of EV public key infrastructure, combined with limited adoption of the vehicle secure charging features, presents an opportunity to establish a preference for quantum-resistant cryptography as a step on the migration path. Delays will stunt the efforts as rapidly accelerating EVs sales and huge infrastructure investments will create large growing bases of long-lived vehicles and infrastructure. Migration preparations can commence while NIST continues the process to standardize post-quantum cryptography (PQC), which are quantum-resilient algorithms designed to be secure against traditional and quantum computing threats. The first step in preparing EV charging is to identify the presence of traditional public key cryptography algorithms and applications. With this objective in mind, this report is intended to advise the vehicle manufacturers, charging station manufacturers, charging station operators, charge network providers and other EV charging stakeholders with information on traditional PKC application and the potential risks when PKC becomes insecure.

This report, the first in a series of reports discussing the topics existing at the confluence of post-quantum cryptography adoption and EV charging, identifies traditional public key applications employed and identifies potential consequences of leaving EV charging infrastructure vulnerable to quantum computing. The focus remains squarely on the of EV charging and infrastructure with respect to PKC and is believed by the authors to complement the NIST SP 1800-38 *Migration to Post-Quantum Cryptography* (Newhouse et al. 2023). While the report is centered on infrastructure, there are implications to vehicles.

Acknowledgments

The Pacific Northwest National Laboratory project team gratefully acknowledges the support provided by Mr. Lee Slezak of the Department of Energy Vehicle Technologies Office. We also extend our thanks to Dr. Frank Tuffner, Pacific Northwest National Laboratory, for his peer review of this document. The accuracy of the information and the perspectives presented in this report are the responsibility of the authors.

Acronyms and Abbreviations

AEAD	authenticated encryption with associated data
AES	Advanced Encryption Standard
CA	certificate authority
CSMS	charging station management system
CSO	charging station operator
CRQC	Cryptographically Relevant Quantum Computer
CS	charging station
DSA	Digital Signature Algorithm
ECC	elliptic-curve cryptography
ECDH	Elliptic-curve Diffie–Hellman
ECDHE	Ephemeral Elliptic-Curve Diffie-Hellman
EV	electric vehicle
EVCC	Electric Vehicle Communication Controller
EVCI	electric vehicle charging infrastructure
EVCS	electric vehicle charging station
EVSE	electric vehicle supply equipment
GCM	Galois/Counter Mode
ISO	International Organization for Standardization
KEM	key exchange mechanism
NEVI	National Electric Vehicle Infrastructure
OCPI	Open Charge Point Interface
OCPP	Open Charge Point Protocol
OCSP	Online Certificate Status Protocol
OEM	Original Equipment Manufacturer
PKC	public key cryptography
PKI	public key infrastructure
PnC	Plug and Charge
PQC	Post-quantum cryptography
RSA	Public key cryptosystem named after its inventors Rivest, Shamir, and Adleman
SECC	Supply Equipment Communication Controller
SHA	Secure Hash Algorithm
TLS	Transport Layer Security
V2G	vehicle to grid
XFC	Extreme Fast Charger

XML	Extensible Markup Language

Table of Contents

Figures

Tables

1.0 Introduction

Electric vehicle (EV) charging is a hyber hybrid system that exhibits characteristics of Internet-of-Things, operational technology, and cloud computing. Cryptography is utilized to protect the confidentiality and integrity of the communication, data, and operations in the EV charging infrastructure (EVCI). As is common in other ecosystems, EV charging systems make extensive use of public key cryptography (PKC). PKC is a method of encrypting or signing data with a key pair. Each key pair comprises a public key and a numerically-related private key. The public key is made available for anyone to use. A public key certificate is a digital document that cryptographically links the public key to the owner. Public key infrastructure (PKI) is then used to manage and distribute the certificates.

The public key and the corresponding private key are mathematically related to one another. While mathematically possible to derive the private key from the public key, the process is assumed computationally hard. A computationally hard problem means an algorithm does not currently exist that can derive the private key before the application rekeys and new keys are generated. The time to expire is often defined as a non-functional requirement and dependent on the purpose of the keys and the cryptography algorithm used. Quantum computing challenges this assumption. Quantum computing is a rapidly emerging compute technology that uses properties of quantum physics to compute and store data, providing the capacity to solve some complex problems more efficiently than traditional computers[1]. A sufficiently large general quantum computer, known as a Cryptographically Relevant Quantum Computer (CRQC), will have the potential to quickly break existing traditional PKC. For public key cryptosystems that are widely utilized, CRQC will derive the private key in the matter of tens to hundreds of hours (Grumbling and Horowitz 2019; Gidney and Ekerå 2021; Webber et al. 2021). Once that epoch has been reached, encrypted data that is considered safe today may be speedily decrypted. Digital signatures will be readily forged, degrading trust, authenticity and source origination. In the case of public key infrastructure (PKI), the efforts to rekey a certificate authority and issue new certificates would exceed the time needed to attack the new signatures.

National Institute of Standards and Technology (NIST), an agency of the U.S. Department of Commerce, is standardizing post-quantum cryptography (PQC) public key cryptosystems to defend against traditional and quantum computing advancements. PQC cryptosystems are substitutes for traditional cryptosystems, serving the same purposes and goals, but they are resistant to CRQC. Efforts to accelerate PQC adoption are underway (Rep. Khanna 2022; The White House 2021; US Department of Energy 2023) even though the PQC standardization process has yet completed. Research continues to harden the PQC primitives against side-channel attacks, where the hardware that the algorithm runs is exploited to gather information that violates the security objectives (Ji et al. n.d.; Ma et al. 2022). While work is underway to guide the PQC transition (National Institute of Standards and Technology 2017a), the push to transition to PQC is driven by lengthy time requirements expected for the transition. This is especially true of vehicles and EVCI that are expected to have long lifespans..

The first step for an orderly PQC transition is to inventory applications of traditional public key cryptosystems (Department of Homeland Security n.d.). The purpose of this report is to identify traditional public key cryptosystem applications employed for EV charging. There are multiple

[1] Traditional public key cryptography is underpinned by how difficult it is to solve the mathematical problems of integer factorization and discrete logarithms over finite fields and elliptic curves. These cryptosystems will be vulnerable to Shor’s algorithm, a quantum algorithm that exponentially speeds factoring compared to traditional computing.

standards used across the EV charging ecosystem; for the sake of this paper, only communication between the EV, the EV charging station (EVCS), and charging station management system (CSMS) are considered. Based on requirements established in the National Electric Vehicle Infrastructure (NEVI) formula program (23 CFR 680), these protocols are Open Charge Point Interface (OCPI), Open Charge Point Protocol (OCPP), International Organization for Standardization (ISO) 15118-2, and ISO 15118-20.

By documenting how current EV charging ecosystems use traditional PKC, this paper enumerates potential consequences of leaving EVCI vulnerable to quantum computing. The most consequential threat is to digital signatures, which are approaches used to ensure the authenticity and integrity of messages and documents. If digital signatures can be efficiently attacked, cryptographic identity, such as X.509 certificates, becomes vulnerable and proving identity becomes susceptible to impersonation. At that junction, an attacker could exploit the system to cause power outages, system shutdown, fire, hardware damage, or even injury and loss of life. Section 5.0 provides a high-level overview of the threats to EVCI and Appendix D contains a detailed table of potential threats and their consequences.

This report is the first in a series examining the EV charging adoption of post-quantum cryptography. It is intended to assist the understanding of the landscape and support future decision making. Our focus remains squarely on the of EV charging and infrastructure with respect to PKC and complements the NIST SP 1800-38 *Migration to Post-Quantum Cryptography* (Newhouse et al. 2023). While the report is centered on infrastructure, there are implications to vehicles.

2.0 EV Infrastructure Primer

Within EVCI, there are three primary components: the EV, the EVCS, and the CSMS (a system used to centrally manage a network of chargers). In addition to the components, there are actors, such as charging network providers, charging station operators (CSOs), charger original equipment manufacturers (OEMs), and payment processing networks, all of which have roles and responsibilities to enable safe, secure, and resilient charging.

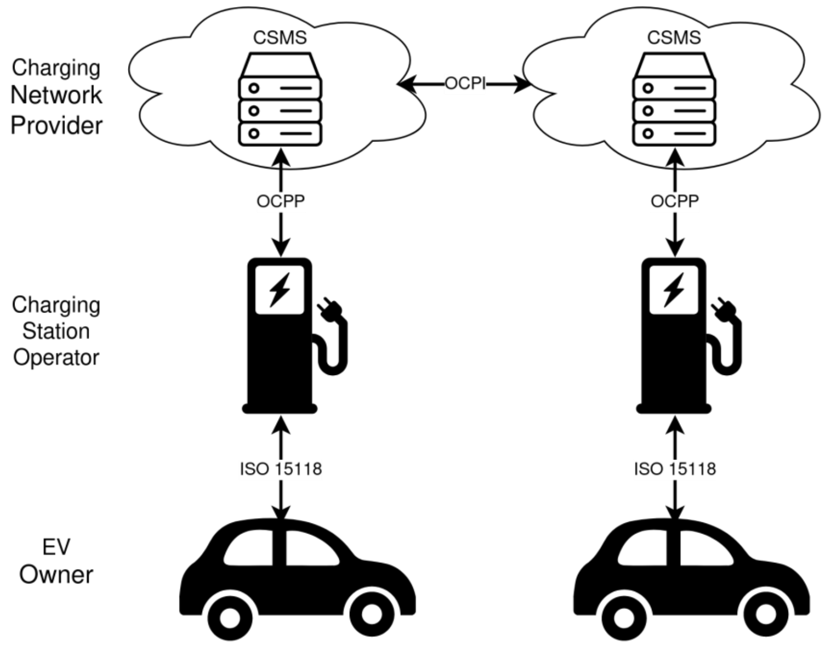

Figure 1 Charging Infrastructure Protocols

As illustrated in Figure 1, each relationship is associated with the protocols ISO 15118, OCPP, or OCPI. These protocols are also the ones established for use in the NEVI formula program. ISO 15118 is a suite of standards that govern the communication between the Electric Vehicle Communication Controller (EVCC), found in the car, and the Supply Equipment Communication Controller (SECC), which is located at the charger. ISO 15118 manages the charging processes, including starting and stopping charge sessions, authenticating devices (e.g., Plug and Charge [PnC], which allows automated authentication and billing process without any further interaction by the vehicle's driver beyond connecting the vehicle), and scheduling and initalizing configuration parameters of the session.

OCPP is an open protocol for EVCS and CSMS communications. OCPP supports functions such as device management, access control, smart charging, energy transactions, logging, metering, and security functions ("OCPP 2.0.1 Part 2 - Specification" 2020).

OCPI is an open protocol for communication among charging networks and can be used by applications to access charging information and services. It allows EV drivers to utilize various EV charging networks based on location, accessibility, and pricing as a product of automated roaming (Open Charge Point Interface 2021).

3.0 Public Key Cryptography

Cryptosystems can be generally categorized into three classes:

- Symmetric key encryption
- Asymmetric key encryption
- Hash functions

Symmetric key and asymmetric key encryption systems are the two primary techniques to encipher data. Symmetric encryption uses a single key to encrypt and decrypt data, while asymmetric encryption techniques employ two distinct, but related, keys to encrypt and decrypt data. Asymmetric encryption and PKC are typically synonymous as one of the keys—the public key—is distributed. Examples of popular symmetric key cryptosystems are Advanced Encryption Standard with Galois/Counter Mode (AES-GCM) and ChaCha20-Poly1305; asymmetric cryptosystems would be Elliptic-Curve Cryptography (ECC), Rivest–Shamir–Adleman (RSA), Diffie-Hellman, and Digital Signature Algorithm (DSA). A hash function, for example Secure Hash Algorithm (SHA), is a one-way function that maps arbitrarily large inputs to small, fixed-sized outputs called hash values. The function is constructed so that the reverse—constructing the preimage from hash—is hard.

While quantum computing can speed up brute force key searches (Grover 1996), little advantage is gained when attacking AES (National Institute of Standards and Technology 2017b), other similarly constructed symmetric key cryptosystems, and standard hash functions. However, a sufficiently sized CRQC will be able to efficiently perform the mathematical operations to attack the foundations of traditional asymmetric key cryptosystems. The Shor's algorithm, a quantum algorithm, promises to factor numbers and compute the discrete logarithm in polynomial time (Shor 1997), a sub-exponential speedup over traditional computing approaches. This means that while inventorying all cryptosystems, it is important to safeguard against future vulnerabilities, and taking stock of public key cryptosystems is immediately critical and should be performed in the near-term.

Public key cryptosystems generally serve three purposes, notionally described below: public key encryption, digital signatures, and key exchange. Section 4.0 gives a high-level overview of how public key cryptosystems are used in the EVCI, and Appendix A provides a more detailed inventory of public key cryptosystems by EV protocol.

3.1 Encryption

Encryption is a method of encrypting data with two distinct but corresponding keys. The keypair (pk, sk) comprises the public key pk, which is widely shared, and the mathematically related private key sk, which must not be disclosed. The function $\mathrm{enc}(m, pk) \rightarrow c$ encrypts a message m under pk, outputting a ciphertext c. The reciprocal function $\mathrm{dec}(c, sk) \rightarrow m$ decrypts a c under sk. A critical property of public key encryption is *indistinguishability*: ciphertexts obtained from encrypting any message must look identical to those from encrypting any other message. This implies that the encryption is a randomized algorithm where no more than message length is learned.

3.2 Digital Signature

A digital signature is a virtual fingerprint on a message or document that is used to provide authenticity protection, integrity protection, and non-repudiation. A signature ensures origin authentication—verifying the source of the information—and that the message was unaltered in transit. Moreover, non-repudiation ensures that the signer cannot deny the authenticity. At its most basic form, when digitally signing a message, the message is encrypted using the sender's private key or certificate, then is sent to the receiver who verifies the message is from the appropriate sender by decrypting using the sender's public key. In practice, it is more common to first hash the plaintext message and then encrypt the hashed message to create the digital signature.

By definition, a digital signature scheme comprises two routines, $\mathrm{sign}(\cdot)$ and $\mathrm{verify}(\cdot)$. The digital signature s of message m signed under sk is computed with $\mathrm{sign}\big(sk, \mathrm{hash}(m)\big) \to s$. The hash function $\mathrm{hash}(\cdot)$ transforms an arbitrary-length input to a n-bit, fixed-length hash value. To verify the authenticity of a message, $\mathrm{verify}(pk, \mathrm{hash}(m'), s) \to 1$ if and only if $\mathrm{hash}(m') = \mathrm{hash}(m)$ and (pk, sk) are a keypair. Otherwise, the function produces 0. The digital signature scheme is constructed to ensure that the signatures are not forgeable, alterable, or reusable.

3.3 Key Exchange

A key exchange mechanism (KEM) is used to securely exchange session keys and other information fundamental to establishing secure communication channels. The mechanism is the means for two different parties to generate the same secret even if someone was eavesdropping on the communication. Key agreement, such as Diffie-Hellman key exchange, can be rephrased in context of a KEM.

As public key encryption is more resource intensive when compared against symmetric cryptography, key exchange is typically employed to establish a shared symmetric key k between two parties using open communications without posing risk to the confidentiality of k. One approach to key exchange is key encapsulation. In key-encapsulation scheme $KEM = (\mathrm{keygen},\mathrm{encaps},\mathrm{decaps})$, $\mathrm{keygen}()$ generates a keypair (pk, sk). The encapsulation algorithm takes the public key pk, $\mathrm{encaps}(pk)$, and produces a ciphertext c and a session k. Finally, the decapsulation algorithm, decaps, takes a secret key sk and a ciphertext c, and outputs key k. At this junction, both parties have the same k.

The algorithm as presented is not an authenticated key exchange, which is an algorithm that authenticates the identities of the participating parties. By reusing blocks with the parties' public keys, an authenticated key exchange can be constructed in a straightforward manner. This KEM construction offers perfect forward secrecy as a (pk, sk) is generated at the beginning of each session, preventing the compromise of long-term secrets from affecting the confidentiality of prior communications.

4.0 Public Key Applications in EV Charging Infrastructure

EV charging uses asymmetric key algorithms to create and verify digital signatures and agree on symmetric session keys. Digital signatures are critical use cases as compromised signatures lead to impersonation and forgery, which undermine the trust in the infrastructure. Key exchange counters the ability to record communications and decrypt at a later junction. The information being communicated is generally ephemeral, which means confidentiality is less of a priority than authenticity and integrity.

PKIs play a critical role in every aspect of confidentiality, integrity, availability, authenticity, and non-repudiation—all important cybersecurity concepts for securing systems. In EV charging there are three roles that employ PKI in ISO 15118 (“ISO 15118-20:2022” 2022): the CSO at the SECC, the Charging Network Provider (CNP) at the EVCC, and the car manufacturer (OEM) at the EVCC. In practice this looks like:

- *Certificates*: A certificate is an electronic identity document that cryptographically links a public key to an identity. The binding of the public key and name (the certificate’s subject) is established by the issuer digitally signing the document. Parties can verify the document by verifying the issuer’s signature. The issuer has a certificate that establishes their identity, which can be verified. The process repeats across the trust chain until a trust anchor, an authoritative entity for which trust is assumed and not derived, is encountered. In the context of X.509 certificates, a trust anchor is a root certificate in which both the identity and the issuer are the same. An in-depth explanation of the certificate authority (CA) hierarchies can be found in 5.0Appendix B.
- *Secure Communications:* The EVCI widely adopts Transport Layer Security (TLS), a cryptographic protocol designed to protect communication from eavesdropping and tampering. Digital signatures are used to identify and authenticate peers, ensuring that they are the intended recipient. KEMs share secret session keys used to bulk encrypt messages and assure authenticity. For more information on how TLS works see 5.0Appendix C.
- *Certificate Installation and Verification*: Clients and servers call on key generation and CA signing for establishing a replacement certificate when one has expired. These certificates are what is used to secure the ecosystem (i.e., enabling TLS, allowing billing messages to be signed, ensuring firmware comes from the approved source). When requesting a new certificate (at the point of certificate expiration), the entire message is signed to prevent an external actor from receiving a certificate. Once the CA generates a new certificate, private key, and public key, the CA signs the new certificate and keys to protect the integrity of the new certificate. For each of these use cases, when a certificate is sent, the certificate is verified up its chain to the root CA.
- *Signed Messages:* Messages are digitally signed to ensure authenticity and non-repudiation.
  - *Load control*: Charge monitoring mechanisms include the verification of certificates encapsulated within a load control message request. The EV certificate is applied to voltage control Extensible Markup Language (XML) message requests. The authorization of these certificates allows the EVCS to have proper load assurance and stability.
  - *Billing (including tariffs and metering status)*: Billing messages utilize a certificate private key to generate a digital signature and to ensure data secrecy and validity.

  - *Signing for sequence target setting and charge scheduling (optional)*: Configuration and scheduling mechanisms may apply digital signing for authenticity protection of charge values. Manipulation of these values could pose serious harm, but signing is optional.
  - *Charge Sessions*: Employ various points of certificate and signature verification to ensure the identity, authentication, and authorization of the EV/EVCS.
- *Firmware, Software, and Settings Verification* ("ISO 15118-20:2022" 2022): Secure update mechanisms utilize firmware digital signatures and associated verification support to ensure the integrity and authenticity of the downloaded code (Cooper et al. 2018). In addition, an EVCS may employ secure boot where digital-signed firmware is verified before loading (Regenscheid 2018).

The identified PKC applications are specific to the EV infrastructure protocols ISO 15518, OCPP, and OCPI, along with applications tied to the charger. Further details are supplied in the appendices regarding specific applications. There will be other instances of PKC found in systems supporting infrastructure such as enterprise and cloud computing, credit card payment, and authenticators.

5.0 Consequences

Because the EVCI relies on cryptography for safe transactions and security, the following information could be made available by any quantum computer user: private keys, digital signatures, messages, credit card payment information, online transactions, and other sensitive data. If key encapsulation fails, an actor can obtain the session keys, allowing them to control the communication channel and directly affect the functionality of the connected components.

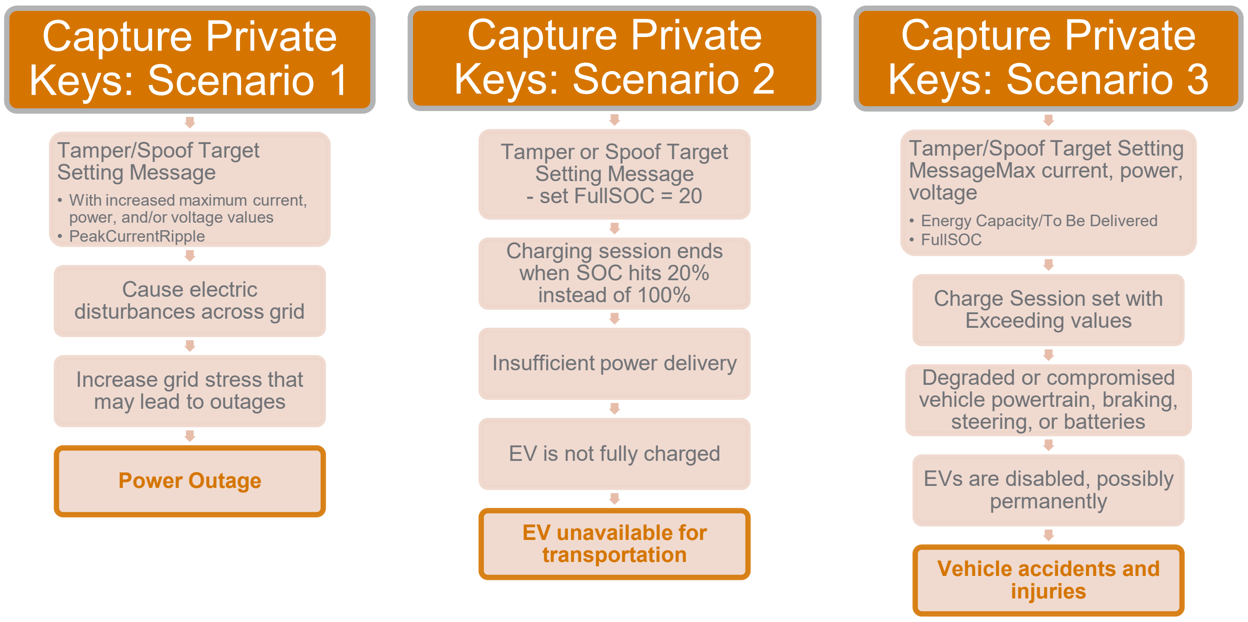
Capture Private Keys: Scenario 1
Tamper/Spoof Target Setting Message
• With increased maximum current, power, and/or voltage values
• PeakCurrentRipple
Cause electric disturbances across grid
Increase grid stress that may lead to outages
Power Outage
Capture Private Keys: Scenario 2
Tamper or Spoof Target Setting Message - set FullSOC = 20
Charging session ends when SOC hits 20% instead of 100%
Insufficient power delivery
EV is not fully charged
EV unavailable for transportation
Capture Private Keys: Scenario 3
Tamper/Spoof Target Setting MessageMax current, power, voltage
• Energy Capacity/To Be Delivered
• FullSOC
Charge Session set with Exceeding values
Degraded or compromised vehicle powertrain, braking, steering, or batteries
EVs are disabled, possibly permanently
Vehicle accidents and injuries

Figure 2 Consequences of Compromising Private Keys

For example, an attacker could maliciously change the infrastructure by integrating malware through the firmware update functionality defined in OCPP. Firmware update messages within EVCI are sent through TLS protocols; if the session keys were able to be captured (i.e., using Shor's algorithm), it would allow an actor with a quantum computer to acquire the tools to break the encryption and inject a faulty update request between the EVCS and CSMS. If the attacker included an illegitimate URL location to the update request, the EVCS/electric vehicle supply equipment (EVSE) would inherently reach out to this location, downloading malicious or incompatible code. Installation of this code could alter local configuration parameters, inducing several High Consequence Events (HCEs) outlined by Idaho National Laboratory: power outages, injury or loss of life, system shutdown, hardware damage, fire, and/or decreased stability/reliability of the grid.[1]

During certificate installation, an attacker could exploit a certificate by deciphering the certificate's private key (i.e., using Shor's algorithm) while the new certificate and private key are sent over TLS via a KEM (SECP256r1). By capturing one certificate's private key, an attacker can also capture certificate information for any new certificates for the OEM and contract certificates, thereby allowing the malicious actor to monitor traffic sent using various certificates in the certificate authority hierarchy, capture private information (such as payment details), and make configuration changes (i.e., changing max voltage). Based on ISO 15118-2, other

[1] Carlson, Richard, et al. 2021. Consequence-driven Cybersecurity for High-Power Electric Vehicle Charging Infrastructure. Energies vol. 14.

opportunities to invalidate the EVCI or cause physical harm would be to change the max current limit, max power limit, energy capacity, current regulation tolerance, peak current ripple, and energy to be delivered.

Within an update firmware request, sent from the charge network operator (CNO) to the EVCS over OCPP, the firmware type variable is signed. This class includes the Uniform Resource Identifier (URI) of the firmware, retrieval and installation date and time, the signing certificate, and the base64 encoded firmware signature. If a malicious actor were to obtain the signing certificate's private key, they would be able to successfully execute a man-in-the-middle (MITM) attack. The actor could swap the firmware type of the original request with a new digitally signed firmware type generated from the actor that contains the altered URI. Another approach would be to replace or edit only the location value of the firmware type rather than generating a completely new class. The figure below depicts this approach in further detail. These URIs could send the EVCS to bugged websites or to download malicious code that could disrupt vital functionality (i.e., software errors, altered configuration settings, backdoor).

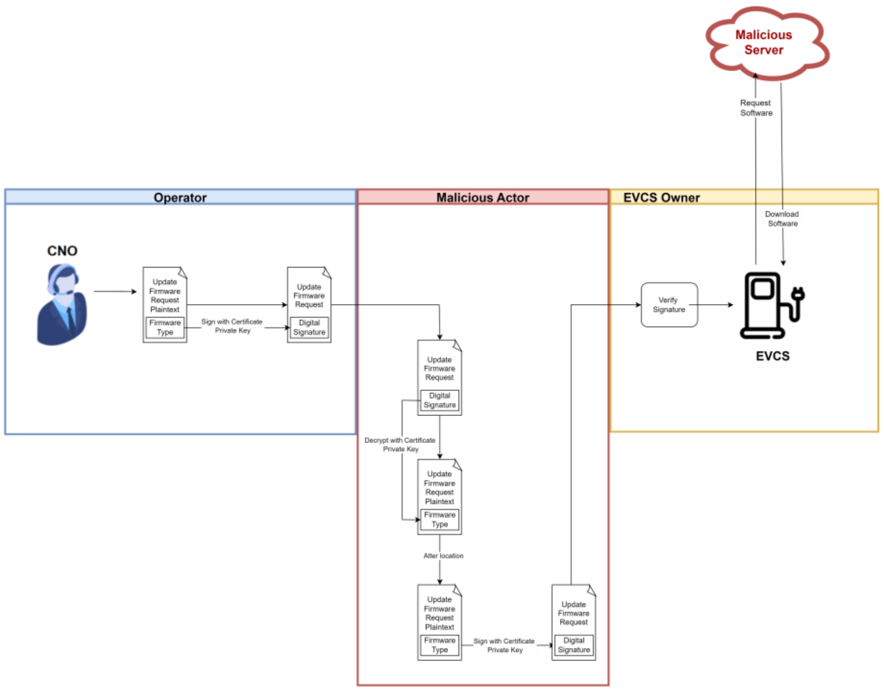

Figure 3 MITM URI Alteration Example

More details on threats to the EVCI can be found in Appendix D.

References

Cooper, David, Andrew Regenscheid, Murugiah Souppaya, Christopher Bean, Mike Boyle, Dorothy Cooley, and Michael Jenkins. 2018. “Security Considerations for Code Signing.” NIST Cybersecurity White Paper.

Department of Homeland Security. n.d. “Post-Quantum Cryptography.” Department of Homeland Security. Accessed June 21, 2023. https://www.dhs.gov/quantum.

Gidney, Craig, and Martin Ekerå. 2021. “How to Factor 2048 Bit RSA Integers in 8 Hours Using 20 Million Noisy Qubits.” *Quantum* 5 (April): 433. https://doi.org/10.22331/q-2021-04-15-433.

Grover, Lov K. 1996. “A Fast Quantum Mechanical Algorithm for Database Search.” In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–19. STOC ’96. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/237814.237866.

Grumbling, Emily, and Mark Horowitz, eds. 2019. *Quantum Computing: Progress and Prospects*. Washington, D.C.: National Academies Press. https://doi.org/10.17226/25196.

“ISO 15118-2 Road Vehicles — Vehicle-to-Grid Communication Interface — Part 2: Network and Application Protocol Requirements.” 2014. ISO. Geneva, Switzerland: International Organization for Standardization.

“ISO 15118-20 Road Vehicles — Vehicle to Grid Communication Interface — Part 20: 2nd Generation Network Layer and Application Layer Requirements.” 2022. ISO. Vernier, Geneva: International Electrotechnical Commission.

“ISO 15118-20:2022.” 2022. International Organization for Standardization. https://www.iso.org/standard/77845.html.

Ji, Yanning, Ruize Wang, Kalle Ngo, Elena Dubrova, and Linus Backlund. n.d. “A Side-Channel Attack on a Hardware Implementation of CRYSTALS-Kyber.” Accessed June 23, 2023. https://eprint.iacr.org/undefined/undefined.

Ma, Haocheng, Shijian Pan, Ya Gao, Jiaji He, Yiqiang Zhao, and Yier Jin. 2022. “Vulnerable PQC against Side Channel Analysis - A Case Study on Kyber.” In *2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 1–6. https://doi.org/10.1109/AsianHOST56390.2022.10022165.

Mosca, Michelle, and Marco Piana. 2022. “Quantum Threat Timeline Report 2022.” Global Risk Institute.

National Institute of Standards and Technology. 2017a. “Post-Quantum Cryptography FAQs.” Computer Security Resource Center. January 3, 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs.

———. 2017b. “Post-Quantum Cryptography FAQs.” NIST Computer Security Resource Center. January 3, 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs.

Newhouse, William, Murugiah Souppaya, William Barker, and Christopher Brown. 2023. “Migration to Post-Quantum Cryptography: Preparation for Considering the Implementation and Adoption of Quantum Safe Cryptography.” NIST Special Publication (SP) 1800-38 (Draft). National Institute of Standards and Technology. https://csrc.nist.gov/pubs/sp/1800/38/iprd.

“OCPP 2.0.1 Part 2 - Specification.” 2020. OCPP. Open Charge Alliance.

Open Charge Alliance. n.d. “OCPP 2.0.1.” Accessed August 11, 2023. https://www.openchargealliance.org/downloads/.

Open Charge Point Interface. 2021. “OCPI 2.2.1.” EVRoaming. https://evroaming.org/app/uploads/2021/11/OCPI-2.2.1.pdf.

Regenscheid, Andrew. 2018. “Platform Firmware Resiliency Guidelines.” 800–193. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-193.

Rep. Khanna, Ro [D-CA-17. 2022. “H.R.7535 - 117th Congress (2021-2022): Quantum Computing Cybersecurity Preparedness Act.” Legislation. Congress.Gov. 2022-04-18. December 21, 2022. https://www.congress.gov/bill/117th-congress/house-bill/7535.

Shor, Peter W. 1997. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.” *SIAM Journal on Computing* 26 (5): 1484–1509. https://doi.org/10.1137/S0097539795293172.

The White House. 2021. “Executive Order on Improving the Nation’s Cybersecurity.” The White House. May 12, 2021. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

US Department of Energy. 2023. “2023 Grid Modernization Lab Call.” April 25, 2023. https://www.energy.gov/gmi/2023-grid-modernization-lab-call.

Webber, Mark, Vincent Elfving, Sebastian Weidt, and Winfried K. Hensinger. 2021. “The Impact of Hardware Specifications on Reaching Quantum Advantage in the Fault Tolerant Regime.” arXiv. http://arxiv.org/abs/2108.12371.

Appendix A – EV Charging Protocol Applications of Public Key Cryptography

A.1.1 ISO 15118

International Organization for Standardization (ISO) 15118, a suite of standards that govern electric vehicle (EV)-to-charger communications, has defined the requirements for authentication, authorization, load control, and billing for EV and EV supply equipment (EVSE) communication (specifically using signatures, x509 certificates, and the public key infrastructure [PKI] model). It defines what Transport Layer Security (TLS) establishment looks like between the EV and electric vehicle charging station (EVCS) and various other security requirements, including supported versions and backward compatibility. ISO 15118-2 utilizes TLS version 1.2 while ISO 15118-20 utilizes TLS version 1.3. ISO 15118-20 incorporates, by reference, ISO 15118-2, and it is assumed that any implementation of ISO 15118-20 will implement any of the requirements of 15118-2 unless otherwise stated.

A.1.1.1 Certificates & Key Management

Supply Equipment Communication Controller (SECC) certificates are used in the TLS connection establishment for the SECC to authenticate to the EV Communication Controller (EVCC). ISO 15118-20 goes further and mandates mutual authentication (mTLS), where the EVCC is also assigned a certificate to prove its identity to the SECC. The contract certificates are used in the application layer and to authenticate against SECC or the secondary actor (SA). Vehicle to grid (V2G) root certificate authority (CA) and one or more intermediate sub-CA certificates are used to certify the SECC certificates and other contract certificates. The original equipment manufacturer (OEM) root CA certificate and vehicle certificates are used in TLS and for the SECC to authenticate the EVCC. Lastly, the OEM provisioning certificates that stem from the OEM root CA certificates may be used for installing and updating the contract certificates.

ISO 15118-2 specifies the follow requirements that govern certificates:

- [V2G2-005] Each V2G Entity shall support Hash-operation SHA-256 (signature process) according to NIST FIPS PUB 180-4 (for Part 1 Use Case Element ID: F1).
- [V2G2-006] For each V2G Entity the signature operation shall be elliptic-curve cryptography (ECC)-based using elliptic curves (secp256r1[SECG notation]) with signature algorithm elliptic-curve Digital Signature Algorithm (ECDSA) (for Part 1 Use Case Element ID: F1).
- [V2G2-007] The key length for ECC-based asymmetric cryptography each V2G Entity uses shall be 256 bit.
- [V2G2-009] The path length constraint of the PKI certificate tree shall be limited to 3.
- [V2G2-010] The size of a certificate in Distributed Energy Resource (DER) encoded form shall be not bigger than 800 Bytes. For transmission, all certificates shall be DER encoded.
- [Annex F – Certificate Profiles] specifies X.509 certificate requirements.

ISO 15118-20 specifies:

- [V2G20-2673] Each V2G entity shall support Hash-operation SHA-512 (signature process) according to NIST FIPS PUB 180-4 (for ISO 15118-1, use case element ID: F1).

- [V2G20-2318] Each V2G entity shall support `SHAKE256` according to NIST FIPS PUB 202 (for ISO 15118-1, use case element ID: F1).
- [V2G20-2674] Each V2G entity shall support signature operations using ECC-based elliptic curve (secp521r1[SECG notation]) with signature algorithm ECDSA (for ISO 15118-1, use case element ID: F1).
- [V2G20-2319] Each V2G entity shall additionally support signature operations with ECC algorithm `Ed448` (for ISO 15118-1, use case element ID: F1), i.e., signature algorithm Edwards-curve Digital Signature Algorithm (EdDSA) using elliptic curve `Curve448` (or `Curve448-Goldilocks)` in Edwards form, see IETF RFC 7748 and IETF RFC 8032.
- [Annex B – Certificate Profiles] specifies X.509 certificate requirements.

A.1.1.2 TLS Connections

ISO 15118-2 and ISO 15118-20 specify TLS for secure communications. As noted above, 15118-2 and 15118-20 specify TLS 1.2 and TLS 1.3, respectively (see Appendix C for more information about TLS and how it works). ISO 15118-20 has backward compatibility with ISO 15118-2 by offering TLS 1.2. The EVCC provides the necessary information for the SECC to respond with the proper TLS version using the supported protocol versions provided in the TLS `ClientHello` message, the first message sent in the TLS handshake. Importantly, the maximum 15118 protocol is selected based on which TLS version is negotiated. Beyond TLS 1.3, ISO 15118-20 introduced other significant changes to its TLS practices: TLS is mandatory for all use cases and identity mechanisms—closing a 15118-2 security loophole where TLS is optional—and mutual authentication (abbreviated mTLS), in which the EVCC and SECC in a connection, authenticate each other.

With respect to TLS authentication and certificate verification, EVCC authenticates SECC using the SECC certificate. This is achieved by SECC having a private key corresponding to the SECC certificate and EVCC having a vehicle to grid (V2G) root certificate and verifying the certificate chain from the V2G root certificate to the SECC certificate. The validity check of the sub-CA certificate in the certificate chain is performed via the Online Certificate Status Protocol (OCSP) response received during the TLS handshake, conveyed using the X.509 `CertificateStatus` extension (also referred to as OCSP stapled responses; for details, refer to IETF RFC 6066). This means the EVCC does not need network access to query the OCSP responder for certificate status. Mechanisms to revoke SECC certificates are not required but they must be short term certificates.

As the SECC is not aware of which V2G root certificate to use (due to multiple valid V2G root certificates being available worldwide), it is necessary for EVCC to provide a list of V2G root certificates that the EVCC possesses during the TLS handshake. The `CertificateAuthorities` extension is used for this purpose. The SECC provides a certificate chain based on the list of V2G root certificate authorities known by the EVCC. The chain also comprises a certificate of each sub-CA.

The EVCC-SECC communication is encrypted using two symmetric keys—the session keys—that were negotiated during the TLS handshake. The symmetric encryption performance is typically better than asymmetric cryptography. TLS ensures communications will not be eavesdropped on, and when combined with the tampering protections, the communication cannot be altered, nor can the session be hijacked.

ISO 15118-2 specifications that govern TLS are:

- [V2G2-067] For the considered use cases, unilateral authentication with TLS version 1.2 according to IETF RFC 5246 with extensions according to IETF RFC 6066 shall be supported by each V2G Entity. The EVCC authenticates the SECC by verifying the SECC Certificate (chain) provided from the SECC to the EVCC.
- [Table 3 — TLS implementation / support for External Identification Means (EIM) Identification Modes] specifies when TLS is used. In public environments, the SECC must provide the option to use TLS. The EVCC decides whether to utilize it or not.
- [Table 6 — TLS authentication] defines TLS authentication requirements.
- [Table 7 — Supported cipher suites] specifies two TLS 1.2 cipher suites, `TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256` and `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`. The first cipher suite utilizes elliptic curve Diffie-Hellman key agreement (ECDH) and Elliptic-Curve Digital Signature Algorithm (ECDSA). The second specifies an ephemeral key agreement (ECDHE), a key agreement variant in which public keys are generated for each session.
- [V2G2-602] The SECC shall support all cipher suites defined in Table 7, if TLS is used.
- [V2G2-603] The EVCC shall support at least one cipher suite as listed in Table 7, if TLS is used.

For 15118-20:

- [NOTE 1, p. 52] In case of ISO 15118-20 communication, TLS 1.3 is used to secure the communication between EVCC and SECC. To secure VAS[1], TLS 1.3 can also be used.
- [V2G20-1264] For the considered use cases mutual authentication with TLS version 1.3 according to IETF RFC 8446 shall be supported by each V2G entity.
- [V2G20-1521] TLS versions higher than 1.3 may also be supported.
- [V2G20-2359] The EVCC or SECC may support TLS version 1.2 as specified in ISO 15118-2 for backward compatibility.
- [Table 5 — Supported communication protocols] maps the corresponding relationship between TLS version and 15118 protocol version.
- [Table 7 — Supported named groups] specifies the default TLS 1.3 groups, `secp521r1` and `x448`, that can be used for key agreement.
- [V2G20-1634] The SECC shall support all named groups defined in Table 7.
- [V2G20-1637] The EVCC shall support all named groups defined in Table 7.

[1] Value Added Service

A.1.1.3 Signing & Message Security

Table 1 Signed XML Messages

XML Message	Certificate Private Key Used	Signing Component	Verification Component
Authorization Request	Contract certificate	EVCC	SECC
Metering Receipt Req (“ISO 15118-2 Road Vehicles — Vehicle-to-Grid Communication Interface — Part 2: Network and Application Protocol Requirements” 2014) Metering Confirmation Request (“ISO 15118-20 Road Vehicles — Vehicle to Grid Communication Interface — Part 20: 2nd Generation Network Layer and Application Layer Requirements” 2022)	Contract certificate	EVCC	SECC
Charge Parameter Discovery Response	Charge Network Provider (CNP) sub-CA 2	CNP	EVCC

Below are direct quotes from ISO 15118-2 and 15118-20 explaining how messages are secured:

> *Security on the Application Layer is provided using signature and encryption of messages. Information targeted for SA[1] services is exchanged using XML[2] data structures and is consequently protected end-to-end using XML Security. In general, two pairs of Security mechanisms are supported:*
>
> - *— Authenticity and Integrity: Signature generation → Signature verification; XML based signature is applied. The entity, which creates the XML message, signs certain or all fields of an XML message. The receiver verifies the signature.*
> - *— Confidentiality: Encryption → Decryption; Asymmetric encryption is applied. The entity, which creates the message, encrypts a single binary field of the XML message. The receiver decrypts that binary field*. (“ISO 15118-20 Road Vehicles — Vehicle to Grid Communication

[1] Secondary Actor
[2] Extensible Markup Language

Interface — Part 20: 2nd Generation Network Layer and Application Layer Requirements" 2022)

XML Signatures are a W3C[1] recommendation that addresses the authenticity requirements of some data fragments (e.g., metering information) of the XML-based V2G. XML Signatures define a mechanism by which messages and message parts can be digitally signed to provide integrity, to ensure that the data is not tampered with and authentication, to verify the identity of the message producer. For protecting confidentiality, a hybrid encryption scheme based on the Diffie-Hellman-Protocol is applied. XML Signatures as defined in W3C XML Signature Syntax and Processing Version 1.1 can be applied to arbitrary digital content (data objects) in the same way as digital signatures are calculated. When applying a digital signature to data objects, the data objects are first digested (hashed), and the result is then signed using an asymmetric algorithm like RSA or ECDSA. In the case of XML, the digest is placed in an XML element, together with additional information. This element is then hashed and cryptographically signed. This standard uses detached XML Signatures according to W3C XML Signature Syntax and Processing Version 1.1. That means the signature and data can be in separate (externally detached) or in the same XML document (internally detached) as sibling elements. The signature may comprise only a part of the XML document referenced by an ObjectID. ("ISO 15118-2 Road Vehicles — Vehicle-to-Grid Communication Interface — Part 2: Network and Application Protocol Requirements" 2014)

XML signature is chosen to protect billing relevant information between EVCC, SECC and/or SA. Moreover, binary encryption provides a confidentiality protected way for private key provisioning to the EVCC without having intermediaries access this private key. Both approaches require asymmetric key material. The credentials for EVCC signing are provided by the Contract Certificate and credentials for EVCC receiving encrypted data are provided by an ECDH key exchange as described in Annex G. Contract Certificate is bound to a EMAID[2] and used in XML signature to authorize the vehicle for charging. The Contract Certificate can be verified even if the SECC is offline. ("ISO 15118-2 Road Vehicles — Vehicle-to-Grid Communication Interface — Part 2: Network and Application Protocol Requirements" 2014)

Signed meter reading approval from the vehicle used for online and semi-online connections are common pieces of information exchanged between EVCC and SECC. The meter readings from the SECC are sent via the TLS protected tunnel. They may include the signature of the electricity meter providing source authentication to protect them additionally. The vehicle in turn signs the meter readings to provide a base for the billing process if local regulations permit it. This approach saves the meter reading signatures on the SECC side. The meter readings are cumulated so that the latest signed meter reading is the base for the billing. Here XML Signatures are used. They ensure integrity protection with the possibility for all intermediate and participating entities to rely on this information. ("ISO 15118-2 Road Vehicles — Vehicle-to-Grid Communication Interface — Part 2: Network and Application Protocol Requirements" 2014)

[1] World Wide Web Consortium
[2] E-Mobility Account Identifier

One differentiation between 15118-2 and 15118-20 is the `CertificateChainType` element, used by the Plug and Charge (PnC) identity mechanism. Both 15118-2 and 15118-20 define the `genChallenge` field to be exactly 128 bits long (V2G2-825 and V2G20-697, respectively); however, 15118-20 enforces that a certificate must be 1,600 characters or less, be base64 encoded, and have up to three sub-certificates (see V2G_CI_CommonMessages.xsd). ISO 15118-2 states the `CertificateChainType` must be 800 characters or less, be base64 encoded, and have up to four sub-certificates (see V2G_CI_MsgDataTypes.xsd). Due to the requirement of the base64 encoding, the raw message (before encoding) must be less than 1,200 (for 15118-20) and 600 (for 15118-2), due to base64 encoding increasing the size of the file by about 33 percent.

A.1.2 OCPI

Version 2.2.1 is the current official version of Open Charge Point Interface (OCPI), but the EVRoaming Foundation is in the process of developing version 3.0. Version 2.2.1 has very little defined security requirements or guidelines, but the EVRoaming Foundation is planning on introducing more in version 3.0.

OCPI in either version’s documentation does not specify certain versions of TLS or any specific algorithms for their signed data or certificates.

A.1.2.1 v2.2.1

OCPI 2.2.1’s only use of public key cryptography (PKC) is their signed meter values in the Charge Detail Records and secure communication using authentication tokens and Secure Sockets Layer (SSL)—specifically, their use of certificates.

A.1.2.2 v3.0

While OCPI 3.0 has not been finalized and released, the business use case document defines additional security measures not included in version 2.2.1. Specifically:

- OCPI will directly handle the ISO 15118 contract handling (including all the necessary security requirements; see Appendix A.1.1 for more details). OCPI will implement a local form of authorization utilizing a pre-established contract between the EV drivers and Mobile Service Providers (MSPs). This may be radio frequency identification tokens, a list of tokens sent to the Charge Point Operator (CPO) as a whitelist, or real-time authorization. Version 3.0 clarifies that, “Certificate revocation is not part of OCPI, OCSP can be used for this” and “at this time Root Certificates retrieval/installation is not seen as something where OCPI plays a role.”
- OCPI implementers must be able to set up and maintain a secure connection between two OCPI platforms.
- OCPI implementers must be able to validate that the data received is not altered and is authenticated based on the signature of the received data.

A.1.3 OCPP 2.0.1

The Open Charge Point Protocol (OCPP) 2.0.1 specification edition 2 document states, “No application layer security measures are included. Based on these considerations, OCPP security is based on TLS and public key cryptography using X.509 certificates. Because the

CSMS[1] usually acts as the server, different users or role-based access control on the CS[2] are not implemented in this standard. To mitigate this, it is recommended to implement access control on the CSMS." The document also clarifies "In some cases (e.g. lab installations, test setups, etc.) one might prefer to use OCPP 2.0.1 without implementing security. While this is possible, it is NOT considered a valid OCPP 2.0.1 implementation."

A.1.3.1 Certificates and Key Management

OCPI 2.0.1 does not define which keys should be used for digital signing, and it notes that the CSMS certificate and CS certificate (used to set up secure TLS connections) may be used for signing but explains that depending on the use case, there would be better keys to use.

Due to a local controller being at risk of having its certificates stolen (due to a hardware attack), the protocol enforces the CSMS certificate on the local controller to only communicate with the attached CS(s) and not with any other CS in the infrastructure.

A.1.3.2 TLS Connections

Secure communication in OCPP consists of three security profiles:

(1) Unsecured Transport with Basic Authentication

(2) TLS with Basic Authentication

(3) TLS with Client-Side Certificates (Mutual Authentication)

OCPP usually coordinates communication involving the CS and other secondary actors to charging, including the CSMS, CNP, and OEM. TLS establishment for the CSMS and CS are similar to the TLS explained in Appendix C. The CS acts as the client and is verified with its certificate (CS certificate). However, in some instances, the CS certificate can be the same as the SECC certificate. The CSMS acts as the server and is verified by its own certificate for authentication. The diagram below is from the OCPP protocol documentation (Open Charge Alliance n.d.).

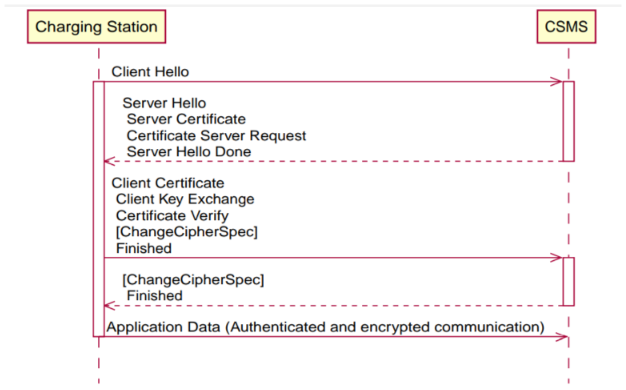

Figure 4 TLS Handshake with Client-Side Certificates in OCPP

[1] Charging Station Management System
[2] Charging Station

Note, data integrity for a connection relies on the underlying Transmission Control Protocol (TCP)/Internet Protocol (IP) TLS mechanisms.

OCPI 2.0.1 requires both the CS and CSMS to use TLS v1.2 or above. It also specifies that the CSMS and CS must support at least the following four cipher suites: `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`, `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, `TLS_RSA_WITH_AES_128_GCM_SHA256`, `TLS_RSA_WITH_AES_256_GCM_SHA384`.

A.1.3.3 Signing

Message signing is not required by OCPP 2.0.1. If the implementer chooses to sign messages, the following must be true:

- JSON Web Signatures (JWS) are used to sign and verify/extract encapsulated OCPP-JSON message. This restricts the implementer to only the supported algorithms for the signing and TLS connections, specifically:
  - ES256: ECDSA using P-256 and SHA-256
  - RS256: RSASSA-PKCS1-v1_5 using SHA-256
  - RS384: RSASSA-PKCS1-v1_5 using SHA-384.

While not enforced, it is recommended to sign critical commands/replies. It is expected that the following are signed:

Table 2 Signed Messages/Fields

Message	Field
UpdateFirmwareRequest	Signature
MeterValuesRequest	SignedMeterValue
TransactionEventRequest	SignedMeterValue
CertificateSignedRequest	certificateChain
SignCertificateRequest	csr

Below are two documented workflows for how OCPP uses the CS certificate, both diagrams are from the OCPP protocol documentation (Open Charge Alliance n.d.).

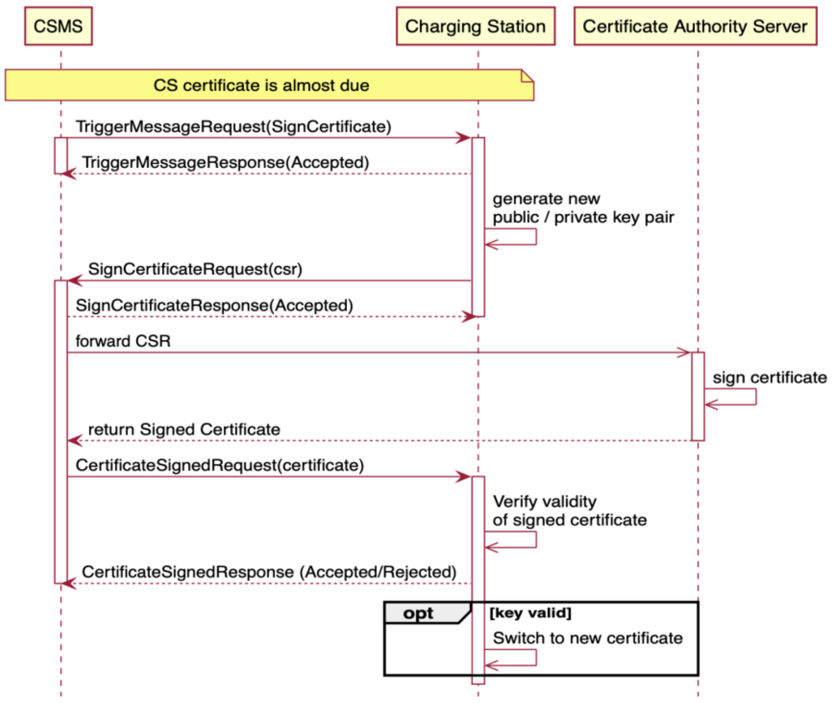
CSMS
Charging Station
Certificate Authority Server
CS certificate is almost due
TriggerMessageRequest(SignCertificate)
TriggerMessageResponse(Accepted)
generate new public / private key pair
SignCertificateRequest(csr)
SignCertificateResponse(Accepted)
forward CSR
sign certificate
return Signed Certificate
CertificateSignedRequest(certificate)
Verify validity of signed certificate
CertificateSignedResponse (Accepted/Rejected)
opt
[key valid]
Switch to new certificate

Figure 5 Update Charging Station Certificate Workflow in OCPP 2.0.1

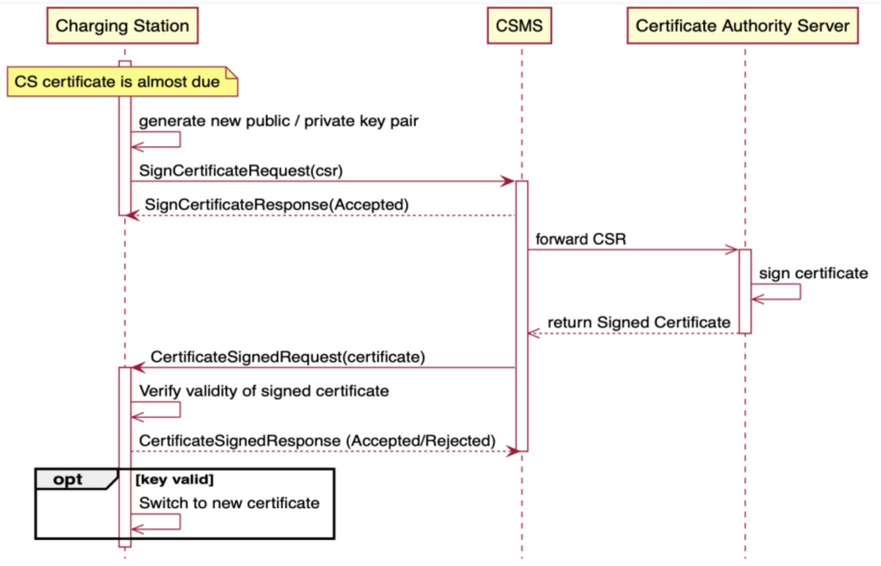
Charging Station
CSMS
Certificate Authority Server
CS certificate is almost due
generate new public / private key pair
SignCertificateRequest(csr)
SignCertificateResponse(Accepted)
forward CSR
sign certificate
return Signed Certificate
CertificateSignedRequest(certificate)
Verify validity of signed certificate
CertificateSignedResponse (Accepted/Rejected)
opt
[key valid]
Switch to new certificate

Figure 6 Update Charging Station Certificate Initiated by Charging Station Workflow in OCPP 2.0.1

Appendix B – Certificate Structure

Certificate	Hierarchy	Root CA	Details
CS certificate	CSO	V2G Root CA	CS – CSMS connection
CSMS certificate	CSO	V2G Root CA	CS – CSMS connection (authenticate CSMS)
SECC Certificate (V2GChargingStationCertificate)	CSO	V2G Root CA	EV – SECC(/CS) TLS connection
Contract Certificate (EVContractCertificate)	CNP	V2G Root CA/ CNP Root CA	Plug & Charge authentication
Vehicle Certificate	OEM	V2G Root CA/ OEM Root CA	EV – SECC TLS connection

Table 3 Certificate Structure

There are four primary certificate hierarchies used in the electric vehicle charging infrastructure (EVCI). The trust anchor would be from the OEM root CA of the component needing the firmware update (e.g., SECC and EVSE). The trust anchor of the firmware signing certificate is generally from the EVCS OEM root CA, although in some cases, an EV and EVCS are under the same OEM. In those instances, the vehicle certificate and firmware signing certificate can generate the same root CA (vehicle OEM).

Supplemental certificates include the cross certificates and OCSP signer certificates. A cross certificate is signed by a root CA of one organization (e.g., V2G root CA) but contains the public key of an existing CA certificate in another organization (e.g., vehicle sub-CA-1). For example, the vehicle certificate chain can contain a cross certificate where a vehicle sub-CA-1 is signed by the V2G root CA. These cross certificates can be within both a SECC and vehicle certificate chain. The OCSP signer certificate signs the OCSP response for the status of certificates including the SECC, contract, vehicle, OEM provisioning, and all of the corresponding sub-CA2 and sub-CA1 (if used). Below is a simplified CA structure diagram showing the different layers of the hierarchy.

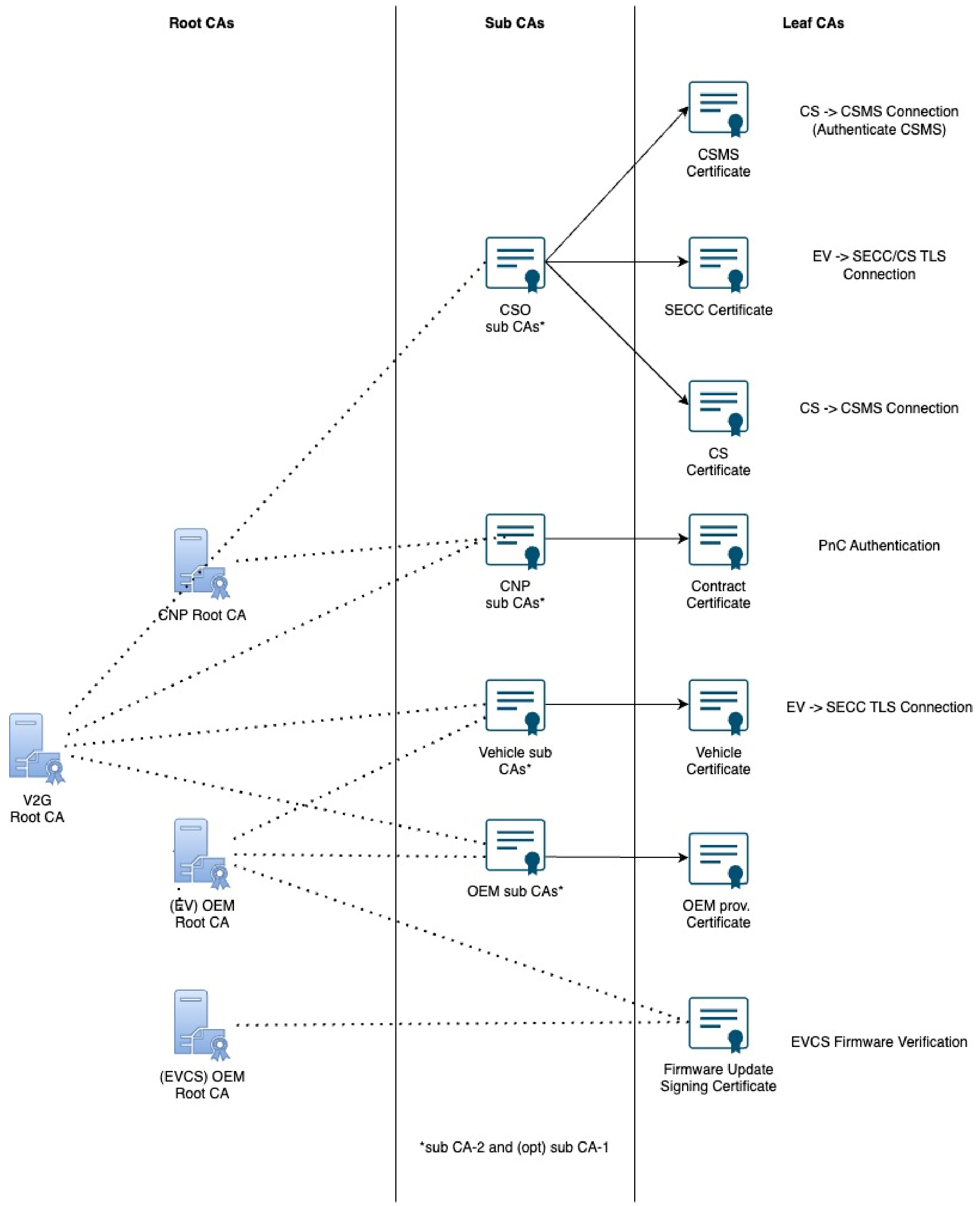

Figure 7 Simplified CA Structure

Appendix C – Transport Layer Security

Transport Layer Security (TLS) is a widely accepted way to secure communication protocols across different fields and applications, and it relies heavily on public key cryptography. In the EVCI, TLS is used to secure communication between components and users, for example, EVSE and SECC (15118), CS and secondary actors (OCPP), and between platforms (OCPI).

TLS involves both asymmetric and symmetric cryptosystems. A prominent feature of this protocol is the key establishment and exchange that is required for a communication session to be established. If an actor can obtain the session keys, they could control the communication channel and connected components. Existing asymmetric cryptosystems are deployed to encapsulate said session keys. The most common cryptosystems used as a key exchange scheme include RSA and ECC. In general, ECC schemes are used in EV charging systems for TLS. The security concept relies on the exponential time required to solve a discrete logarithm problem (ECC) or to factor a product of two large prime numbers (RSA). The time and money required to break it would be too great to warrant the reward. By the time one could decipher the key, the session would have already ended, making the information obsolete.

C.1 Cipher Suites (Symmetric Algorithms)

To encrypt and decrypt messages via symmetric algorithms, ISO 15118-2 and ISO 15118-20 specify different cipher suites. A cipher suite is a set of algorithms—key exchange, bulk encryption, key derivation—that TLS uses to secure communications. There are two cipher suites that are supported by the EVCC and SECC for ISO 15118-20; `AES-256-GCM-SHA384` and `CHACHA20-POLY1305-SHA256`. However, the previous version, ISO 15118-2, requires support for ECDH-ECDSA and ECDHE-ECDSA with `AES-128_CBC_SHA256`. `AES-256-GCM-SHA384` is the primary cipher suite used in 15118-20 where messages are hashed with SHA384 and then encrypted with AES-256-GCM.

Table 4 TLS Cipher Suites Indicated in ISO 15118-2 and 15118-20

Cipher Suite	Key Exchange	Key Derivation Function (KDF)[1] Hash Algorithm	Bulk Encryption Algorithm	Message Authentication Code
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	ECDH[2]	SHA256	AES-128-CBC	HMAC
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE	SHA256	AES-128-CBC	HMAC

[1] In TLS 1.2, the hash algorithm serves the pseudo random function TLS key derivation mechanism.
[2] IETF is in the process of deprecating key exchanges that do not offer practical forward secrecy. See https://datatracker.ietf.org/doc/draft-ietf-tls-deprecate-obsolete-kex/

Cipher Suite	Key Exchange	Key Derivation Function (KDF)[1] Hash Algorithm	Bulk Encryption Algorithm	Message Authentication Code
TLS_AES_256_GCM_SHA384	ECDHE	SHA384	AES-256-GCM	AEAD
TLS_CHACHA20_POLY1305_SHA256	ECDHE	SHA256	CHACHA20-POLY1305	AEAD

ISO 15118-2 and 15118-20 specify TLS versions TLS 1.2 and TLS 1.3, respectively. There were notable changes between TLS 1.2 and TLS 1.3. The two most significant changes are the support of: (1) authenticated encryption associated data (AEAD) and (2) practical forward secrecy (PFS). AEAD is a modern cryptographic primitive that simultaneously ensures confidentiality (message contents cannot be observed by third parties) and authenticity (message cannot be forged). PFS is when key agreement protocols guarantee confidentiality of previously encrypted messages even if private keys are exposed. TLS 1.3 dropped support for cipher suites that were not AEAD and PFS.

Appendix D – EV Charging Ecosystem Threat Model

Table 5 maps Idaho National Laboratory's High Consequence Events (HCEs) to cybersecurity threat categories. HCEs are high-priority outcomes that could be caused by cyber sabotage. The mapping is based on the STRIDE framework, an industry-based approach for threat modeling. STRIDE is an acronym for six threat categories: **S**poofing of identities, **T**ampering with messages and data, **R**epudiating an action, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege.

Table 5 High Consequence Events are Mapped to STRIDE Threat Categories.

Event Description	Systems attacked	Spoof	Tamper	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Power outage(s) due to sudden load shed from multiple Extreme Fast Chargers (XFCs).	CSMS, EVSE		X			X	X
Injury or loss of life due to electrocution, electrical shock, or burns from exposed conductors due to failed insulation of the XFC cable or connector.	CSMS, EVSE		X				
Power outage(s) due to sudden load shed or increase from on-site energy storage system manipulation.	CSMS, EVSE, DER, Utility		X			X	
(Wireless Power Transfer Only) Implanted medical device failure or injury caused by exposure of high electromagnetic field.	EVSE		X				
Damage to equipment within the feeder distribution area (transformers, switch gear, harmonics, overload capacitor bank, high reactive power).	CSMS, EVSE, UTILITIY		X				
The XFC and Distributed Energy Resource (DER) at the site are not able to provide grid services (e.g., curtailment) when needed, causing decreased stability/reliability of the grid.	CSMS, EVSE, DER, Utility, Aggregator	X	X			X	
System shutdown (XFC or charging site) due to a software error state.	CSMS, EVSE		X				
System shutdown due to network outage (Wi-Fi, cellular, or other communications outage).	CSMS, EVSE, eMSP, Aggregator		X			X	
Hardware damage to the charger over very long durations of elevated temperature.	EVSE		X				
(WPT Only) Induced voltage (high V/m) on vehicle components or electrical harnesses may damage harness or electrical components not associated with WPT system. Vehicle components that are not rated or shielded from high magnetic field levels may heat up.	EVSE		X				
Theft or alteration of personally identifiable information (PII) data transmitted between vehicle, XFC, EV driver, network operator, etc.	CSMS, EVSE, eMSP, Aggregator		X	X	X		
Vehicle fire due to vehicle battery overcharge	EVSE		X				
(WPT Only) Vehicle electrical component damage due to over-voltage condition of the vehicle side WPT components.	EVSE		X				
Hardware damage to the XFC(s).	EVSE		X				
Loss of system control/visibility due to communications interruption.	CSMS, EVSE		X			X	