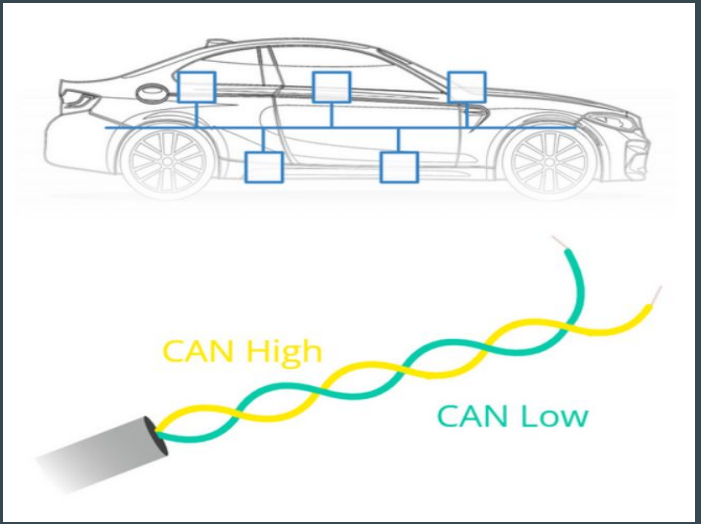
Mobile Vehicle Cybersecurity with Onboard Key Management

Iowa State University: ECpE sdmay23-15

Aayush Chanda, Alexander Freiberg, Baganesra Bhaskaran, Brian Goode, Chau Wei Lim, Michael Roling

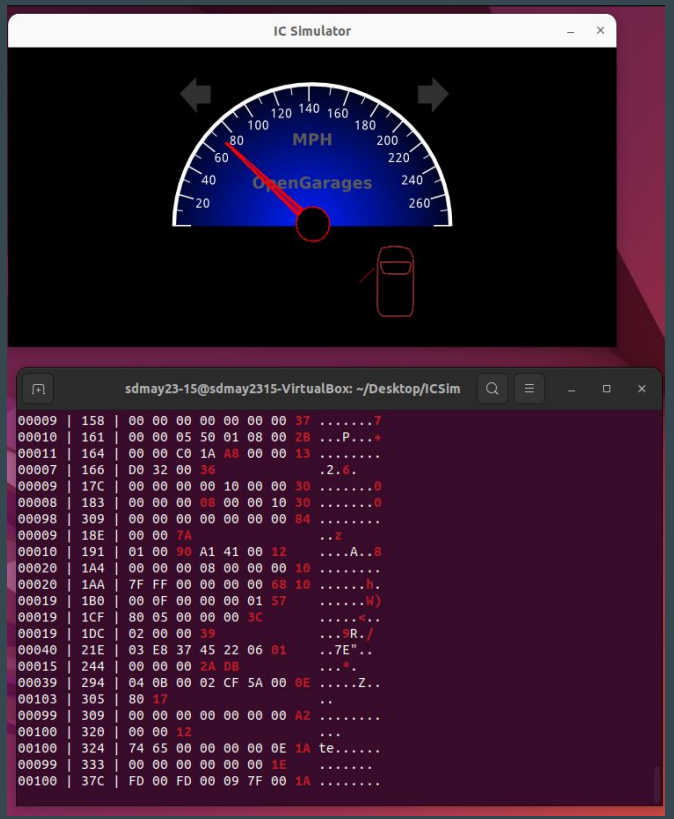
Introduction

- Vehicle controllers communicate via Controller Area Network (CAN) Bus
  - Open and non-secure network
  - Follow SAE J1939 communication protocols
- Interest of all parties for authentic data
  - OEMs, owners, and 3rd party producers
- Safety concerns
  - Physical devices/sniffer tools
  - Virtual software development



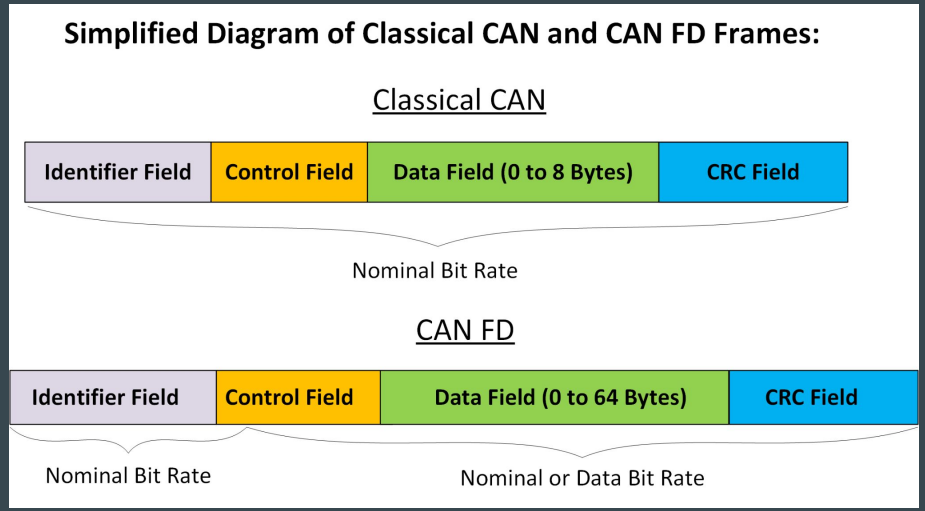
Implementation Architecture

- CAN Simulator (C)
  - Ubuntu
  - CAN Tx/Rx
  - CAN Sniffer
- J1939 Protocols
  - CAN FD; 2-5 MBits/Sec.
    - CAN; 1 MBit/Sec.
- TweetNaCl
  - Encryption/Decryption
  - 40 bytes of overhead
    - 24 byte nonce
    - 16 byte message authentication code (MAC)
- Hardware: outside the scope of the project



Milestones

- CAN Tx/Rx
  - CAN Socket; Python and C
  - Software development to ensure sequential messages
- CAN FD and J1939
  - Extending CAN Frames
  - Posed issues with encryption
- TweetNaCl
  - Implementing nonce and MAC
  - Ensuring timely asynchronous communication



Contributions

- *Aayush (Advisor Liaison):* developed functionality for CAN FD frames, integrated TweetNaCl encryption, and served to effectively communicate between the advisor and team.
- *Alexander (Client Liaison):* initialized the Virtual CAN environment where all software was implemented, developed TweetNaCl encryption, and ensured proper communication with the client.
- *Baganesra (GitLab Administrator):* developed transmission/reception functionality among the virtual ECUs, aided in the design of encryption protocols, and managed growth within GitLab.
- *Brian (Team Organizer):* aided in the design of a manifest list for key exchanges, implementation of a nonce, and helped organize the team for weekly development.
- *Chau (Strategist):* developed software to facilitate communication for each ECU, managed setting up the CAN network at each node, and led the development for each week's schedule.
- *Michael (Documentor):* oversaw the implementation of pertinent J1939 protocols, code review for integration of TweetNaCl, and created documentation for project's development.

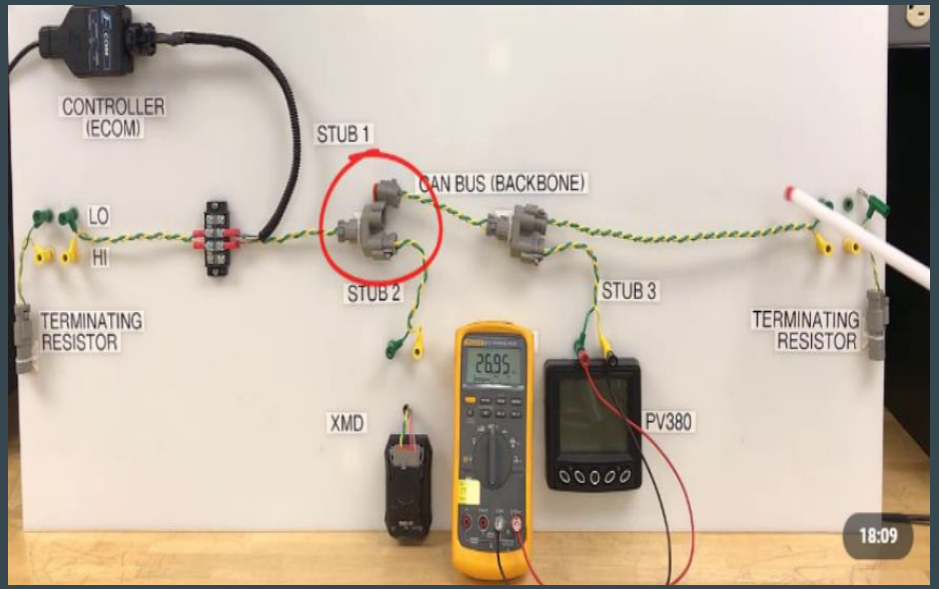
Setbacks and Solutions

- Learning curve of CAN
  - J1939 Protocols
- Virtual simulation environment
  - Ubuntu
- Incorrect Initial Design Implementation
  - Lightweight Authentication using CRC bit field in CAN Frame
- CAN Socket in C
  - CAN Tx/Rx
  - Multiple nodes on the CAN Bus
- CAN FD
- TweetNaCl encryption
  - Box Function; nonce and MAC

```
File vehiclePubKey.txt : Found
Key from File vehiclePubKey.txt: Read
File vehiclePrivKey.txt : Found
Key from File vehiclePrivKey.txt: Read
Public/Private Key Pair Loaded
Socket Initialized
Socket switched to CAN FD
Socket bound to Interface
Initializing CAN FD Frame

***** Reading Mode 1 *****

Timeout occured: Nothing to receive

***** Checking if there is data to send *****
Nonce Loaded: B988D80129C05DCE4BAB84A106B6A8FF59848998B86CF430
Message to be Encrypted (In Hex): 726F6C6C636C6F6E65733132
Encrypting Message...
Encrypted Message Length: 28
Encrypted Message: 6F83B40C7F5D9D9DFA8489745E7472C4C47C67BA148A947EB4121E56
Total Payload Length (Padded): 52
Sending Frame...
CAN FD Frame SENT (ID = 0x123, Length = 52 Bytes)
Payload: B988D80129C05DCE4BAB84A106B6A8FF59848998B86CF4306F83B40C7F5D9D9DFA8489745E74
72C4C47C67BA148A947EB4121E56

***** Reading Mode 2 *****

Data available
Scanning for Frames...
Frame Received!
Frame Data: B988D80129C05DCE4BAB84A106B6A8FF59848998B86CF4306F83B40C7F5D9D9DFA8489745
E7472C4C47C67BA148A947EB4121E56
Nonce Extracted: B988D80129C05DCE4BAB84A106B6A8FF59848998B86CF430
Ciphertext Extracted: 6F83B40C7F5D9D9DFA8489745E7472C4C47C67BA148A947EB4121E56
Decrypting Message...
Message: Decrypted!
Plaintext Message: rollclones12
Frame ID              Length of Frame Data          Length of Plaintext Message
     Message (Hex)
0x123                 52                            12
     72 6F 6C 6C 63 6C 6F 6E 65 73 31 32

***** Reading Mode 3 *****
^C
```

Continued Development

- Hardware development; outside scope of the project
  - Flashing valid ECUs on CAN Bus
  - Testing invalid devices on CAN Bus
- Software development
  - Manifest containing Protocol Group Numbers (PGNs) to map to CAN IDs
  - Increase Maximum Transmission Unit (MTU) to fit larger messages
  - Implement Software/Scripts to manage each ECU
  - Integration of messages as found in J1939 protocol.



Conclusion

- Effectively met client's requirements
  - Technical ability to handle CAN-FD segments
    - Sequential Tx/Rx CAN messages (<5 mS)
  - Implementation of key management protocols (J1939)
  - Generated key to handle encryption/decryption of messages
    - TweetNaCl
- Strong safety applications to the vehicle industry
  - OEM manufacturers, vehicle owners, and 3rd party producers
- Applications after graduation
  - Network security at the enterprise level
  - Design of electric drives on CAN Bus
  - Engineering controllers with CAN Bus functionality

Works Cited

- CAN Bus Diagram
- CAN_FD_image
- sdmay23-15 • Mobile Vehicle Cybersecurity with On-board Key Management (iastate.edu)

Appendix

Client: John Roberts

Advisor: Dr. Joseph Zambreno