November 2023

Kenneth Rohde
Jonathan Chugg
Jake Guidry
Barney Carlson
Cybersecurity Researchers

Access Capabilities for CCS (AcCCS)

Exposing Vehicle to Charger Communications

Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

Idaho National Laboratory

INL/CON-23-74780

What is the goal?

- Future Electric Vehicles (EVs) will support Plug-and-Charge
  - Vehicle stores your payment information
  - Vehicle stores your charge network credentials (e.g. Electrify America)
  - Vehicle authenticates you via PKI certificates

- Future EVs will support two-way power transfer
  - Vehicle to Home
  - Vehicle to Grid

- Vehicle charge ports and EVSE cordsets are typically not locked

- **We need a inexpensive and flexible communications test suite for CCS…**

Vehicle Types

- HEV: (hybrid electric vehicle)
  - No plug; all power comes from gasoline
  - internal combustion engine (ICE) and motor(s) work together for improved efficiency
- PEV (plug-in electric vehicle) are defined as any vehicle that connects or plugs in to the grid to fully recharge the traction battery pack
  - BEVs: battery electric vehicle
    - no internal combustion engine (ICE)
    - no fuel tank
  - EREVs: extended range electric vehicles
    - operates on electricity first
    - when electric range has been exceeded, operates like a hybrid electric vehicle
  - PHEVs: plug-in hybrid electric vehicles
    - blended electric and ICE operations in various schemes

Photo courtesy of Nissan

Photo courtesy of General Motors

Photo courtesy of Ford

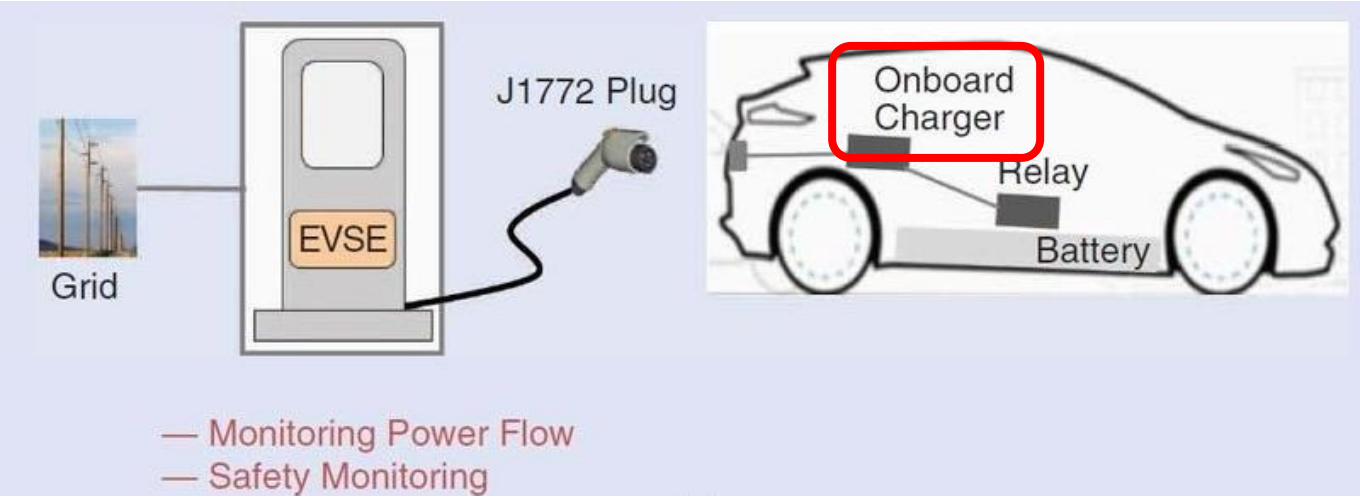
Electric Vehicle Supply Equipment (EVSE)

- Level 1 EVSE (≤1.4 kW):
  - AC 110/120V electric vehicle supply equipment
  - SAE J1772 standard
- Level 2 EVSE (≤19 kW):
  - AC 208/240V electric vehicle supply equipment
  - SAE J1772 standard

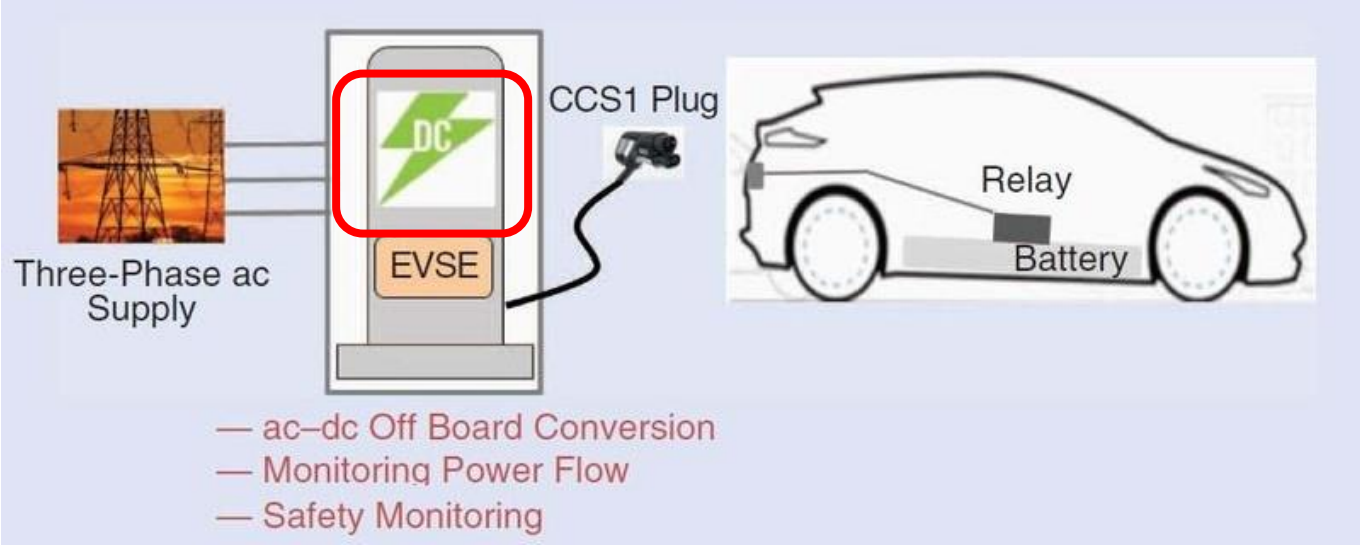
- DC fast chargers (22 kW to 350kW) for light duty vehicles
  - 3 predominate standards (for light duty)
    - SAE Combo Connector (CCS)
    - CHAdeMO
    - Tesla/NACS

AC Charging vs. DC Charging

- AC Charging
  - The charge station provide AC voltage to the vehicle
  - The vehicle converts AC to DC via onboard charge module



- DC Charging
  - The charge station converts AC to DC
  - The vehicle requests DC current
  - The charge station provides the requested DC current
  - Communications is required for control



Current Technology: DC Fast charging

- CHAdeMO and SAE CCS
  - 50 kW power transfer
    - Input: 480 VAC 3ϕ (75 A circuit)
    - Output: ~<500V DC up to 200A
- Tesla DC charging
  - 120kW power transfer
    - a few stations capable of 240kW

SAE J1772 / CCS

Tesla

Pocatello, ID photo courtesy of Jim Francfort

Current Technology: Xtreme Fast Charging (XFC)

- Nearly all uses CCS standard connector
  - Up to 400kW output power
    - <920VDC and 500A
      - Liquid cooled cable utilized for reduced weight and improved cable flexibility
      - Modular power conversion stages for scalable design, decreased cost, but increased controls complexity
  - Most designs separate the power electronics from the user interface (charge dispenser)

Plug and Charge (ISO 15118)

- The EV identifies itself to the EVSE when plugged in
- Charge session start without the need for credit card, RFID tag, app., etc.
- Transfers info between EV to EVSE

https://www.switch-ev.com/knowledgebase/basics-of-plug-and-charge

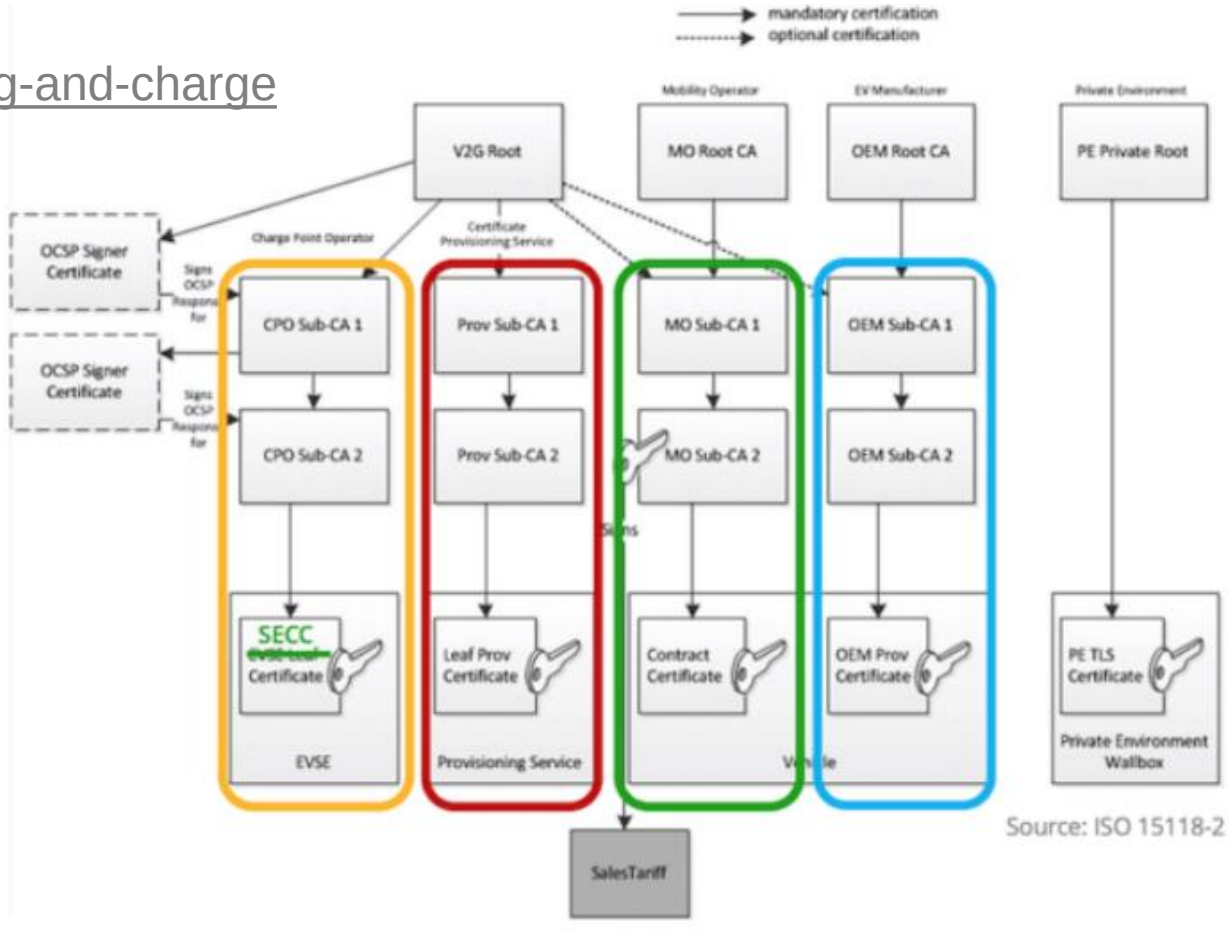
One PKI for each Plug & Charge market role

Charge point operator (CPO)
Operates and maintains the charging stations via its backend IT system

Certificate provisioning service (CPS)
Facilitates the installation of a new contract certificate for the EV through a digital signature

Mobility operator (MO)
Provides a legal e-mobility contract and issues contract certificates associated with that legal contract

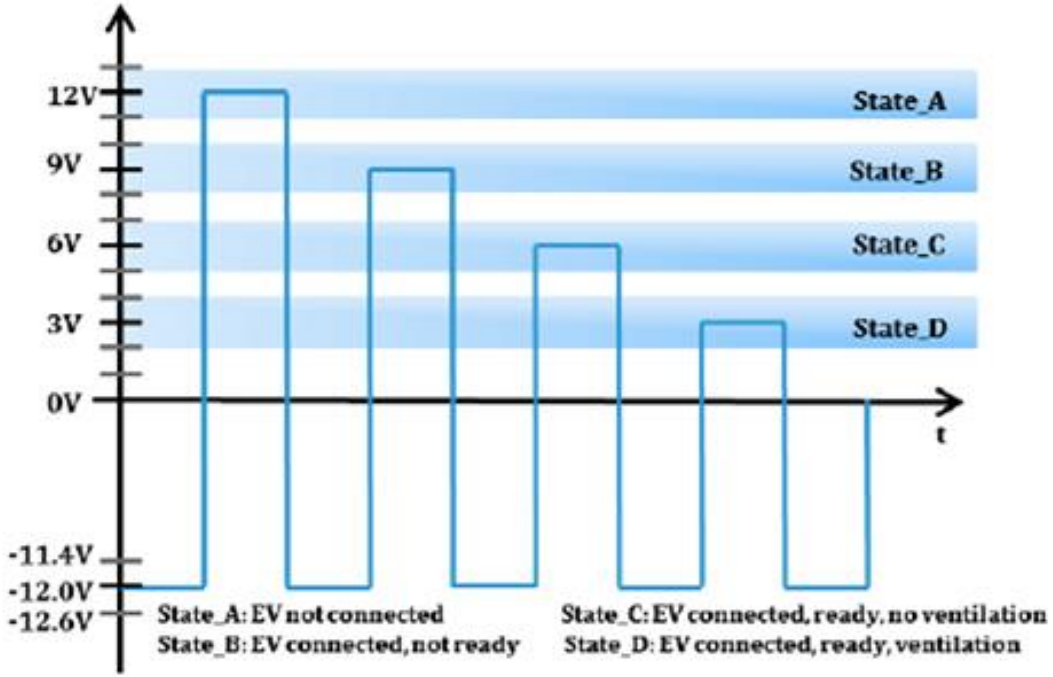
Car manufacturer (OEM)
Issues the unique OEM provisioning certificate needed to install a new contract certificate for Plug & Charge



Source: ISO 15118-2

SAE J1772 Control Pilot

- PWM duty cycle is 1 KHz
  - Every 1ms is a full cycle



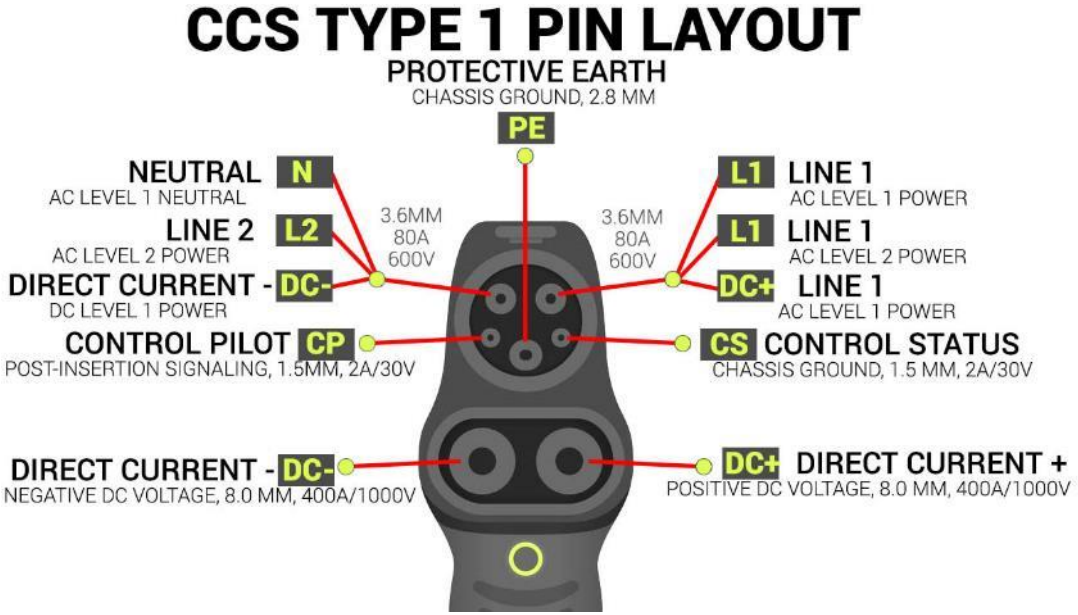
PWM	SAE Continuous	SAE Short Term
50%	30 A	36 A Peak
40%	24 A	30 A Peak
30%	18 A	22 A Peak
25%	15 A	20 A Peak
16%	9.6 A	
10%	6 A	
5%	0 A (PLC, TCP/IP)	DC Charging

AC Charging

TCP/IP Where?!?

- Control Pilot (CP) pin carries an embedded signal
  - PWM + HomePlug Green Phy (HPGP)
  - IPv6 communications
  - DIN 70121, ISO 15118-2, ISO 15118-20

SAE J1772 / CCS

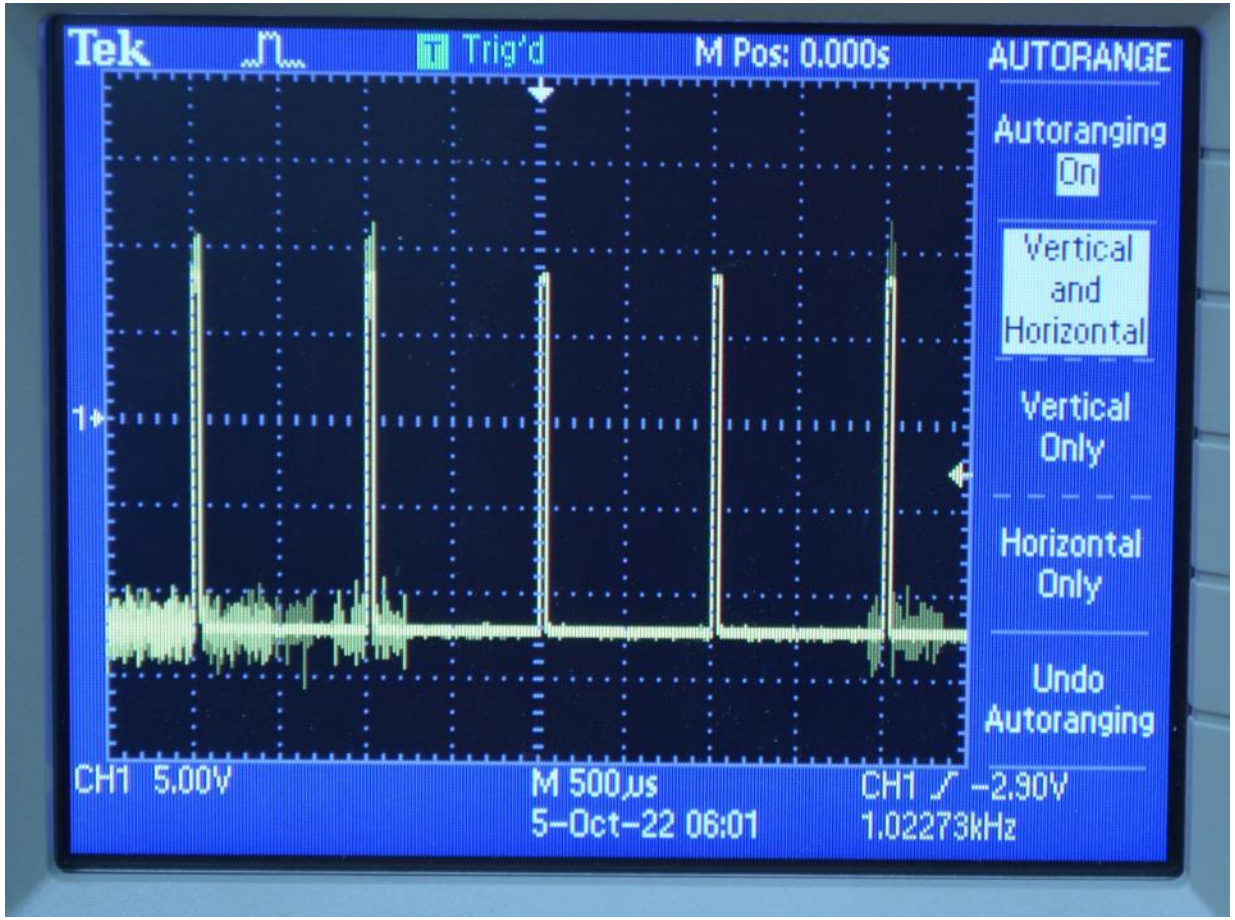


Consumer Home Powerline Extenders

- Power Line Carrier (PLC) communications
- Embed high frequency signals onto a few wires
- HomePlug AV2 specification
  - HomePlug Green Phy is a subset of AV2
- “Secure” network with a touch of a button

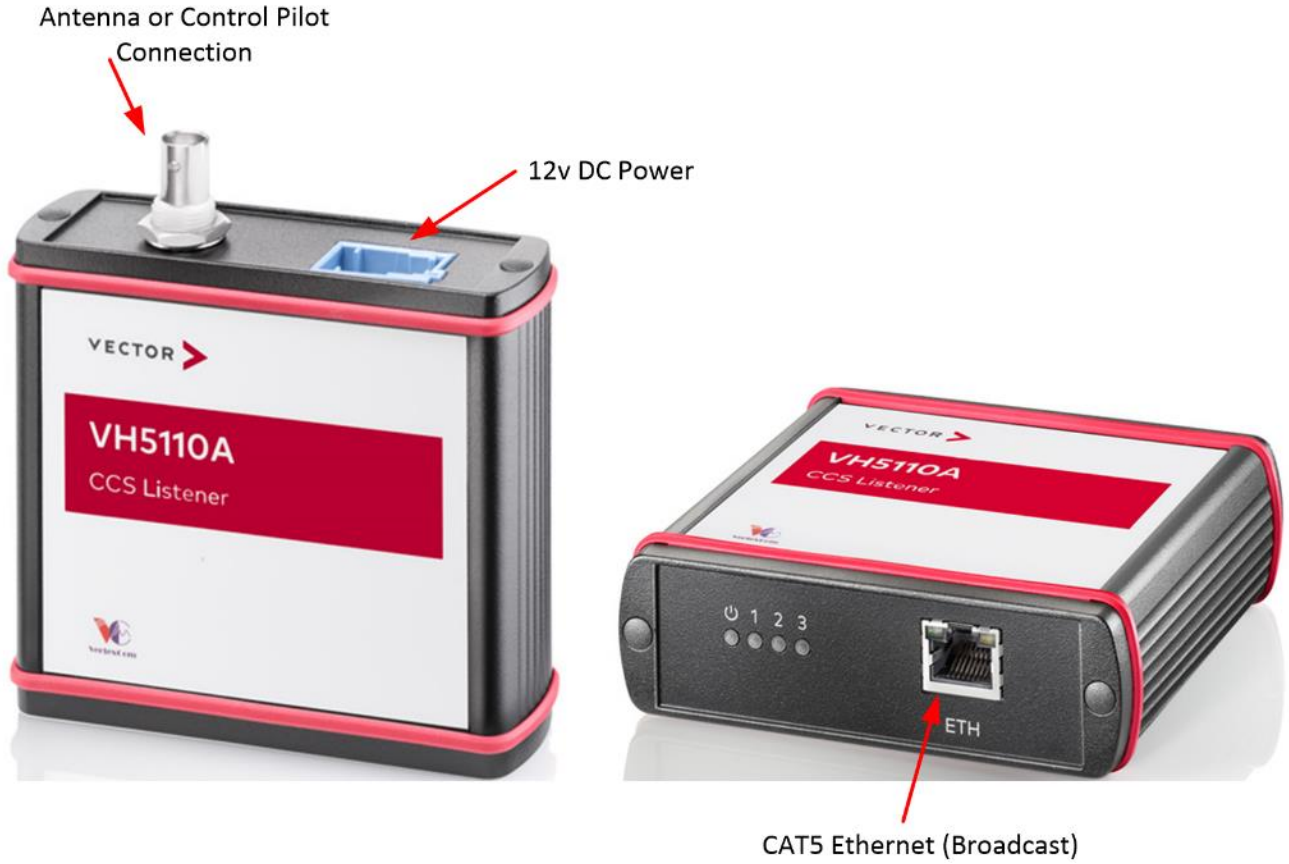
What this looks like….

- HPGP signal is embedded on the PWM



Sniffing HomePlug Green Phy (HPGP)

- Vector VH5110A
- Connected directly to the Control Pilot wire or a good antenna
- Passively sniffs traffic
  - The physical nature of this networking is easy to sniff



Wireshark Capture (DIN SPEC 70121)

- Vehicle: BMW i3
- Charger: ABB TerraHP (350 KW)

- The vehicle is the “client” and the EVSE is the “server”
- Layer 2 network setup and negotiation (HPGP)
- IPv6/UDP multicast on port 15118
- Full IPv6/TCP communications with charge data
- Data is Efficient XML Interchange (EXI) binary encoded XML (yuck)
  - The only good encoder/decoder freely available is in Java (barf)

Layer 2 – HPGP

No.	Time	Source	Destination	Protocol	Length	Info
10	8.200385649	LeopoldK_00:6b:1c	eGauge_01:61:79	HomePlug AV	60	CC_ASSOC.REQ
11	8.800371246	LeopoldK_00:6b:1c	eGauge_01:61:79	HomePlug AV	60	CC_ASSOC.REQ
12	9.341552836	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	60	CM_SLAC_PARM.REQ
13	9.344256622	Microchi_60:99:2e	LeopoldK_00:6b:1d	HomePlug AV	60	CM_SLAC_PARM.CNF
14	9.400433511	LeopoldK_00:6b:1c	eGauge_01:61:79	HomePlug AV	60	CC_ASSOC.REQ
15	9.612412043	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	60	CM_START_ATTEN_CHAR.IND
16	9.642083879	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	60	CM_START_ATTEN_CHAR.IND
17	9.672471234	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	60	CM_START_ATTEN_CHAR.IND
18	9.712143473	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
19	9.742738223	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
20	9.772210392	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
21	9.802769871	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
22	9.833114292	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
23	9.863242967	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
24	9.892455354	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
25	9.923047394	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
26	9.952534828	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
27	9.983823276	LeopoldK_00:6b:1d	Broadcast	HomePlug AV	71	CM_MNBC_SOUND.IND
28	10.000541883	LeopoldK_00:6b:1c	eGauge_01:61:79	HomePlug AV	60	CC_ASSOC.REQ
29	10.239467476	Microchi_60:99:2e	LeopoldK_00:6b:1d	HomePlug AV	129	CM_ATTEN_CHAR.IND
30	10.253068367	LeopoldK_00:6b:1d	Microchi_60:99:2e	HomePlug AV	70	CM_ATTEN_CHAR.RSP
31	10.894231672	LeopoldK_00:6b:1d	Microchi_60:99:2e	HomePlug AV	85	CM_SLAC_MATCH.REQ
32	10.897122712	Microchi_60:99:2e	LeopoldK_00:6b:1d	HomePlug AV	1[illegible]	CM_SLAC_MATCH.CNF
33	12.148241806	LeopoldK_00:6b:1c	devolo_f2:0a:7b	HomePlug AV	60	CC_ASSOC.REQ
34	12.150678514	devolo_f2:0a:7b	LeopoldK_00:6b:1c	HomePlug AV	60	CC_SET_TEI_MAP.IND
35	12.150678649	devolo_f2:0a:7b	Broadcast	HomePlug AV	60	CC_SET_TEI_MAP.IND
36	12.152129617	devolo_f2:0a:7b	LeopoldK_00:6b:1c	HomePlug AV	60	CC_ASSOC.CNF
37	12.200915822	LeopoldK_00:6b:1c	devolo_f2:0a:7b	HomePlug AV	491	CM_ENCRYPTED_PAYLOAD.IND (Encrypted Payload Indicate)
38	12.203632975	devolo_f2:0a:7b	LeopoldK_00:6b:1c	HomePlug AV	491	CM_ENCRYPTED_PAYLOAD.IND (Encrypted Payload Indicate)
39	12.210049505	LeopoldK_00:6b:1c	devolo_f2:0a:7b	HomePlug AV	60	CM_STA_INDENTIFY.IND
40	12.211317770	devolo_f2:0a:7b	LeopoldK_00:6b:1c	HomePlug AV	60	CM_STA_INDENTIFY.RSP
41	12.607960615	::	ff02::1:ff00:6b1d	ICMPv6	78	Neighbor Solicitation for fe80::f27f:cff:fe00:6b1d
42	13.611820873	fe80::f27f:cff:fe00:6b1d	ff02::2	ICMPv6	62	Router Solicitation

There are serious problems here…
NMK: Network Membership Key
DAK: Direct Access Key

Layer 3/4 - UDP port 15118

- Supply Equipment Communication Controller (SECC)
  - SECC Discovery Protocol (SDP)

layers > SECC.py > SECC_ResponseMessage

```python
# -*- coding: utf-8 -*-

#    V2GInjector is a tool to penetrate V2G network through PowerLine, monitor and i
#    Copyright (C) 2019  Sebastien Dudek (@FlUxIuS)
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <https://www.gnu.org/licenses/>.

from scapy.all import *

class SECC_RequestMessage(Packet):
    name = "SECC_RequestMessage"
    fields_desc = [ ByteField("SecurityProtocol", 0x0),
                    ByteField("TransportProtocol", 0x0),
                  ]

class SECC_ResponseMessage(Packet):
    name = "SECC_ResponseMessage"
    fields_desc = [ IP6Field("TargetAddress", "::"),
                    ShortField("TargetPort", 0),
                    ByteField("SecurityProtocol", 0x0),
                    ByteField("TransportProtocol", 0x0),
                  ]

SECC_types = {  0x9000 : "SECC_RequestMessage",
                0x9001 : "SECC_ResponseMessage",
        }

class SECC(Packet):
    name = "SECC"
    fields_desc = [ ByteField("Version", 0x01),
                    ByteField("Inversion", 0xfe),
                    ShortEnumField("SECCType", 0, SECC_types),
                    IntField("PayloadLen", 0),
                  ]

bind_layers( UDP, SECC, { "dport" : 15118 } )
bind_layers( UDP, SECC, { "sport" : 15118 } )
bind_layers( SECC, SECC_RequestMessage, { "SECCType" : 0x9000 } )
bind_layers( SECC, SECC_ResponseMessage, { "SECCType" : 0x9001 } )
```

LeopoldK_00:6b:1c	devolo_f2:0a:7b	HomePlug AV	60	CM_STA_INDENTIFY.IND
devolo_f2:0a:7b	LeopoldK_00:6b:1c	HomePlug AV	60	CM_STA_INDENTIFY.RSP
::	ff02::1:ff00:6b1d	ICMPv6	78	Neighbor Solicitation for fe80::f27f:cff:fe00:6b1d
fe80::f27f:cff:fe00:6b1d	ff02::2	ICMPv6	62	Router Solicitation
fe80::f27f:cff:fe00:6b1d	ff02::1	UDP	72	49701 → 15118 Len=10
fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	UDP	90	15118 → 49701 Len=28
fe80::f27f:cff:fe00:6b1d	ff02::1:ff60:992e	ICMPv6	86	Neighbor Solicitation for fe80::821f:12ff:fe60:992e fro
fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	ICMPv6	86	Neighbor Advertisement fe80::821f:12ff:fe60:992e (sol,
fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	78	56325 → 53537 [SYN] Seq=0 Win=3232 Len=0 MSS=1432
fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	78	53537 → 56325 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0 MSS
fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	74	56325 → 53537 [ACK] Seq=1 Ack=1 Win=3232 Len=0
fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	116	56325 → 53537 [PSH, ACK] Seq=1 Ack=1 Win=3232 Len=42

```
e (720 bits), 90 bytes captured (720 bits) on interface eth0, id 0
i_60:99:2e (80:1f:12:60:99:2e), Dst: LeopoldK_00:6b:1d (f0:7f:0c:00:6b:1d)
 6, Src: fe80::821f:12ff:fe60:992e, Dst: fe80::f27f:cff:fe00:6b1d
rc Port: 15118, Dst Port: 49701

e80000000000000821f12fffe60992e…
```

```
0000  f0 7f 0c 00 6b 1d 80 1f  12 60 99 2e 86 dd 60 00   ····k··· ·`····`·
0010  00 00 00 24 11 ff fe 80  00 00 00 00 00 00 82 1f   ···$···· ********
0020  12 ff fe 60 99 2e fe 80  00 00 00 00 00 00 f2 7f   ···`···· ********
0030  0c ff fe 00 6b 1d 3b 0e  c2 25 00 24 d1 c0 01 fe   ····k;·· ·%·$····
0040  90 01 00 00 00 14 fe 80  00 00 00 00 00 00 82 1f   ******** ********
0050  12 ff fe 60 99 2e d1 21  10 00                     ···`···! ··
```

```
Packet:  43
=== Starting a new charge session ===
Found Car Broadcast from (IPv6) fe80::f27f:cff:fe00:6b1d
###[ SECC ]###
  Version   = 1
  Inversion = 254
  SECCType  = SECC_RequestMessage
  PayloadLen= 2
###[ SECC_RequestMessage ]###
     SecurityProtocol= 16
     TransportProtocol= 0

Packet:  44
Found Charger Reply from (IPv6) fe80::821f:12ff:fe60:992e
###[ SECC ]###
  Version   = 1
  Inversion = 254
  SECCType  = SECC_ResponseMessage
  PayloadLen= 20
###[ SECC_ResponseMessage ]###
     TargetAddress= fe80::821f:12ff:fe60:992e
     TargetPort= 53537
     SecurityProtocol= 16
     TransportProtocol= 0
```

Layer 3/4 – EXI Encoded XML – DIN 70121

```
Packet:  50
EV --> EVSE
01fe8001000000228000dbab9371d3234b71d1b981899189d191818991d26b9b3a232b30020000040040
<?xml version="1.0" encoding="UTF-8"?><ns4:supportedAppProtocolReq xmlns:ns4="urn:iso:15118:2:2010:A
ppProtocol" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns3="http://www.w3.org/2001/
XMLSchema"><AppProtocol><ProtocolNamespace>urn:din:70121:2012:MsgDef</ProtocolNamespace><VersionNumb
erMajor>2</VersionNumberMajor><VersionNumberMinor>0</VersionNumberMinor><SchemaID>1</SchemaID><Prior
ity>1</Priority></AppProtocol></ns4:supportedAppProtocolReq>
Packet:  51
Packet:  52
EVSE --> EV
01fe80010000000480400040
<?xml version="1.0" encoding="UTF-8"?><ns4:supportedAppProtocolRes xmlns:ns4="urn:iso:15118:2:2010:A
ppProtocol" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns3="http://www.w3.org/2001/
XMLSchema"><ResponseCode>OK_SuccessfulNegotiation</ResponseCode><SchemaID>1</SchemaID></ns4:supporte
dAppProtocolRes>
```

50	14.432248606	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	116	56325 → 53537 [PSH, ACK] Seq=1 Ack=1 Win=3232 Len=42
51	14.439313470	fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	74	53537 → 56325 [ACK] Seq=1 Ack=43 Win=2878 Len=0
52	14.545640085	fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	86	53537 → 56325 [PSH, ACK] Seq=1 Ack=43 Win=2878 Len=12
53	14.555105306	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	74	56325 → 53537 [ACK] Seq=43 Ack=13 Win=3232 Len=0
54	14.572002061	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	97	56325 → 53537 [PSH, ACK] Seq=43 Ack=13 Win=3232 Len=23
55	14.584724751	fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	99	53537 → 56325 [PSH, ACK] Seq=13 Ack=66 Win=2855 Len=25
56	14.594990724	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	74	56325 → 53537 [ACK] Seq=66 Ack=38 Win=3101 Len=0
57	14.612532799	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	98	56325 → 53537 [PSH, ACK] Seq=66 Ack=38 Win=3232 Len=24
58	14.624620633	fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	101	53537 → 56325 [PSH, ACK] Seq=38 Ack=90 Win=2831 Len=27
59	14.634535457	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	74	56325 → 53537 [ACK] Seq=90 Ack=65 Win=3232 Len=0
60	14.652377627	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	98	56325 → 53537 [PSH, ACK] Seq=90 Ack=65 Win=3232 Len=24
61	14.665647902	fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	96	53537 → 56325 [PSH, ACK] Seq=65 Ack=114 Win=2807 Len=22
62	14.675447834	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	74	56325 → 53537 [ACK] Seq=114 Ack=87 Win=3232 Len=0
63	14.692896834	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	95	56325 → 53537 [PSH, ACK] Seq=114 Ack=87 Win=3232 Len=21
64	14.705677784	fe80::821f:12ff:fe60:992e	fe80::f27f:cff:fe00:6b1d	TCP	97	53537 → 56325 [PSH, ACK] Seq=87 Ack=135 Win=2786 Len=23
65	14.715102637	fe80::f27f:cff:fe00:6b1d	fe80::821f:12ff:fe60:992e	TCP	74	56325 → 53537 [ACK] Seq=135 Ack=110 Win=3232 Len=0

```
+ Frame 50: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface eth0, id 0
+ Ethernet II, Src: LeopoldK_00:6b:1d (f0:7f:0c:00:6b:1d), Dst: Microchi_60:99:2e (80:1f:12:60:99:2e)
+ Internet Protocol Version 6, Src: fe80::f27f:cff:fe00:6b1d, Dst: fe80::821f:12ff:fe60:992e
+ Transmission Control Protocol, Src Port: 56325, Dst Port: 53537, Seq: 1, Ack: 1, Len: 42
- Data (42 bytes)
    Data: 01fe8001000000228000dbab9371d3234b71d1b981899189…
    [Length: 42]
```

```
0000  80 1f 12 60 99 2e f0 7f  0c 00 6b 1d 86 dd 60 00   ···`····  ··k···`·
0010  00 00 00 3e 06 ff fe 80  00 00 00 00 00 00 f2 7f   ···>····  ********
0020  0c ff fe 00 6b 1d fe 80  00 00 00 00 00 00 82 1f   ····k···  ********
0030  12 ff fe 60 99 2e dc 05  d1 21 0c 70 b9 39 04 90   ···`····  ·!·p·9··
0040  42 3f 50 18 0c a0 2a 53  00 00 01 fe 80 01 00 00   B?P···*S  ********
0050  00 22 80 00 db ab 93 71  d3 23 4b 71 d1 b9 81 89   ·"·····q  ·#Kq····
0060  91 89 d1 91 81 89 91 d2  6b 9b 3a 23 2b 30 02 00   ********  k·:#+0··
0070  00 04 00 40                                        ···@
```

https://en.wikipedia.org/wiki/Efficient_XML_Interchange

Current Hardware

Devolo dLAN Green Phy Evaluation boards

Qualcomm QC7000 daughter board

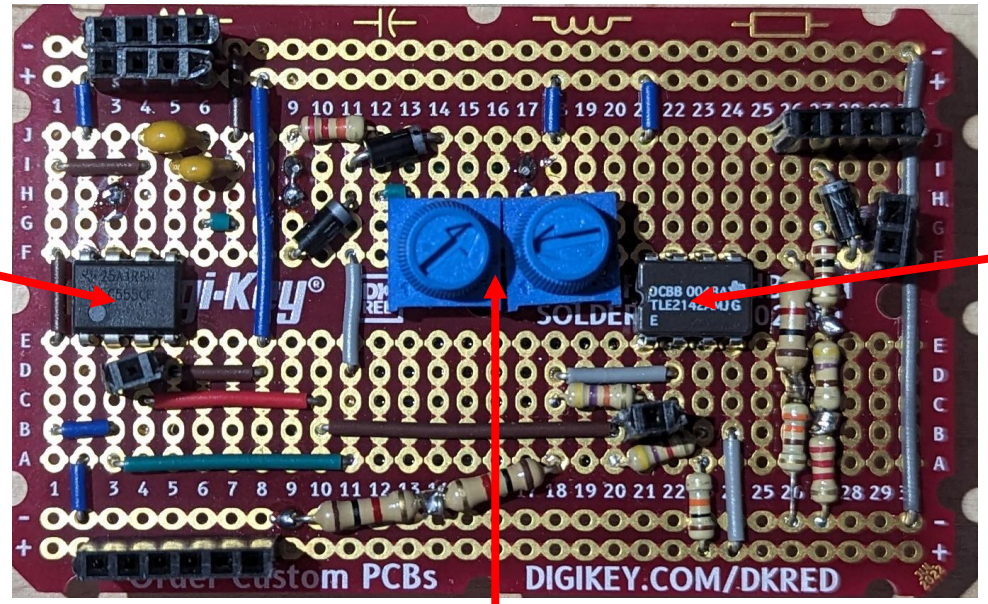
Current Hardware

1. Raspberry Pi 4
2. EVSE Configured Devolo
3. EV Configured Devolo
4. PWM Circuit
5. Relays
6. Power Isolators

Current Hardware

PWM Circuit

555 Timer (1 kHz)

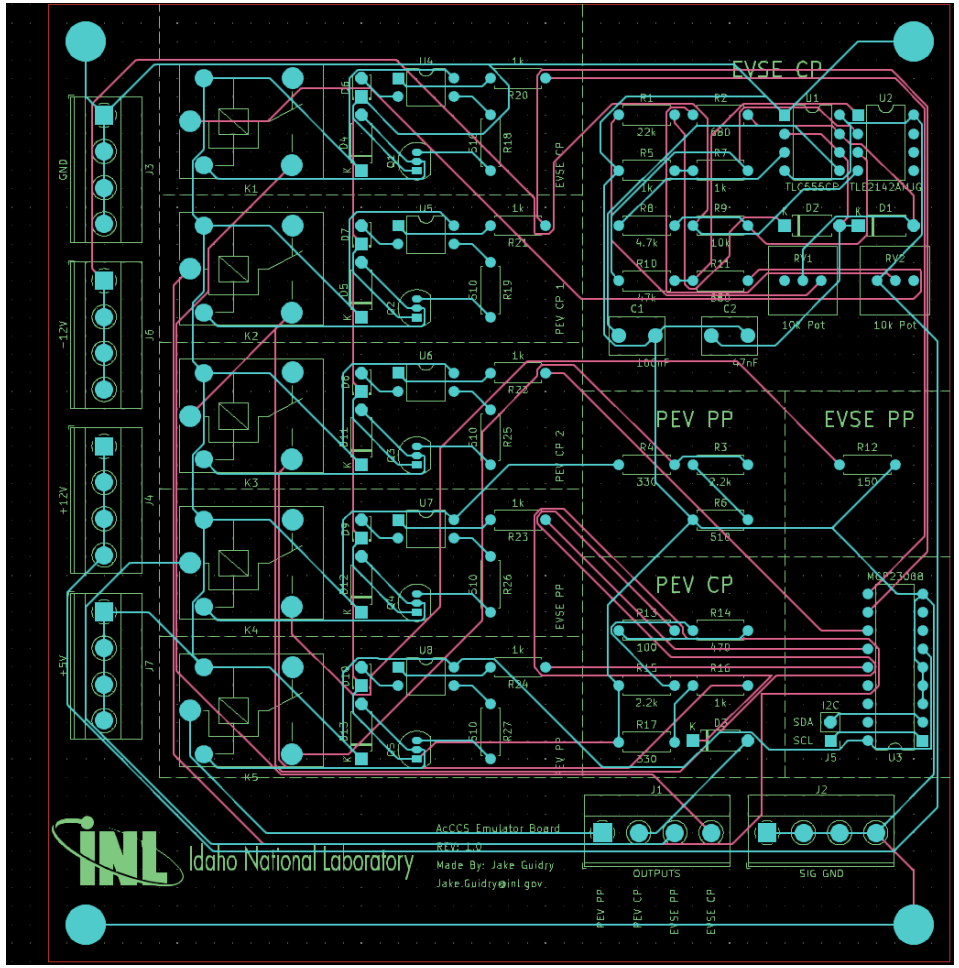


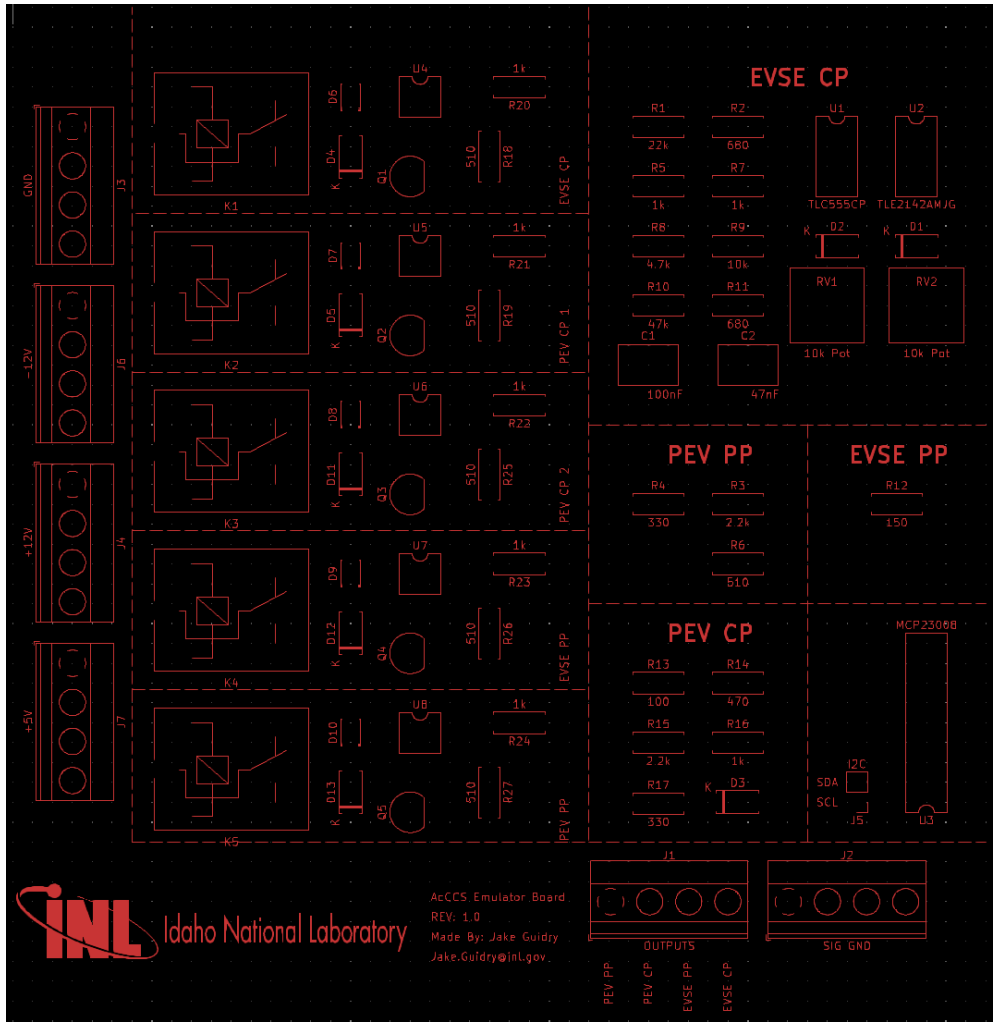
Dual Op-Amp (+/- 12v)

POTs Adjust Frequency and Duty-Cycle

Current Hardware

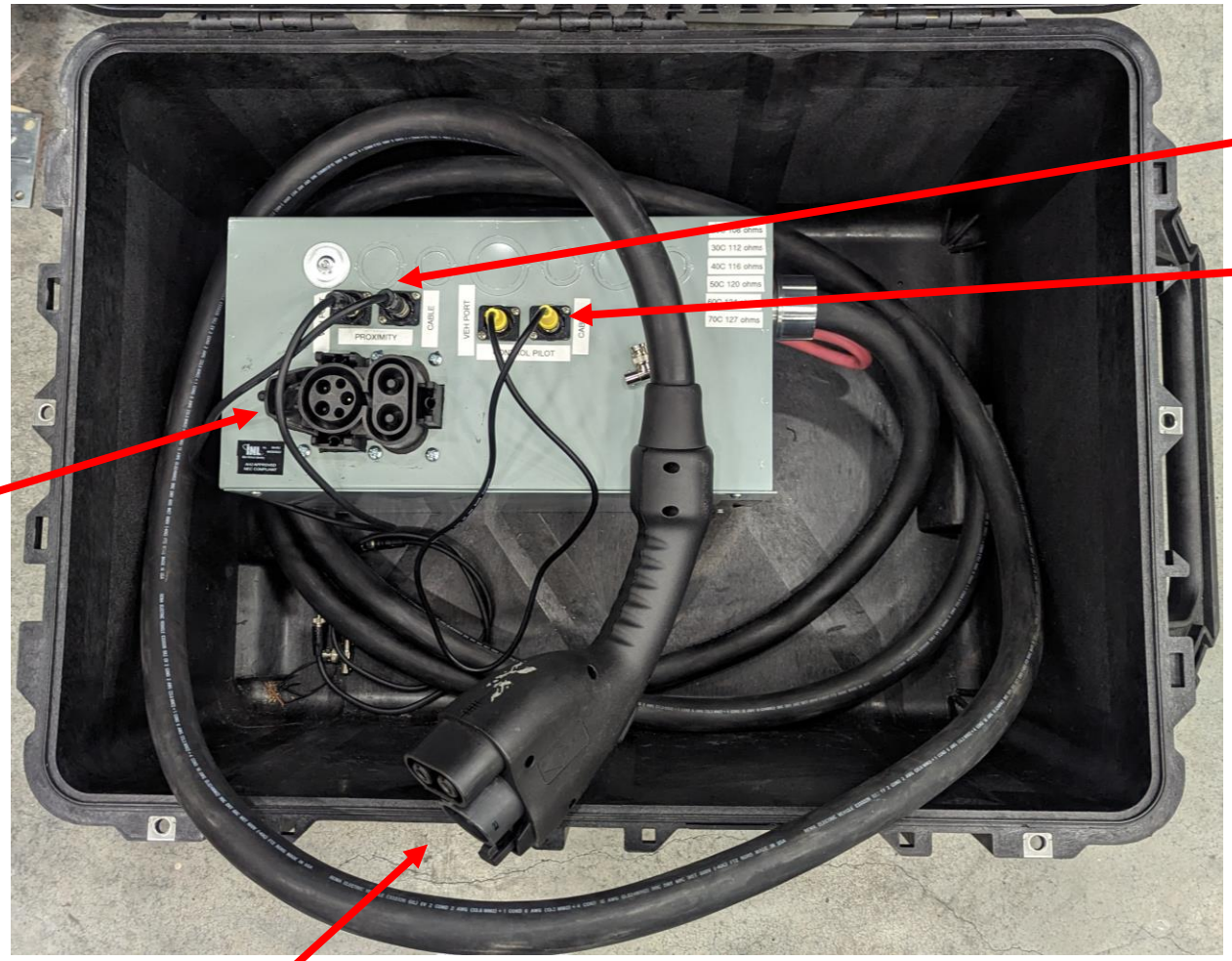
PCB PWM Circuit





Current Hardware

MITM Box



Proximity Pin

Control Pilot

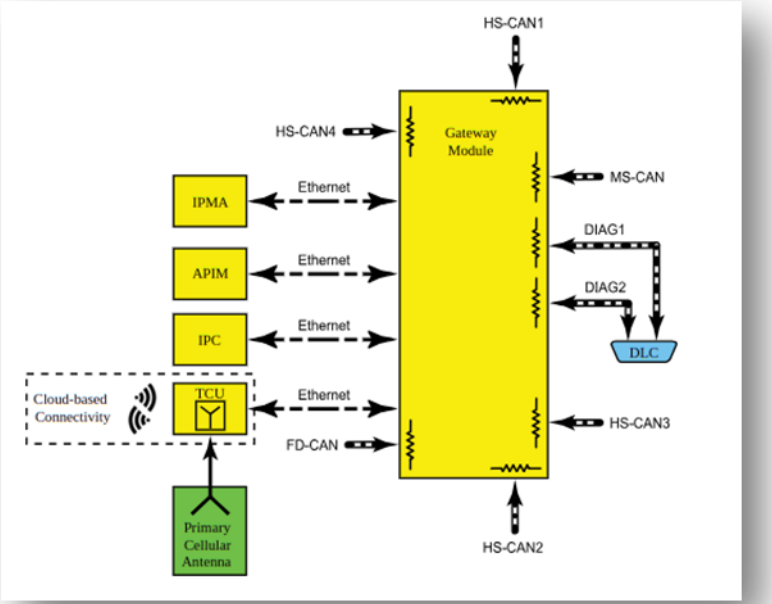
From EVSE

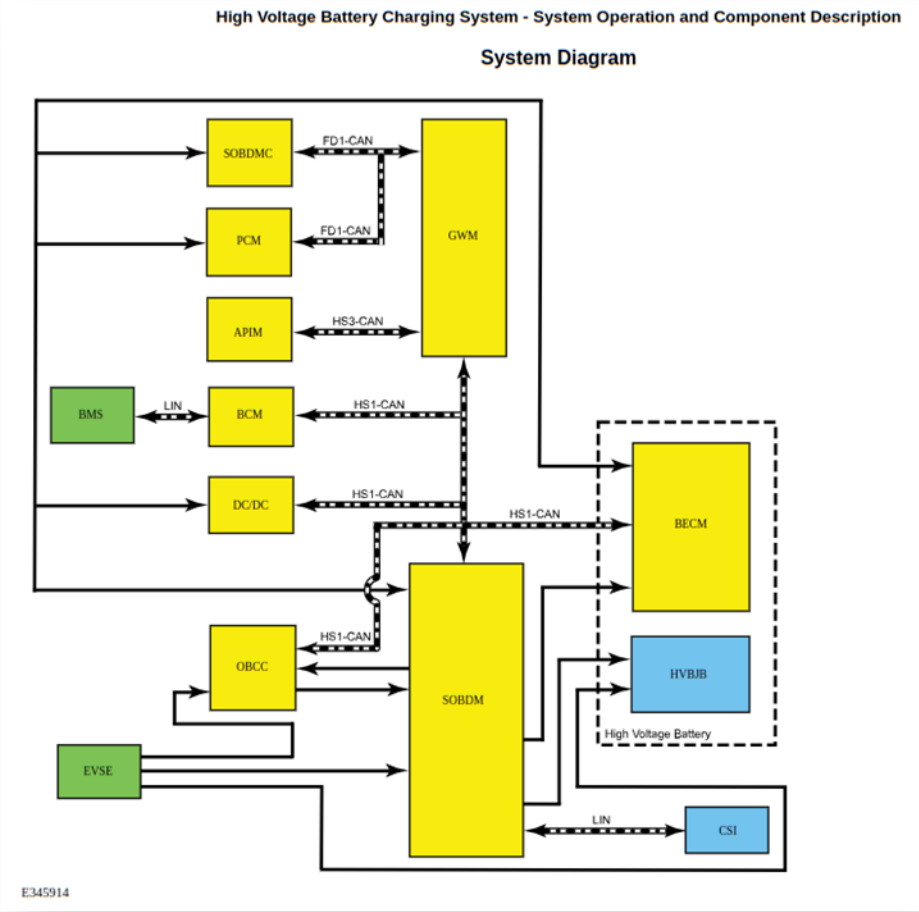
To Vehicle

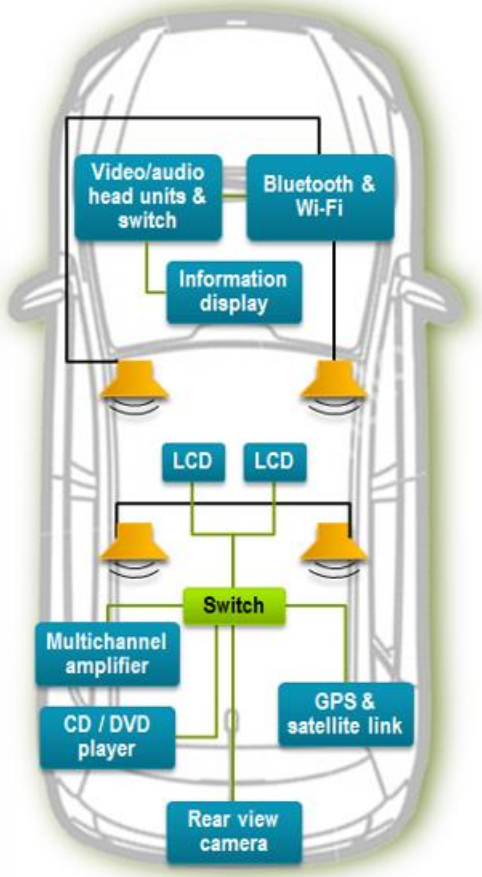
Scanning the Vehicle

Central Gateway Security

VLAN Segmentation



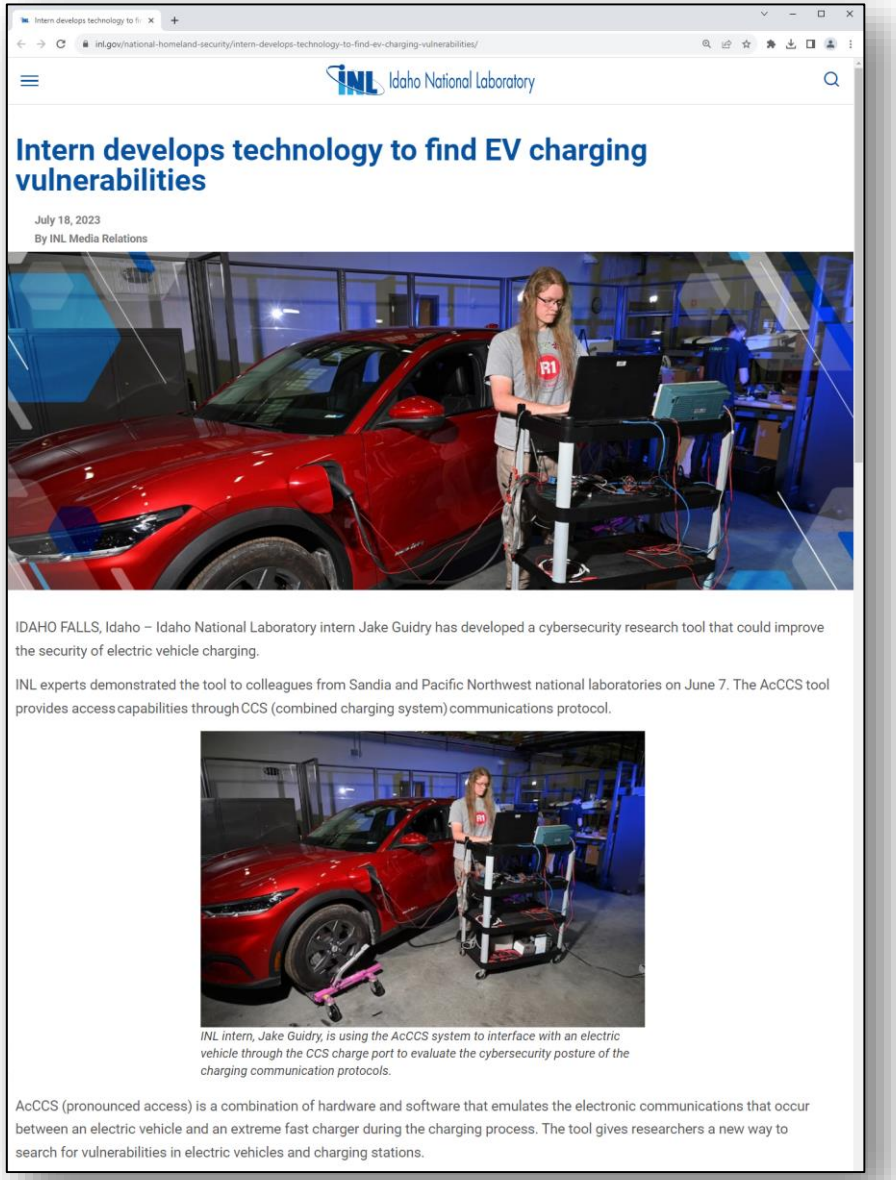




Scanning the Vehicle

- Establish the vehicle connection
- Keep the vehicle in a “PendingAuthorization” state
- Port scan (slowly) for open TCP or UDP ports
  - Restart the network and resume scanning

Scanning the Vehicle



Idaho National Laboratory

Intern develops technology to find EV charging vulnerabilities

July 18, 2023
By INL Media Relations

IDAHO FALLS, Idaho – Idaho National Laboratory intern Jake Guidry has developed a cybersecurity research tool that could improve the security of electric vehicle charging.

INL experts demonstrated the tool to colleagues from Sandia and Pacific Northwest national laboratories on June 7. The AcCCS tool provides access capabilities through CCS (combined charging system) communications protocol.

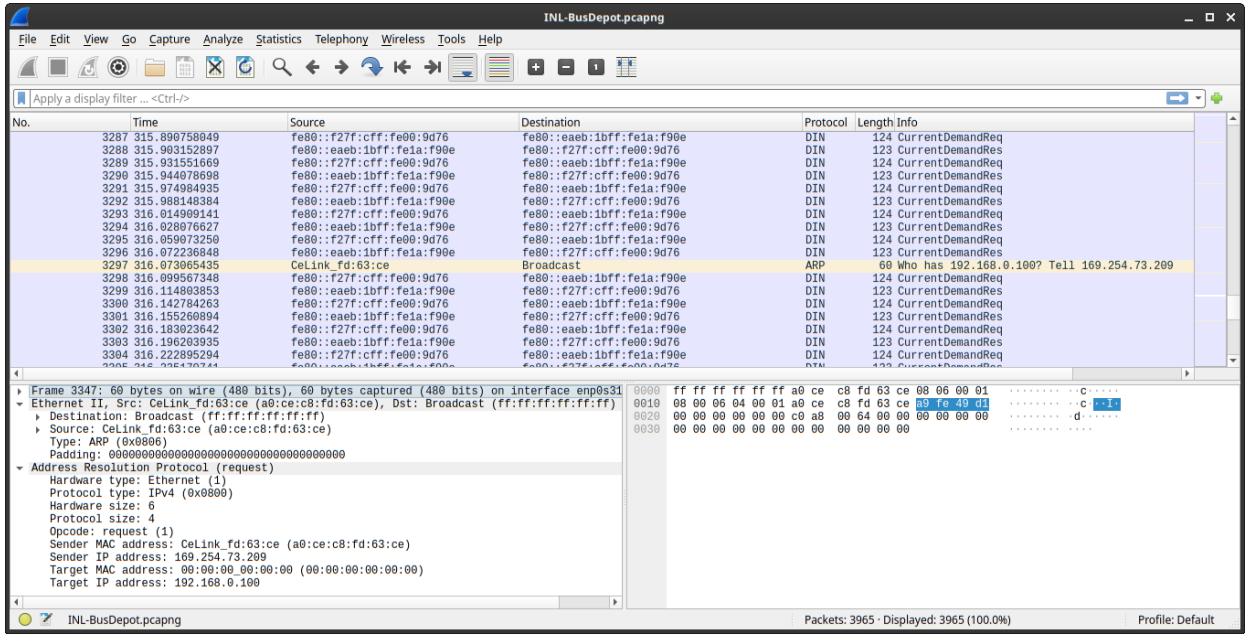
INL intern, Jake Guidry, is using the AcCCS system to interface with an electric vehicle through the CCS charge port to evaluate the cybersecurity posture of the charging communication protocols.

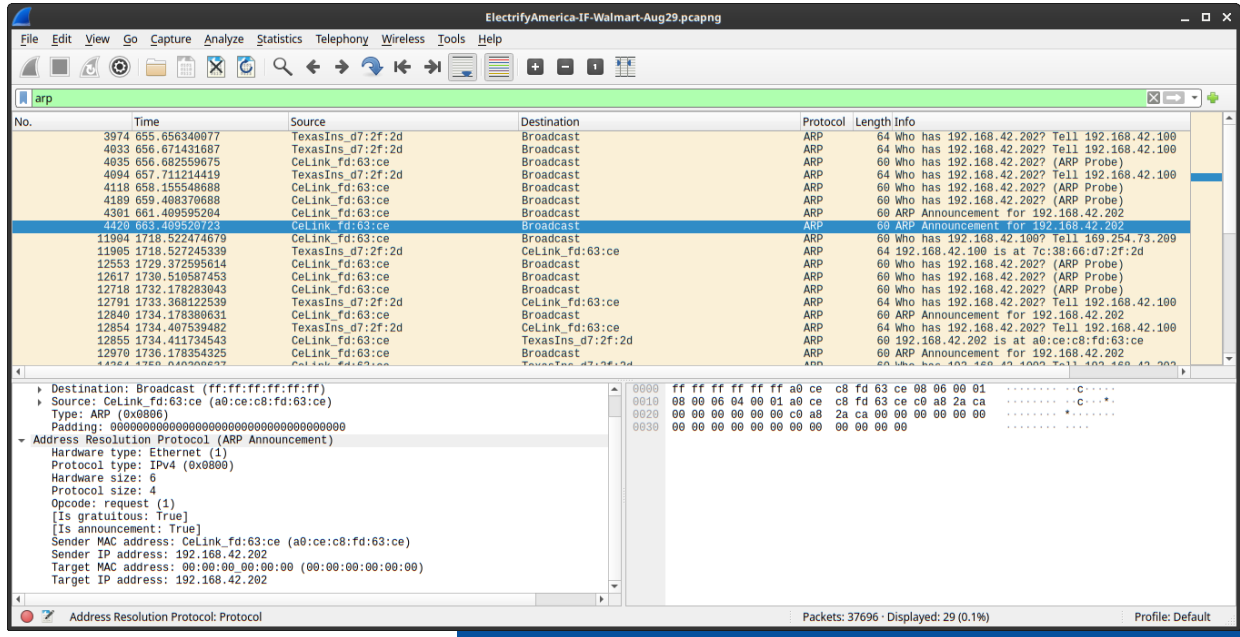
AcCCS (pronounced access) is a combination of hardware and software that emulates the electronic communications that occur between an electric vehicle and an extreme fast charger during the charging process. The tool gives researchers a new way to search for vulnerabilities in electric vehicles and charging stations.

TCP Reset vs. ICMP Destination Unreachable?

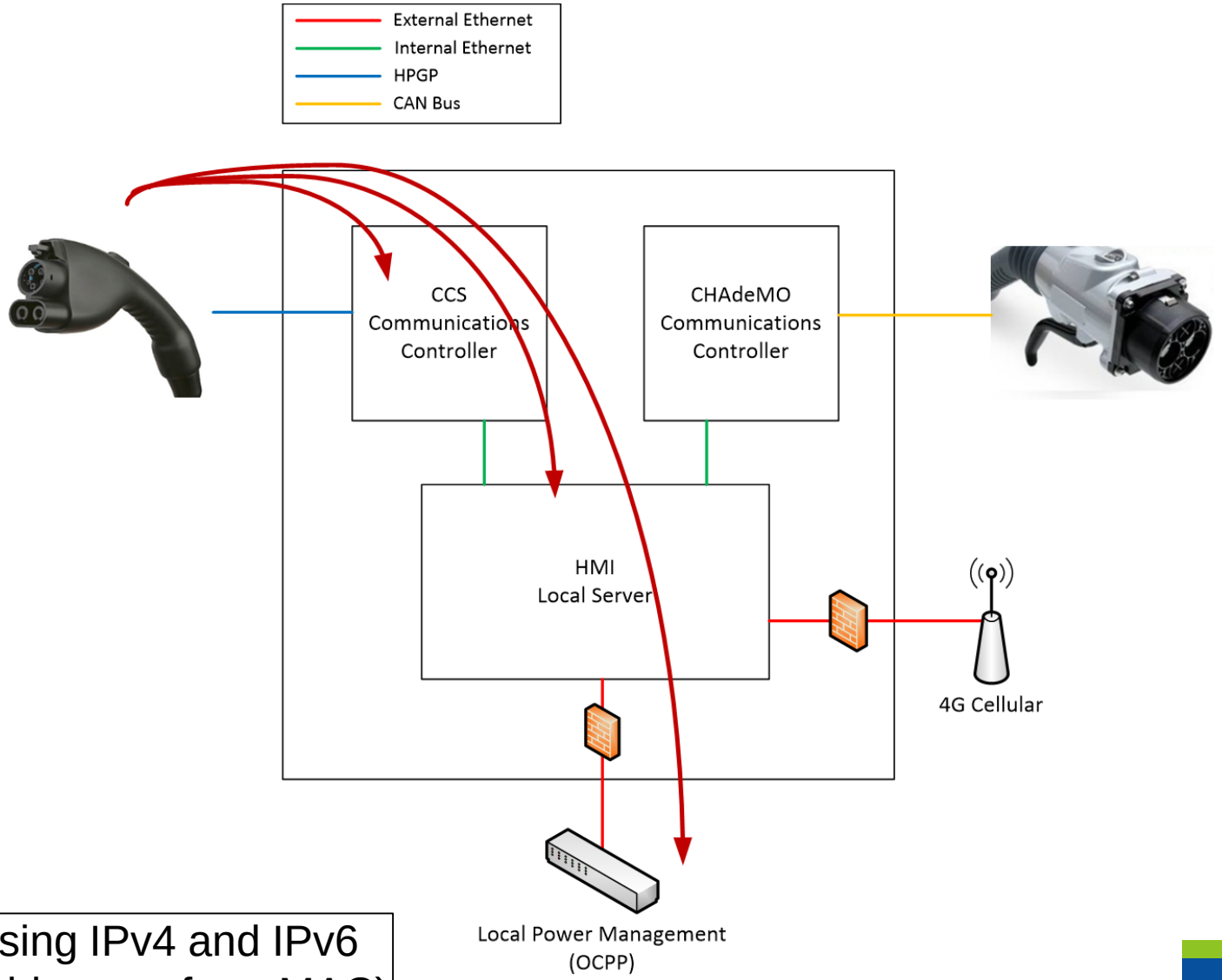
Scanning the Charging Station

- Establish the EVSE connection
- Keep the EVSE communicating with “CurrentDemandReq” “ContractAuthenticaionReq” messages
  - Or request power (no payment necessary!?!)
- Port scan for open TCP or UDP ports
- Listen for other network packet “leakage”



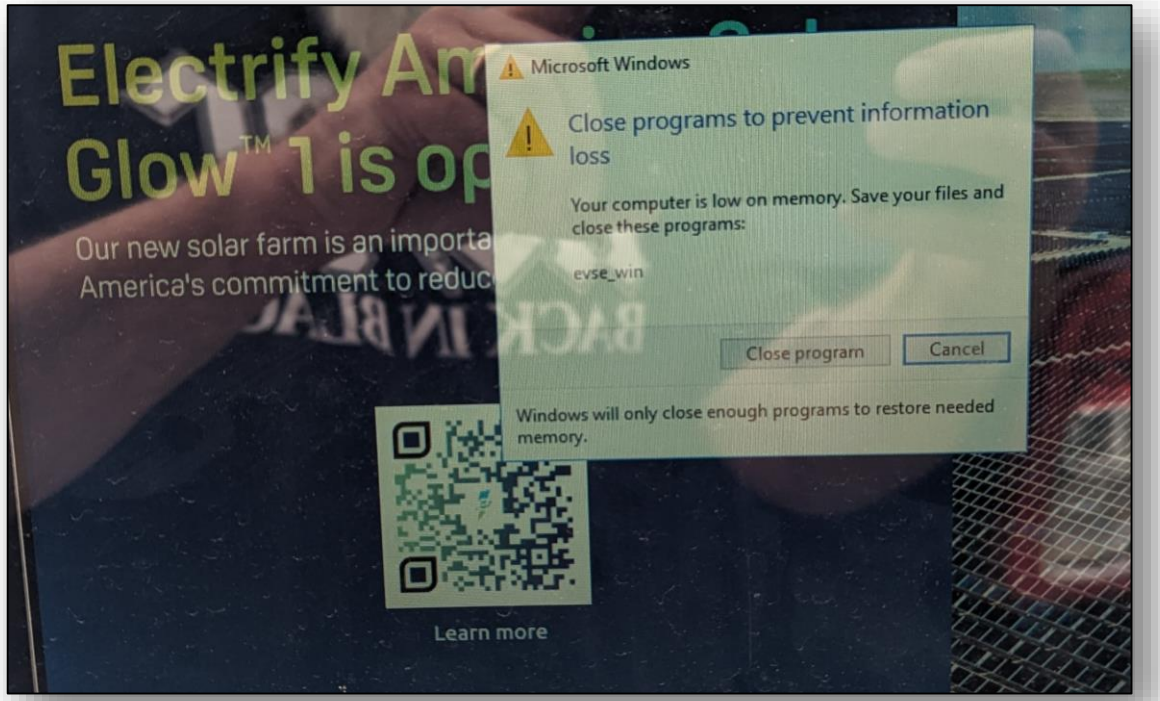


Scanning the Charging Station



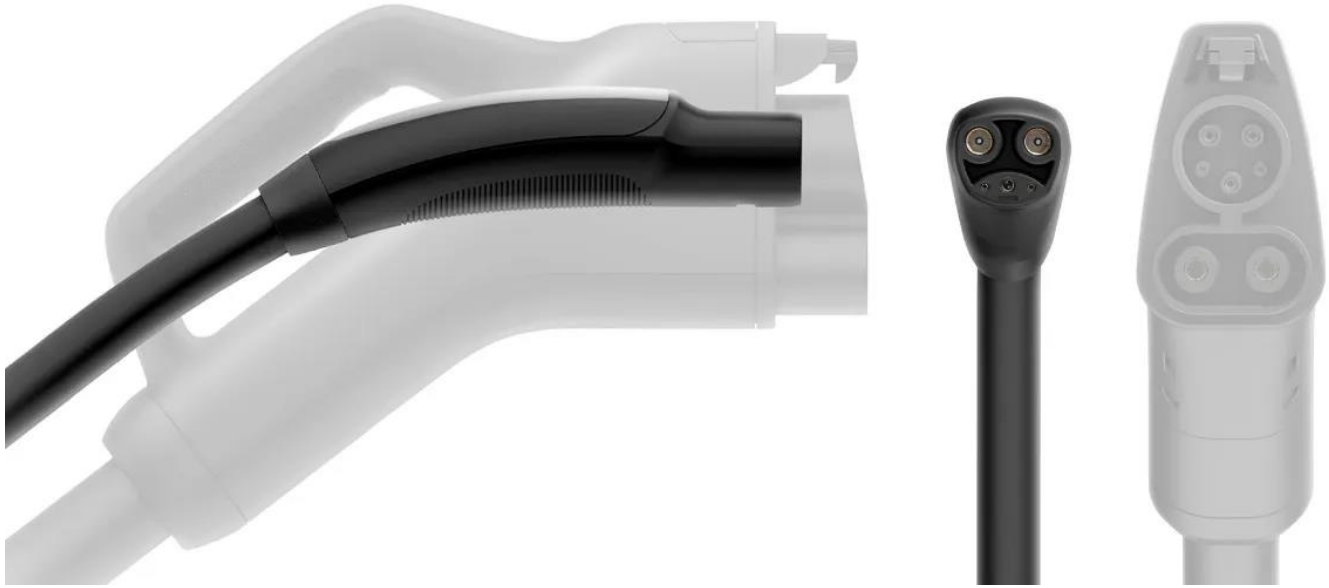
Scan devices using IPv4 and IPv6 (link-local IPv6 addresses from MAC)

Scanning the Charging Station



What about Tesla and NACS?

The CP, or Control Pilot, pin is used as a digital communication path between the charging system and vehicle. The control pilot uses pulse width modulation to communicate charging state and current in accordance with IEC 61851. **Power Line Communication** is superimposed onto the control pilot line while DC charging.



Answer: It's basically the same thing!

Credits

- V2GInjector
  - https://github.com/FlUxIuS/V2GInjector
- CODICO Devolo dLAN Green Phy Eval Boards
  - https://www.codico.com/en/pev-side-configured-dlan-green-phy-eval-board
  - https://www.codico.com/en/evse-side-configured-dlan-green-phy-eval-board
- Green Phy Module SDK
  - https://github.com/devolo/dlan-greenphy-sdk
- QCA700 Configuration
  - https://github-wiki-see.page/m/zoological/caribou/wiki/dlan-green-phy-module
- OpenEVSE project
  - https://github.com/OpenEVSE/OpenEVSE_PLUS

Idaho National Laboratory

Battelle Energy Alliance manages INL for the U.S. Department of Energy's Office of Nuclear Energy. INL is the nation's center for nuclear energy research and development, and also performs research in each of DOE's strategic goal areas: energy, national security, science and the environment.