OAK RIDGE National Laboratory

CAN-D: Controller Area Network Decoder

Team: Robert A. Bridges, Miki Verma, Mike Iannacone, Sam Hollifield

PI: Stacy Prowell

Cybersecurity Research Group
Oak Ridge National Laboratory

U.S. DEPARTMENT OF ENERGY

Controller Area Network (CAN)

- Modern vehicles have scores of electronic control units (ECUs)
- CAN is a message-based protocol used in
  - Industrial facilities
  - Operating rooms
  - Airplanes
  - **Vehicles**

OBD-II Port

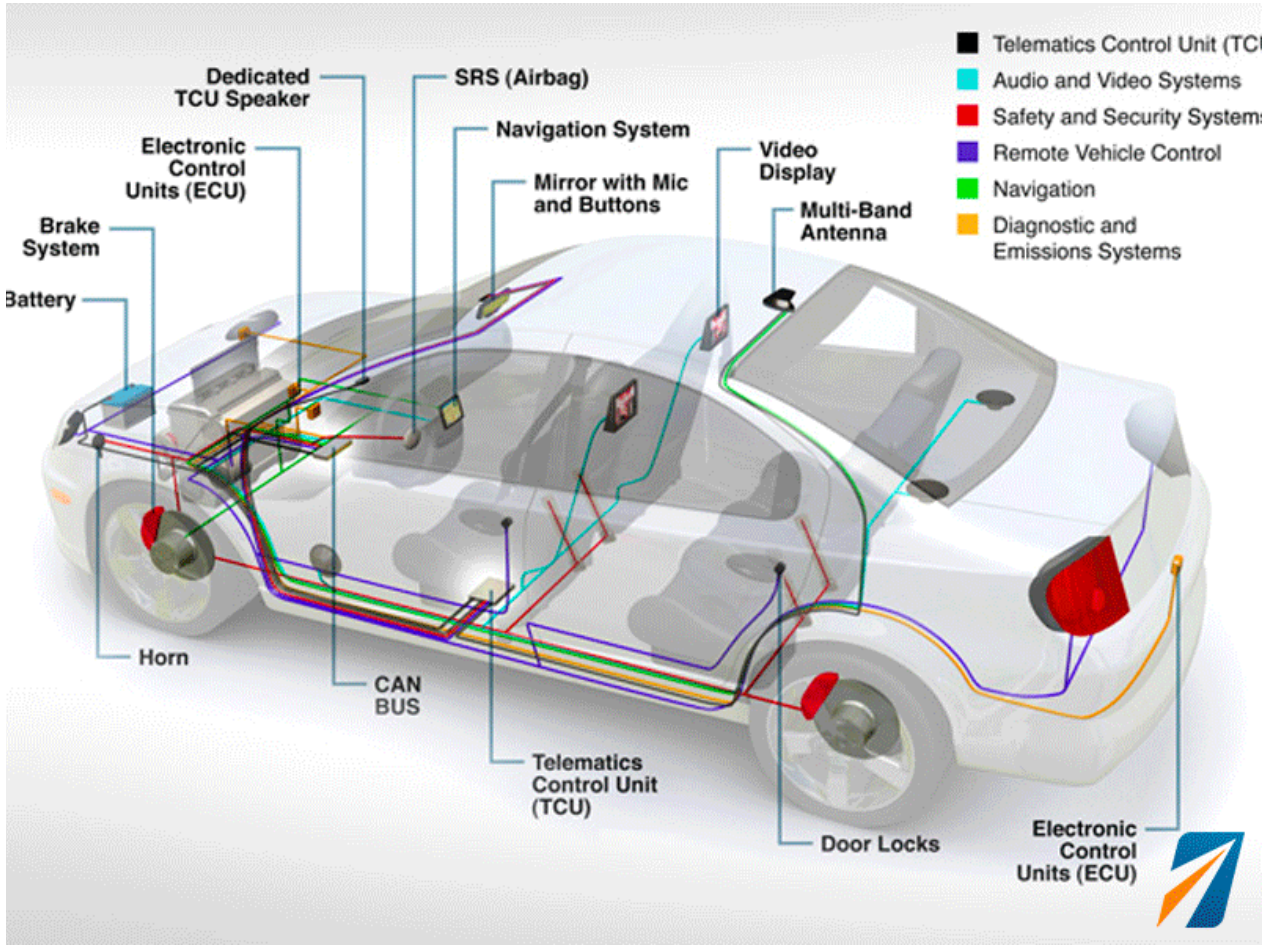
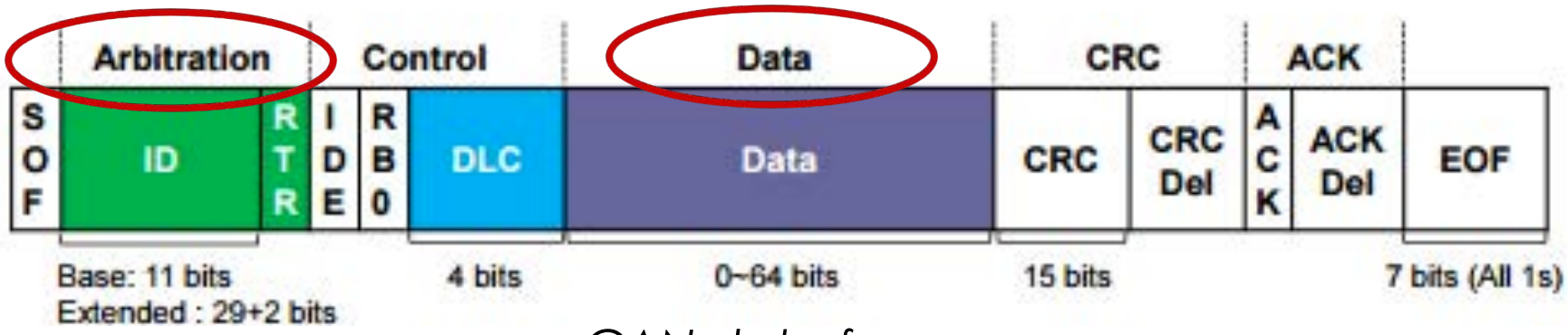


Photo credit: www.fleetistics.com/wp-content/uploads/2016/11/Telematics-Vehicle-Network-System.png

Controller Area Network (CAN)

- CAN Data Frame format:
  - Arbitration ID (AID): indicating priority and indexing contents
  - Data field (or message): containing up to 8 bytes of data



CAN data frame

Photo credit: Cho & Shin, *CCS 2016*

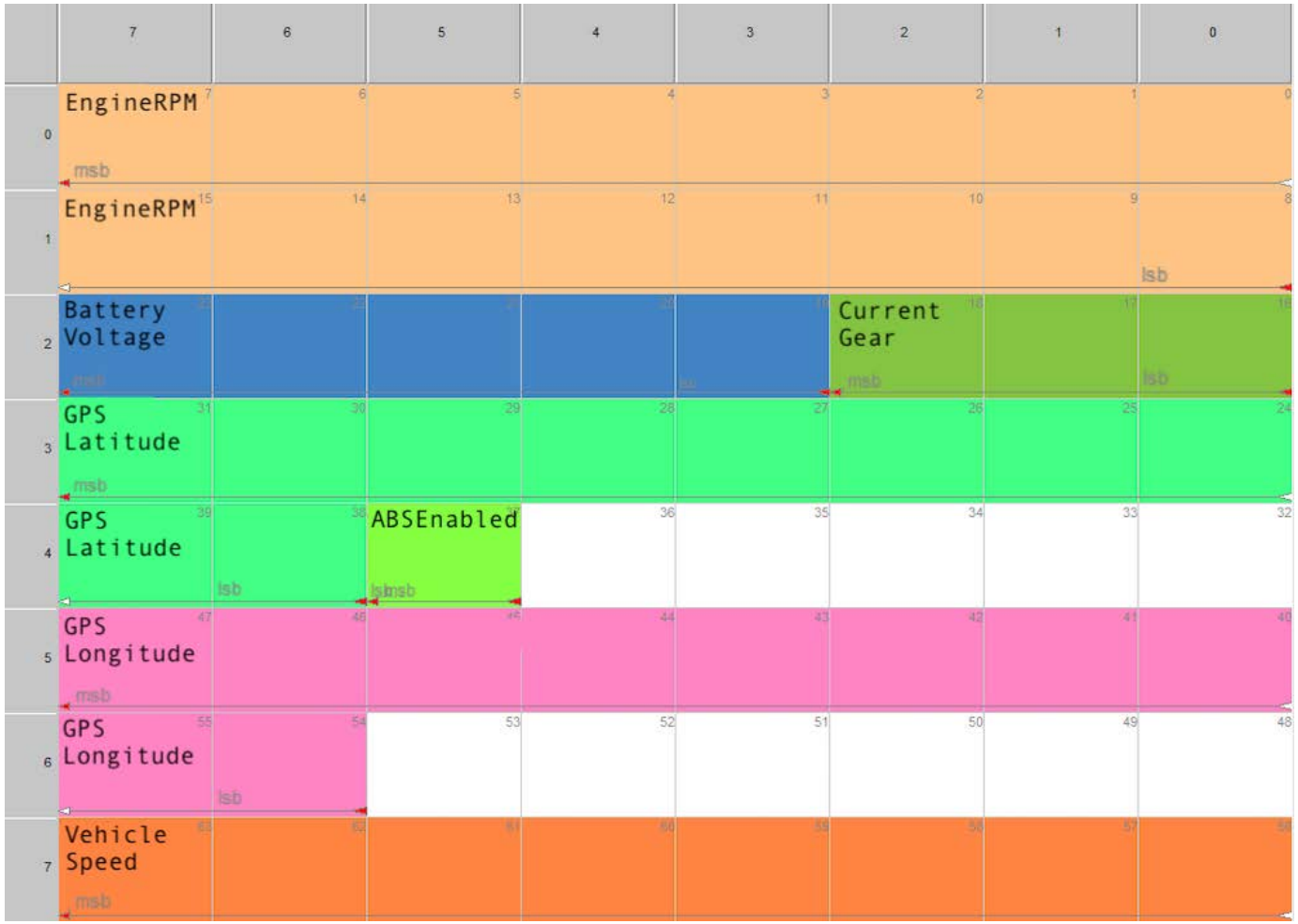
CAN data example

Timestamp AID # data (in hex)

```
(1536160252.263295) can0 223#002000000000004D
(1536160252.263296) can0 3B7#0000000000000000
(1536160252.263296) can0 2C1#0800E2007DC700F9
(1536160252.265273) can0 2D0#06F601007000047
(1536160252.268251) can0 020#000007
(1536160252.269250) can0 0B4#0000000054000010
(1536160252.269252) can0 024#01FE020762007F15
(1536160252.269252) can0 025#0FEE00178787893
(1536160252.270291) can0 2C4#038D001F0080615E
(1536160252.271312) can0 262#0000000069
(1536160252.271313) can0 4DC#1D00010700000000
```

CAN Decoding Problem

- 'Key' for reading data field is proprietary
- Way to read data field differs for every vehicle model, make, year, trim



64-bit CAN data field description defined in DBC: CAN signals (color), unused bits (white)

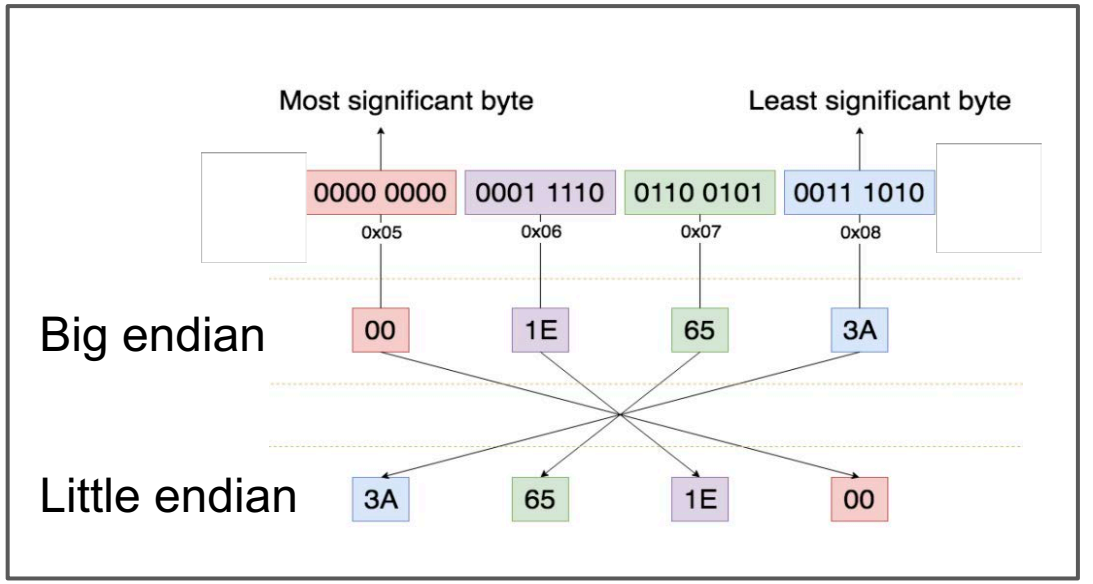
Photo Credit: hackaday.com/2013/10/22/can-hacking-the-in-vehicle-network/

Goal is to reverse engineer signal definitions as in `.dbc` file

1. Bit Position (Start bit index and length)

	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31
	32	33	34	35	36	37	38	39
	40	41	42	43	44	45	46	47
Start bit = 48	**48**	49	50	51	52	53	54	55
	56	57	58	59	60	61	62	63

} *Length = 16*



2. Endianness (Byte Order)

Big endian: 48 is MSB→ 55 → 56 → 63 LSB
Little endian: 56 is MSB→ 63→ 48→ 55 is LSB

3. Signedness (Bit-to-Integer Encoding)

Unsigned: Usual base 2 encoding
Signed: two's complement encoding

Bits	Unsigned Value	Two's Complement Value
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

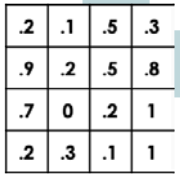
4. Interpretation & Linear Conversion

Physical meaning and units (e.g. speed in MPH, engine temp in F)
Scale and offset factor (to convert the 2^n value to appropriate range)

CAN Decoding Pipeline

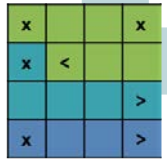
Learn signal boundary probabilities

Get probability of cut between each bit in AID trace using various classification methods



Optimize custom penalty function to learn endianness & signal boundaries

Run Token Packing Optimization

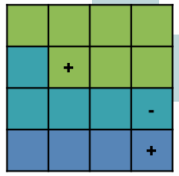


Signed/unsigned classification

Classify each token as signed/unsigned using various classification methods

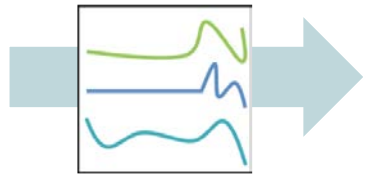
Tokenization & Translation

Interpretation



Diagnostic matching

Determine if any token signal is linearly related to any diagnostic signal



CAN-D Research Results

CAN-D is the **only technology** that can reverse engineer **all four parts** of a signal definition

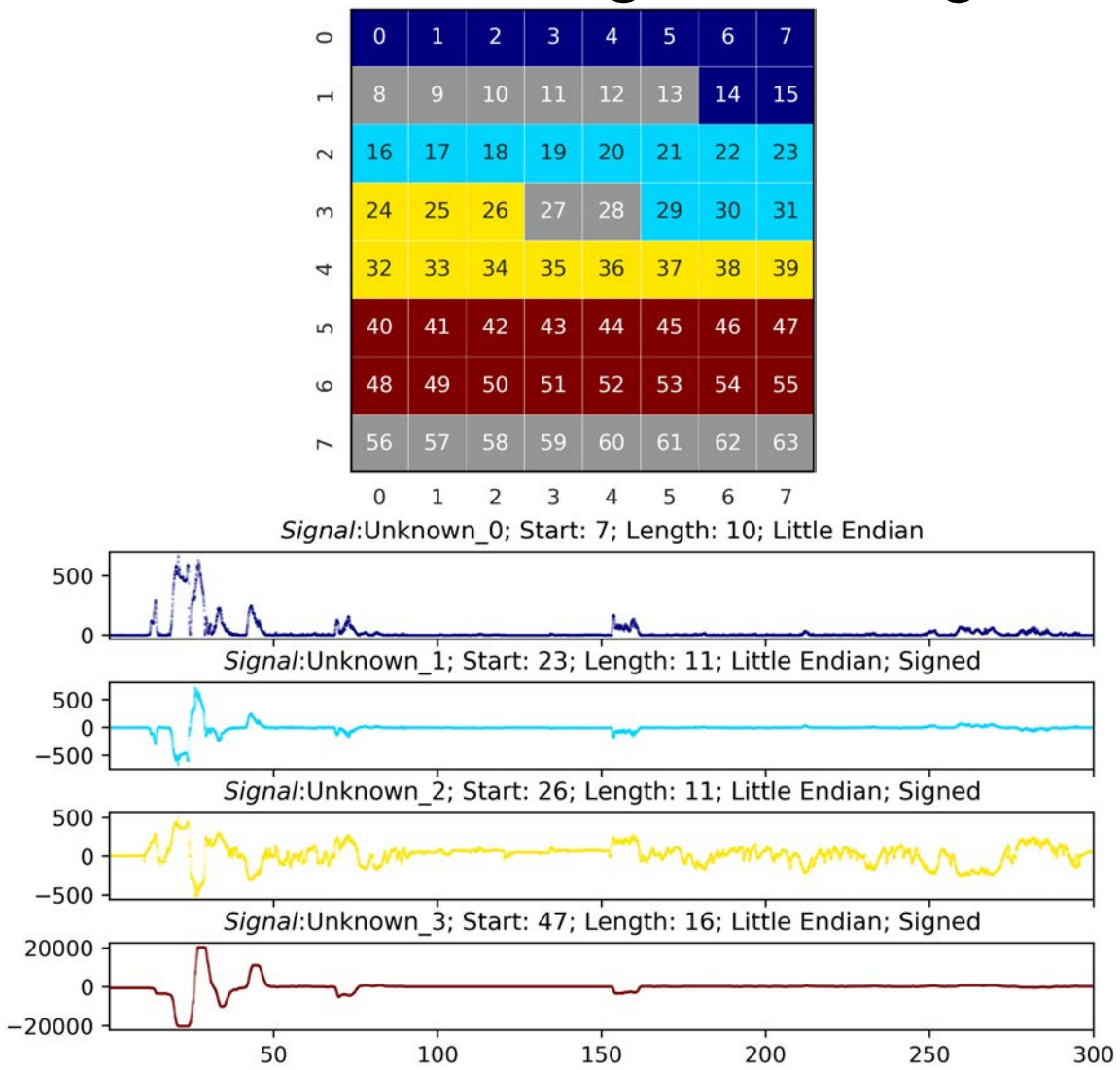
TABLE II: Automotive CAN signal reverse engineering algorithms for each of the four signal properties. CAN-D is the only comprehensive algorithm, determining all four properties.

	Boundary	Endianness	Signedness	Interpretation
Jaynes et al. (2016) [8]	○	○	○	◐
Markowitz & Wool (2017) [2]	●	○	○	○
Huybrechts et al. (2017) [1]	◐	○	○	◐
Nolan et al.'s **TANG** (2018) [3]	●	○	○	○
Marchetti & Stabili's **READ** (2018) [4]	●	○	○	○
Verma et al.'s **ACTT** (2018) [5]	●	○	○	●
Pesé et al. **LibreCAN** (2019) [6]	●	○	○	●
Young et al. (2020) [7]	○	○	○	◐
CAN-D	●	●	●	●

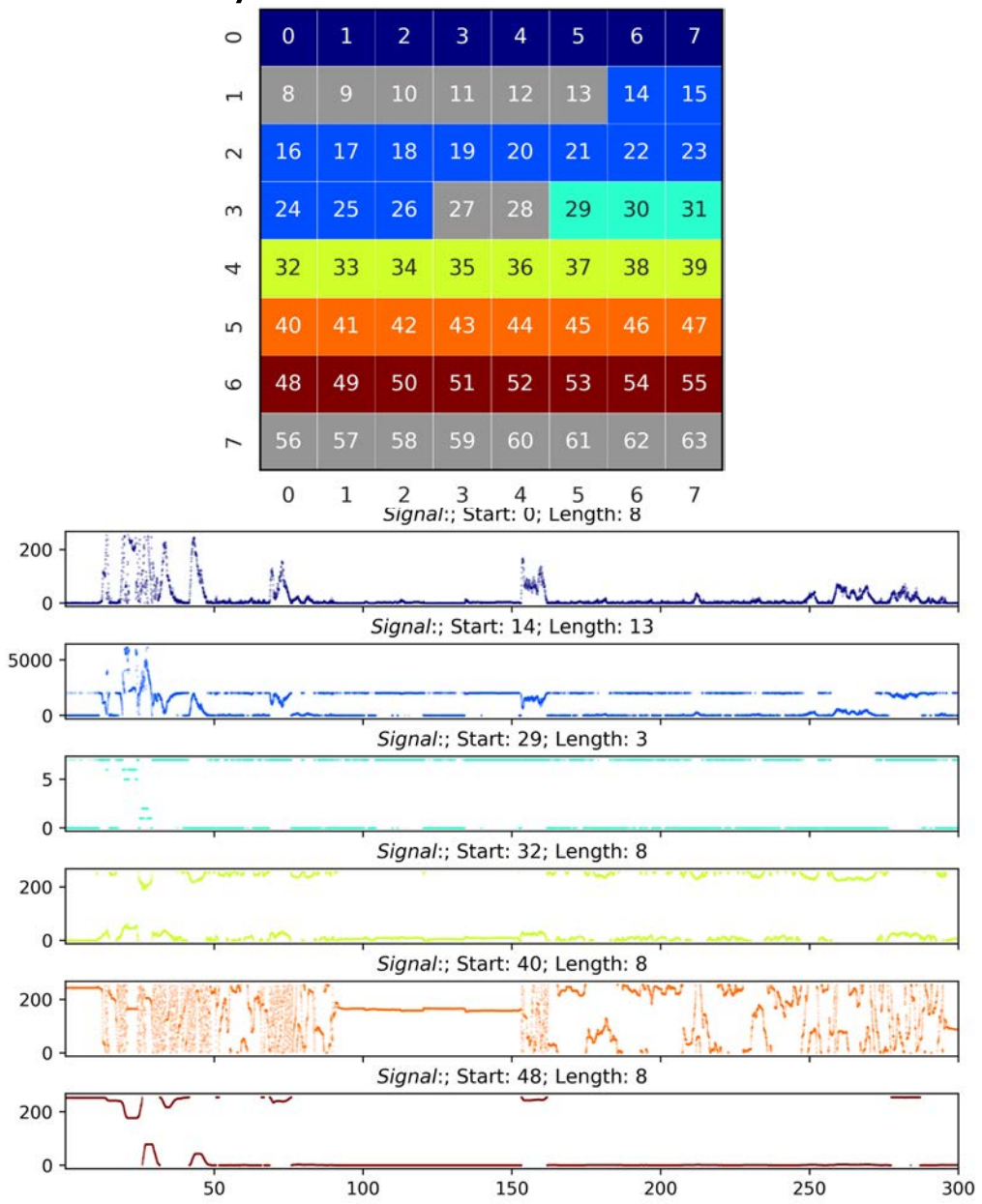
			Baseline	TANG [4]	ACTT[6]	READ [5]	LibreCAN [7]	CAN-D Heuristic	CAN-D ML
Signal Boundary Classif.	F	c	0.0	67.1	*	71.1	78.7	91.6	**93.5**
		f−	0.0	45.6	14.5	63.2	70.6	89.6	**91.2**
		f+	90.1	84.9	89.6	94.4	94.2	98.1	**98.4**
	P	c	0.0	62.6	*	94.8	86.8	**97.2**	96.4
		f−	0.0	31.8	35.2	80.6	64.2	**89.9**	87.6
		f+	**100.0**	75.7	96.5	97.6	92.4	98.2	97.6
	R	c	0.0	72.4	*	56.8	71.9	86.7	**90.8**
		f−	0.0	80.3	9.1	51.9	78.4	89.4	**95.2**
		f+	82.0	96.5	83.7	91.4	96.1	98.1	**99.1**
Mean ℓ^1 Signal Error	CAN Log	#1	18.9	30.0	18.9	20.9	25.4	4.0	**3.9**
		#2	11.2	16.3	12.7	9.7	11.1	**0.7**	1.9
		#3	20.9	25.6	20.9	19.8	20.5	**1.3**	2.1
		#4	4.4	13.8	6.4	2.3	3.5	**1.3**	1.7
		#5	11.9	8.9	11.9	8.8	3.7	3.3	**2.3**
		#6	11.0	18.4	11.0	10.6	9.7	4.9	**3.4**
		#7	6.1	16.4	7.9	5.5	7.6	**0.0**	**0.0**
		#8	9.4	16.7	9.4	7.9	10.2	**0.2**	1.5
		#9	11.1	14.2	10.6	12.0	12.8	1.5	**1.2**
		#10	9.8	14.1	11.3	7.9	9.9	**0.7**	0.9
	Average		11.5	17.4	11.9	10.5	11.4	**1.8**	1.9

CAN-D is the most accurate signal reverse engineering technology by far (**80% less average error**)

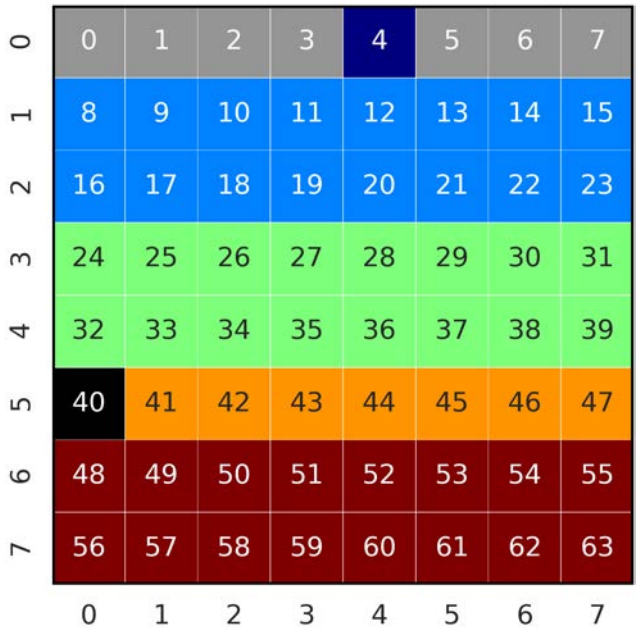
Our algorithm's translation of an AID with 4 steering-related signals

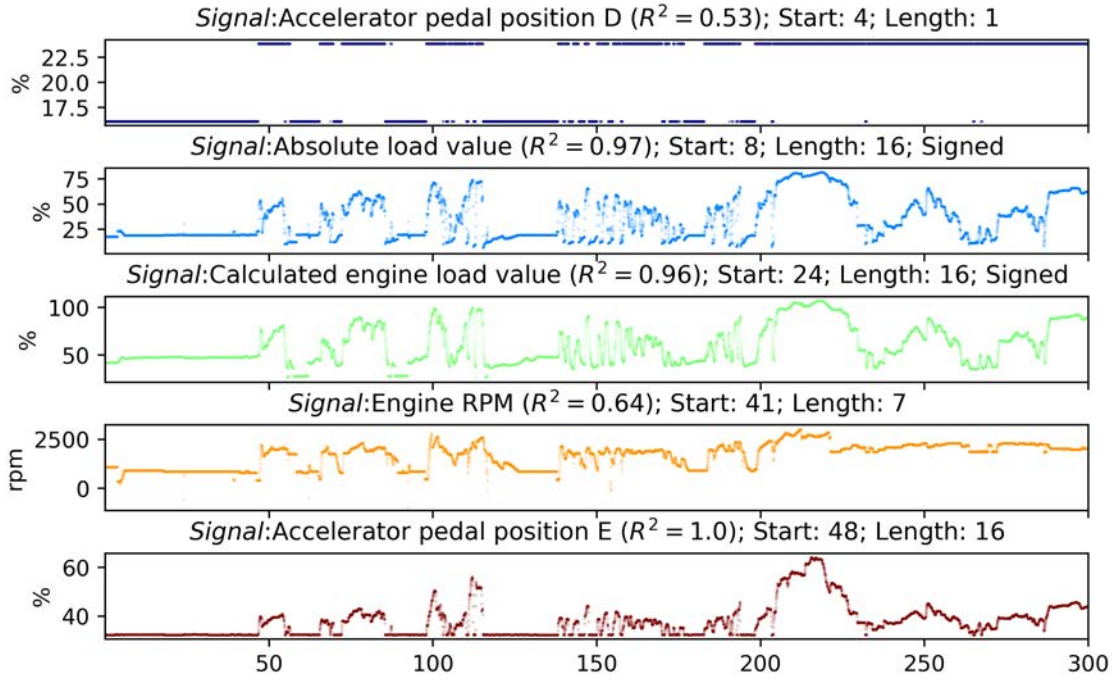


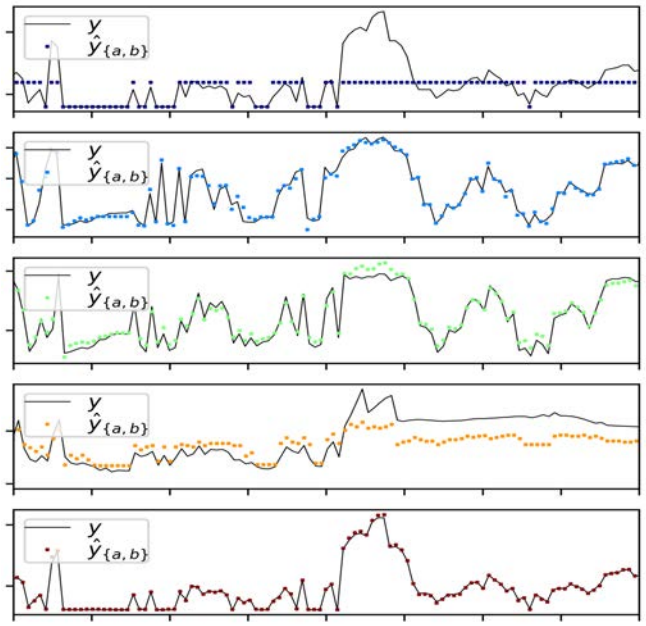
Competitors' translation (all nearly the same)



Our Signal Interpretation







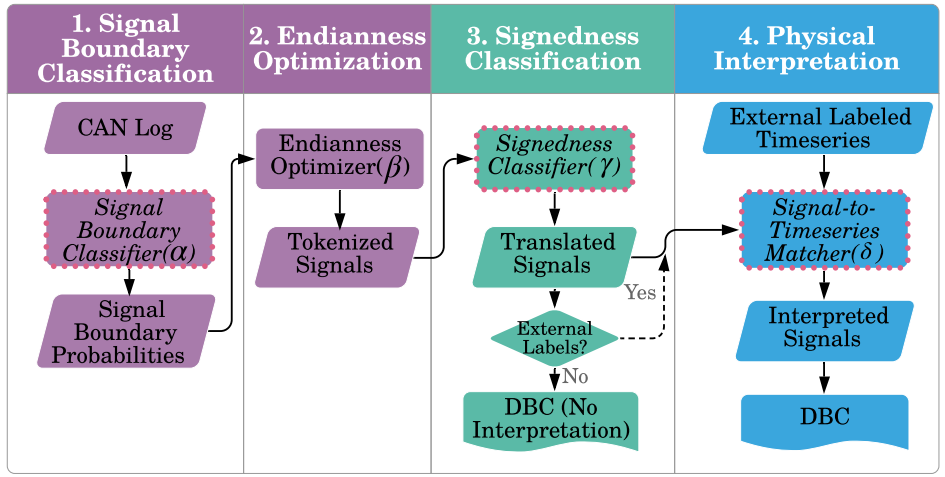
- 5 engine-related payloads in this AID's data field translated
- Unified Diagnostic Services (DIDs) used to automatically interpret the translated signals (middle plots with labels)
- Translated time series interpolated with DID time series (right plots)

CAN-D: Controller Area Network Decoder

CAN data

```
(1536160252.263295) can0 223#0020000000000004D
(1536160252.263296) can0 3B7#0000000000000000
(1536160252.263296) can0 2C1#0800E2007DC700F9
(1536160252.265273) can0 2D0#06F6010070000047
(1536160252.268251) can0 020#000007
(1536160252.269250) can0 0B4#0000000054000010
(1536160252.269252) can0 024#01FE020762007F15
(1536160252.269252) can0 025#0FEE000178787893
(1536160252.270291) can0 2C4#038D001F0080615E
(1536160252.271312) can0 262#0000000069
(1536160252.271313) can0 4DC#1D00010700000000
```

Algorithmic Pipeline



Signal Definitions `.dbc` file

Step 1: One-time per vehicle type training

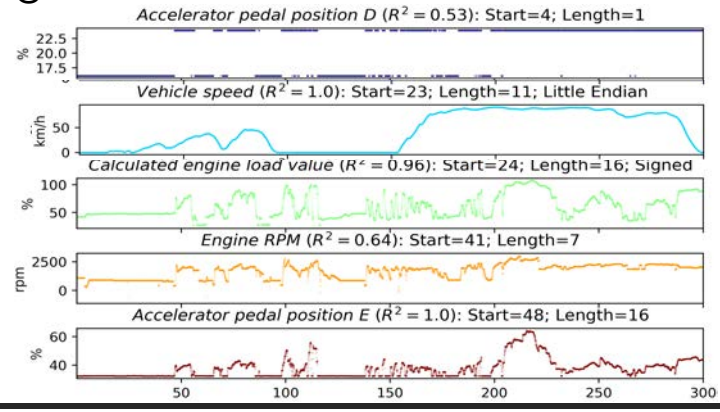
Step 2: Real-time or post-drive decoding

CAN data

```
(1536160252.263295) can0 223#0020000000000004D
(1536160252.263296) can0 3B7#0000000000000000
(1536160252.263296) can0 2C1#0800E2007DC700F9
(1536160252.265273) can0 2D0#06F6010070000047
(1536160252.268251) can0 020#000007
(1536160252.269250) can0 0B4#0000000054000010
(1536160252.269252) can0 024#01FE020762007F15
(1536160252.269252) can0 025#0FEE000178787893
(1536160252.270291) can0 2C4#038D001F0080615E
(1536160252.271312) can0 262#0000000069
(1536160252.271313) can0 4DC#1D00010700000000
```

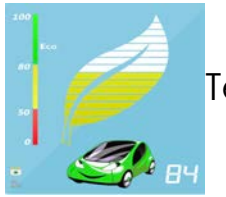
Signal Definitions `.dbc` file

Translated Signals



Enabling technologies, for example

Cyber Security Technologies



Efficiency Technologies

Performance Tuning Technologies

Fleet Management Technologies

OAK RIDGE National Laboratory

Jordan Sosnowski

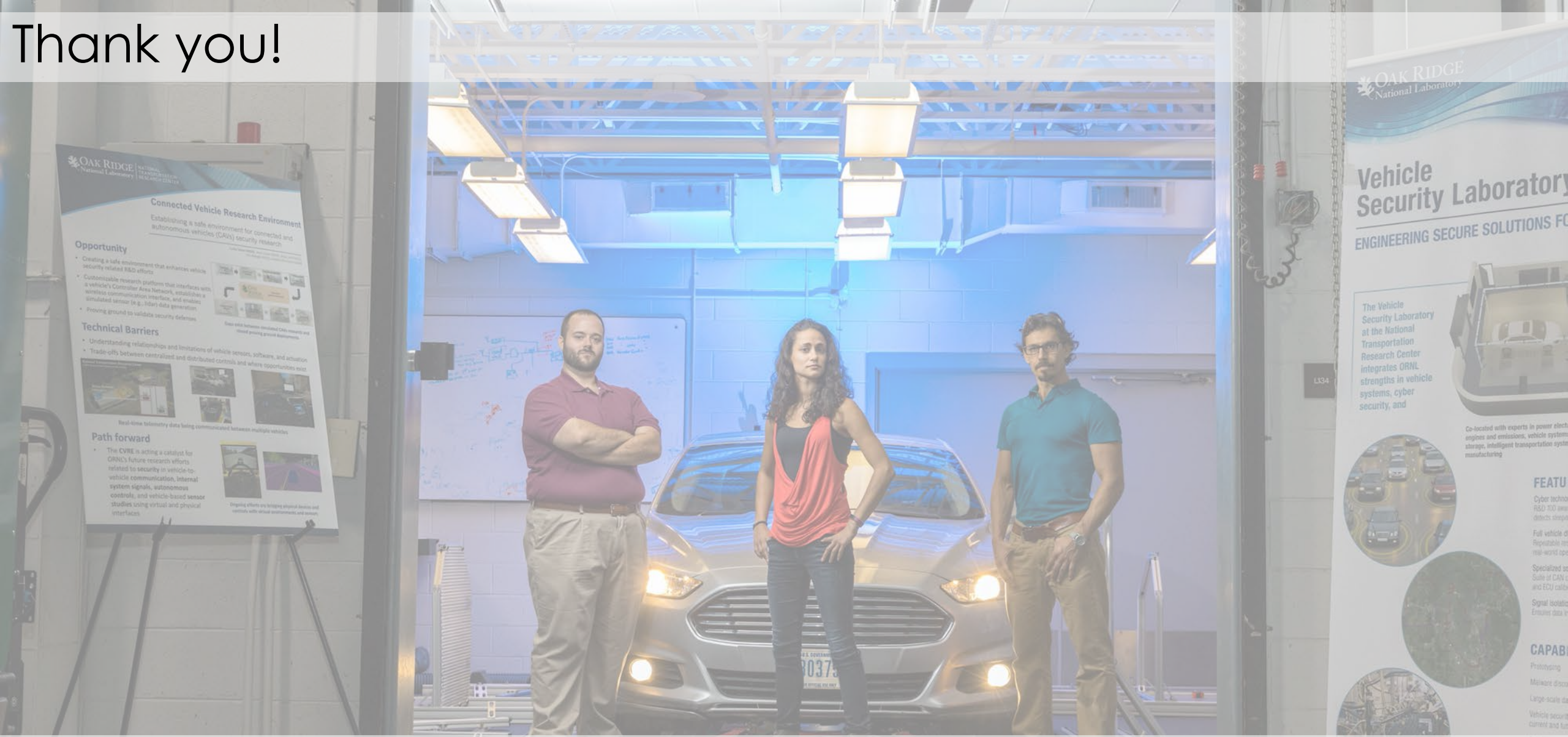
Technology Solution

- Modular 4-step algorithmic pipeline to reverse engineer signal definitions from raw CAN data in a once-per-vehicle-type training
- Outputs are CAN signal definitions in a standard format (`.dbc` file)
- Can be used in real-time translation or post-drive analysis
- TRL 5
- Demo videos: One-Time Training & In-Situ Decoding
- Detailed documentation `arxiv.org/abs/2006.05993`

Two plug-in prototypes

- Raspberry PI + CANberry 2.0 (pictured)
- Nvidia Jetson TX2

Thank you!



Questions?

Stacy Prowell, prowellsj@ornl.gov