SAE J1939-specific Cyber Security for Medium and Heavy-Duty Vehicles

Subhojeet Mukherjee

Ph.D. Final Defense

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Organization

- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Medium and Heavy-Duty (MHD) Vehicles

- Part of the US Critical Infrastructure
  - Goods transport
  - Package delivery
  - Emergency services
  - School transport
  - Public transport
- About six different classes of vehicles weighing 14,001 pounds and above
  - About 13,000,000 registered trucks and buses on road in 2019

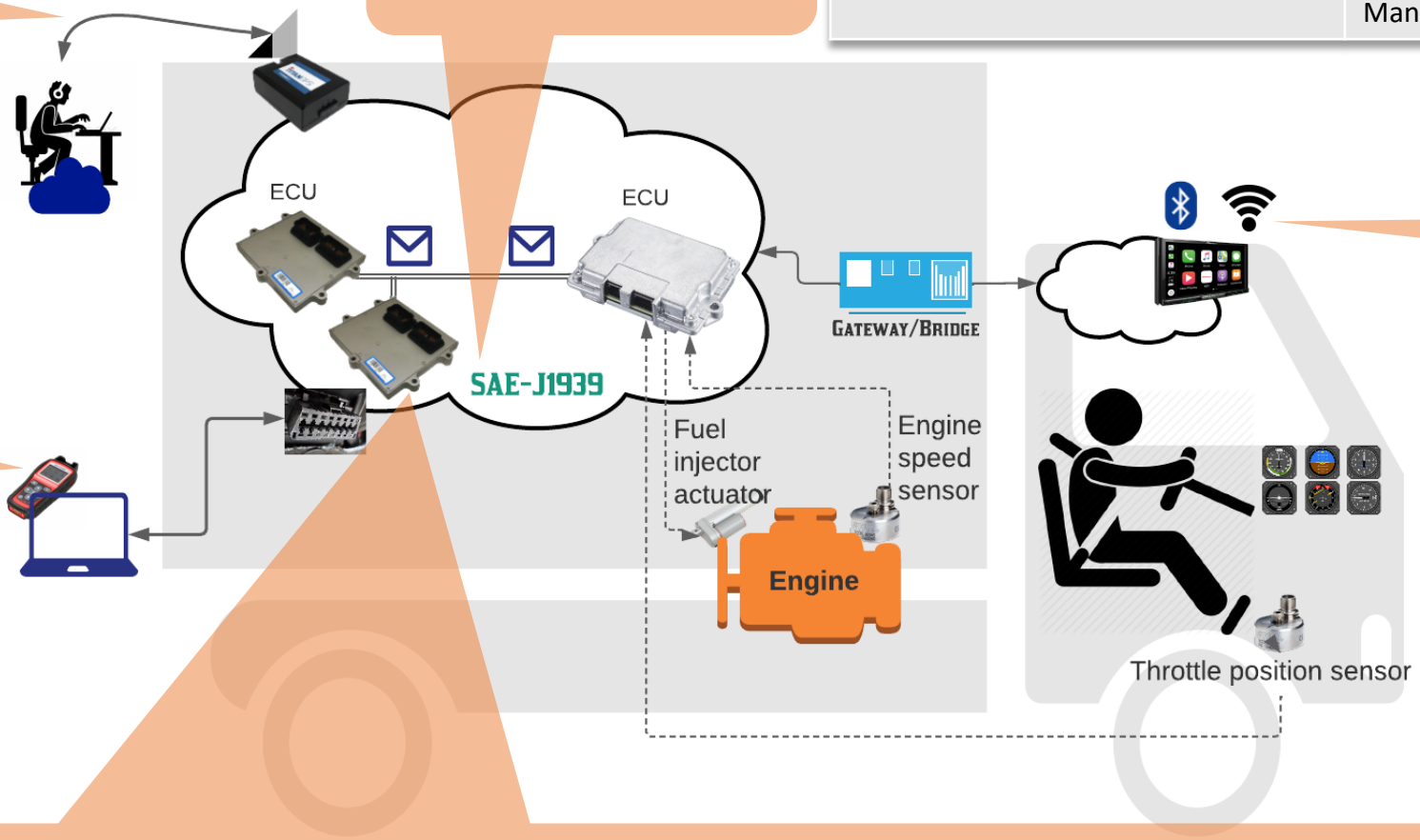
Modern Electronification

SAE J1939 Documents	Layer
SAE J1939-11, 13, 14, 15, 16	Physical
SAE J1939-21	Data-Link
SAE J1939-31	Network
SAE J1939-71, 73, 74, 75, Digital Annex	Application
SAE J1939-81, 82, 84	Network Management

Controller Area Network (CAN 2.0)

Layered networking standard: SAE J1939

Long distance telematics



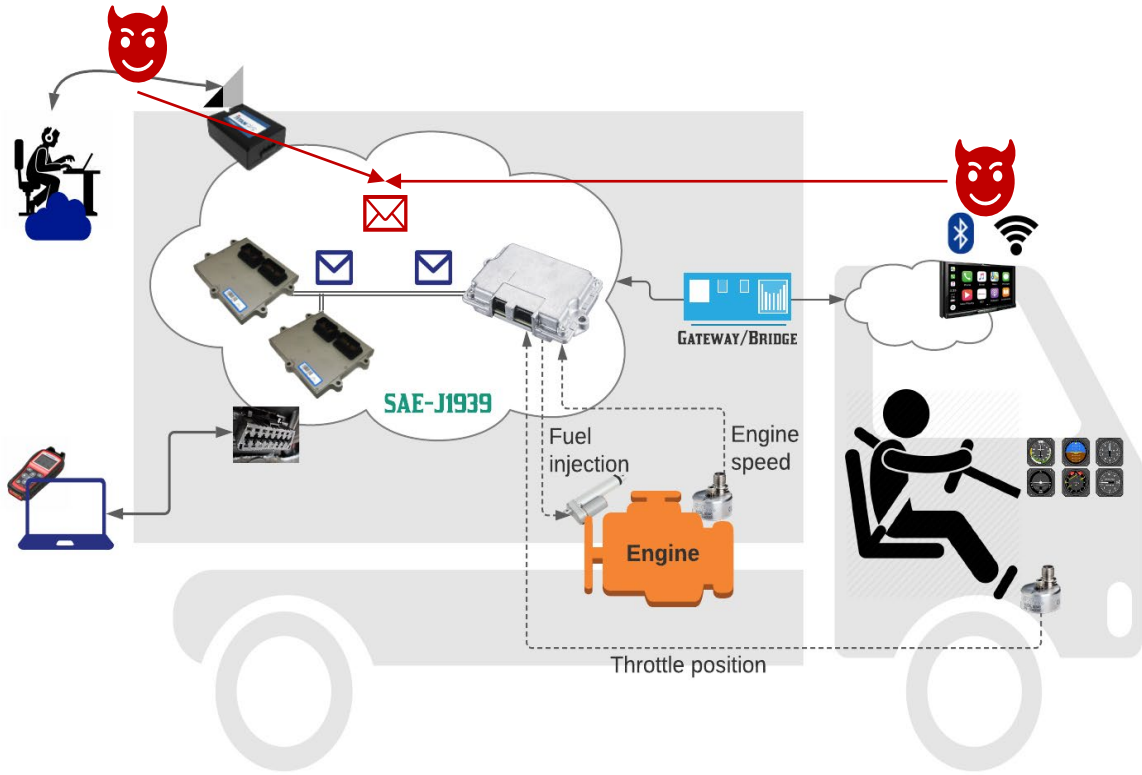
Wireless entertainment

Physical diagnostics

Mechanical operations controlled by low voltage embedded electronic control units (ECU)

Remote Threats to In-Vehicle Electronics

- Remote access can be obtained through open and vulnerable channels [Chec11, Mill14]
- Openly available standards can be leveraged to craft attacks
- Messages can be transmitted to
  - Control vehicle operations
  - Disrupt vehicle operations
  - Spoof vehicle information to the driver, fleet manager, etc.
- Practical attacks have been demonstrated at the application and network management layer of the SAE J1939 specifications

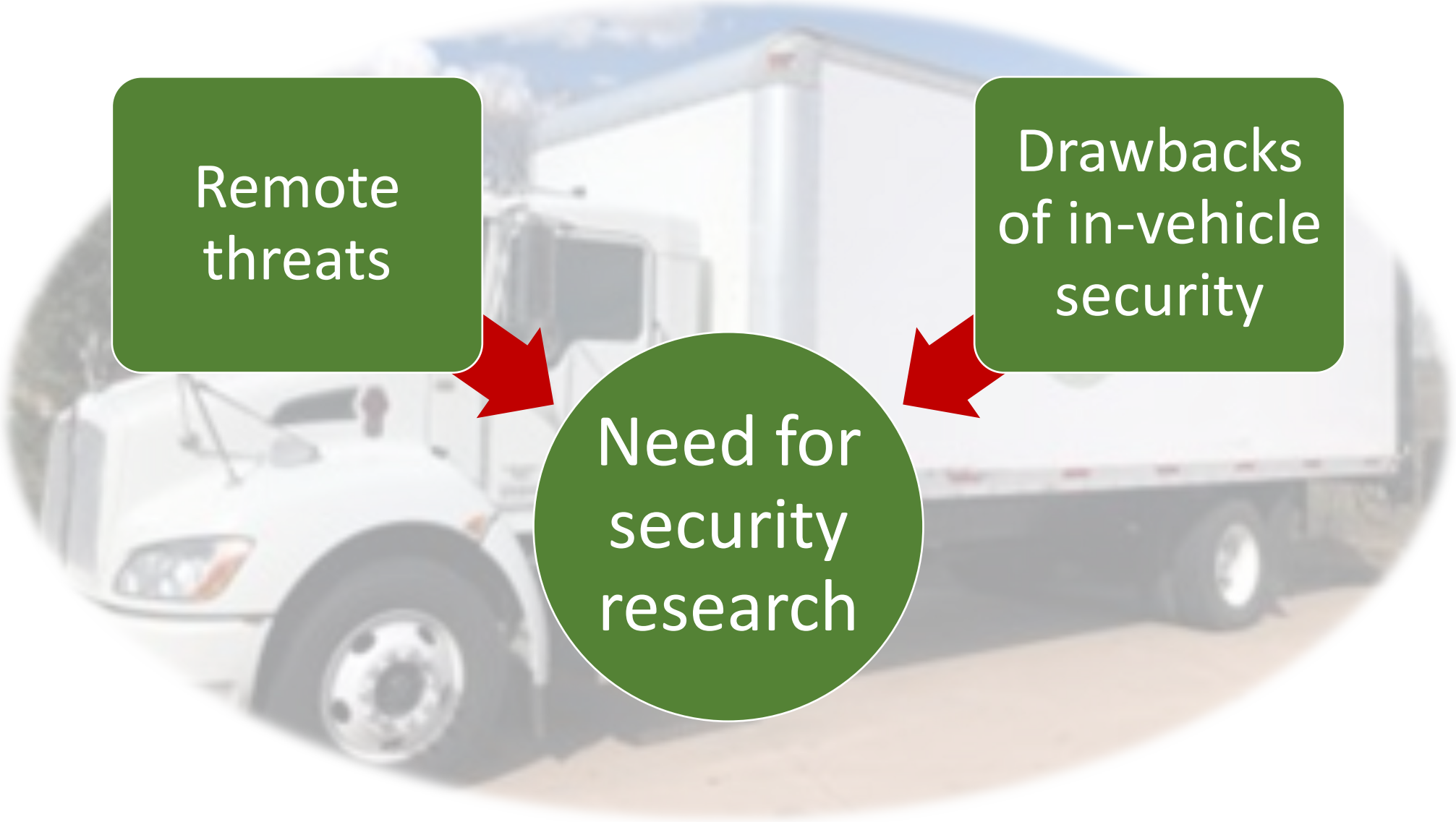


State-of-the-art in In-Vehicle Security

Mostly for passenger vehicles

Four directions of research having drawbacks

1. Behavioral anomaly-based
   - Current solutions are offline trained
2. Rule-based
   - Current solutions use rules based on message content only
3. Sender authentication-based
   - Attacks can be security enforcement abiding
4. Specification-based
   - Attacks can be security enforcement abiding


Remote threats
Drawbacks of in-vehicle security
Need for security research

Research Questions And Contributions

Can **weaknesses** in the **data-link layer** specifications of SAE J1939 be exploited to **attack in-MHD vehicle ECUs**?

Can a system be designed to **detect network anomalies** on an SAE J1939 network in an **online manner**?

Can a **rule-based system** be designed to **detect threatening SAE J1939 messages as they are being transmitted based on features other than message content**?

Three denial-of-service attacks on the data-link layer specifications that can disrupt normal operations of an ECU

An **online anomaly-based intrusion detection system** that

- Models network behavior through SAE J1939 specified concepts
- Flags abnormal deviations from normal behavior as security infringements

A **rule-based intrusion detection and prevention system** is presented that

- Allows identifying **malicious messages using features other than message content only**
- Is **real-time** and can be used to disrupt malicious messages in transmission

Organization

Introduction

Background

- Primer on SAE J1939 and Controller Area Network (CAN)
- Research Trucks
- Threat Model
- Review of In-Vehicle Security Solutions

Contribution 1

Contribution 2

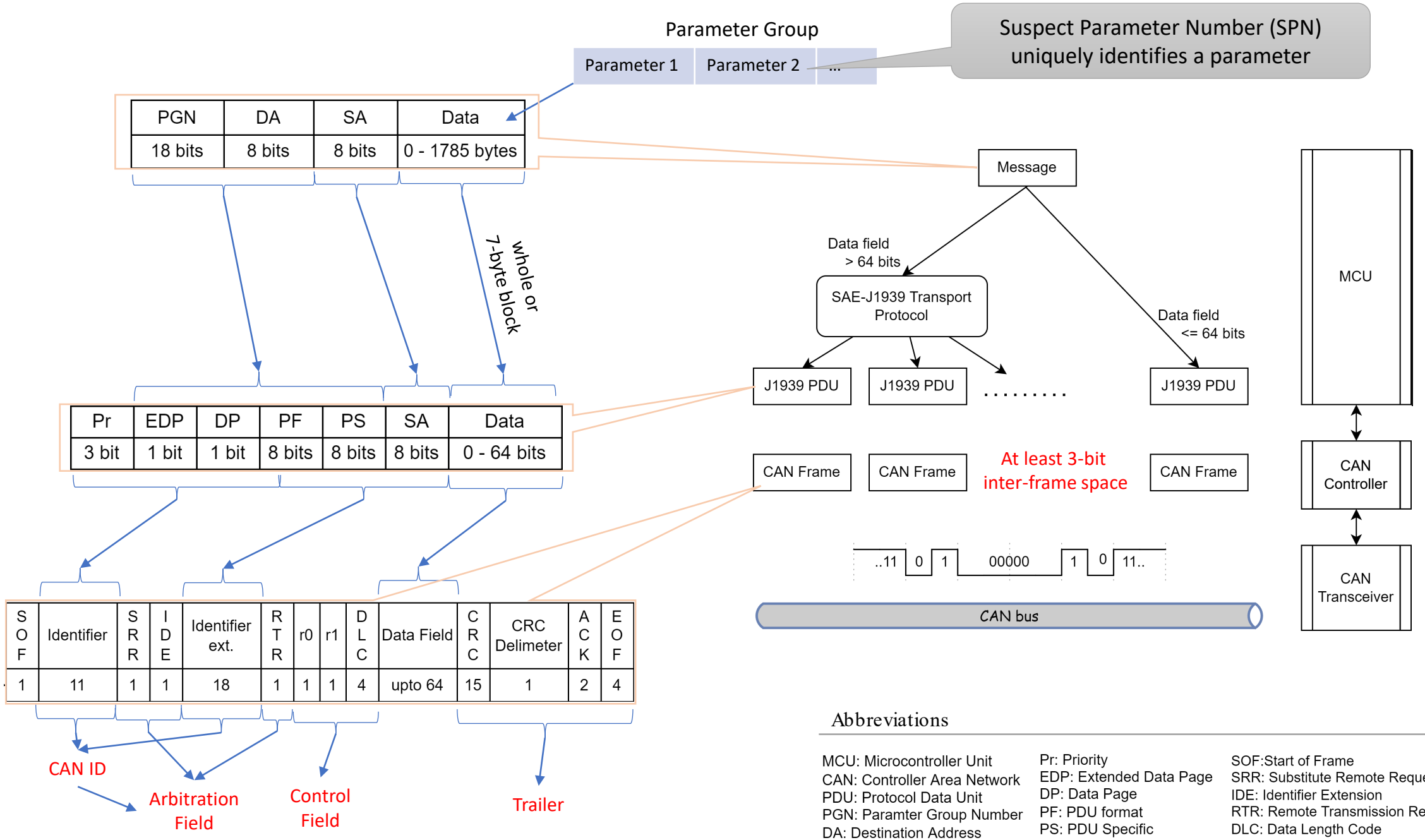
Contribution 3

Conclusion and Future Work

Organization

- Introduction
- Background
  - Primer on SAE J1939 and Controller Area Network (CAN)
  - Research Trucks
  - Threat Model
  - Review of In-Vehicle Security Solutions
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

SAE J1939 Message Processing



Lowest CAN ID wins bus arbitration

Abbreviations

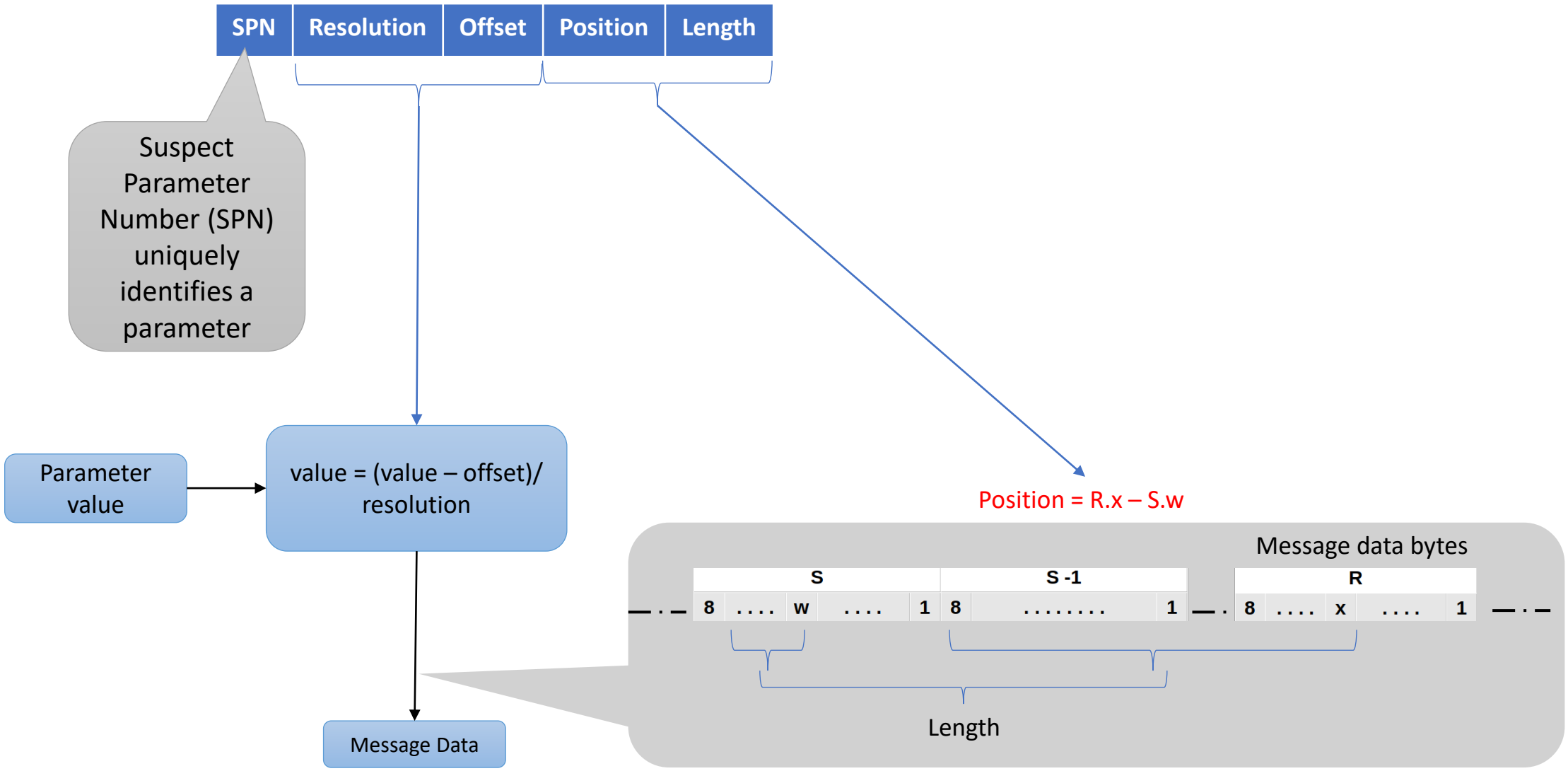
MCU: Microcontroller Unit
CAN: Controller Area Network
PDU: Protocol Data Unit
PGN: Paramter Group Number
DA: Destination Address
SA: Source Address

Pr: Priority
EDP: Extended Data Page
DP: Data Page
PF: PDU format
PS: PDU Specific
SA: Source Address

SOF:Start of Frame
SRR: Substitute Remote Request
IDE: Identifier Extension
RTR: Remote Transmission Request
DLC: Data Length Code
CRC: Cyclic Redundancy Check
ACK: Acknowledgement
EOF: End of Frame

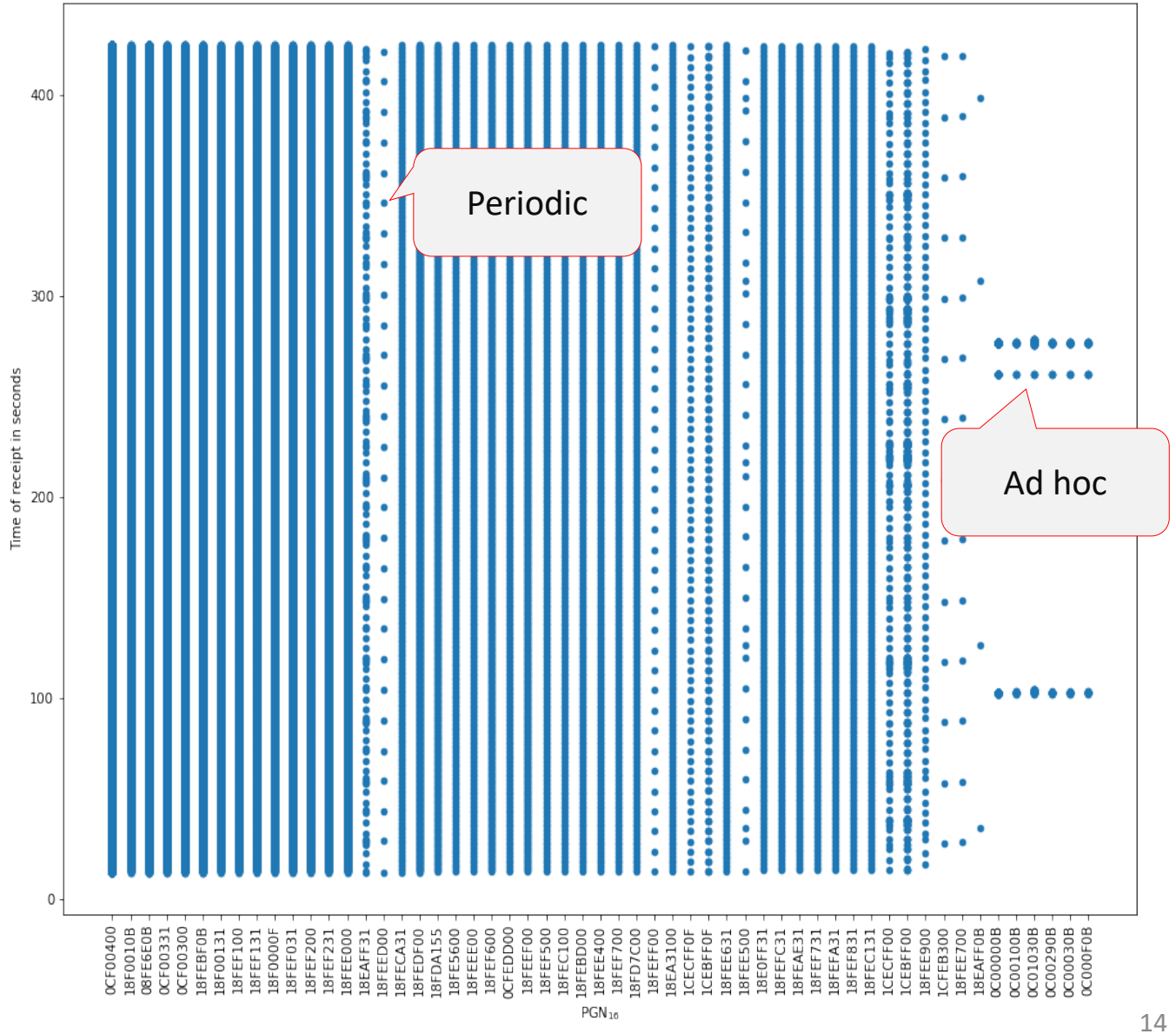
Parameter Placement

SAE J1939 Digital Annex



Parameter Group Transmission

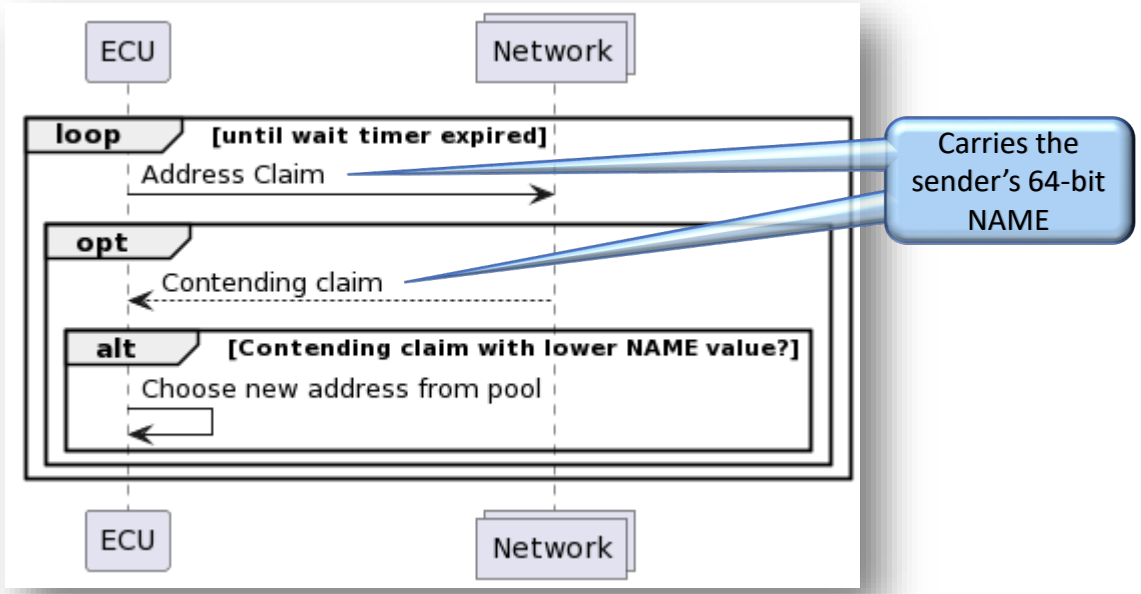
- Two types of transmission types
  - Periodic
    - Most messages on the network are transmitted periodically
  - Ad hoc
    - E.g. Request-responses, commands



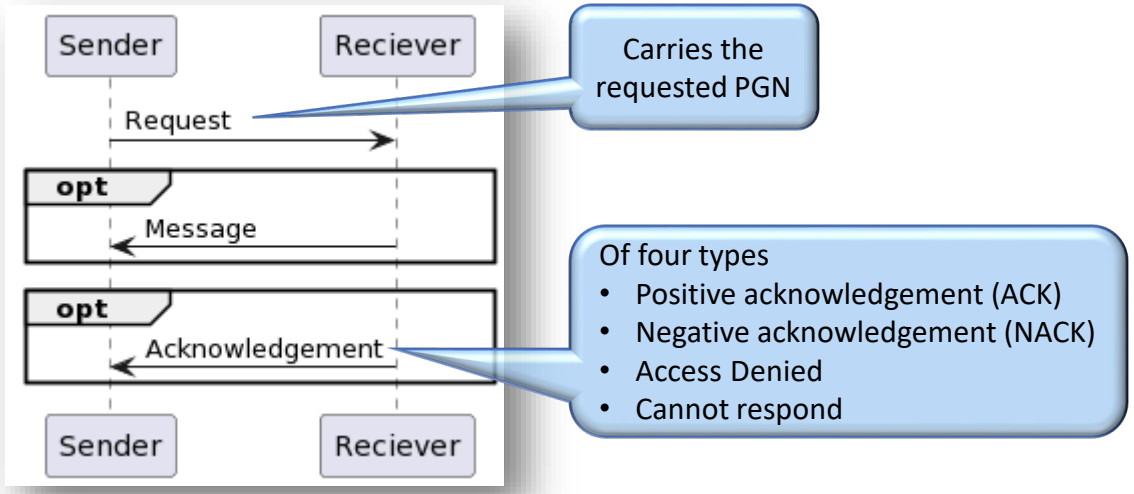
Protocols

Abbreviations
RTS: Request to Send
CTS: Clear to Send
DT: Data Transfer
EoMA: End of Message Acknowledgment

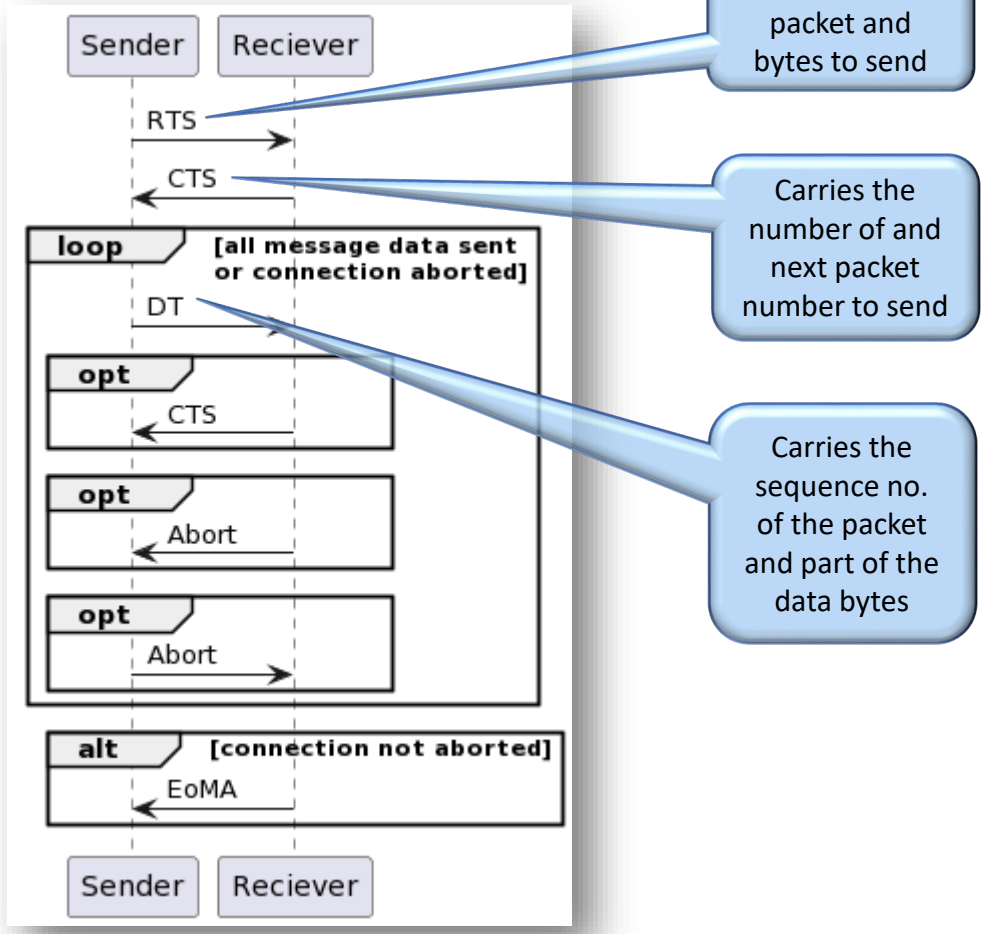
Address Claiming



Parameter Requesting



Destination-specific Multipacket Message Transfer



Organization

- Introduction
- Background
  - Primer on SAE J1939 and Controller Area Network (CAN)
  - Research Trucks
  - Threat Model
  - Review of In-Vehicle Security Solutions
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Research Trucks

- PACCAR PX-7-powered 2014 Kenworth T270 truck
  - Readily accessible for experimentation
- Accessed SAE J1939 network description
  - 3 ECUs
    - Cummins 2350 engine control module
      - Hosts engine controller and engine retarder applications
    - Bendix EC-60 brake ECU
    - Allison RDS-2000 transmission ECU
  - 250 kbps CAN bus
- 600 RPM idle engine speed

- PACCAR MX-13-powered 2015 Kenworth T660 truck
  - No physical access
- Access to recorded traffic
  - ~ 7 minutes long drive around an industrial block
    - Included three hard braking events
  - 137318 messages
  - Five transmitting controllers
    - Engine, brake, retarder, cab, diesel particulate filter
  - 41 unique parameter groups, 35 periodic and 6 ad hoc

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Primer on SAE J1939 and Controller Area Network (CAN)

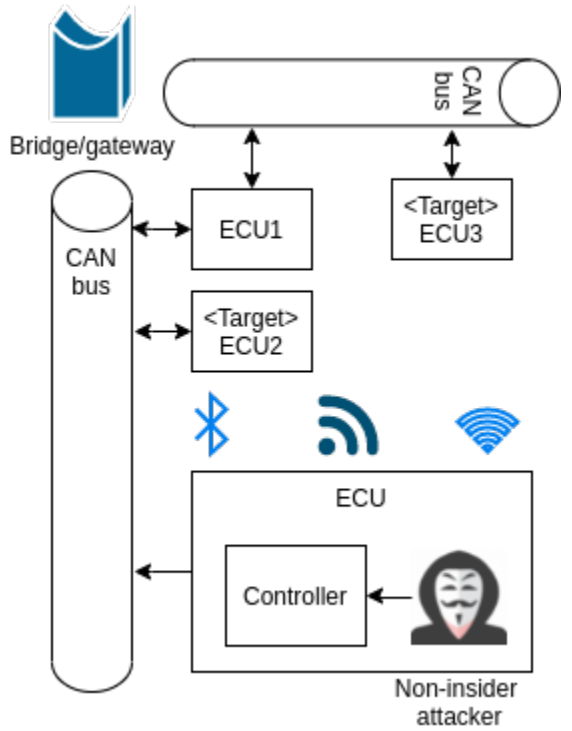
Research Trucks

Threat Model

Review of In-Vehicle Security Solutions

Attacker Capabilities

- Cannot
  - Physically access the truck
  - Bypass CAN controller
  - Manipulate bits in transit
  - Exercise insider capabilities
    - E.g. reprogram ECUs
- Can
  - Read, interpret and write SAE J1939 messages
    - Spoof ECU addresses while sending
  - Communicate with ECUs on the same or different networks to which they have access to



Attack Strategies: High Volume Denial-of-service (HVDoS)

https://youtu.be/ICl6v8SIjOY

Description

Consume the resources available to the ECUs through rapid injection of SAE J1939 messages

Demonstration

Published Attacks

- Network overload attack [Mill13]
  - Gain exclusive access to the bus by sending a high volume of frames with an ID 0 to win as many arbitrations as possible

Attack Strategies: Low Volume Denial-of-service (LVDoS)

https://youtu.be/HE-JZi_nS7I

Demonstration

Description

Disable ECU services by injecting messages at a normal rate

Published Attacks

- Address claim attack [Murv18]
  - Claim ECU address(es) by sending address claim message with lower NAME value

Attack Strategies: Command and Control (CnC)

Description

Sending SAE-J1939 defined messages to command cyber-physical functions of ECUs

https://youtu.be/ZfXkDPU3WMQ

Demonstration

Published Attacks

- Engine control attack [Bura16]
  - Control engine speed by sending engine control request messages with PGN 0x00000
- Retarder jam attack [Bura16]
  - Disable engine retardation by sending 0% torque to the engine retarder in messages with PGN 0x00000 when vehicle is at speeds below 30 mph
- Pedal jam attack [Bura16]
  - Disable acceleration by sending very low (< 0%) torque to the engine controller in messages with PGN 0x00000

Attack Strategies: Fuzzing

Description

Sending random CAN ID and data bytes sent at high rates to understand ECU capabilities and/or invoke erratic behavior

Published Attacks

- No published instances on medium and heavy-duty vehicles
  - Gauges on the instrument cluster moved in an erratic manner on the Kenworth T270 research truck but no physical impact noticed

Organization

- Introduction
- Background
  - Primer on SAE J1939 and Controller Area Network (CAN)
  - Research Trucks
  - Threat Model
  - Review of In-Vehicle Security Solutions
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Classification	Method	Pros	Cons
Sender Authentication-based	• Authenticate messages through digital signatures created using pre-shared keys	No false positives	• Resource intensive [Aliw18] • Unable to detect attacks from legitimate but compromised senders • Introduces communication overhead • Key management can be challenging
Behavioral Anomaly-based	• Learn normal behavior from offline collected data • Flag abnormal deviations from normal as attack	Can detect unknown (0-day) attacks	• Does not account for normal behavior that is not encountered during the training phase [Stac19]
Specification-based	• Build reference model for normal behavior using manufacturer specifications • Flag deviations from normal as attack	Can detect unknown (0-day) attacks	• Unable to detect attacks that obey specifications
Rule-based	• Create a database of attack patterns based on CAN ID and data • Flag frames as malicious if matching patterns are found in the database	Low false positives	• Not all malicious CAN frames can be identified based on their ID and data

Organization

- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Three denial-of-service attacks on the data-link layer specifications

- The Testing Setup
- The Request Overload Attack
- The Connection Exhaustion Attack
- The False Request To Send (RTS) Attack
- Summary

Organization

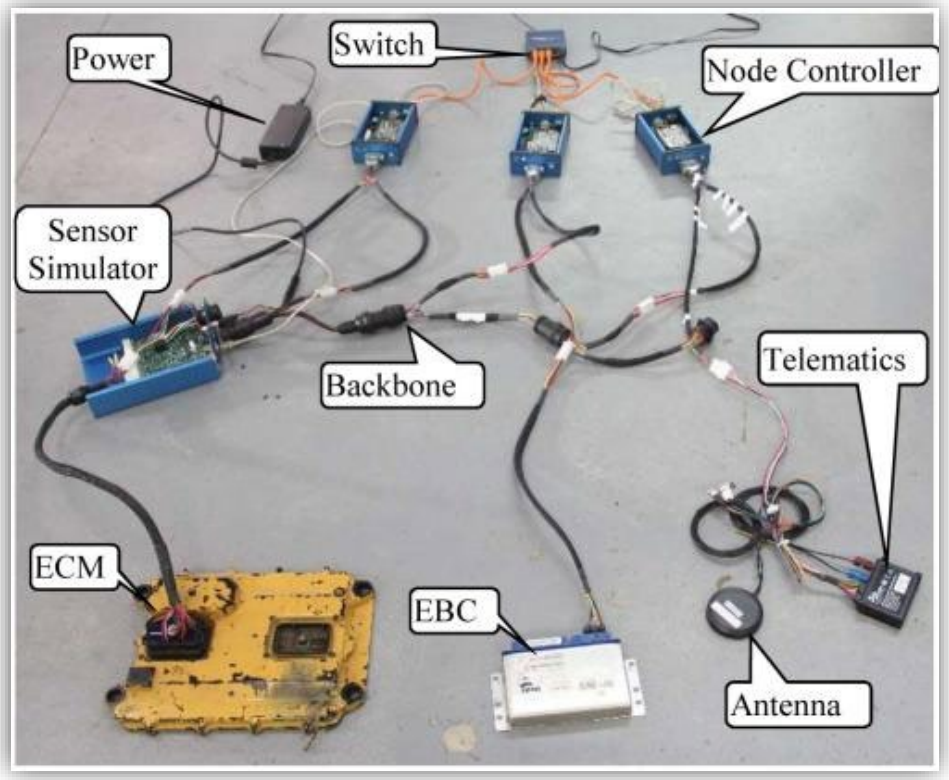
- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Three denial-of-service attacks on the data-link layer specifications

- The Testing Setup
- The Request Overload Attack
- The Connection Exhaustion Attack
- The False Request To Send (RTS) Attack
- Summary

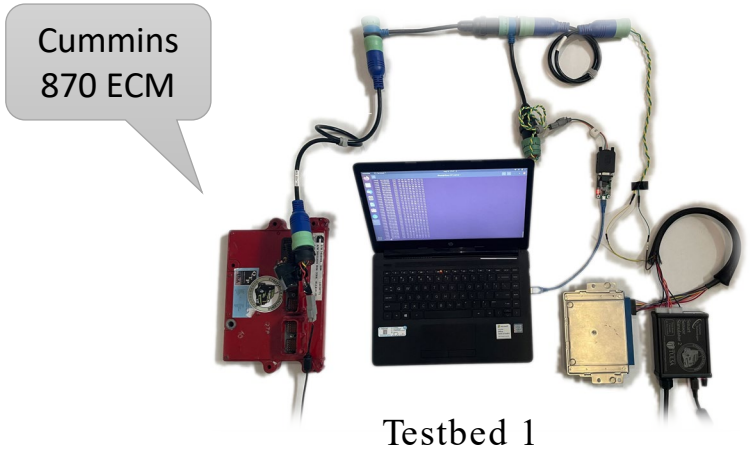
Testing Setup 1 (Remote Testbench)

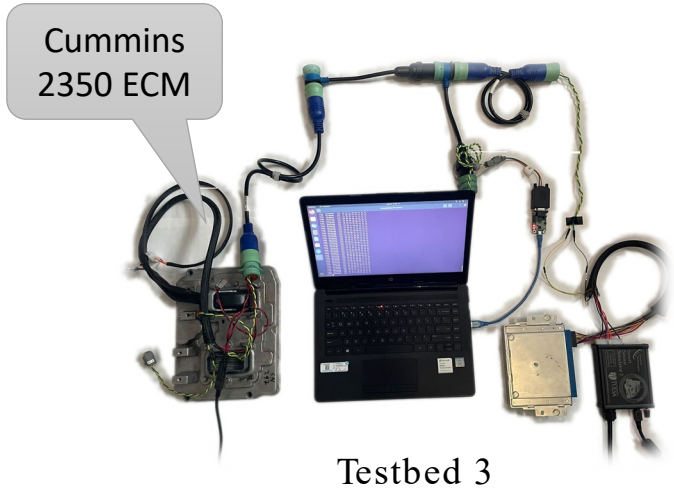
- Setup by collaborators at the University of Tulsa
- Configuration
  - BeagleBone Black node controllers
  - Bendix Electronic Brake Controller (EBC)
  - Caterpillar 2000 engine control module (ECM)
    - Hosts an engine controller and a retarder controller application
- Features
  - ECM transmitting majority of the messages
  - ~ 30% high priority messages on each configuration
  - ~ 100% high priority traffic transmitted by ECM on each configuration

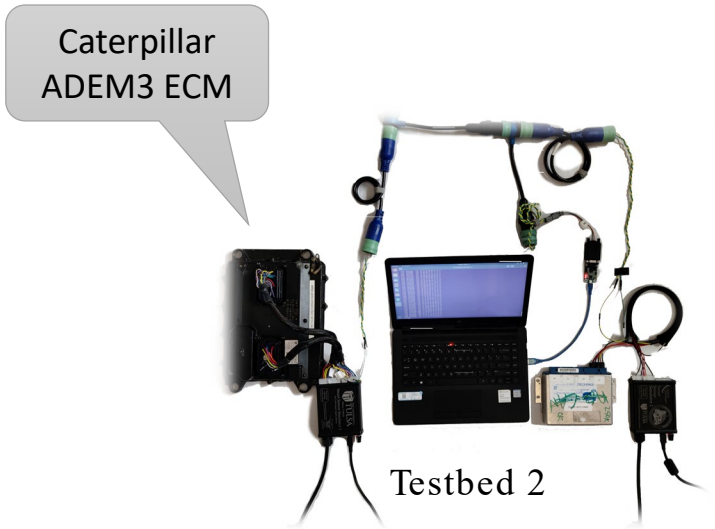


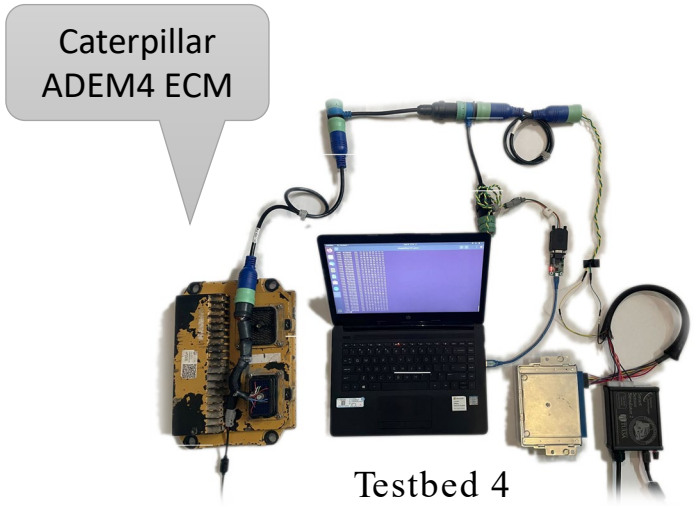
Testing Setup 2 (Local Testbench)

- Four variations
- Configuration
  - Bendix Electronic Brake Controller (EBC)
  - Four different engine control modules (ECM)
    - Each hosts an engine controller and a retarder controller application
- Features
  - ECM transmitting majority of the messages
  - ~ 50% high priority messages on each configuration
  - ~ 70% high priority traffic transmitted by ECM on each configuration









Organization

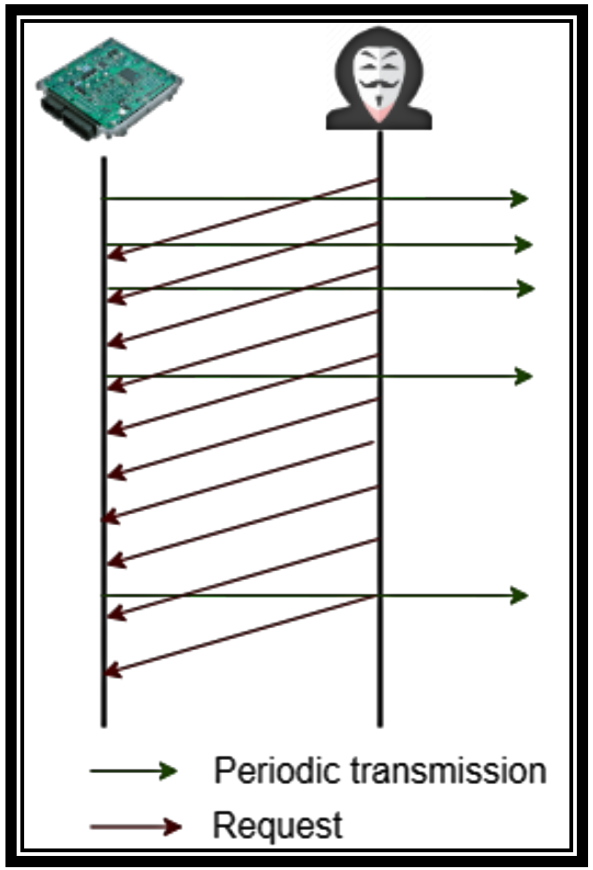
- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Three denial-of-service attacks on the data-link layer specifications

- The Testing Setup
- The Request Overload Attack
- The Connection Exhaustion Attack
- The False Request To Send (RTS) Attack
- Summary

Hypothesis

- Specification
  - All directed requests to an ECU must be processed.
- Attack
  - Send a high volume of SAE J1939 requests to the target ECU
- Expected result
  - In an attempt to serve the sent requests, the ECU fails to perform regular, more critical tasks like transmission of periodic messages

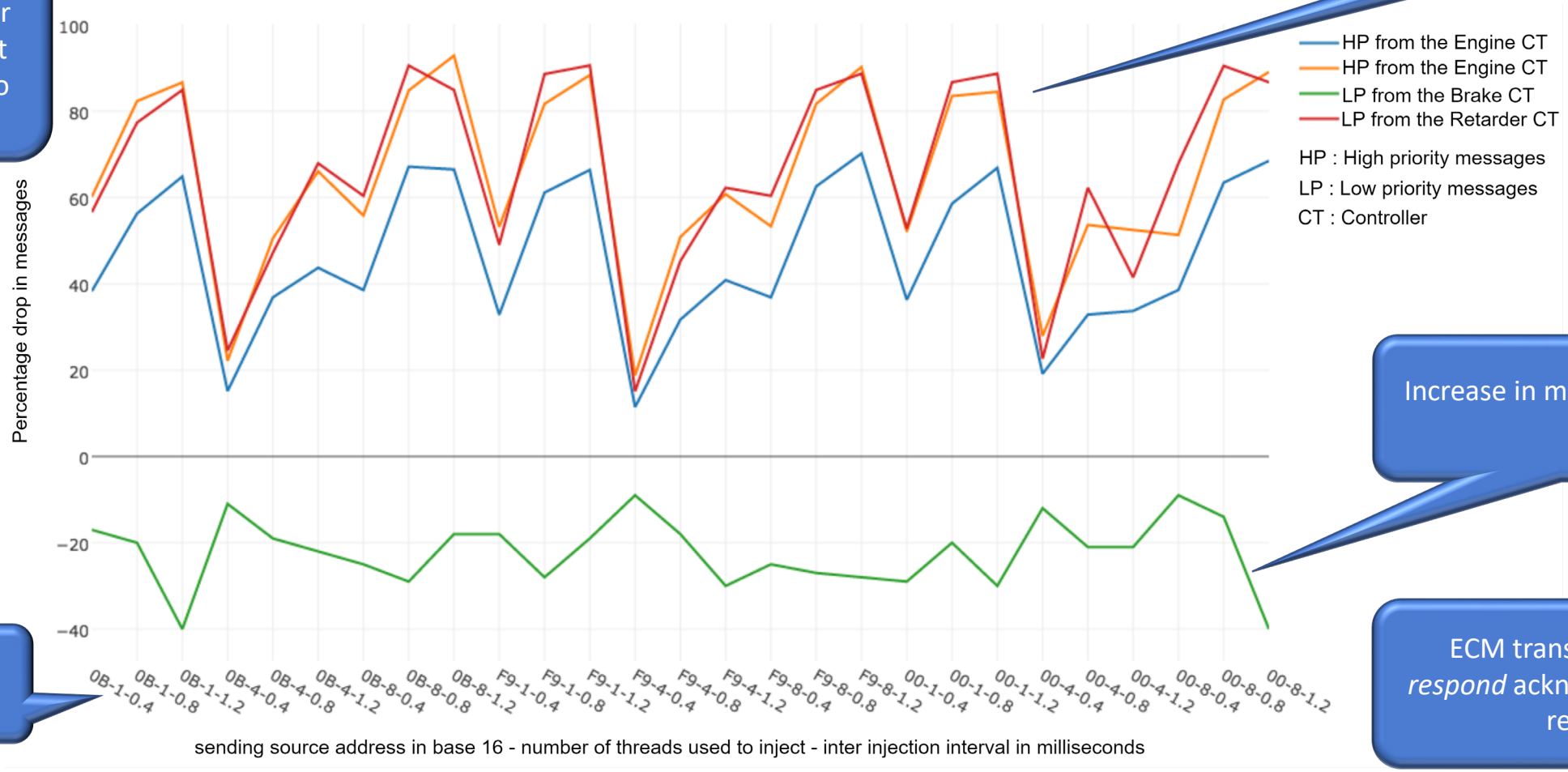


Testing on Remote Testbench

Observations

Rapidly sent requests for component identifier to ECM

Decrease in message count from ECM



Increase in message count from EBC

Experiment Settings

ECM transmitted *cannot respond* acknowledgements on requests

Testing on Local Testbench

Line color significance:

Red: On flooding with messages of ID 00000000_{16}

Blue: On overloading with valid request messages

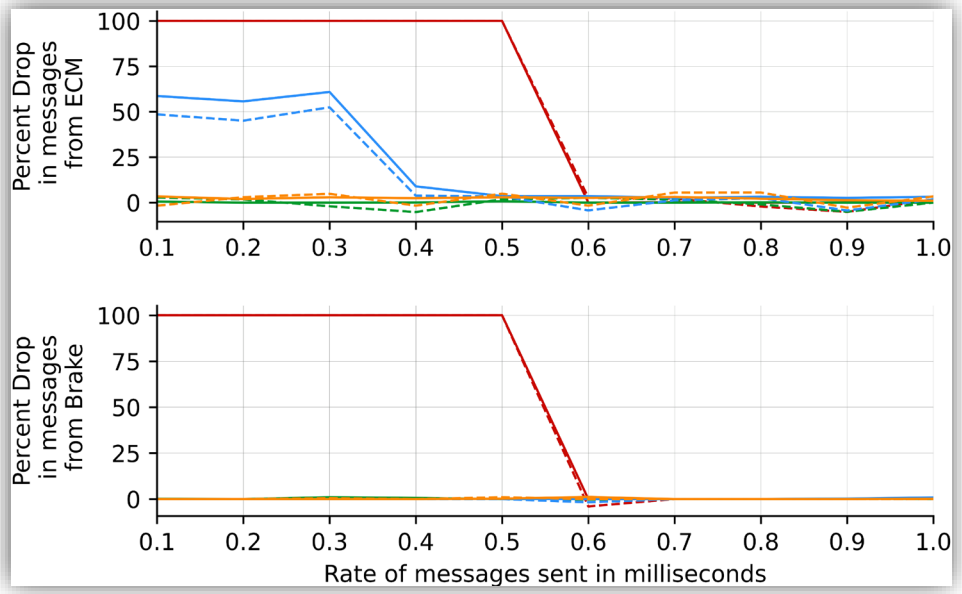
Orange: On overload with invalid request messages

Green: On flooding with messages of ID $1C000000_{16}$

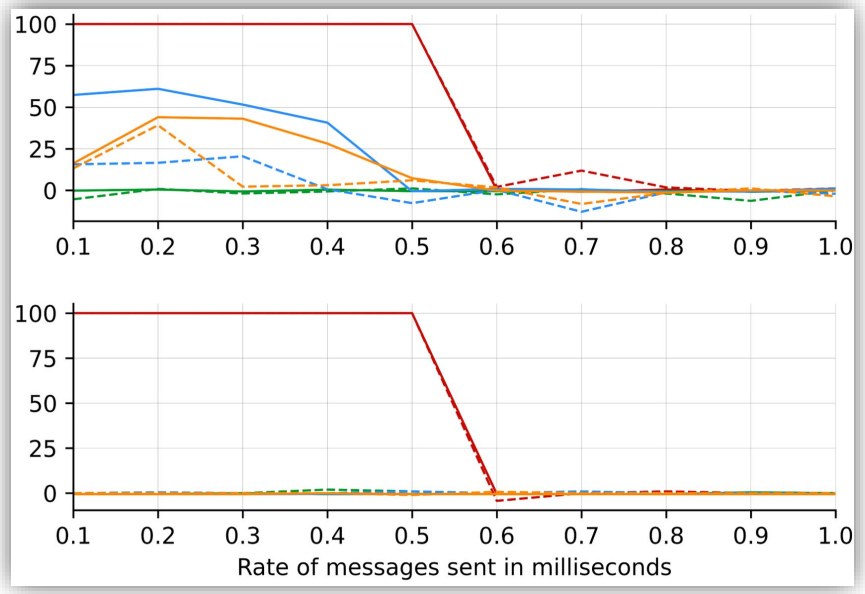
Line shape significance:

Solid: High priority ([0,3]) messages

Dashed: Low priority ([4,7]) messages

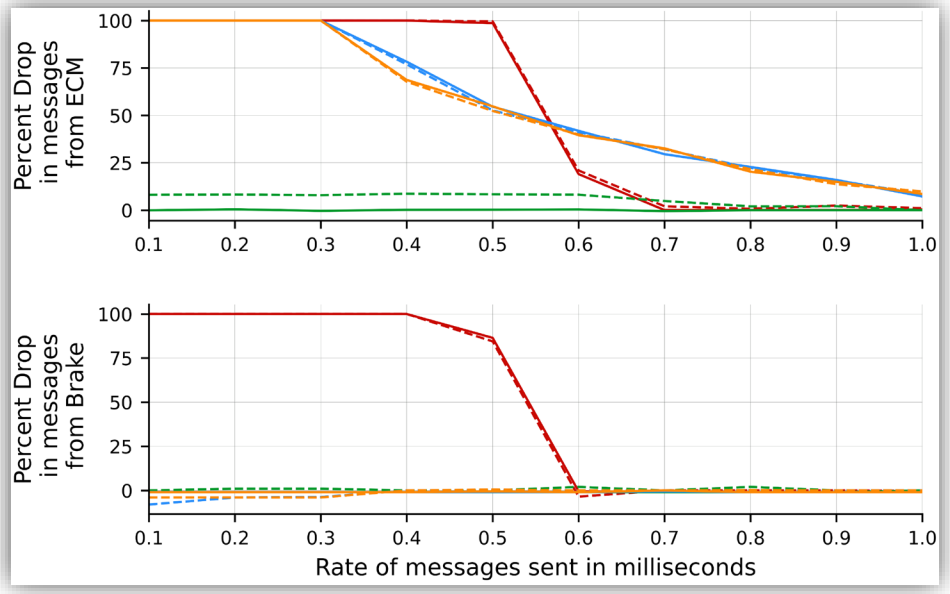


On Testbed 1

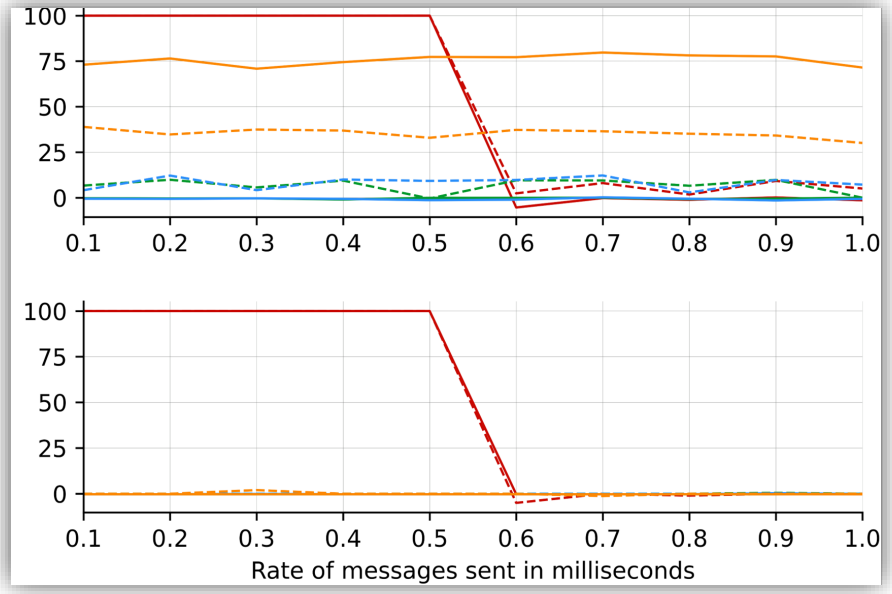


On Testbed 2

Drop in message count from ECM but not from the EBC on Request Overload



On Testbed 3



On Testbed 4

Discussion

- High-volume DoS attack
- Effect on the Kenworth T270 research truck
  - All transmission from the ECM stopped at 3 millisecond injection interval
  - Transmission did not shift gears
  - Engine speed remained high
- Possible defense
  - Not to process more than a certain number of requests in a millisecond on the host
  - Requires ECU firmware change

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Three denial-of-service attacks on the data-link layer specifications

The Testing Setup

The Request Overload Attack

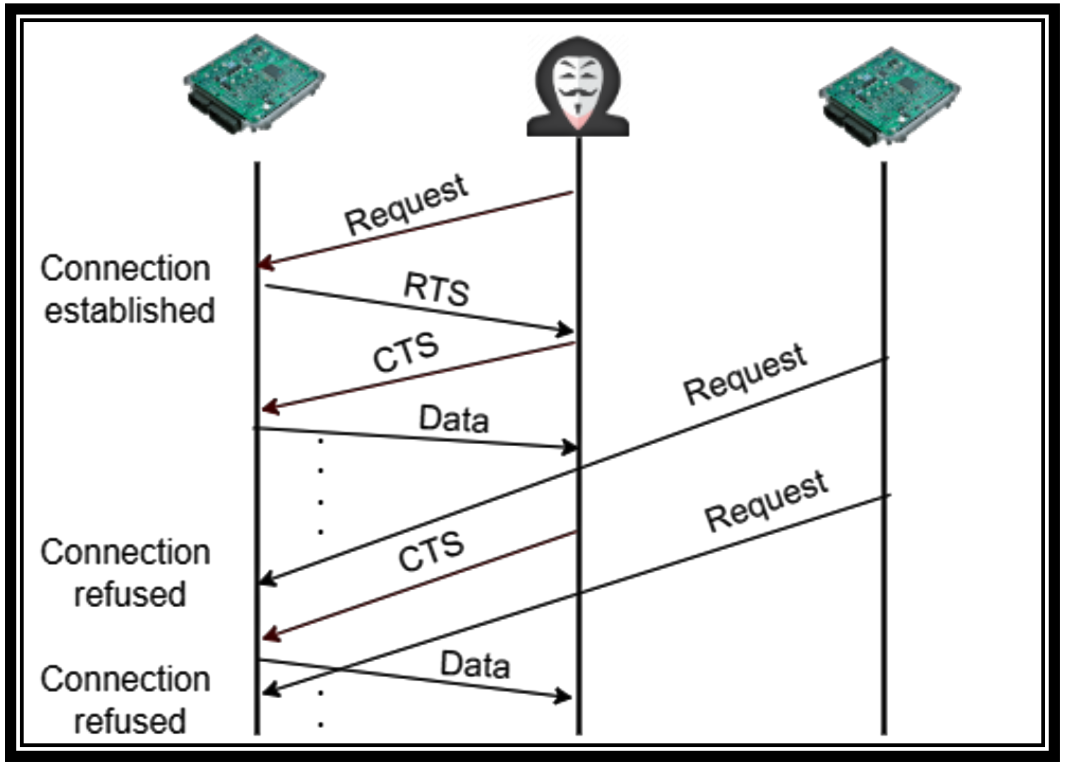
The Connection Exhaustion Attack

The False Request To Send (RTS) Attack

Summary

Hypothesis

- Specification
  - Exactly one established connection for unidirectional transfer
  - Connection can be kept open for 1250 milliseconds by not sending the end of message acknowledgment
  - CTS message can be sent to request message retransmission
- Attack
  - Create multiple spoofed connections
  - Keep connections open by
    - Sending CTS at intervals less than 1250 ms
    - Not sending of end of message acknowledgement
- Expected result
  - Denial of legitimate connection attempts to the target



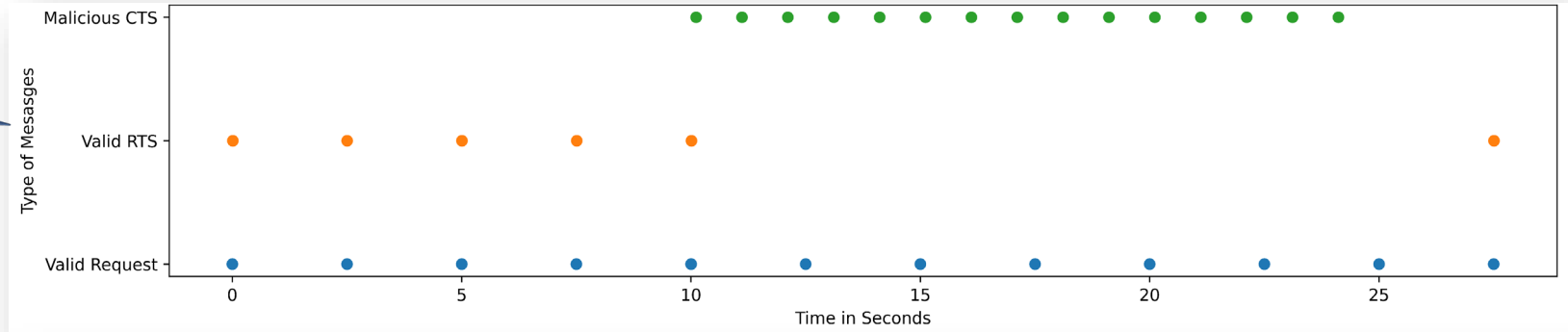
Testing on Remote Testbench

- Malicious node (BeagleBone Black) BB1
- Spoofed controller source addresses 11_{16} and $0B_{16}$
  - Simulated on (BeagleBone Black) BB2

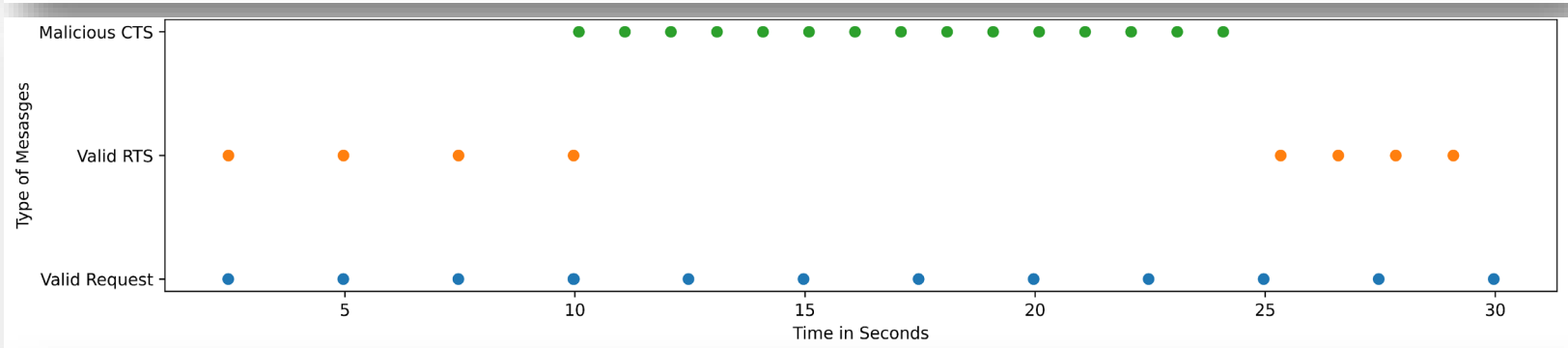
Illegitimate connection initiation	BB1->Engine-\#1	request	00EA0011	EB FE 00 00 00 00 00 00
	Engine-\#1->BB1	RTS	18EC1100	10 2C 00 07 FF EB FE 00
	BB1->Engine-\#1	CTS	00EC0011	11 07 01 FF FF EB FE 00
	BB1->Engine-\#1	request	00EA000B	EB FE 00 00 00 00 00 00
	Engine-\#1->BB1	RTS	18EC0B00	10 2C 00 07 FF EB FE 00
	BB1->Engine-\#1	CTS	00EC000B	11 07 01 FF FF EB FE 00
Illegitimate connection data transfer	Engine-\#1->BB1	Data Transfer	18EB1100	01 43 4D 4D 4E 53 2A 36
	Engine-\#1->BB1	Data Transfer	18EB1100	02 43 20 75 30 37 44 30
	Engine-\#1->BB1	Data Transfer	18EB1100	03 38 33 30 30 30 30 30
	Engine-\#1->BB1	Data Transfer	18EB1100	04 30 30 2A 30 30 30 30
	Engine-\#1->BB1	Data Transfer	18EB1100	05 30 30 30 30 2A 78 30
	Engine-\#1->BB1	Data Transfer	18EB1100	06 36 42 42 42 42 42 42
	Engine-\#1->BB1	Data Transfer	18EB1100	07 42 2A FF FF FF FF FF
	Engine-\#1->BB1	Data Transfer	18EB0B00	01 43 4D 4D 4E 53 2A 36
	Engine-\#1->BB1	Data Transfer	18EB0B00	02 43 20 75 30 37 44 30
	Engine-\#1->BB1	Data Transfer	18EB0B00	03 38 33 30 30 30 30 30
	Engine-\#1->BB1	Data Transfer	18EB0B00	04 30 30 2A 30 30 30 30
	Engine-\#1->BB1	Data Transfer	18EB0B00	05 30 30 30 30 2A 78 30
	Engine-\#1->BB1	Data Transfer	18EB0B00	06 36 42 42 42 42 42 42
	Engine-\#1->BB1	Data Transfer	18EB0B00	07 42 2A FF FF FF FF FF
legitimate connection refusal	BB2->Engine-\#1	request	00EA0011	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA000B	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA0011	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA000B	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA0011	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA000B	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA0011	EC FE 00 00 00 00 00 00
	BB2->Engine-\#1	request	00EA000B	EC FE 00 00 00 00 00 00
Illegitimate connection keep alive	BB1->Engine-\#1	CTS	00EC0011	11 07 01 FF FF EB FE 00
	BB1->Engine-\#1	CTS	00EC000B	11 07 01 FF FF EB FE 00

Testing on Local Testbench

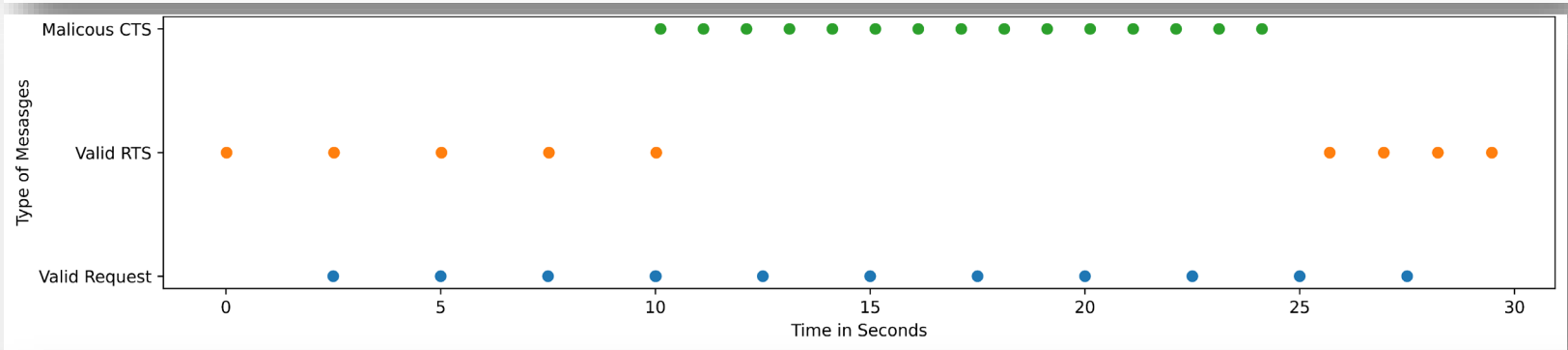
Attack successful on Testbed 1



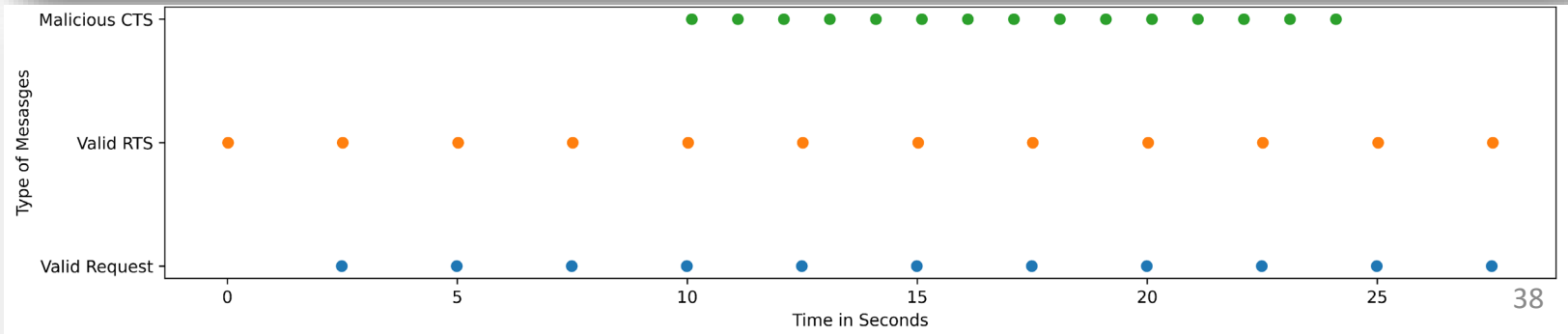
Attack successful on Testbed 2



Attack successful on Testbed 3

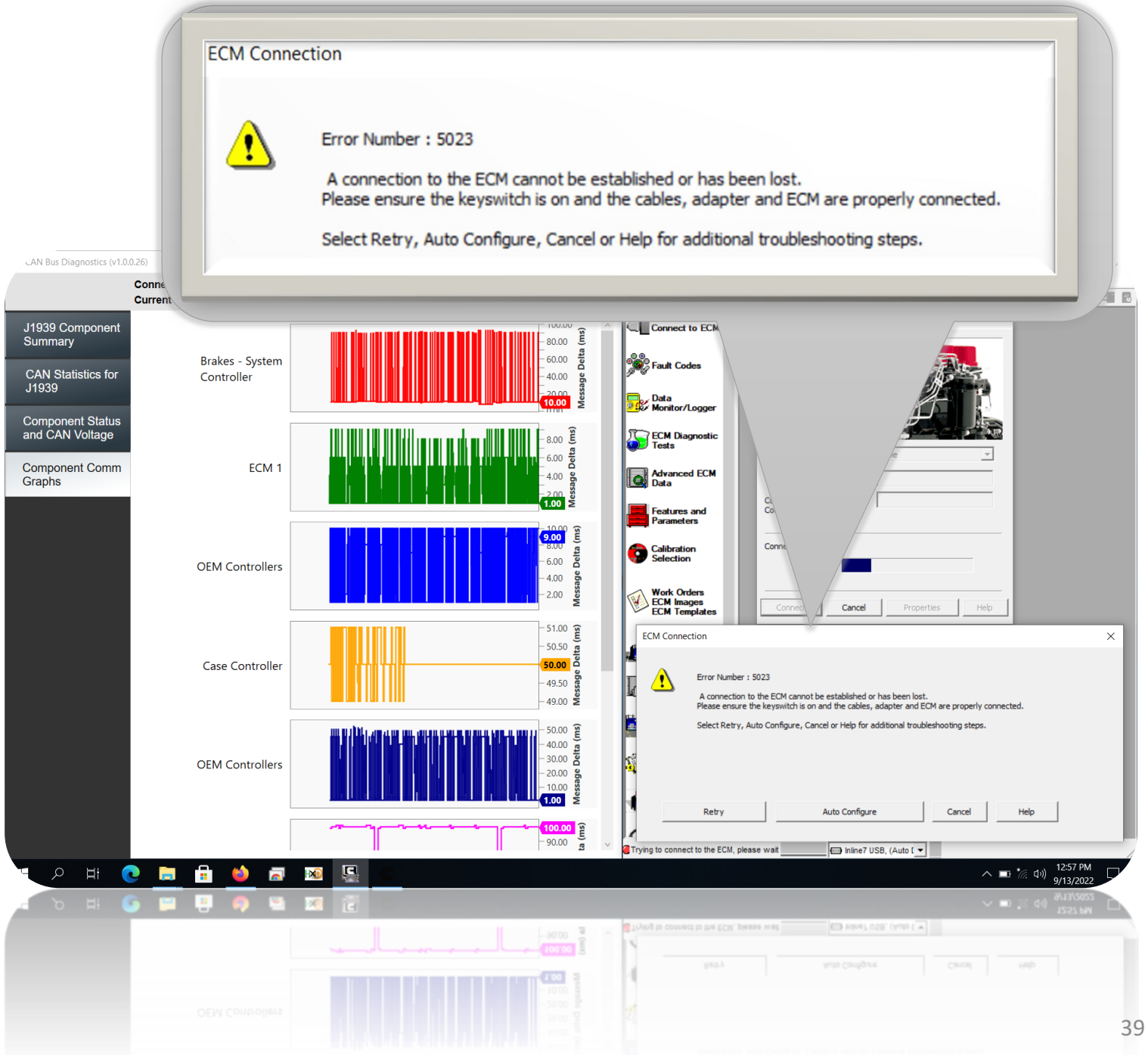


Attack **un**successful on Testbed 4



Discussion

- Low-volume DoS attack
- Effect on the Kenworth T270 research truck
  - No physical impact
  - Proprietary communication can be hampered
  - Diagnostic sessions can be hampered
- Possible defense
  - Not to respond to more than a certain number of CTS retransmit requests
  - Requires ECU firmware change



Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Three denial-of-service attacks on the data-link layer specifications

The Testing Setup

The Request Overload Attack

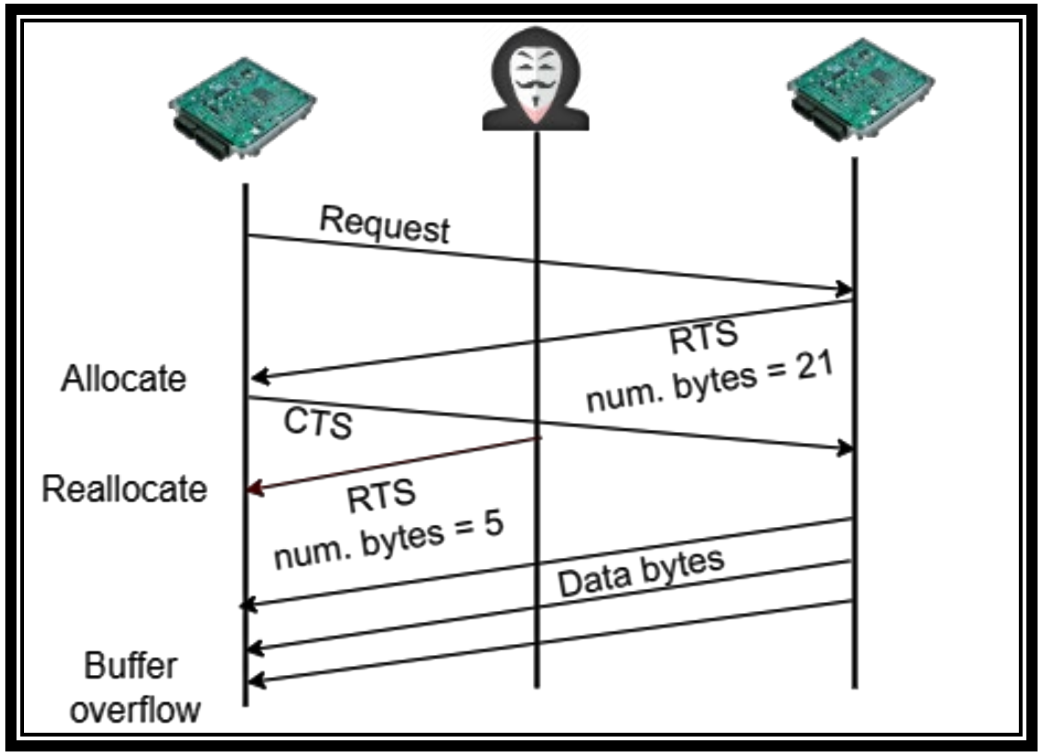
The Connection Exhaustion Attack

The False Request To Send (RTS) Attack

Summary

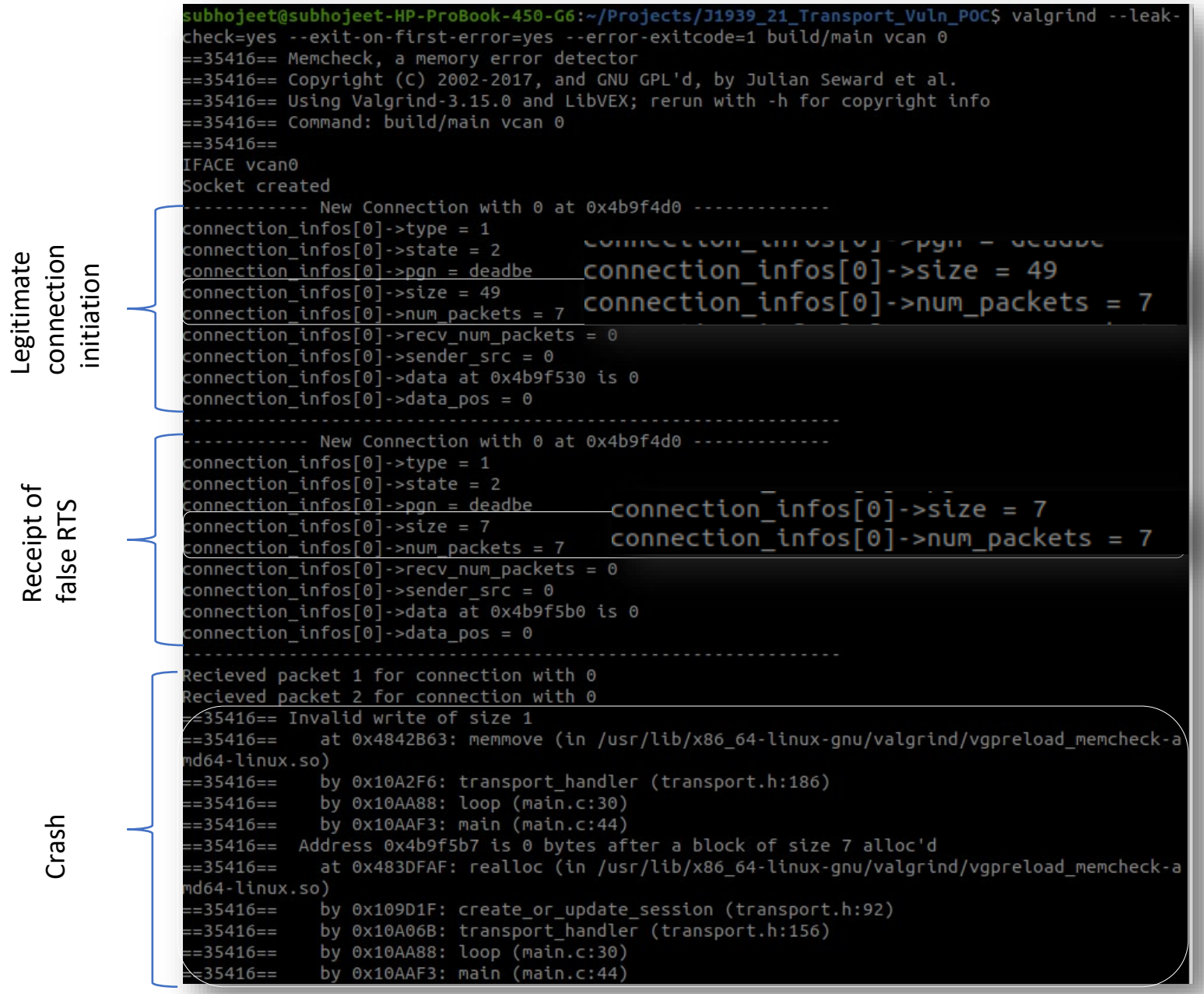
Hypothesis

- Specification
  - If multiple RTS messages are received from the same source address, the most recently received shall be considered without notifying the sender of the first RTS
- Attack
  - Spoof connection creator
  - Send second RTS with smaller data size
- Expected result
  - Data buffer reallocation and buffer overflow



Testing

- ECUs in either testbed did not accept connection requests
  - Possibly an authentication issue
- Vulnerable simulation created
  - Execution with Valgrind memory profiler shows heap overflow



Discussion

- Effect on the Kenworth T270 research truck
  - No evidence of success
    - ECUs did not accept connection attempts
- Basis for test software generation in ECU micro patching
  - Challenge problems for multiple ECU micro patching hackathons
- Possible defense
  - Mismatching number of bytes and packets in the RTS must be checked before reallocation
  - Second (falsifiable) RTS can be ignored

Organization

- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Three denial-of-service attacks on the data-link layer specifications

- The Testing Setup
- The Request Overload Attack
- The Connection Exhaustion Attack
- The False Request To Send (RTS) Attack
- Summary

Summary

Research Question

Can weaknesses in the data-link layer specifications of SAE J1939 be exploited to attack in-MHD vehicle ECUs?

Contribution

Three denial-of-service attacks that utilize protocol specifications made in the SAE J1939 standards

- Noticeable impacts on network communication from target ECUs
- Noticeable impact on a research truck for first two attacks

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Online anomaly-based intrusion detection system

Behavioral Feature Engineering

Network Behavior Modelling Through Feature Value Time Series

Detecting Anomalous Behavior at Runtime

Performance Analysis

Summary

Organization

- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Online anomaly-based intrusion detection system

- Behavioral Feature Engineering
- Network Behavior Modelling Through Feature Value Time Series
- Detecting Anomalous Behavior at Runtime
- Performance Analysis
- Summary

Report Precedence Graph

Message m_t

PGN	DA	SA	Parameter Values
pgn_{m_t}		sa_{m_t}	$\{(spn,\ v_{spn})\}$

Report

$\{(spn, d(v_{spn}), pgn_{m_t}, sa_{m_t})\}$

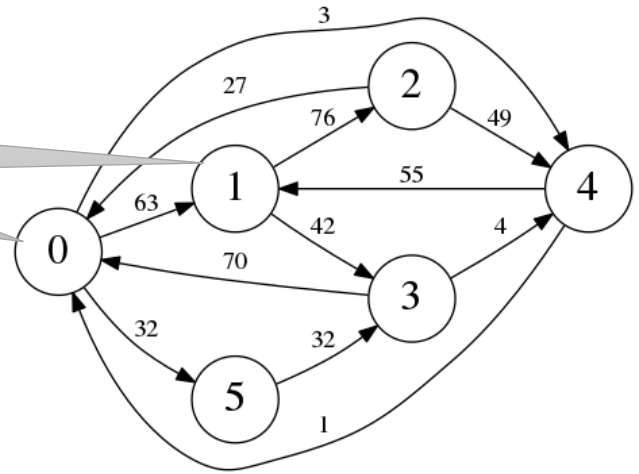
Discretization of v_{spn}

Discrete parameter

v_{spn}

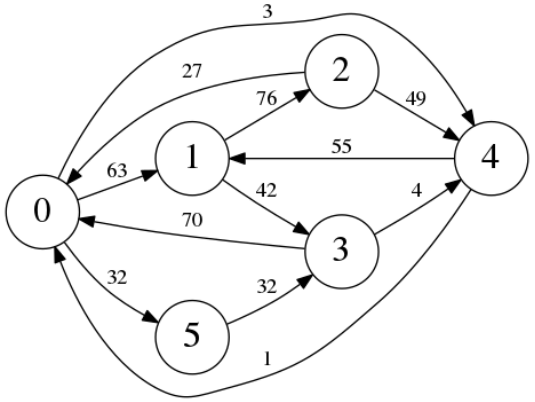
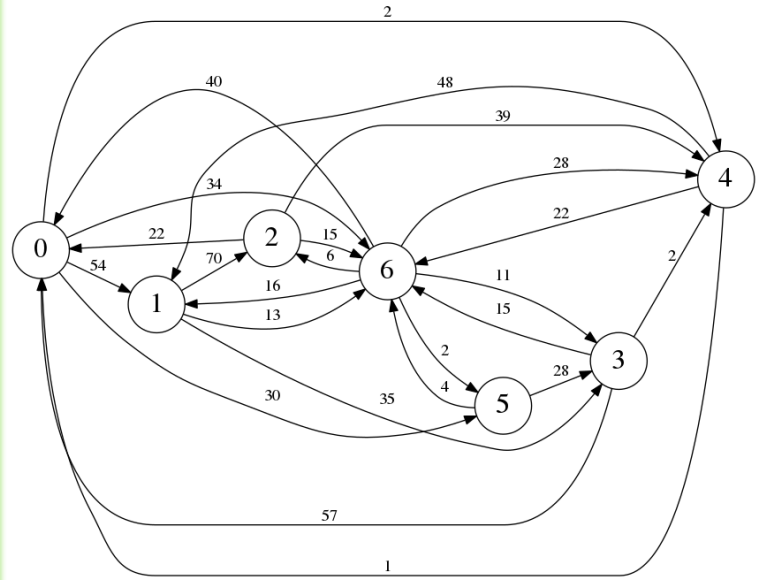
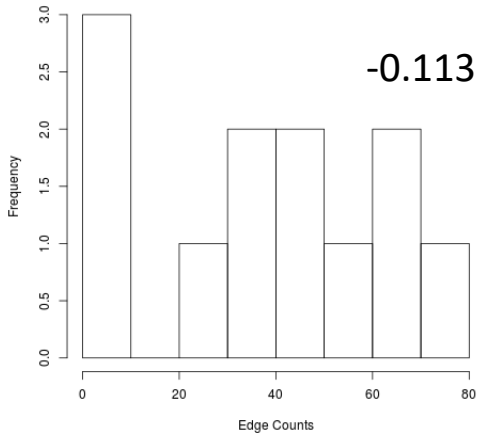
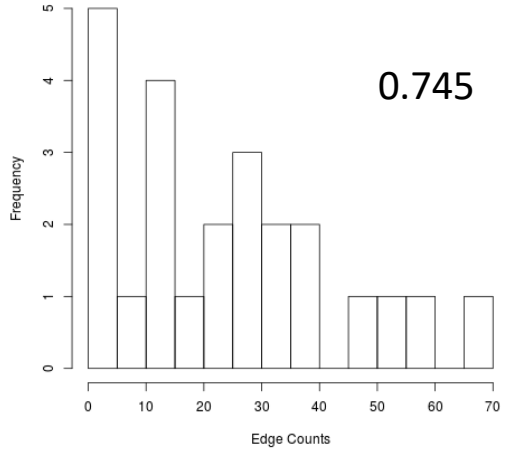
Continuous parameter

$$\left\lfloor length_{spn} * \frac{(v_{spn} - offset_{spn})}{(resolution_{spn} * (2^{length_{spn}} - 1))} \right\rceil$$



Definition 1 (Report Precedence Graph) *The Report Precedence Graph (RPG) is a labeled directed graph* $G = (R, T, L)$ *where each node* $r \in R$ *is a report and each edge* $\langle r_i, r_j \rangle \in T$ *denotes report* r_j *is transmitted after report* r_i ($r_i \prec r_j$ *with no other* r_k *such that* $r_i \prec r_k$ *and* $r_k \prec r_j$), *a total of* $l_{i,j} \in L$ *number of times.*

Behavioral Features

	Normal RPG	RPG Under Attack
Report Precedence Graph (RPG) = *(R, T, L)*		
Flux Capacity of a node r_i $= \lvert(r_j, r_i)\rvert * \lvert(r_i, r_k)\rvert \ \forall(r_j, r_i), (r_i, r_k) \in T$ $= d^{in}(r_i) * d^{out}(r_i)$		
Normalized Graph Flux Capacity (NGFC) of the RPG $= \frac{1}{\lvert R\rvert^3} \sum_{i=1}^{\lvert R\rvert} fc_{r_i}$	0.12	0.37
Edge Weight Skewness (EWS) of the RPG	 -0.113	 0.745

Organization

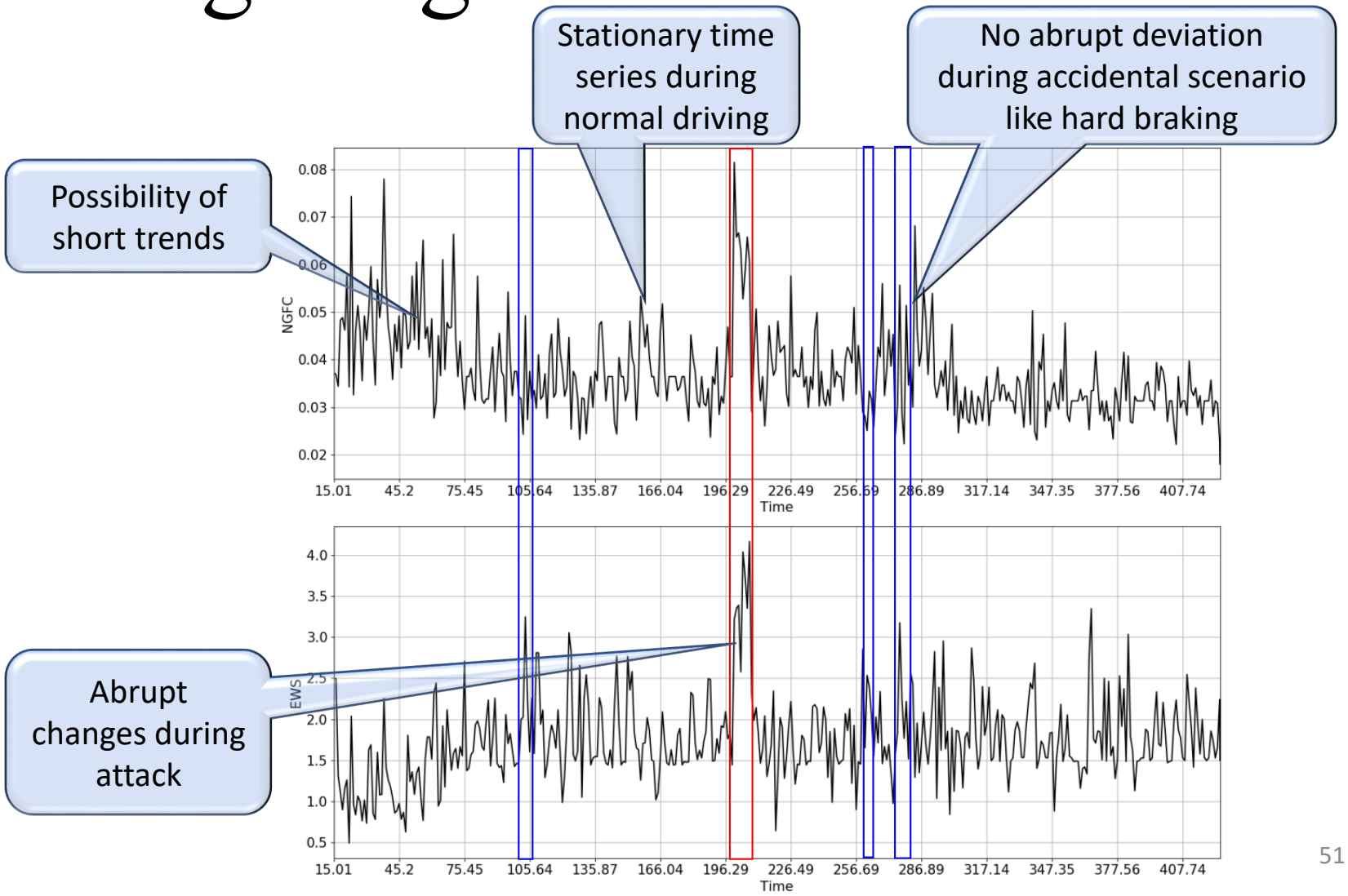
- Introduction
- Background
- Contribution 1
- **Contribution 2**
- Contribution 3
- Conclusion and Future Work

Online anomaly-based intrusion detection system

- Behavioral Feature Engineering
- **Network Behavior Modelling Through Feature Value Time Series**
- Detecting Anomalous Behavior at Runtime
- Performance Analysis
- Summary

Network Behavior Modelling Through Feature Value Time-Series

- Attack detection hypothesis
  - Under normal driving conditions NGFC and EWS time-series are usually stationary with the possibility of short trends
  - But upon malicious message injections, they exhibit significant abrupt changes that can be detected



Time-series of Feature Values calculated on 1 sec (Sampling Window) RPGs

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Online anomaly-based intrusion detection system

Behavioral Feature Engineering

Network Behavior Modelling Through Feature Value Time Series

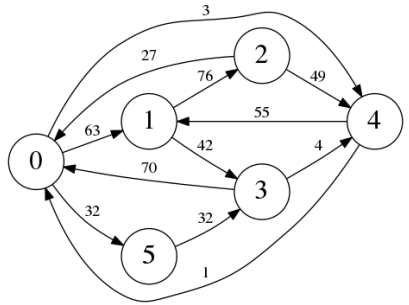
Detecting Anomalous Behavior at Runtime

Performance Analysis

Summary

time ≥ startTime + samplingWindow

Sampling window (1 sec) expired ?



Build RPG

Create and add report to RPG

Add edge, update edge count

```
1 r_i ← getReport(J1939 Message)
2 R ← R ∪ r_i
3 if r_i ≠ r_current ∧ r_current ≠ null
    then
4 |   T ← T ∪ ⟨ r_current, r_i ⟩
5 |   Update L, set l_{i,j} ← l_{i,j} + 1
6 end
7 r_current ← r_i
```

Calculate feature values

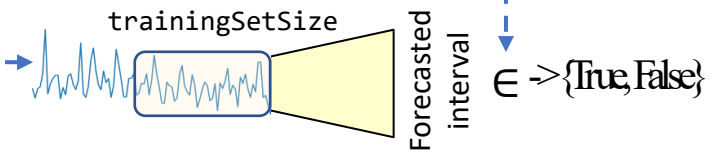
If feature value queues are full

Check for attack

Accumulate feature values in circular queue if not attack is detected

Reinitialize RPG

```
1  NGFC ← NFGC(R,T)
2  EWS ← EWS(L)
3  is_attack ← False
4  if number of items queued =
     trainingSetSize then
5  |   is_attack ← sig-inc(NGQ,
   |     NGFC) ∧ sig-inc(EWQ,
   |     EWS)
6  |   if is_attack then
7  |   |   RAISE ALARM
8  |   end
9  end
10 if is_attack = False then
11 |   NGQ.push(NFGC),
   |     EWQ.push(EWS)
12 end
13 startTime ← time
14 R,T,L ← ∅
15 r_current ← null
```



Check for trend in time-series

Check if input feature value in predicted range [Brut2000]

```
1 Procedure sig-inc(buf,
    current)
2 |   has_trend ← Aug-
  |     mented_Dickey–Fuller_test(buf).p-
  |     value ≤ (100 -
  |     ADFConfidence)/100
3 |   forecast_interval ← HoltWin-
  |     ters(buf,HoltsConfidence,has_trend)
4 |   return value ∉
  |     forecast_interval
```

Organization

- Introduction
- Background
- Contribution 1
- **Contribution 2**
- Contribution 3
- Conclusion and Future Work

Online anomaly-based intrusion detection system

- Behavioral Feature Engineering
- Network Behavior Modelling Through Feature Value Time Series
- Detecting Anomalous Behavior at Runtime
- **Performance Analysis**
- Summary

Experiment Method (Attack Detection)

Attacks
Address claim
Network Overload
Fuzzing attack without data fuzzing
Fuzzing attack with data fuzzing
Request Overload
Engine Control

Execute attacks on the research truck

Kenworth T270 driven on an airstrip

$$\text{Precision} = 100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ flagged\ windows}$$

50 iterations for different experiment configurations

Format
Timestamp,
CAN ID,
CAN DATA.
Attack ongoing?

30 second Network logs

2 logs per attack

Operator

PGN, SA, {(SPN, V_{SPN})}

NGFC, EWS, Attack?

Attack Detection Procedure

Result logs

Format
trainingSetSize
HoltsConfidence,
ADFConfidence,
Precision

Feature logs

Format
Timestamp
NGFC,
EWS

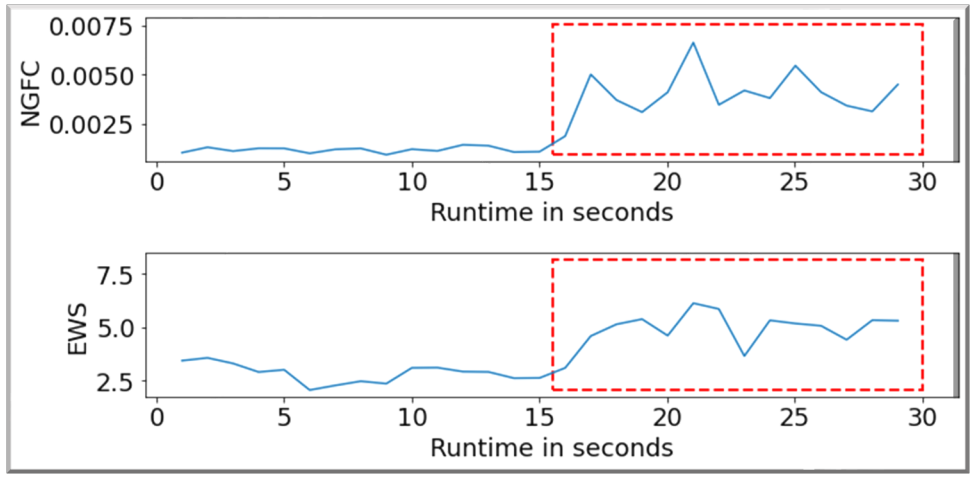
Experiment Factors
trainingSetSize: 5, 10
HoltsConfidence: 10, 30, 50, 70, 90
ADFConfidence: 10, 30, 50, 70, 90

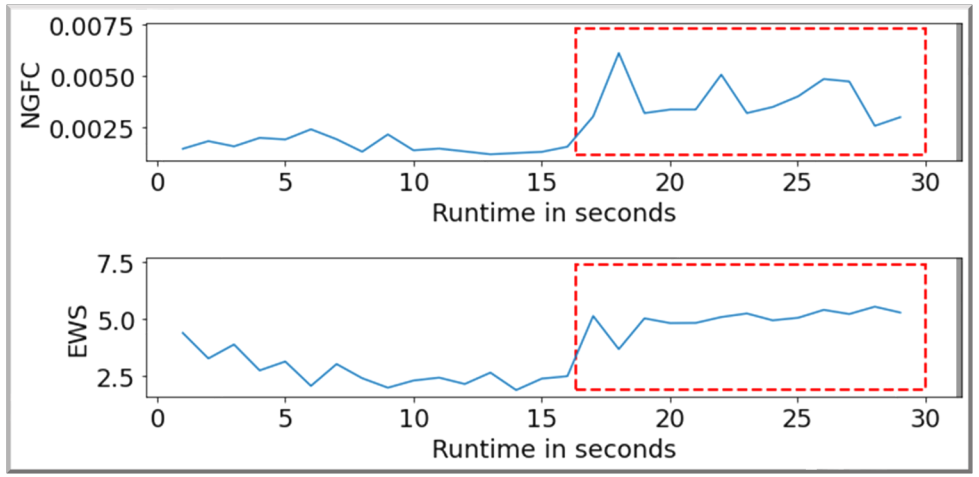
Attack Detection Results For Attack Claim Attack on Kenworth T270

First Run

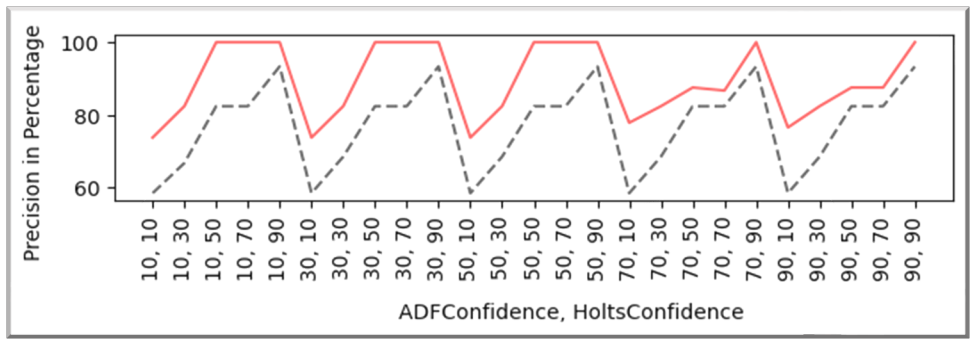
Second Run

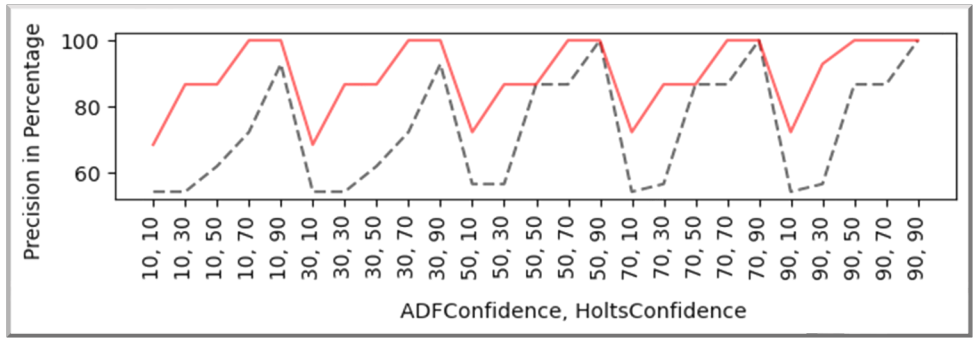
Abnormal feature value deviations during attack





Highest precision at 90% HoltsConfidence and 10 TrainingSetSize





Precision = $100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

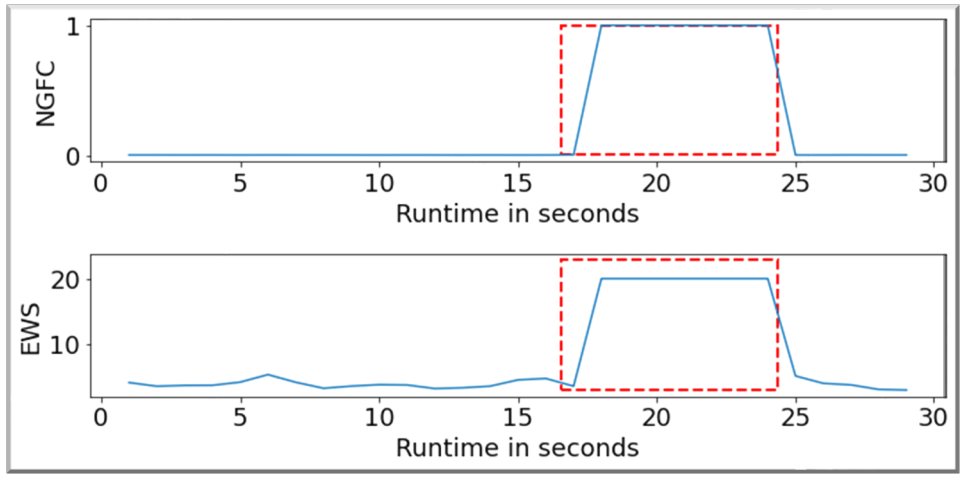
trainingSetSize = 10

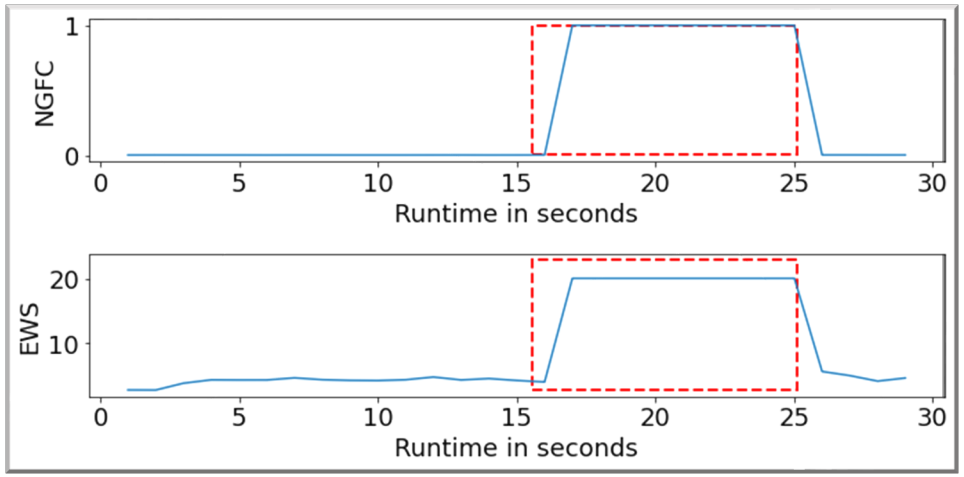
Attack Detection Results For Network Overload Attack on Kenworth T270

First Run

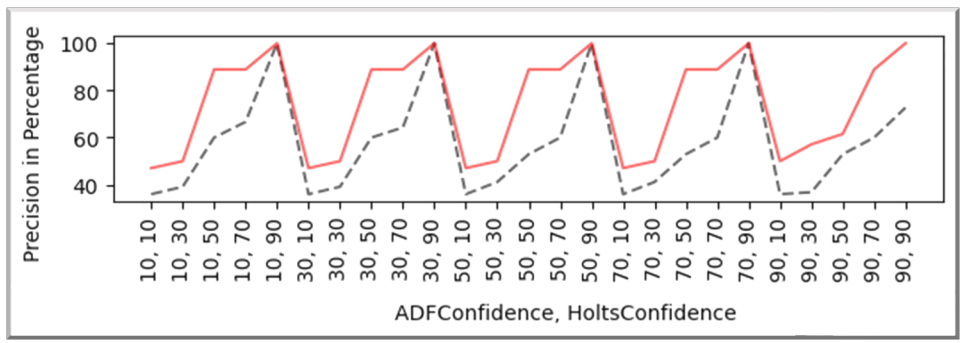
Second Run

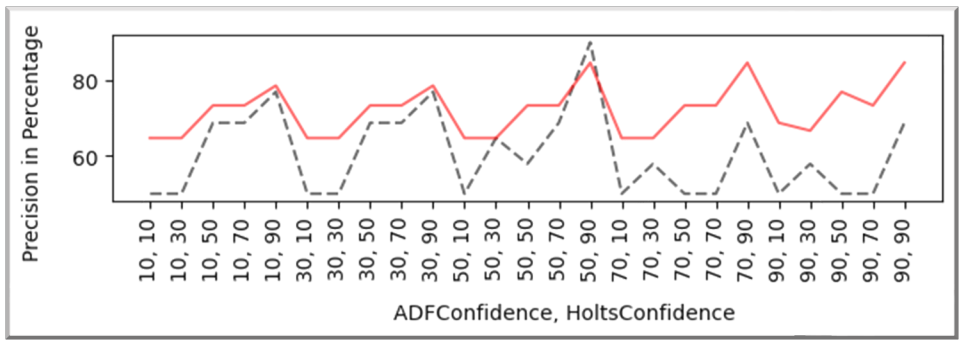
Abnormal feature value deviations during attack





Highest precision at 90% HoltsConfidence and 10 TrainingSetSize





Precision = $100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

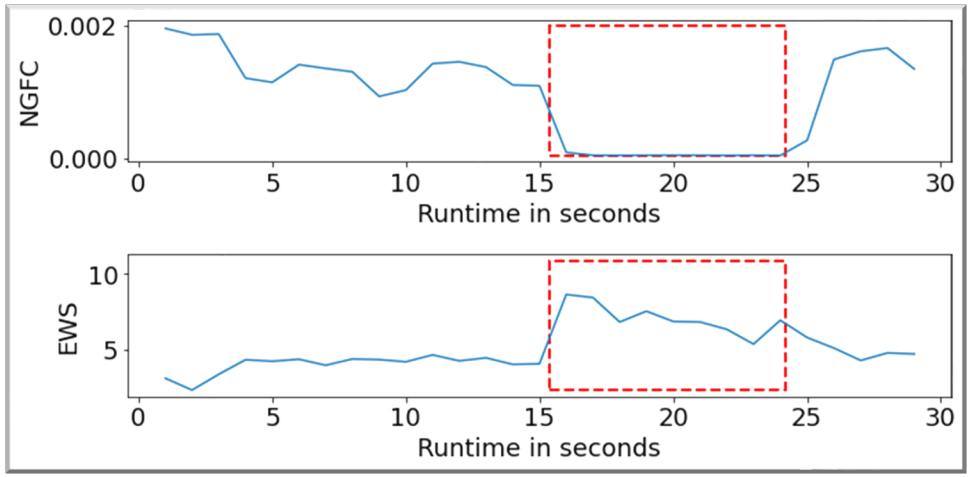
trainingSetSize = 10

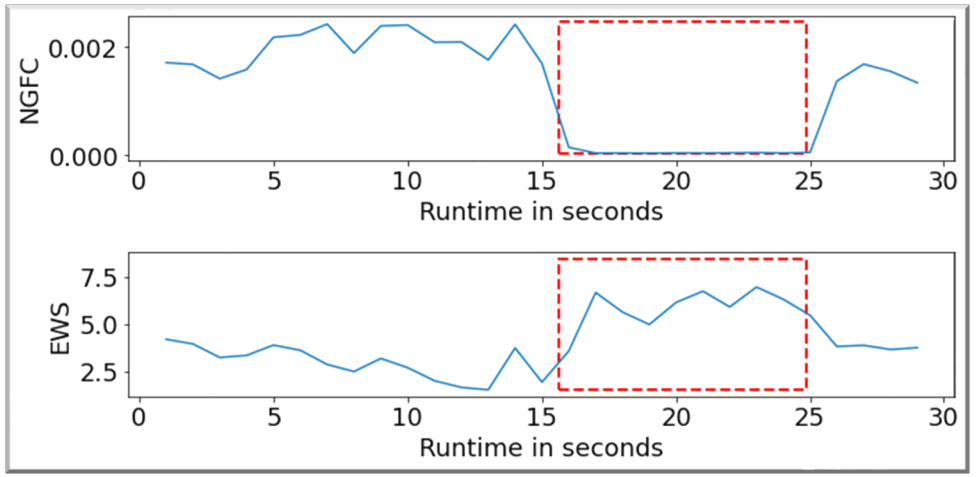
Attack Detection Results For Fuzzing Attack (Without Data Fuzzing) on Kenworth T270

First Run

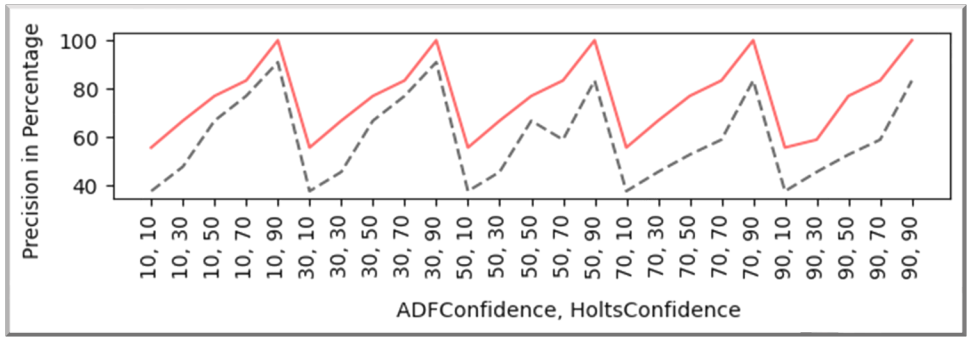
Second Run

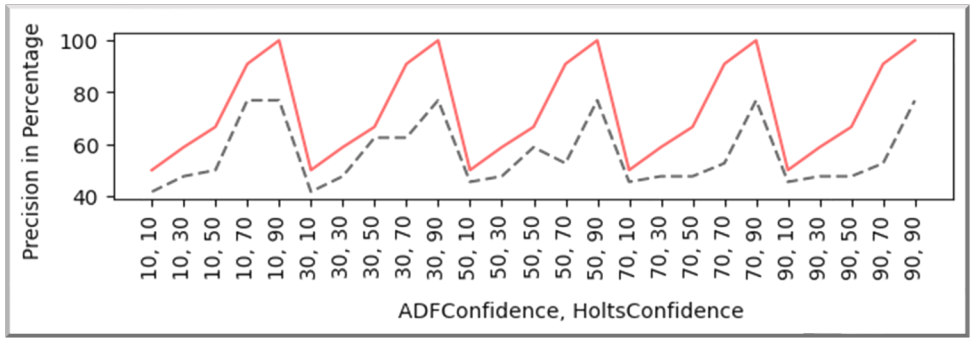
Abnormal feature value deviations during attack





Highest precision at 90% HoltsConfidence and 10 TrainingSetSize





Precision = $100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

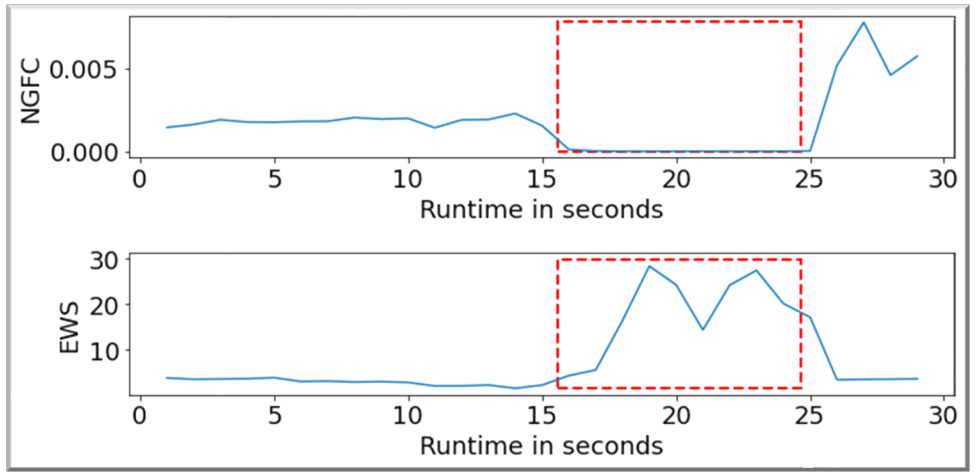
trainingSetSize = 10

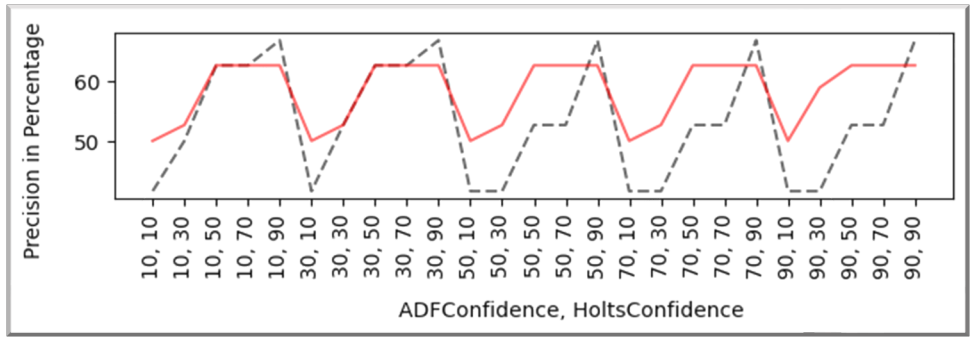
Attack Detection Results For Fuzzing Attack (With Data Fuzzing) on Kenworth T270

Abnormal feature value deviations during attack

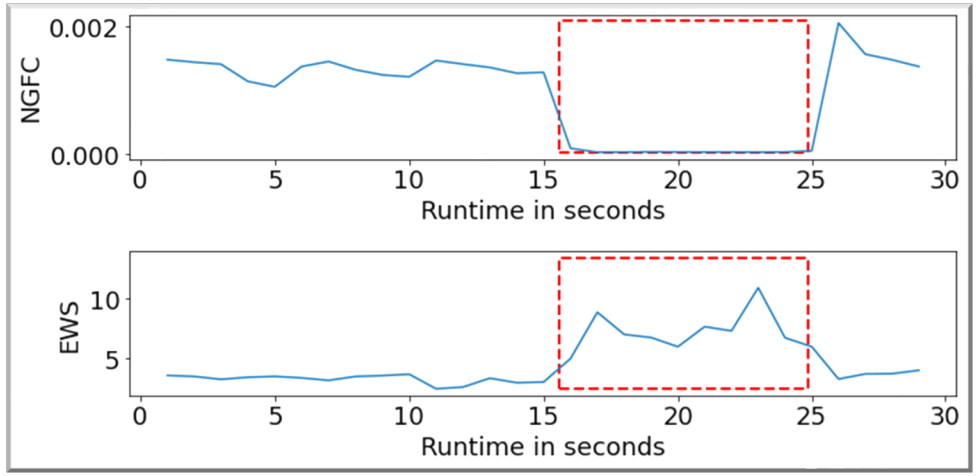
Highest precision at 90% HoltsConfidence and 10 TrainingSetSize

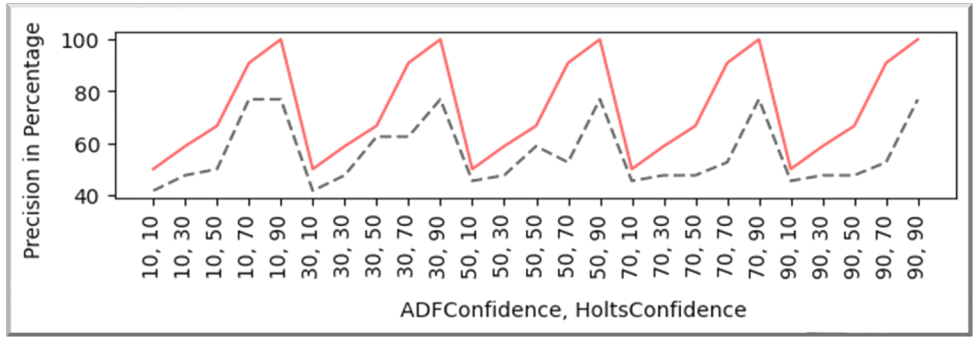
First Run





Second Run





Precision = $100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

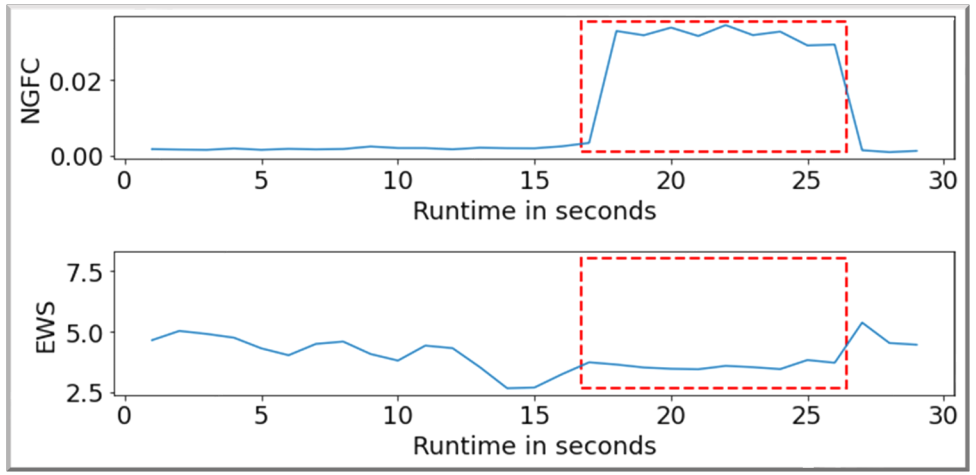
trainingSetSize = 10

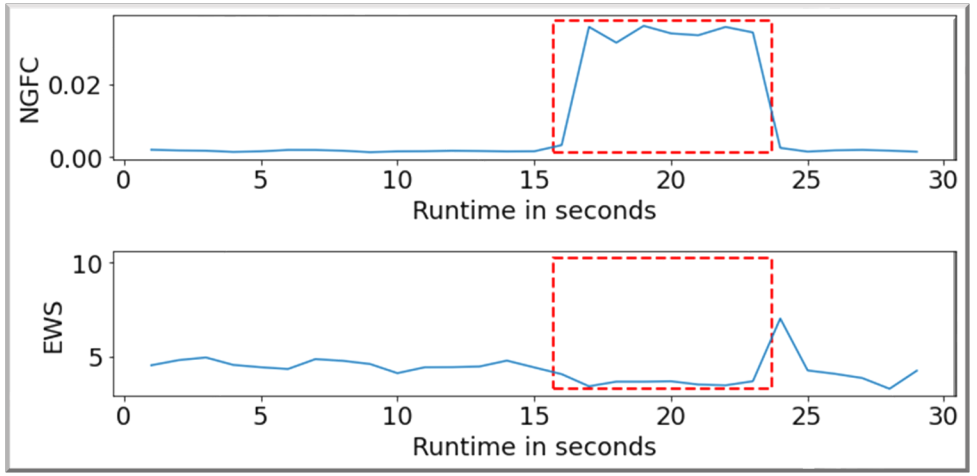
Attack Detection Results For Request Overload Attack on Kenworth T270

First Run

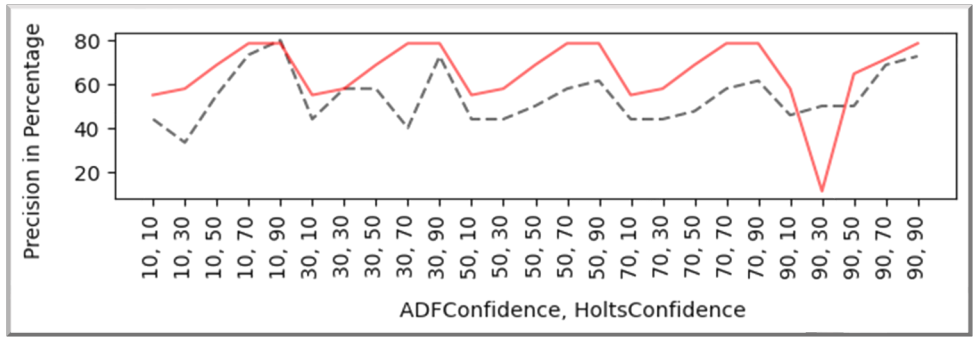
Second Run

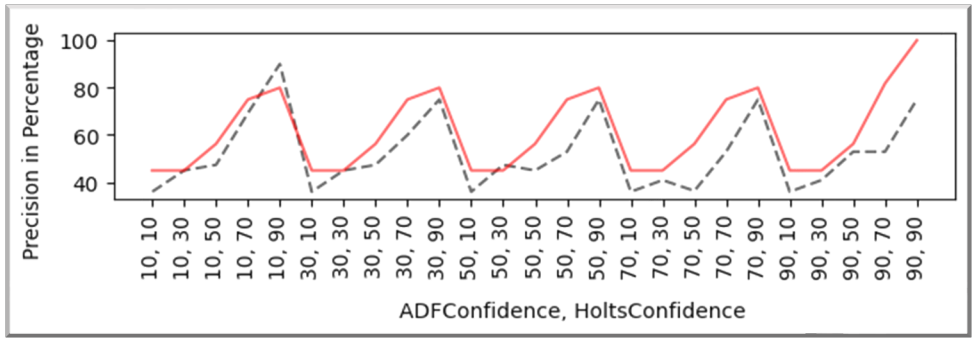
Abnormal NGFC deviations during attack, not EWS





Highest precision at 90% HoltsConfidence and 10 TrainingSetSize





Precision = $100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

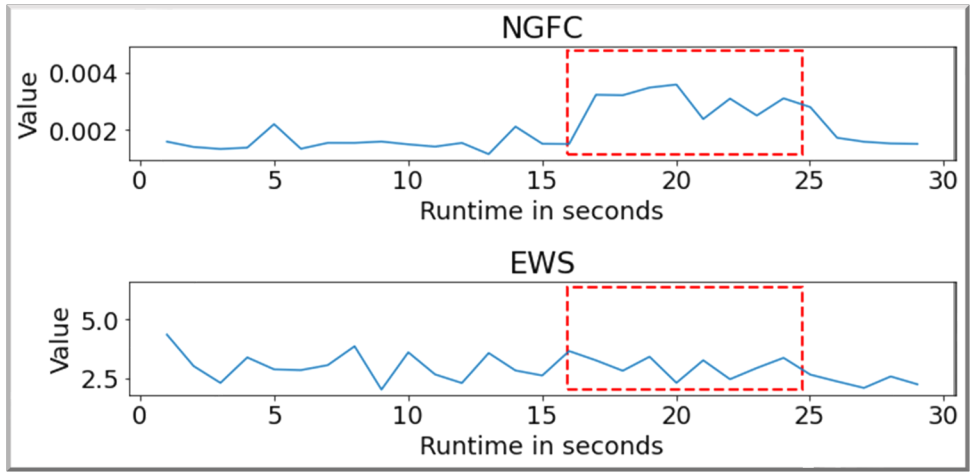
trainingSetSize = 10

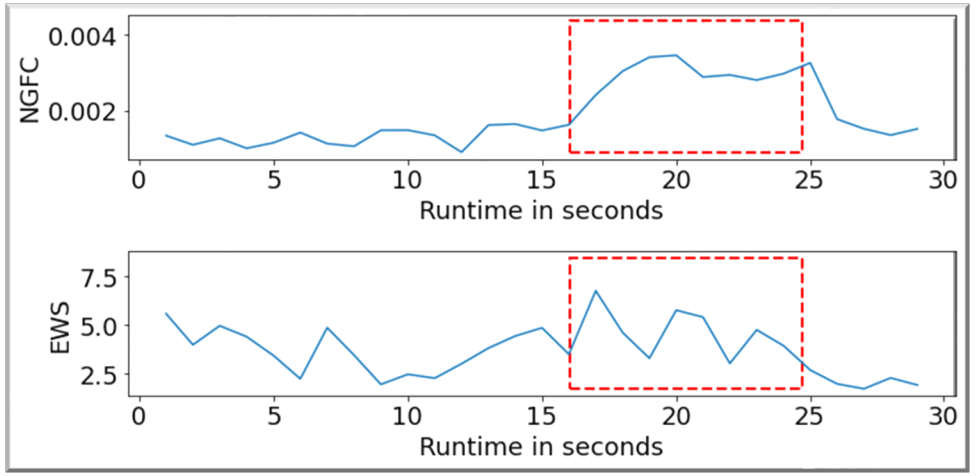
Attack Detection Results For Engine Control Attack on Kenworth T270

First Run

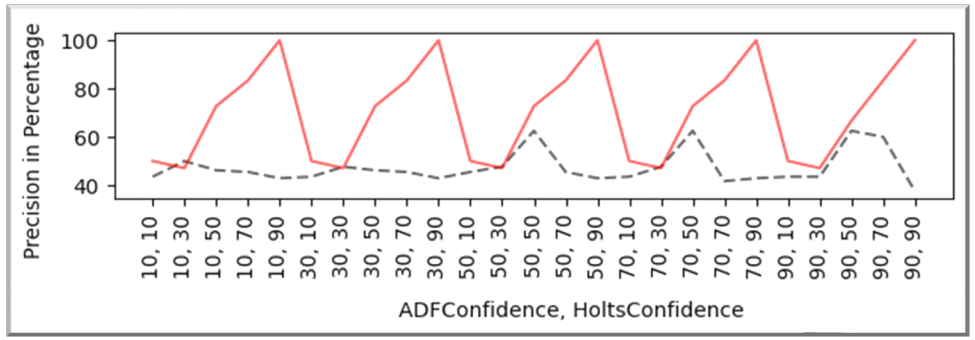
Second Run

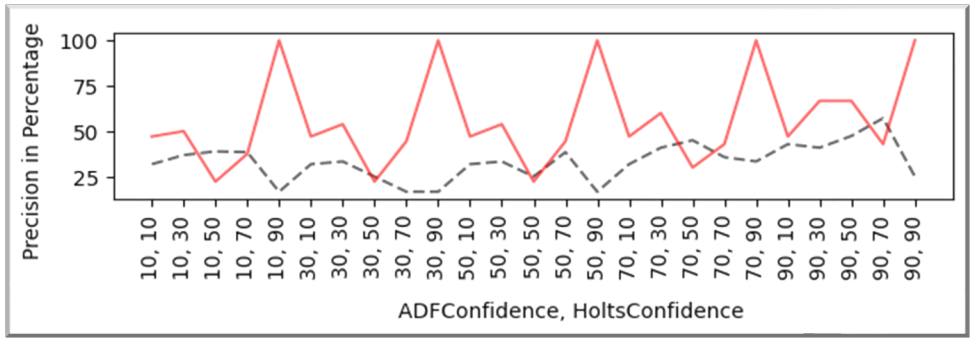
Abnormal NGFC deviations during attack, not EWS





Highest precision at 90% HoltsConfidence and 10 TrainingSetSize





Precision = $100 * \frac{No.of\ flagged\ attack\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

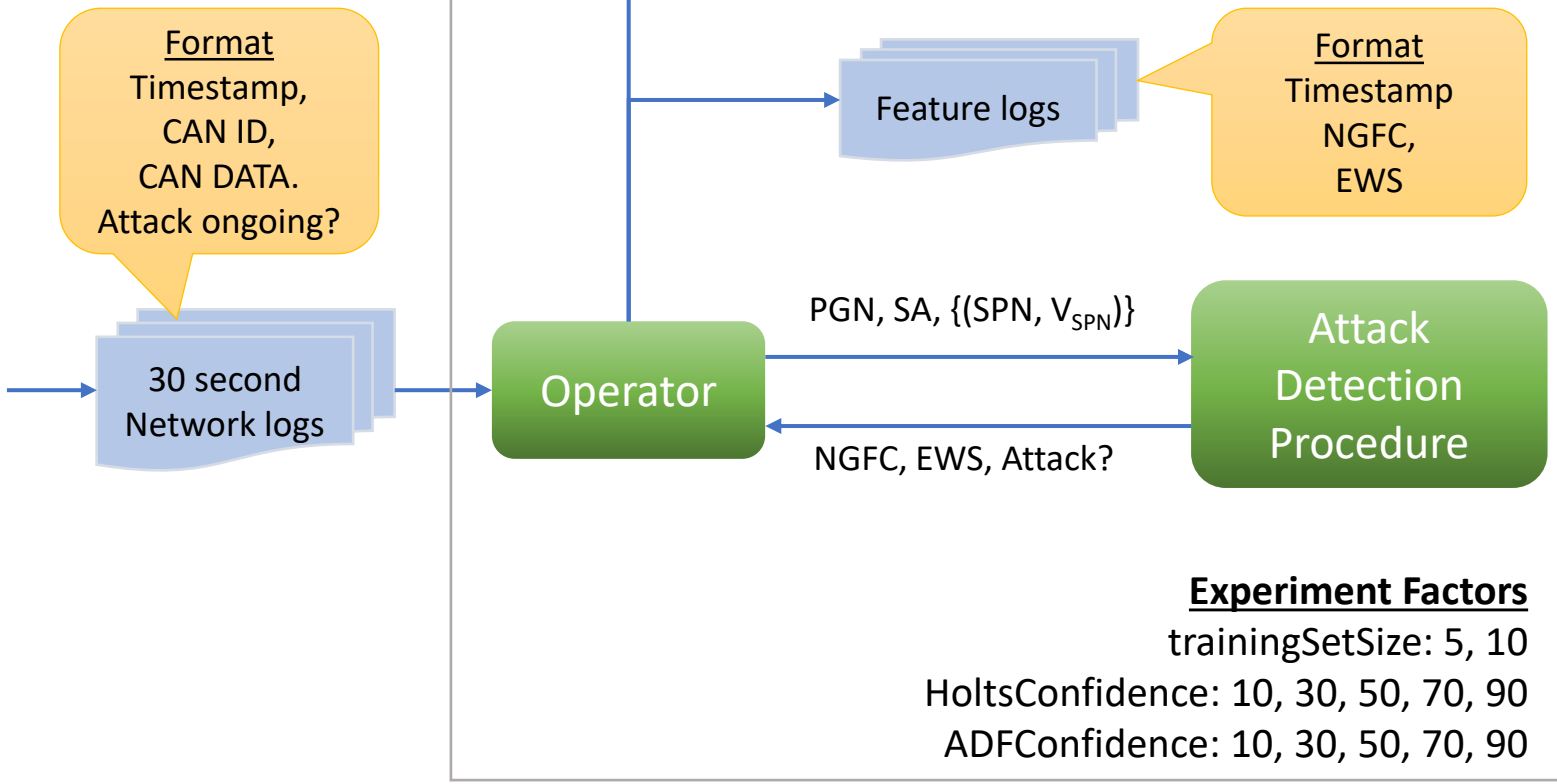
trainingSetSize = 10

Experiment Method (False Alarm Detection)

False Alarm Rate = $100 * \frac{No.of\ flagged\ windows}{Total\ no.of\ windows}$

50 iterations for different experiment configurations

Kenworth T270 driven on an airstrip and on a cross-country trip



Experiment Factors

trainingSetSize: 5, 10

HoltsConfidence: 10, 30, 50, 70, 90

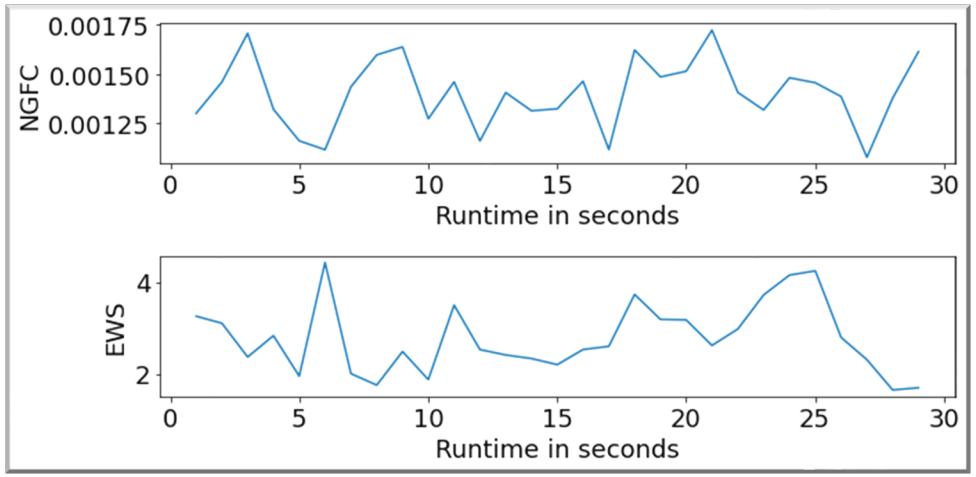
ADFConfidence: 10, 30, 50, 70, 90

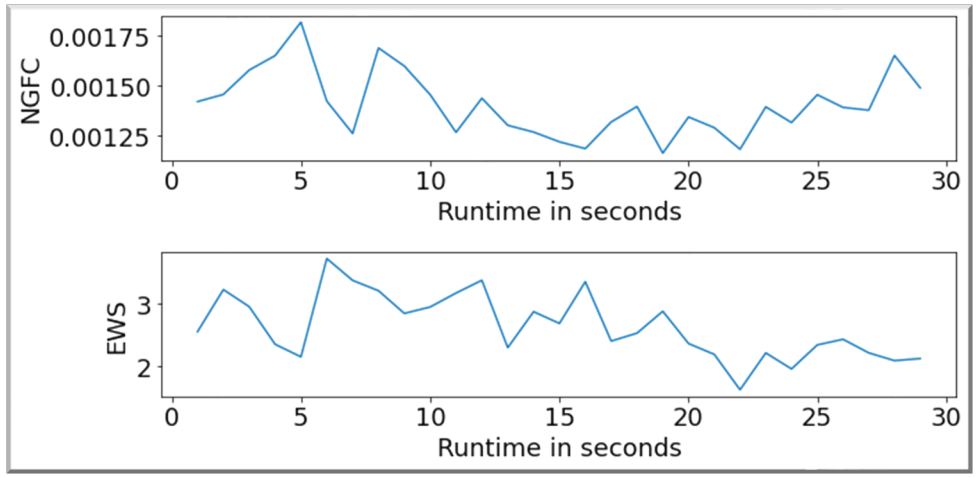
False Alarm Detection Results For Normal Driving on Kenworth T270

Normal Driving on Airstrip

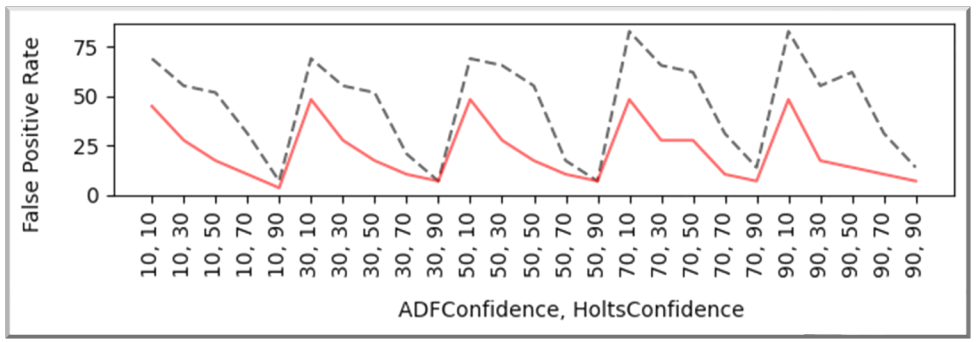
Normal Driving During Cross-Country Trip

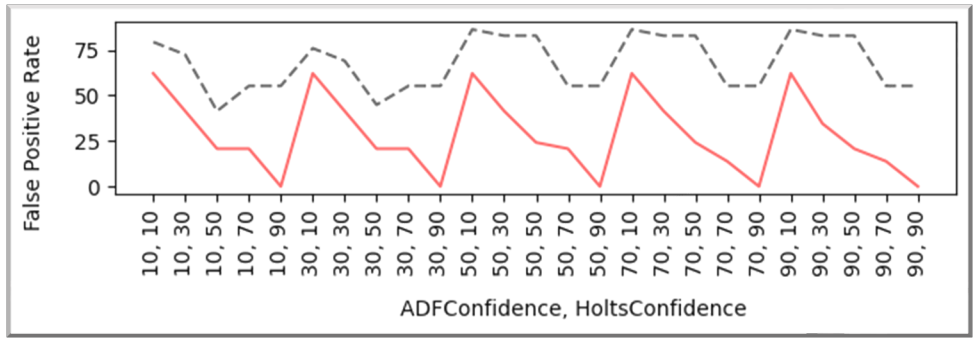
No abnormal feature value deviations





Lowest false positive rate at 90% HoltsConfidence and 10 TrainingSetSize





False Alarm Rate = $100 * \frac{No.of\ flagged\ windows}{Total\ no.of\ windows}$

trainingSetSize = 5

trainingSetSize = 10

High-level Overview of Observations

Less than 4% false positives and 100% detection accuracy on most occasions when

- 90% HoltsConfidence
- 10 trainingSetSize

Organization

- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
- Conclusion and Future Work

Online anomaly-based intrusion detection system

- Behavioral Feature Engineering
- Network Behavior Modelling Through Feature Value Time Series
- Detecting Anomalous Behavior at Runtime
- Performance Analysis
- Summary

Summary

Research Question

Can a system be designed to detect network anomalies on an SAE J1939 network in an online manner?

Contribution

Real-time SAE J1939-based intrusion detection system that does not require offline training

- Uses time series forecasting using minimal historical data to predict an interval of expected behavioral feature values and compares them with the latest values to flag anomalies
- 100% detection precision during most attack experiments
- Less than 4% false positive rate during normal driving experiments

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Performance Analysis

Real-World Demonstration and Discussion

Summary

Organization

Real-time rule-based intrusion detection and prevention system

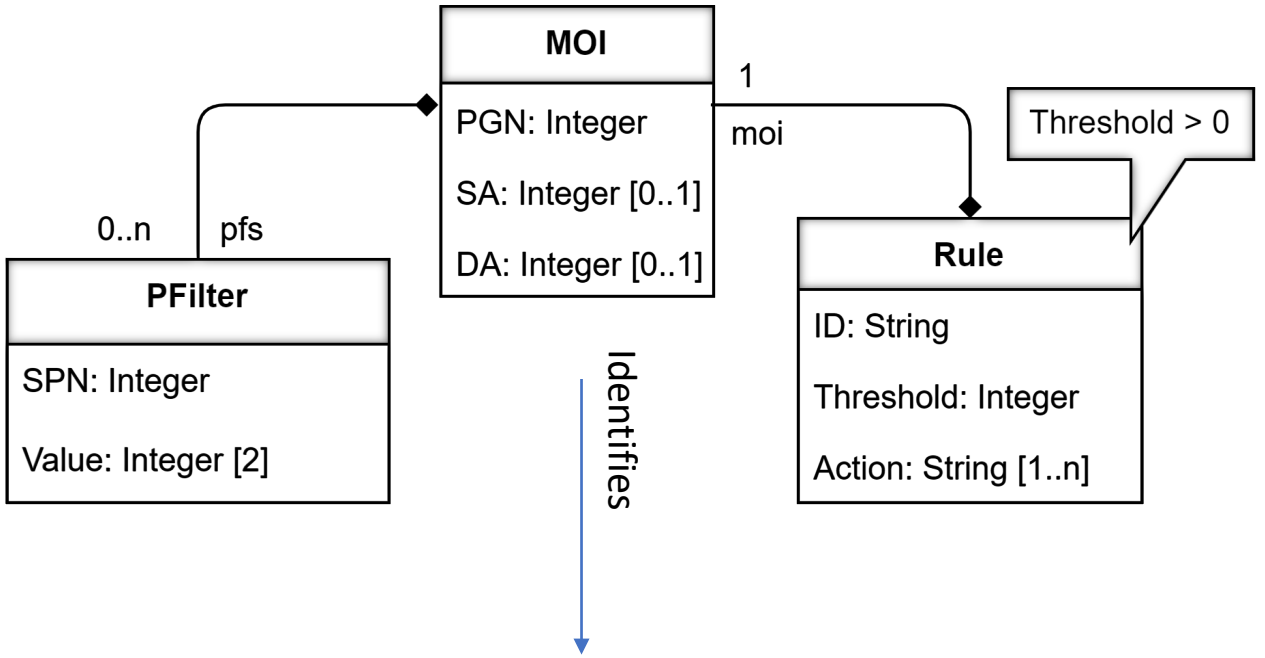
- Introduction
- Background
- Contribution 1
- Contribution 2
- Contribution 3
  - Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks
  - Rule Enforcement
  - Performance Analysis
  - Real-World Demonstration and Discussion
  - Summary
- Conclusion and Future Work

Detection Feature: Message Content

- Invalid PGN, DA, SA
  - For example
    - Spoofed source address
    - Unsupported PGN used for fuzzing
- Hazardous parameter values
  - For example
    - Very low torque (less than 0%) request to the engine controller

Rule Type: *Rule*

Rule is **triggered** when message of interest is being found

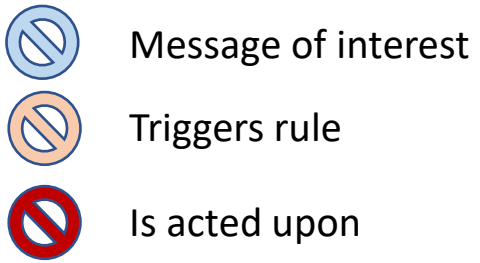


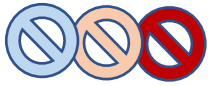
Message of interest is **acted upon** if it triggers rule and the previous *k* message of interest have all triggered rule, *k* being >= Threshold -1.

PGN	DA	SA	Parameters
= rule.moi.PGN	= rule.moi.DA if moi.DA is specified	= rule.moi.SA if moi.SA is specified	= {< spn, v_{spn}>} such that pf.spn[0] ≤ v_{spn} ≤ pf.spn[1], spn = pf.spn ∀ pf ∈ rule.moi.pfs

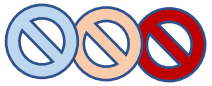
Message of interest

Example *Rule* Enforcement

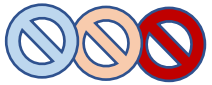




PGN	DA	SA	SPN 518
0	0	0	-125



PGN	DA	SA	SPN 518
0	0	0	-125

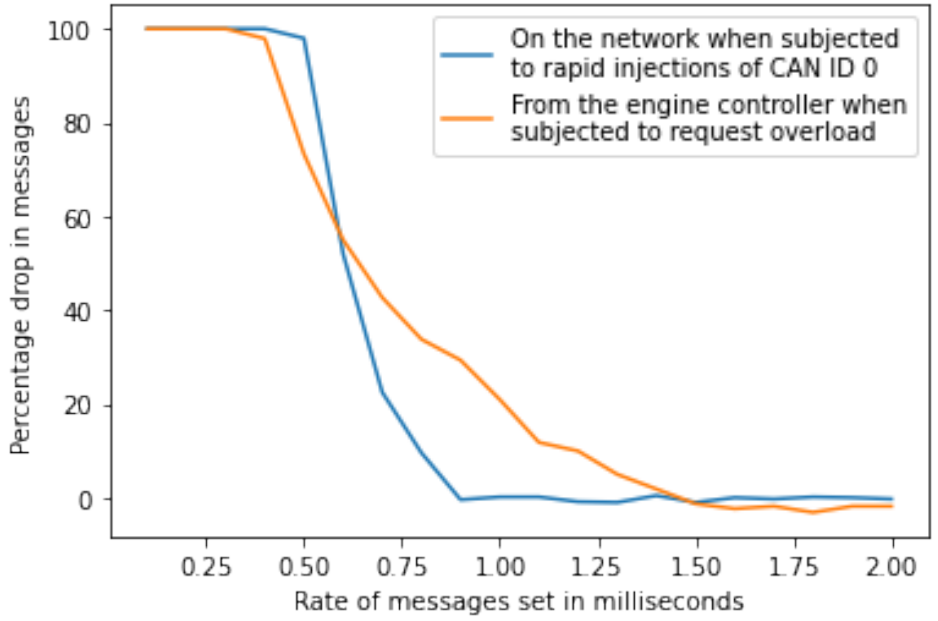


PGN	DA	SA	SPN 518
0	0	0	-125

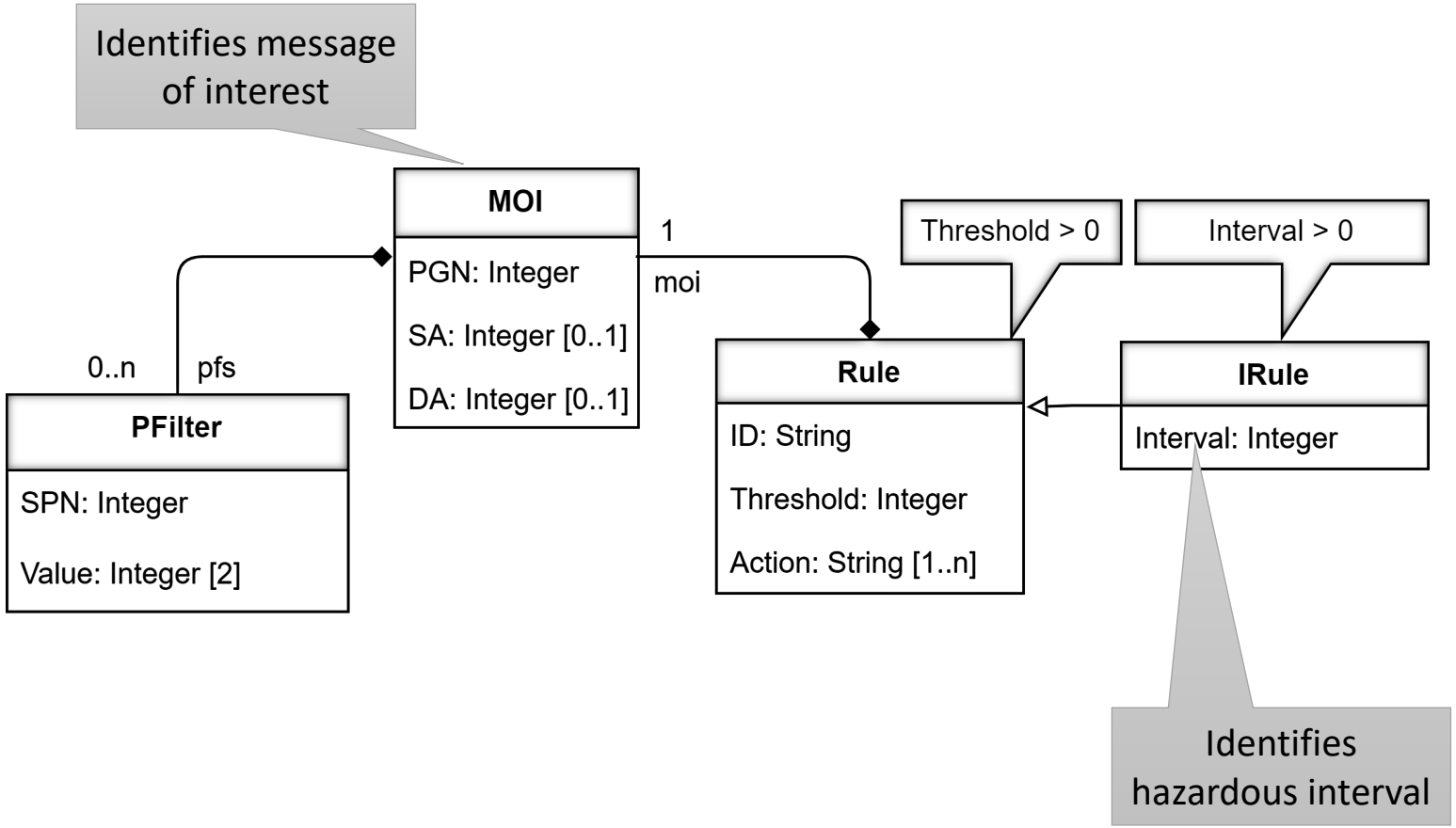
Type	Description	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
Rule	Very low torque request	1	N/A	moi	0	0		518	[-125,-125]

Detection Feature: Transmission Interval

- Messages inserted at very small intervals can lead to denial-of-service (HVDoS)
  - For example
    - Network overload with CAN ID 0
    - Request overload
- Attack is not effective if transmission interval is above certain value
- Messages transmitted below a hazardous interval can be flagged



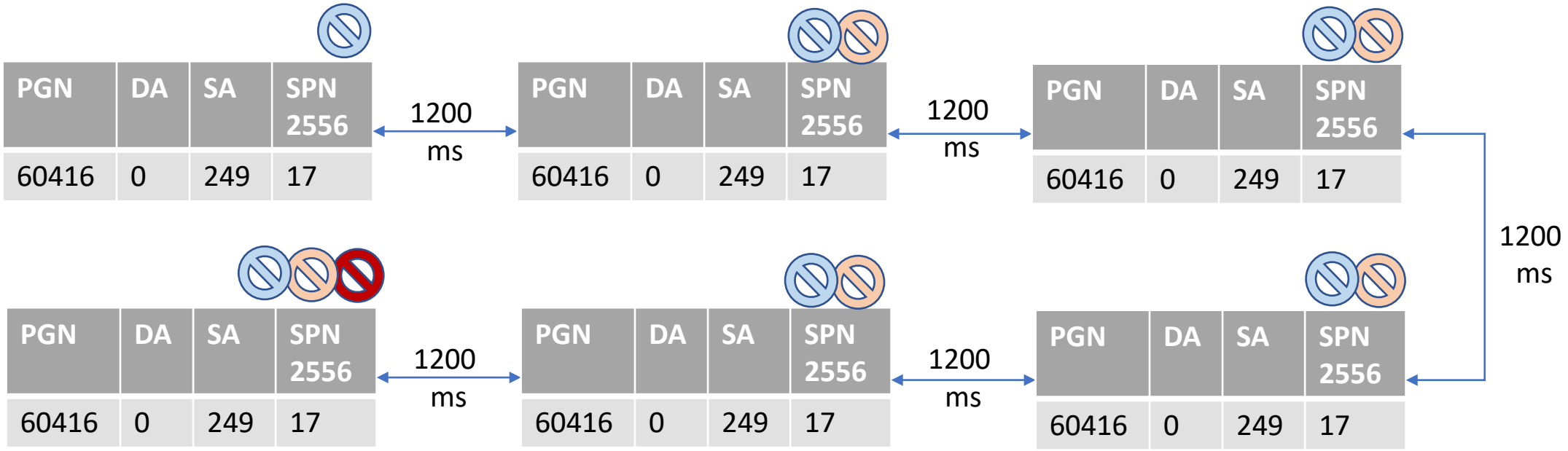
Rule Type: *IRule*



Rule is **triggered** when message of interest is being transmitted at an interval less than the specified from the previous message of interest

Message of interest is **acted upon** if it triggers rule and the previous *k* message of interest have all triggered rule, *k* being >= Threshold - 1.

Example *IRule* Enforcement

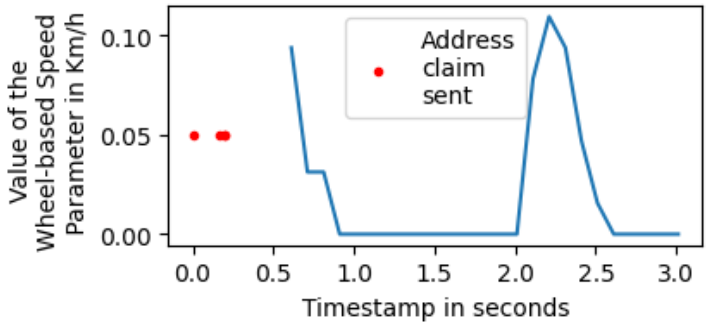


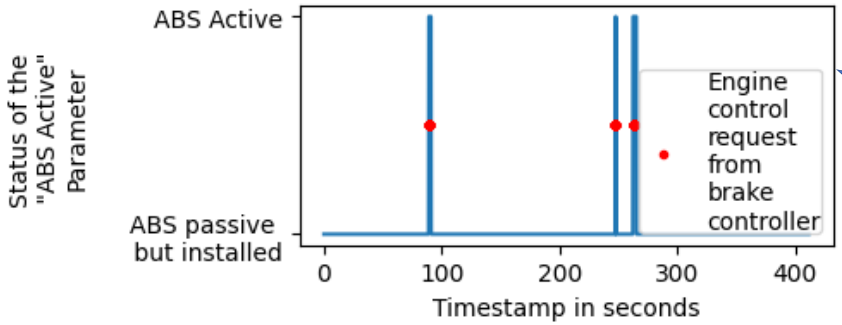
Type	Description	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	Connection exhaustion from a diagnostic tool on engine controller	5	1250	moi	60416	0	249	2556	[17,17]

Detection Feature: Transmission Context

- Some messages can be hazardous when transmitted in certain context
  - Address claim or request to unlock door when vehicle is in motion
  - 0% torque request when vehicle is at speeds below 30 mph
- Some adhoc messages can only be transmitted under certain context
  - Out of context transmissions used for attack can be flagged

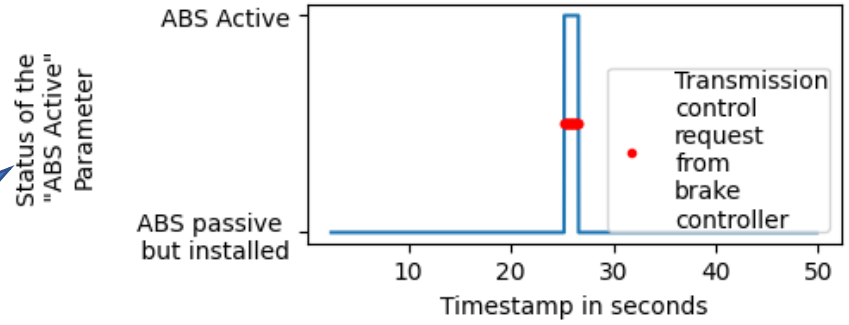
Data collected from the Kenworth T270

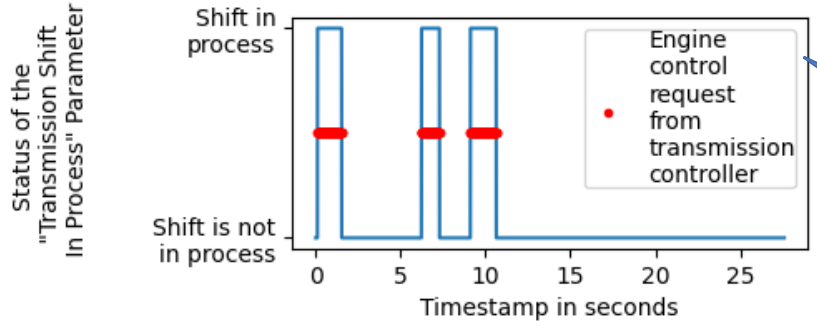




Data collected from the Kenworth T660

Data collected from the Kenworth T270





Data collected from the Kenworth T270

Formalities of Context

If C is a context then it is a set where $\forall c \in C$, c is a two tuple $< p, V >$, where V is a range of values of parameter p and no parameter is repeated in C.

We say that a parameter is *active* in a context C if $\exists c \in C$ such that the first element of c is the parameter whose latest transmitted value lies within the second element c.

We say that a context is *active* if all the constituent parameters are active.

Example Context *C*:

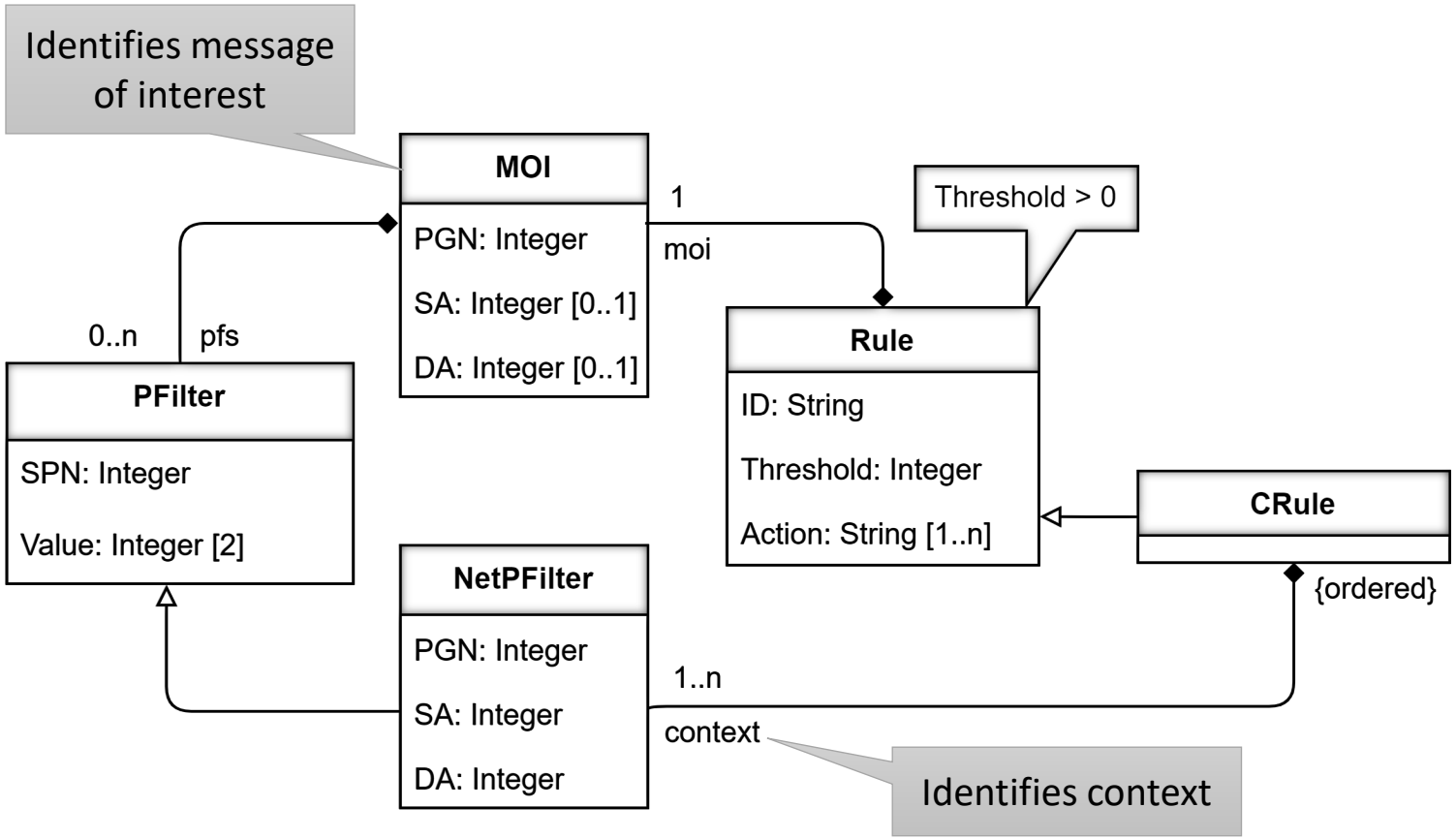
- Vehicle speed within [0, 30] km/h
- Electronic brakes pressed i.e. status between [1,1]

- Vehicle speed is active in *C* if it is within [0, 30] km/h
- Status of electronic brakes is active in C if it is between [1,1]

Context C is active if

- Vehicle speed is active in C
- Status of electronic brakes is active in C

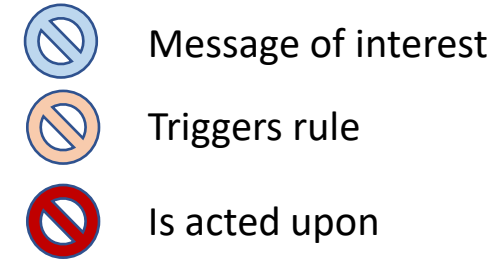
Rule Type: *CRule*



Rule is **triggered** when message of interest is being transmitted and context is active

Message of interest is **acted upon** if it triggers rule and the previous *k* message of interest have all triggered rule, *k* being >= Threshold -1.

Example *CRule* Enforcement



- Parameter “ABS active” deactivated
- Context deactivated

PGN	DA	SA	SPN 563
61441	255	11	1

PGN	DA	SA
0	0	11

- Parameter “ABS active” activated
- Context activated

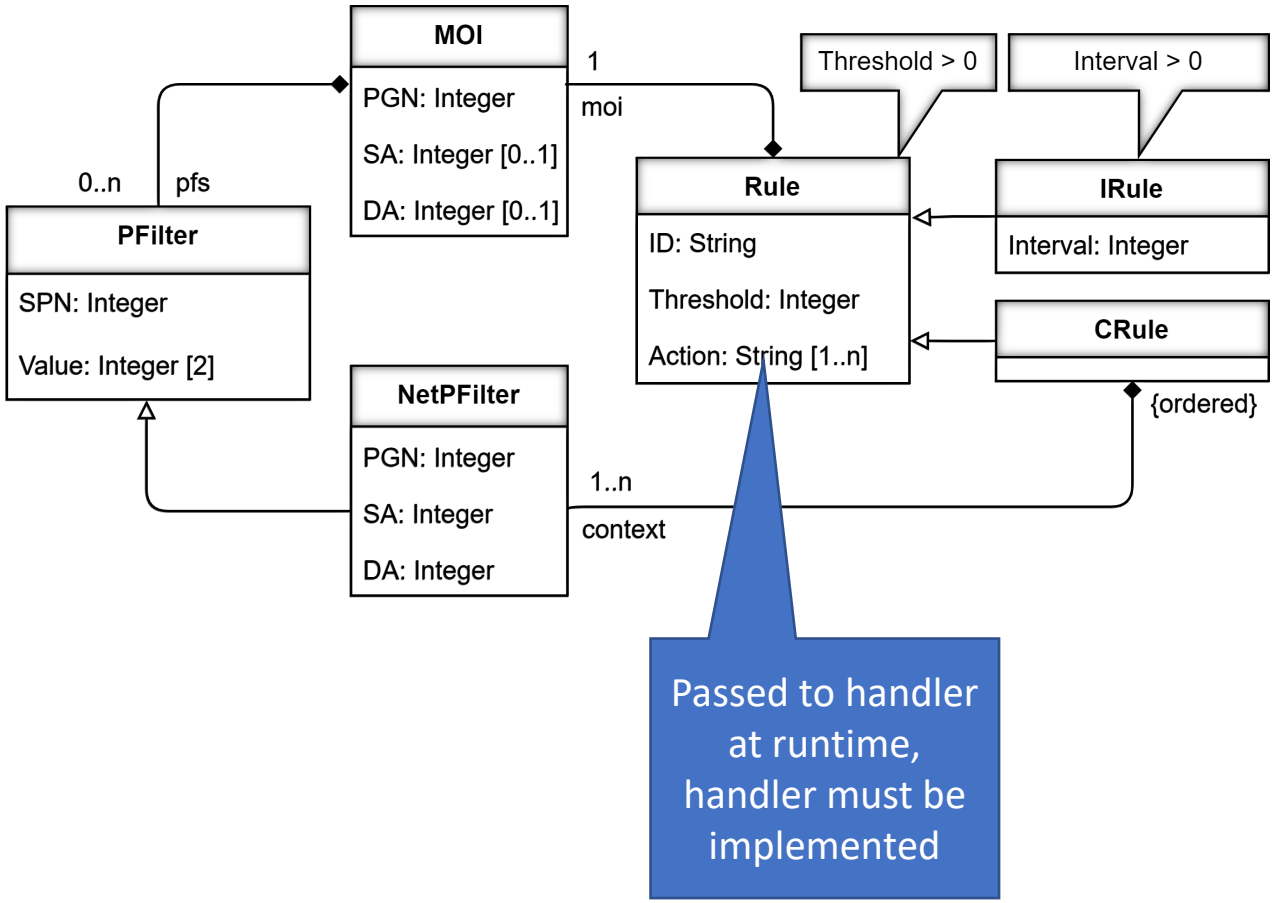
PGN	DA	SA	SPN 563
61441	255	11	0

PGN	DA	SA
0	0	11

Type	Description	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
CRule	Engine control request from ABS when it is not active	1	N/A	moi	0	0	11		
				context	61441	255	11	563	[0,0]

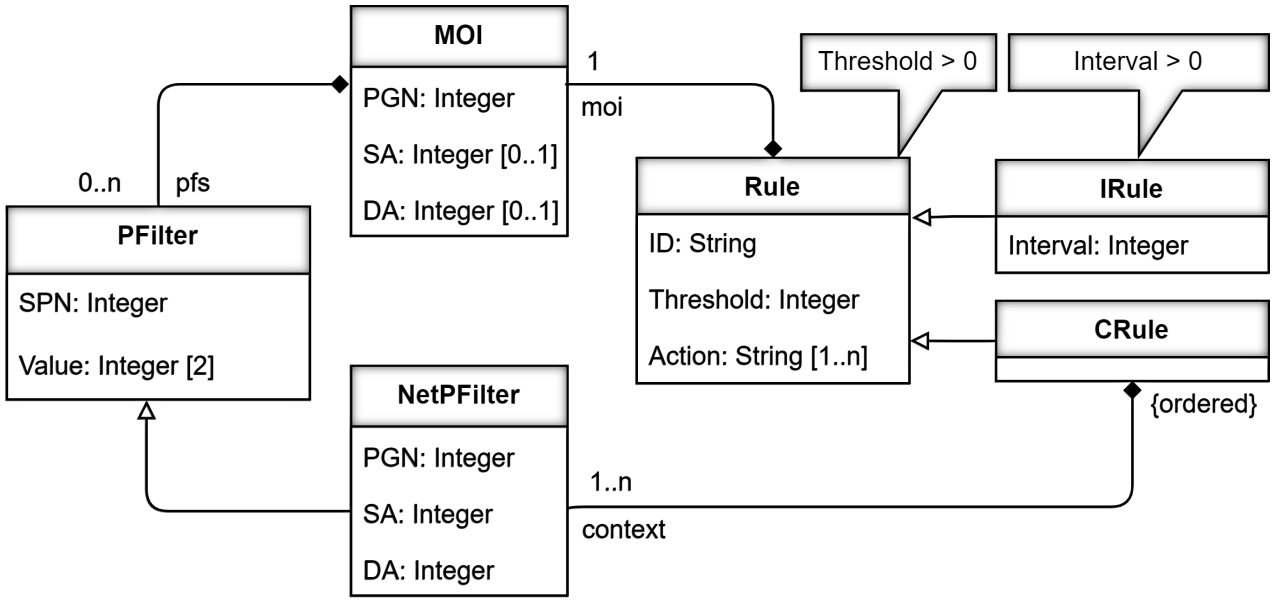
Rule Actions

- Abstractified in this work
- Possible action strategies
  - Raising an alarm
  - Message disruption
    - Transmission of 6 consecutive 0s when bit stuffing is applied
  - Event logging
  - Vehicle restart



Rule Constraints

- If a PGN is used in an MOI object, it cannot be used in a NetPFilter object and vice-versa
- For any MOI object if SA is specified, then so should DA
- No two rules of the same type can exist without PFilters but with the same PGN, DA and SA
  - Implies that maximum number of rules with the same PGN, DA, SA but no Pfilters can be two, one of type Rule and another of type IRule



Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Performance Analysis

Real-World Demonstration and Discussion

Summary

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Performance Analysis

Real-World Demonstration and Discussion

Summary

Preprocessing

Runtime approach

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Performance Analysis

Real-World Demonstration and Discussion

Summary

Preprocessing

Runtime approach

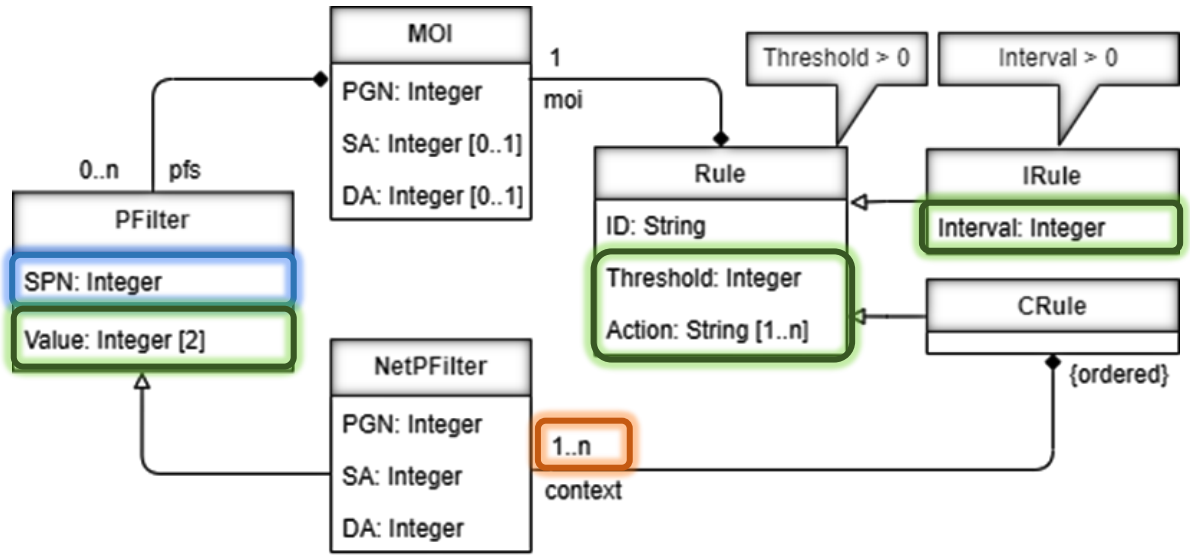
Rules

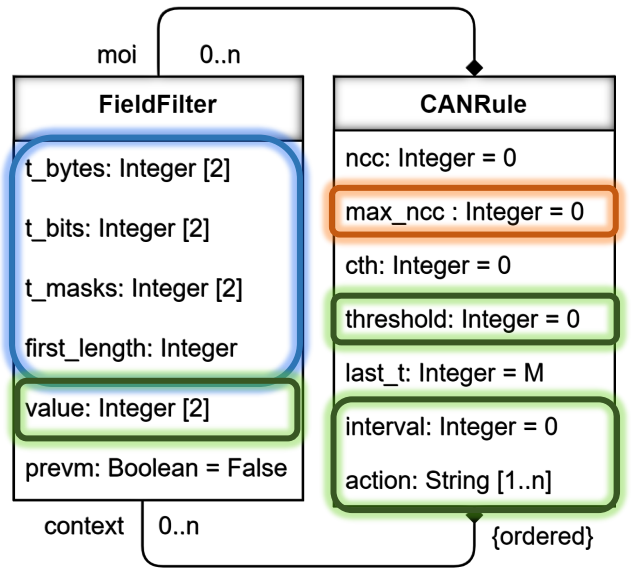
Type	ID	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	R1	2	9	moi	00000_{16}	00_{16}			
Rule	R2	1	N/A	moi	00000_{16}	00_{16}	3116		
Rule	R3	1	N/A	moi	00000_{16}	00_{16}	$0F_{16}$	898	[50,100]
CRule	R4	5	N/A	moi	00000_{16}	00_{16}	$0F_{16}$		
				context	$0FEE1_{16}$	FF_{16}	00_{16}	597	[1,1]
IRule	R5	1	5	moi	$0EA00_{16}$	$0F_{16}$	$0B_{16}$	2540	[65259, 65259]

CANRules

ID	ncc, max_ncc	cth, threshold	last_t, interval	FieldFilter					
				Rela-tion	t_bytes	t_bits	t_masks	first_length	value	prevm
R1	0,0	0,2	M,9							
R2	0,0	0,1	M,0							
R3	0,0	0,1	M,0	moi	2,3	1,1	ff_{16}, ff_{16}	8	[400, 800]	False
R4	0,1	0,5	M, 0	moi						
				context	4,4	5,5	30_{16}, 30_{16}	2	[1,1]	False
R5	0,0	0,1	M, 5	moi	1,3	1,1	ff_{16}, ff_{16}	8	[65259, 65259]	False

** M = Maximum integer value supported by the deployment device architecture





Rules

Type	ID	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	R1	2	9	moi	00000_{16}	00_{16}			
Rule	R2	1	N/A	moi	00000_{16}	00_{16}	31_{16}		
Rule	R3	1	N/A	moi	00000_{16}	00_{16}	$0F_{16}$	898	[50,100]
CRule	R4	5	N/A	moi	00000_{16}	00_{16}	$0F_{16}$		
				context	$0FEE1_{16}$	FF_{16}	00_{16}	597	[1,1]
IRule	R5	1	5	moi	$0EA00_{16}$	$0F_{16}$	$0B_{16}$	2540	[65259, 65259]

Temporary Lookup Table

PGN	DA	SA	Rule	Relation	Indexes
00000_{16}	00_{16}		R1	moi	[]
00000_{16}	00_{16}	31_{16}	R2	moi	[]
00000_{16}	00_{16}	$0F_{16}$	R3	moi	[]
			R4	moi	[]
$0FEE1_{16}$	FF_{16}	00_{16}	R4	context	[0]
$0EA00_{16}$	$0F_{16}$	$0B_{16}$	R5	moi	[]

Keys Values

CANRules

ID	ncc, max_ncc	cth, threshold	last_t, interval	FieldFilter						
				Rela-tion	t_bytes	t_bits	t_masks	first_length	value	prevm
R1	0,0	0,2	M,9							
R2	0,0	0,1	M,0							
R3	0,0	0,1	M,0	moi	2,3	1,1	ff_{16}, ff_{16}	8	[400, 800]	False
R4	0,1	0,5	M, 0	moi						
				context	4,4	5,5	30_{16}, 30_{16}	2	[1,1]	False
R5	0,0	0,1	M, 5	moi	1,3	1,1	ff_{16}, ff_{16}	8	[65259, 65259]	False

** M = Maximum integer value supported by the deployment device architecture

Rules

Type	ID	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	R1	2	9	moi	00000_{16}	00_{16}			
Rule	R2	1	N/A	moi	00000_{16}	00_{16}	31_{16}		
Rule	R3	1	N/A	moi	00000_{16}	00_{16}	$0F_{16}$	898	[50,100]
CRule	R4	5	N/A	moi	00000_{16}	00_{16}	$0F_{16}$		
				context	$0FEE1_{16}$	FF_{16}	00_{16}	597	[1,1]
IRule	R5	1	5	moi	$0EA00_{16}$	$0F_{16}$	$0B_{16}$	2540	[65259, 65259]

CANRules

ID	ncc, max_ncc	cth, threshold	last_t, interval	FieldFilter						
				Rela-tion	t_bytes	t_bits	t_masks	first_length	value	prevm
R1	0,0	0,2	M,9							
R2	0,0	0,1	M,0							
R3	0,0	0,1	M,0	moi	2,3	1,1	ff_{16}, ff_{16}	8	[400, 800]	False
R4	0,1	0,5	M, 0	moi						
				context	4,4	5,5	30_{16}, 30_{16}	2	[1,1]	False
R5	0,0	0,1	M, 5	moi	1,3	1,1	ff_{16}, ff_{16}	8	[65259, 65259]	False

** M = Maximum integer value supported by the deployment device architecture

Temporary Lookup Table

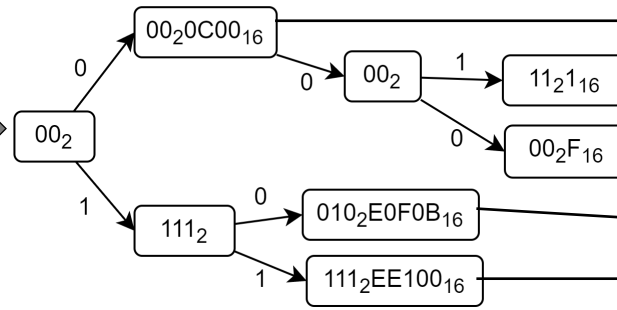
PGN	DA	SA	Rule	Relation	Indexes
00000_{16}	00_{16}		R1	moi	[]
00000_{16}	00_{16}	31_{16}	R2	moi	[]
00000_{16}	00_{16}	$0F_{16}$	R3	moi	[]
			R4	moi	[]
$0FEE1_{16}$	FF_{16}	00_{16}	R4	context	[0]
$0EA00_{16}$	$0F_{16}$	$0B_{16}$	R5	moi	[]

Keys | Values

Arbitration Field Strings

$00_2 00_2 0_2 C00_{16}$
$00_2 00_2 0_2 C00_{16} 00_2 11_2 1_{16}$
$00_2 00_2 0_2 C00_{16} 00_2 00_2 F_{16}$
$00_2 111_2 010_2 E0F0B_{16}$
$00_2 111_2 111_2 EE100_{16}$

Radix Tree



Targets

Rules without PFilters

Rules with PFilters

last_byte	np_can rules	rlinks		
		canrule	relation	indexes
0	[R1]			
0	[R2]			
3		R3	moi	[]
		R4	moi	[]
3		R5	moi	[]
4		R4	context	[0]

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Preprocessing

Runtime approach

Performance Analysis

Real-World Demonstration and Discussion

Summary

SOF

IDLE state

Attack detected?

targets

No

TRACE state

- Accumulate bits
- Compare with node labels
- Retrieve targets
- Check for attack
- Accumulate targets with rlinks
- Change node if possible
- Cannot trace further ?
  - Targets with rlinks retrieved?

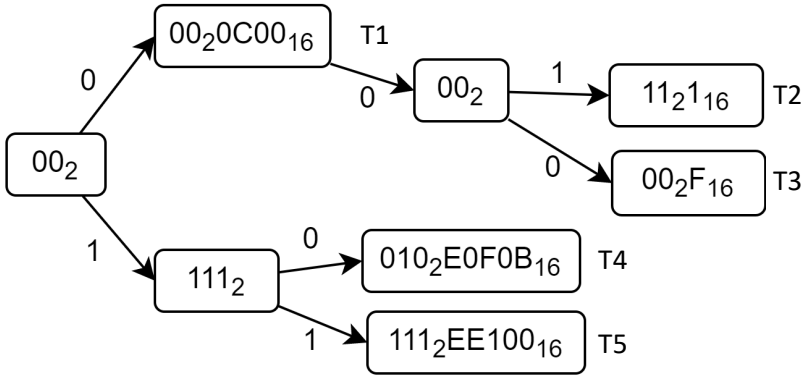
Yes

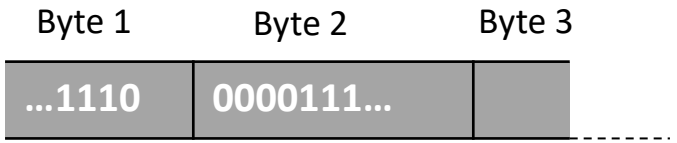
BUFFER state

- Accumulate bytes
- Buffered upto maximum last_byte of accumulated targets?
  - Process accumulated targets with rlinks

Targets

	last _byte	np_can rules	rlinks		
			canrule	relation	indexes
T1	0	[R1]			
T2	0	[R2]			
T3	3		R3	moi	[]
			R4	moi	[]
T4	3		R5	moi	[]
T5	4		R4	context	[0]





Start of frame (SOF)

Start of CAN data

SOF

IDLE state

Attack detected?

targets

No

TRACE state

- Accumulate bits
- Compare with node labels
- Retrieve targets
- **Check for attack**
- Accumulate targets with rlinks
- Change node if possible
- Cannot trace further ?
  - Targets with rlinks retrieved?

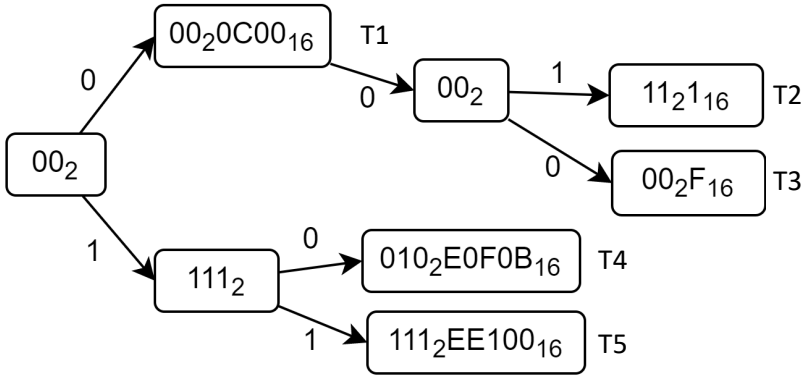
Yes

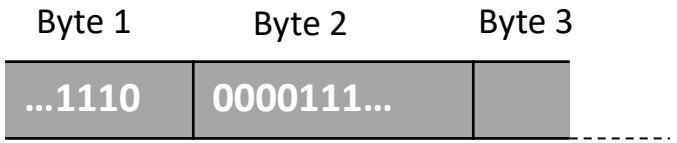
BUFFER state

- Accumulate bytes
- Buffered upto maximum last_byte of accumulated targets?
  - Process accumulated targets with rlinks

Targets

	last _byte	np_can rules	rlinks		
			canrule	relation	indexes
T1	0	[R1]			
T2	0	[R2]			
T3	3		R3	moi	[]
			R4	moi	[]
T4	3		R5	moi	[]
T5	4		R4	context	[0]





Start of frame (SOF)

Start of CAN data

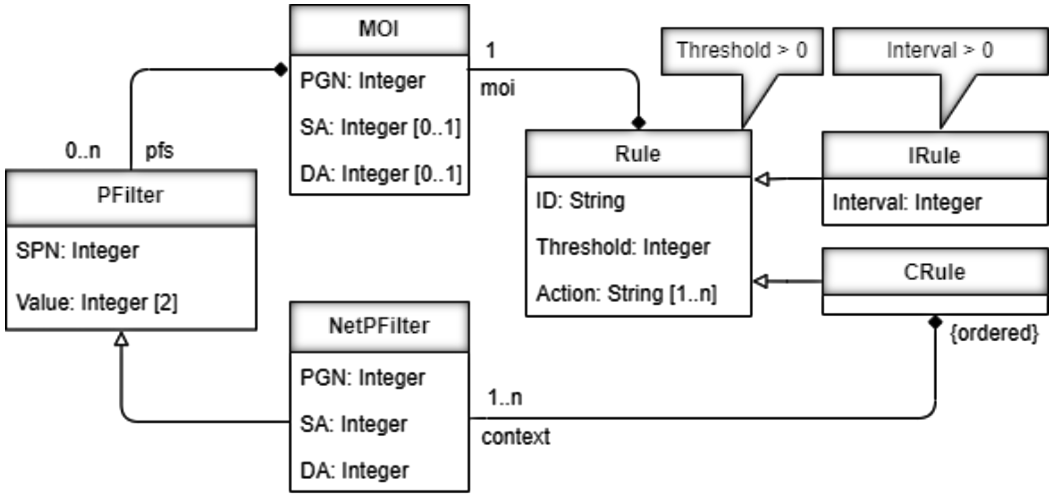
Attack Detection in TRACESState

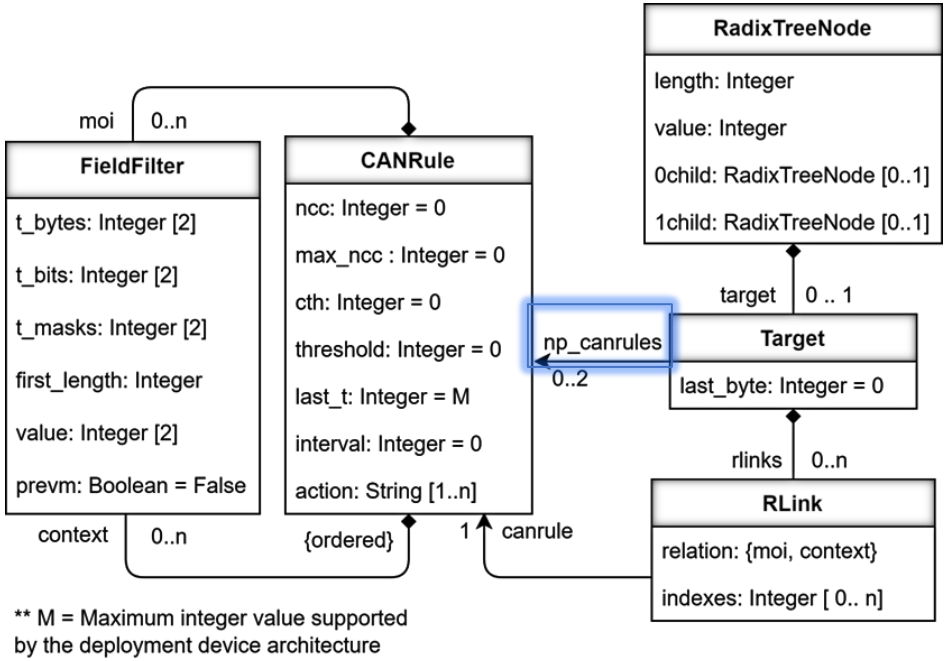
Assume rule is triggered

Check if rule is not triggered

This and the previous (threshold – 1) messages of interest have all triggered rule

```
forall canrule ∈ target.np_canrules do
    canrule.cth ++
    if canrule.interval > 0 then
        time ← get milliseconds from start
        if time - canrule.last_t > canrule.interval then
            canrule.cth ← 0
        end
        canrule.last_t ← time
    end
    if canrule.cth ≥ canrule.threshold then
        act(canrule.action)
        return True
    end
end
return False
```





SOF

IDLE state

Attack detected?

targets

No

TRACE state

- Accumulate bits
- Compare with node labels
- Retrieve targets
- Check for attack
- Accumulate targets with rlinks
- Change node if possible
- Cannot trace further ?
  - Targets with rlinks retrieved?

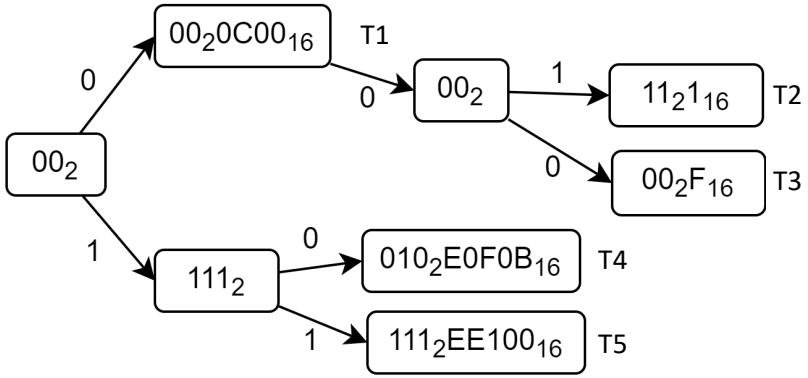
Yes

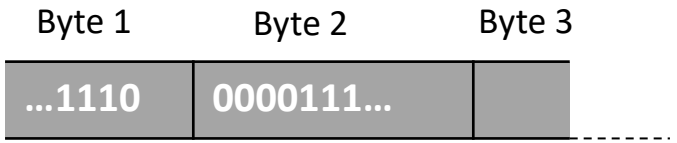
BUFFER state

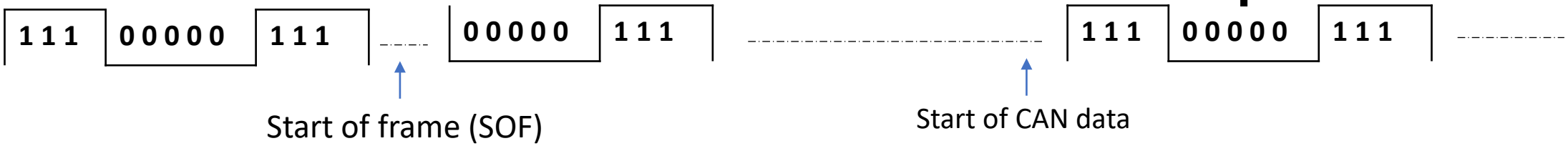
- Accumulate bytes
- Buffered upto maximum last_byte of accumulated targets?
  - **Process accumulated targets with rlinks**

Targets

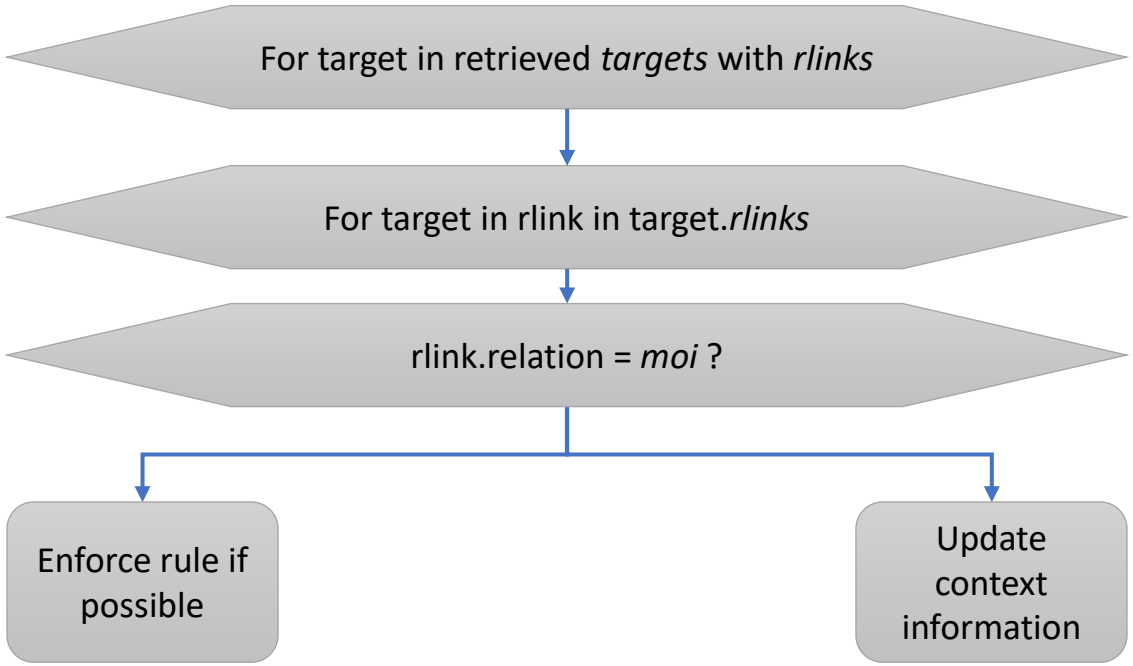
	last _byte	np_can rules	rlinks		
			canrule	relation	indexes
T1	0	[R1]			
T2	0	[R2]			
T3	3		R3	moi	[]
			R4	moi	[]
T4	3		R5	moi	[]
T5	4		R4	context	[0]

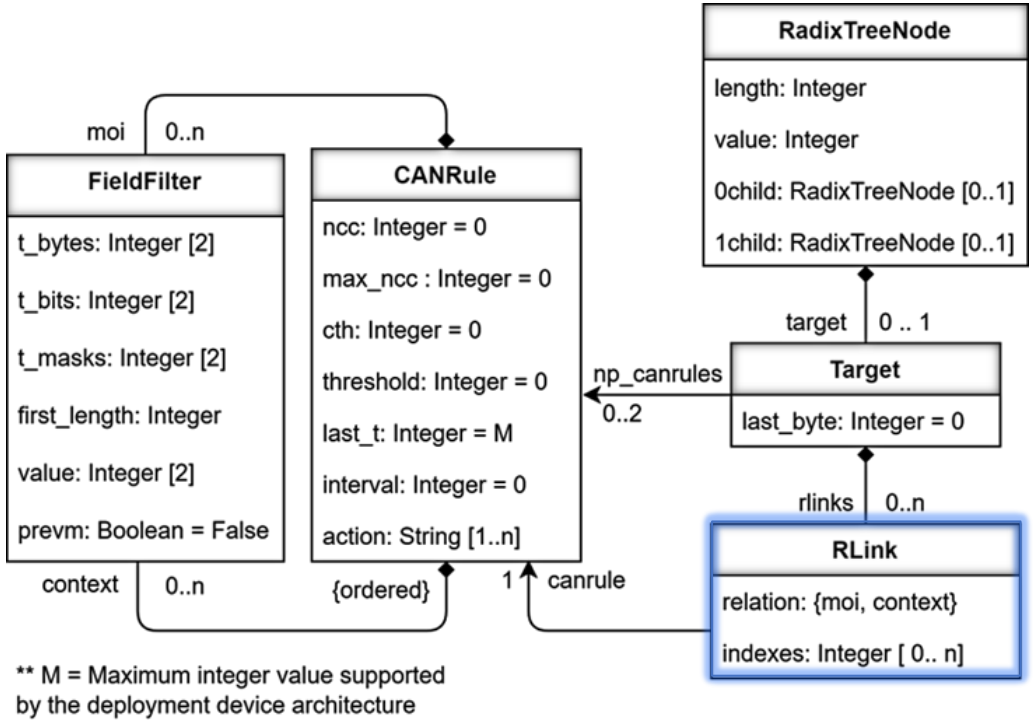




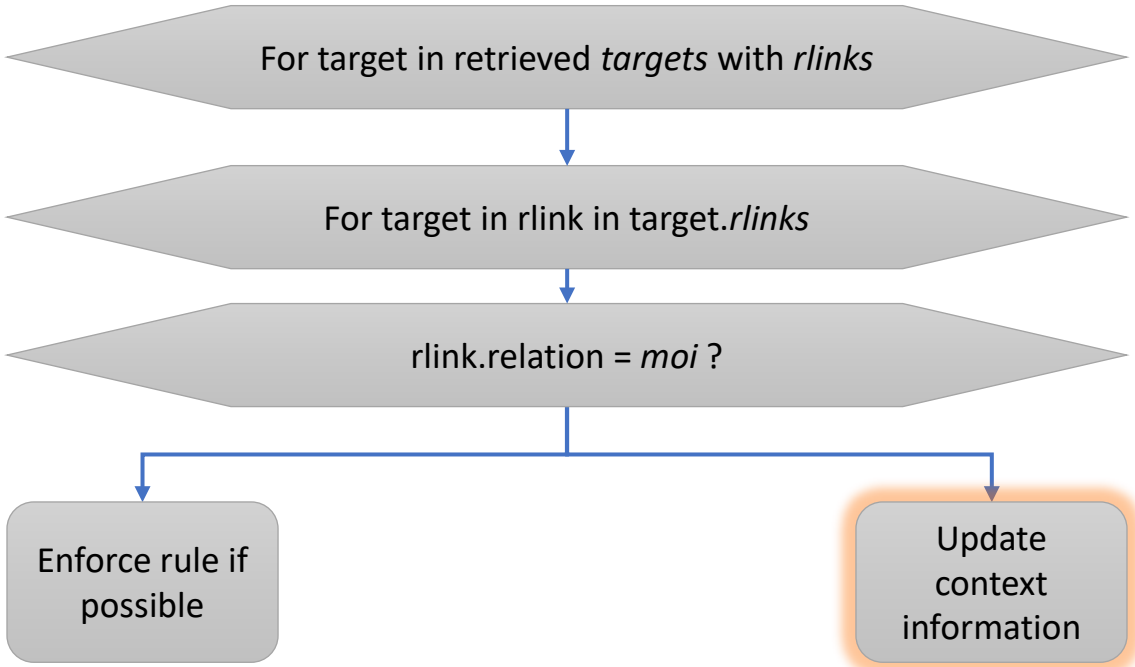


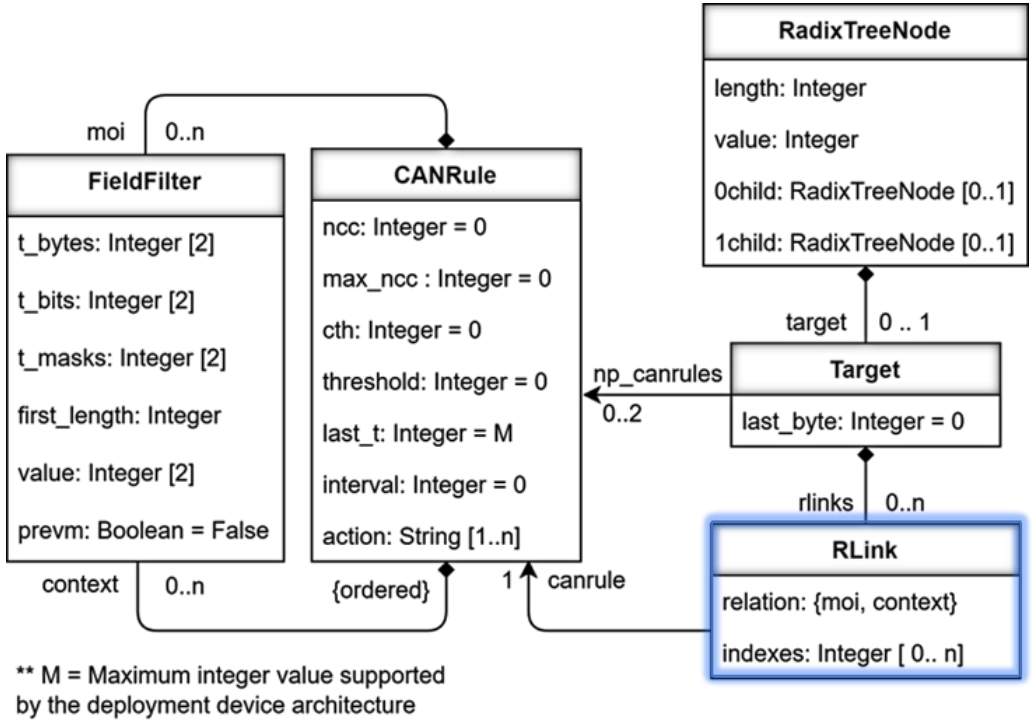
Process Retrieved Targets with Rlinks





Process Retrieved Targets with Rlinks





Context Update

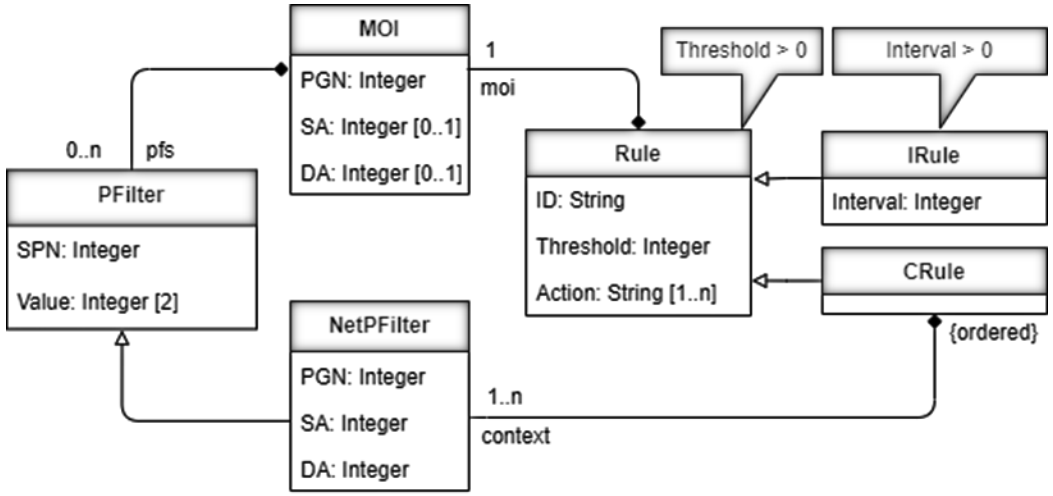
Check if parameter is active

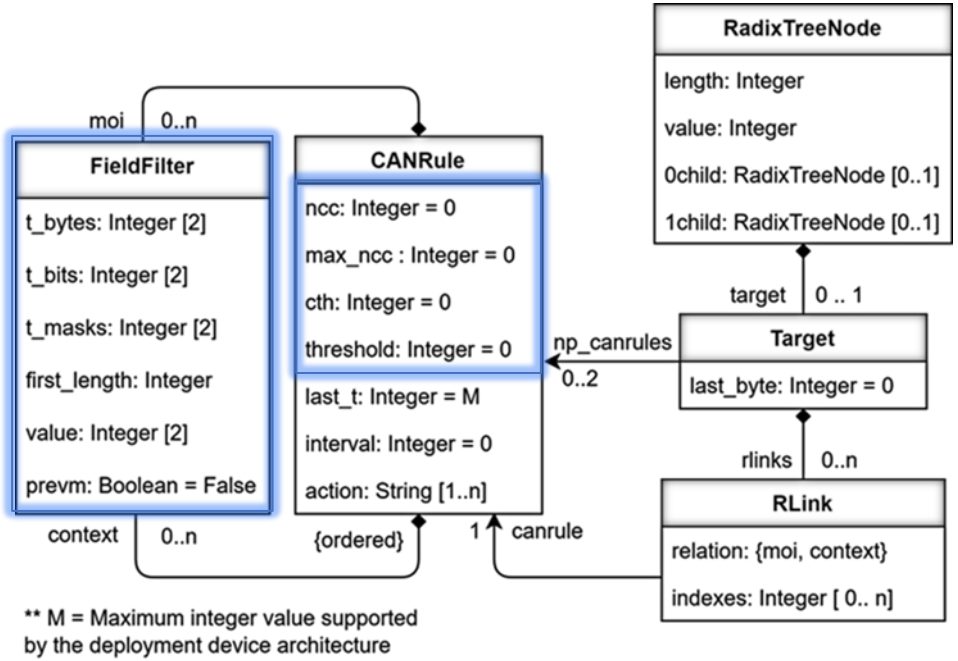
Check if parameter is just activated/ deactivated

Update active parameter count

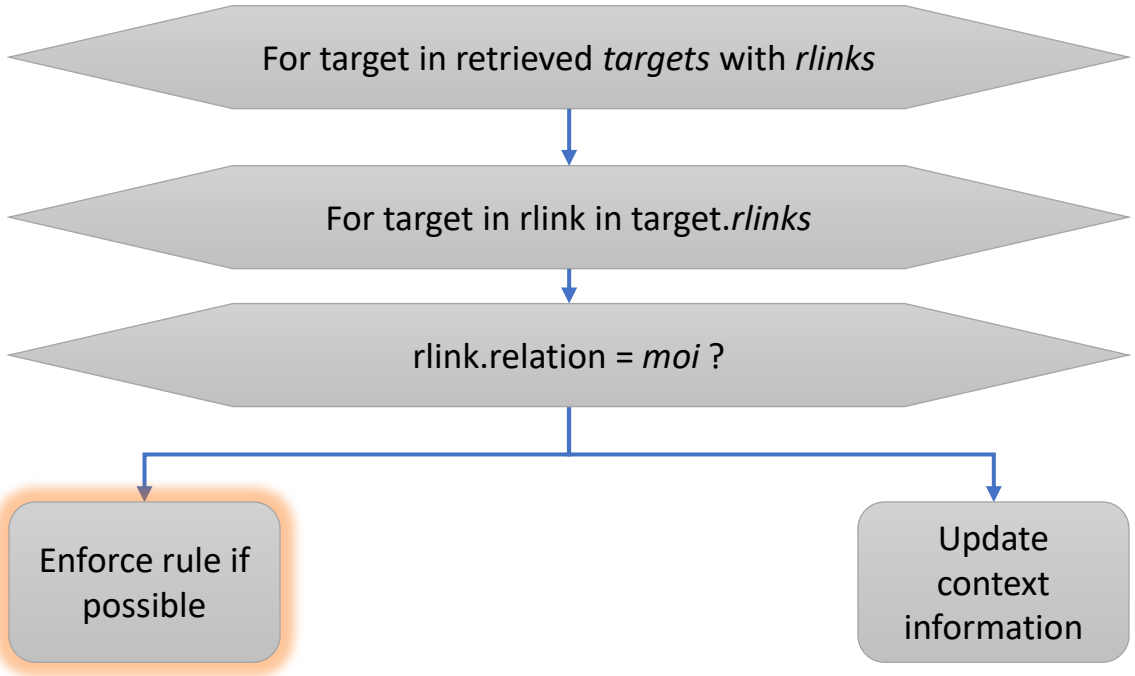
Maintain parameter active status

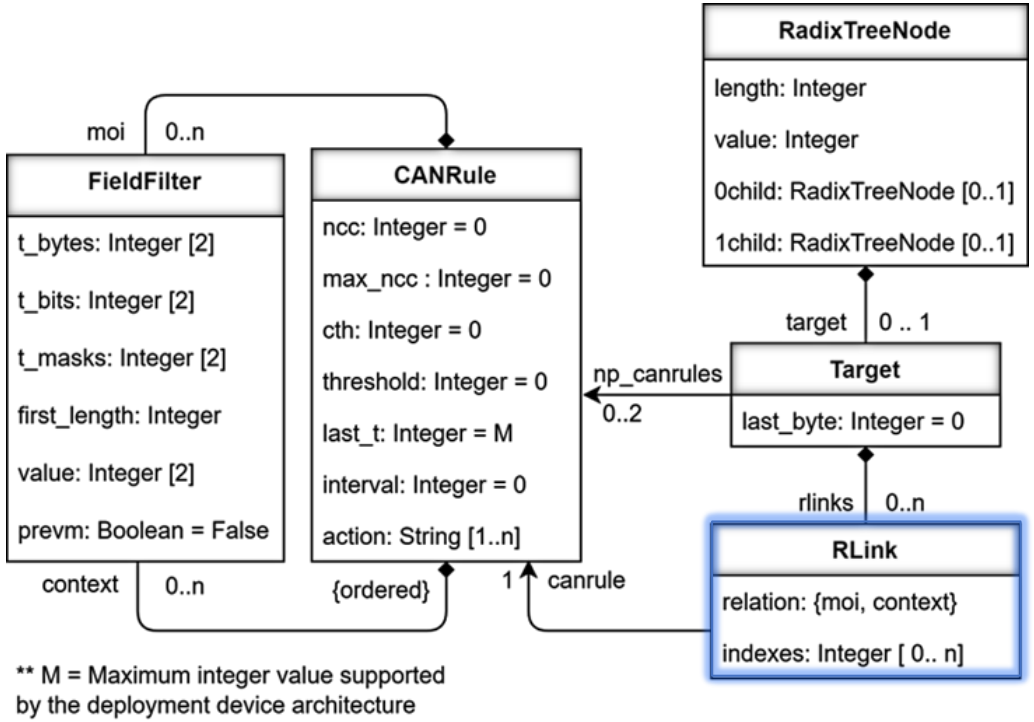
```
for i in rlink.indexes do
    m ← (rlink.canrule.context[i].value[0] ≤
    get_value(buffered data bytes,
    ff) ≤ rlink.canrule.context[i].value[1])
    if m ≠ rlink.canrule.context[i].prevm
     then
        if m = True then
            rlink.canrule.ncc ++;
        end
        else
            rlink.canrule.ncc –;
        end
        rlink.canrule.context[i].prevm ← m
    end
end
```





Process Retrieved Targets with Rlinks





Rule Enforcement

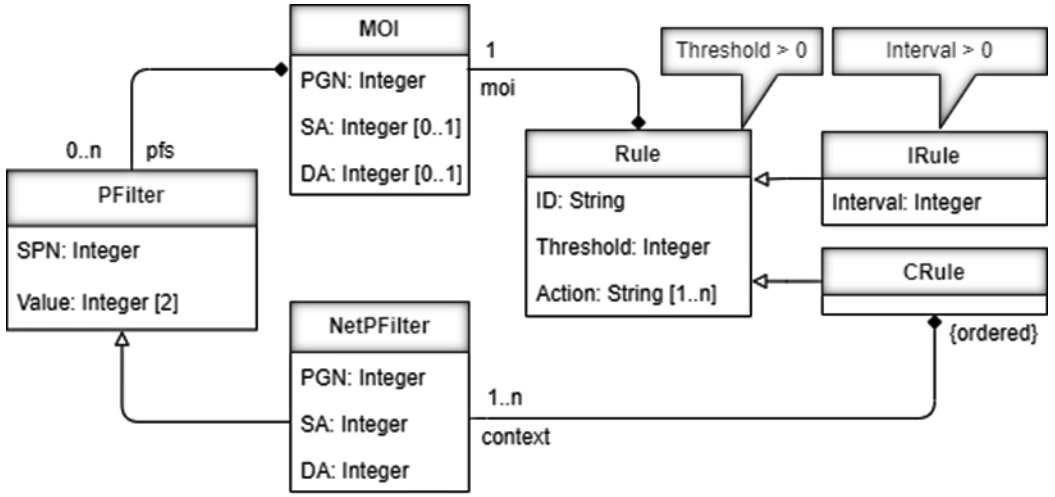
Check if message of interest

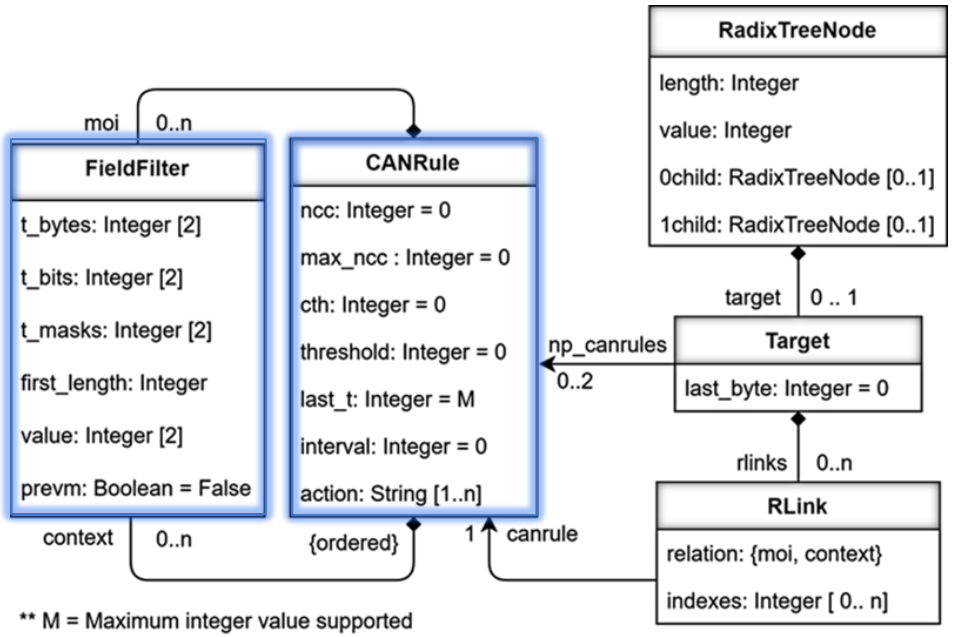
Assume rule is triggered

Check if rule is not triggered

This and the previous (threshold – 1) messages of interest have all triggered rule

```
m ← True
for ff ∈ rlink.canrule.moi do
    m ← m AND (ff.value[0] ≤
    get_value(buffered data bytes, ff) ≤
     ff.value[1]);
end
if m = True then
    rlink.canrule.cth ++
    if rlink.canrule.interval > 0 then
        time ← get milliseconds from start
        if time - rlink.canrule.last_t >
        rlink.canrule.interval then
            rlink.canrule.cth ← 0;
        end
        rlink.canrule.last_t ← time
    end
    if rlink.canrule.ncc <
     rlink.canrule.max_ncc then
        rlink.canrule.cth ← 0;
    end
    if rlink.canrule.cth ≥
     rlink.canrule.threshold then
        act(rlink.canrule.action)
        return
    end
end
```





Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Performance Analysis

Real-World Demonstration and Discussion

Summary

Preliminaries

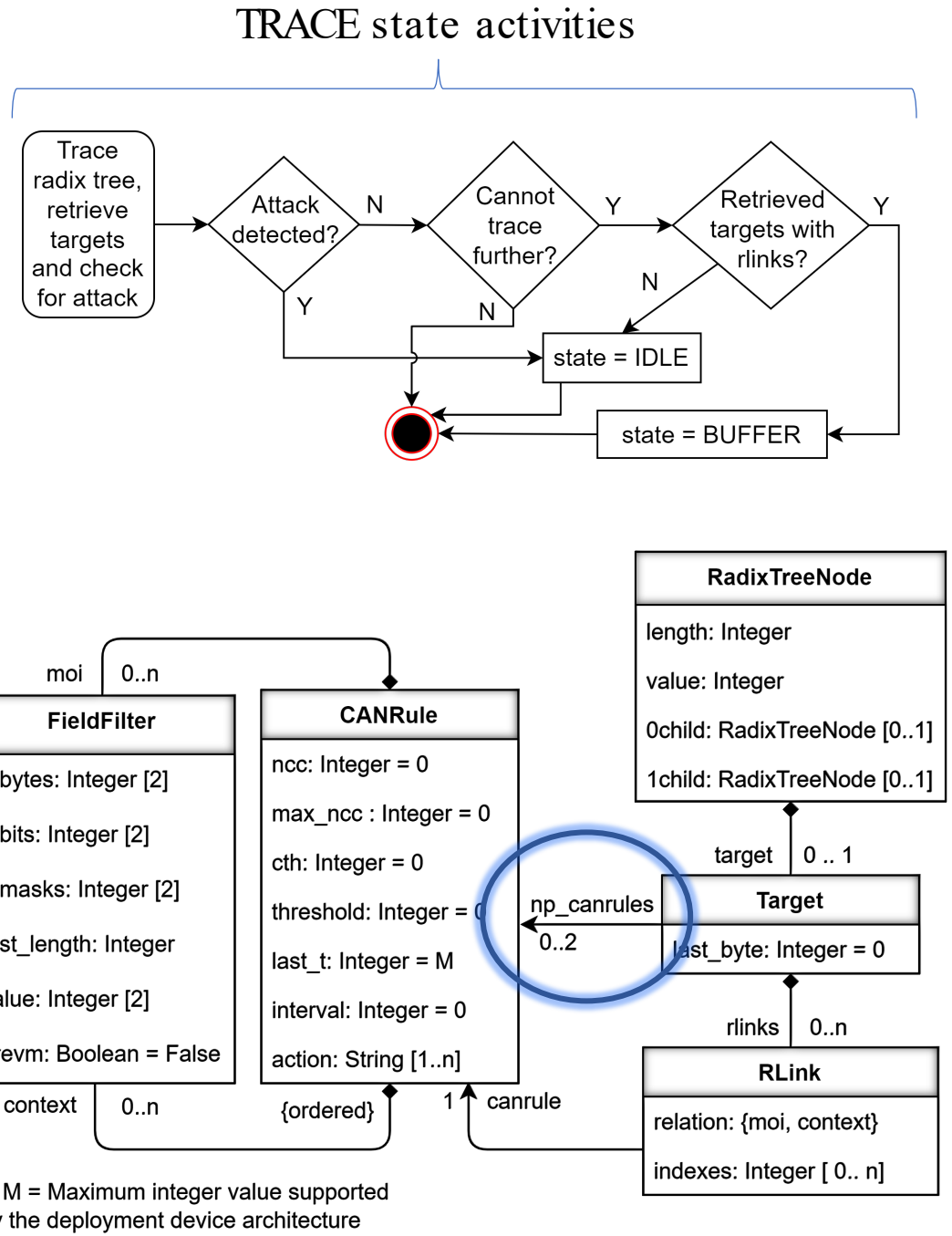
- Analysis question: How many rules can be enforced in the worst-case?
  - How many rules without Pfilters can be enforced in the TRACE state?
  - How many rules with Pfilters can be enforced in the BUFFER state ?
- Experiment platform: Embedded Automotive Application Development Boards
  - Teensy 3.6
    - 180MHz ARM Cortex-M4 processor
    - 256 KB of RAM
  - Teensy 4.1
    - 600MHz ARM Cortex-M7 processor
    - 1024 KB of RAM

Analysis Question

- How many rules can be enforced in the worst-case?
  - **How many rules without Pfilters can be enforced in the TRACE state?**
  - How many rules with Pfilters can be enforced in the BUFFER state ?

Analysis

- Constant time activities in TRACE state
  - Rules without Pfilters are accessed via *np_canrules*
  - Maximum 2 *np_canrules* per Target
- Finish in less than 2 microseconds when 2 *np_canrules* are accessed
  - 2 microseconds is the bit-width on a 500 kbps CAN bus
  - Real-time irrespective of the number of rules



Analysis Question

- How many rules can be enforced in the worst-case?
  - How many rules without Pfilters can be enforced in the TRACE state?
  - **How many rules with Pfilters can be enforced in the BUFFER state ?**

Analysis Methodology

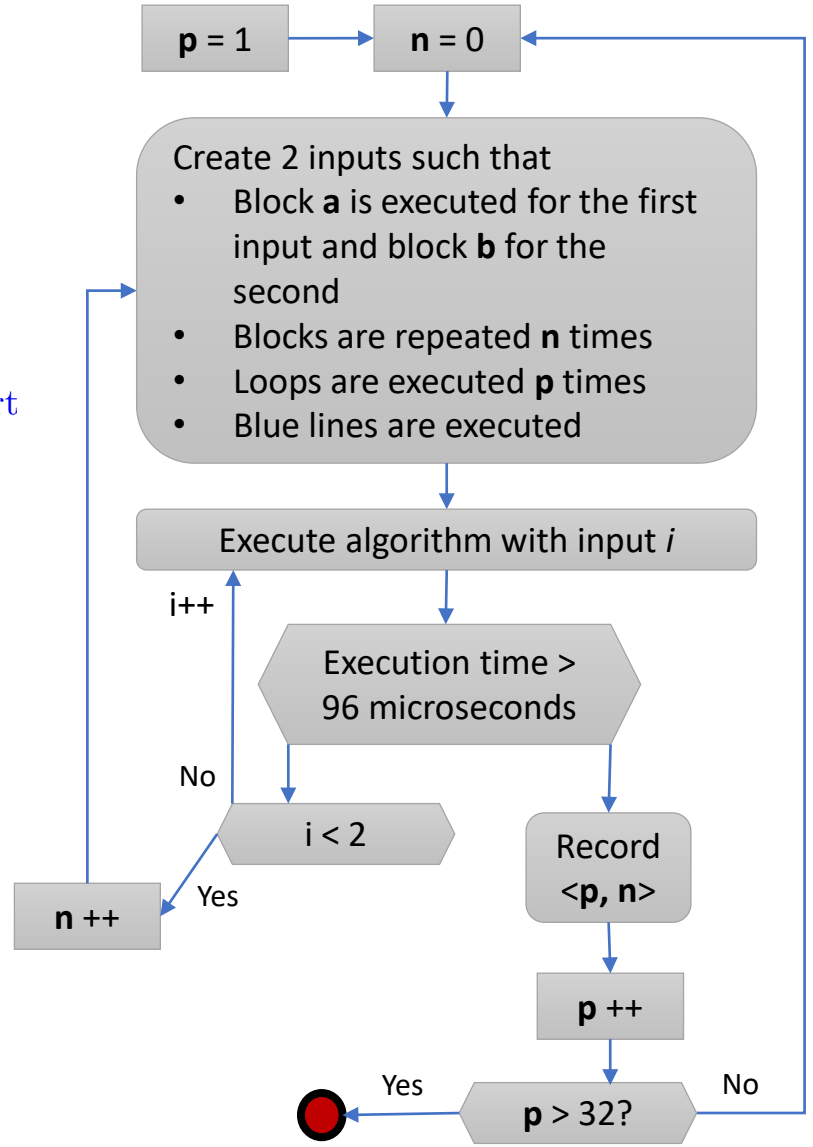
Process retrieved targets with rlinks (*targets*)

```
forall target ∈ targets do
    forall rlink ∈ target.rlinks do
        if rlink.relation = moi then
        end
        else
        end
    end
end
```

Block a

```
m ← True
for ff ∈ rlink.canrule.moi do
    m ← m AND (ff.value[0] ≤
    get_value(buffered data bytes, ff) ≤
     ff.value[1]);
end
if m = True then
    rlink.canrule.cth ++
    if rlink.canrule.interval > 0 then
        time ← get milliseconds from start
        if time - rlink.canrule.last_t >
        rlink.canrule.interval then
            rlink.canrule.cth ← 0;
        end
        rlink.canrule.last_t ← time
    end
    if rlink.canrule.ncc <
     rlink.canrule.max_ncc then
        rlink.canrule.cth ← 0;
    end
    if rlink.canrule.cth ≥
     rlink.canrule.threshold then
        act(rlink.canrule.action)
        return
    end
end
```

loop



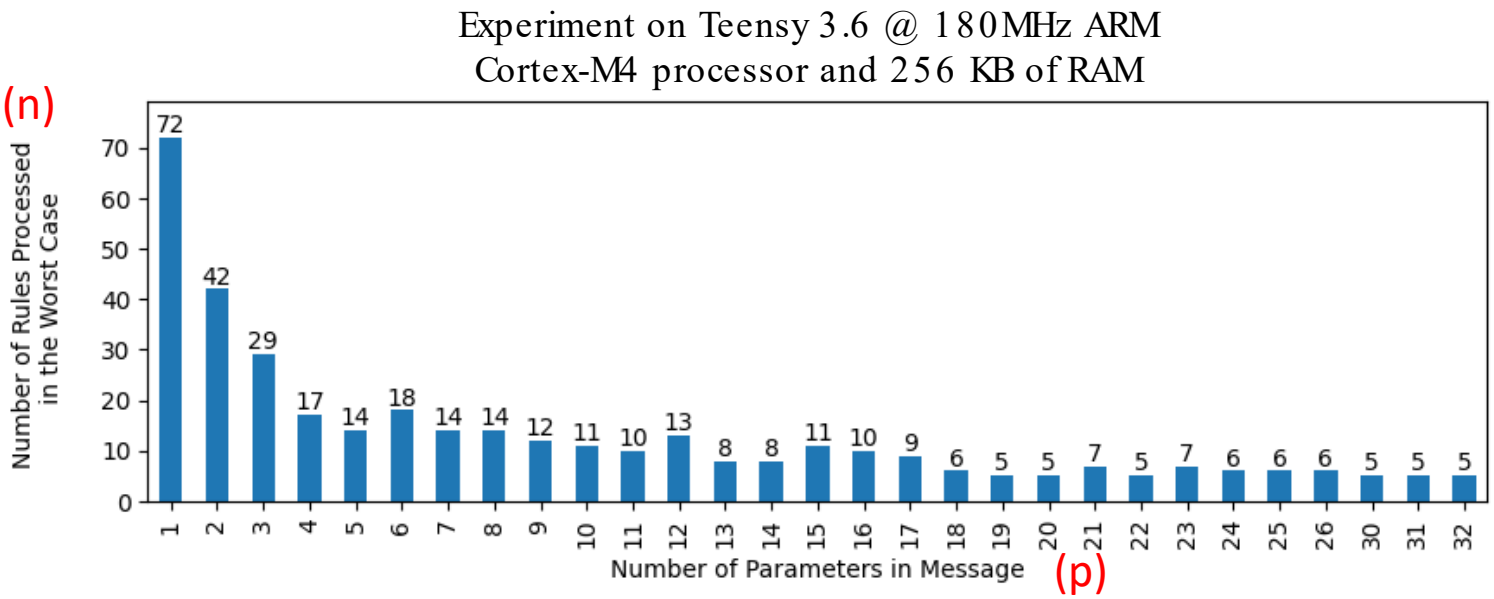
Block b

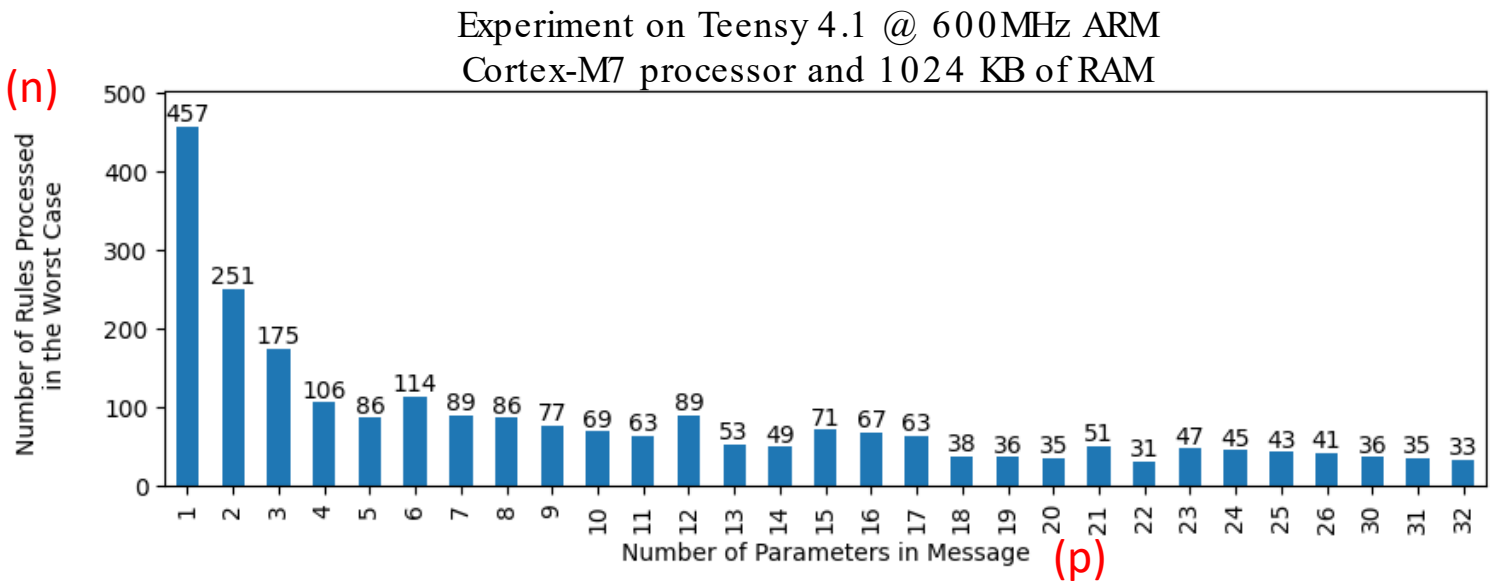
```
for i in rlink.indexes do
    m ← (rlink.canrule.context[i].value[0] ≤
    get_value(buffered data bytes,
    ff) ≤ rlink.canrule.context[i].value[1])
    if m ≠ rlink.canrule.context[i].prevm
     then
        if m = True then
            rlink.canrule.ncc ++;
        end
        else
            rlink.canrule.ncc –;
        end
        rlink.canrule.context[i].prevm ← m
    end
end
```

loop

Analysis Results

- Theoretical worst-case
  - 5 and 33
    - Maximum number of parameters in an SAE J1939 message can be 32
- Worst-case on research trucks
  - 9 and 63
    - Maximum number of parameters carried by a message on the Kenworth T270 is 17
  - 13 and 89
    - Maximum number of parameters carried by a message on the Kenworth T660 is 12
- Worst-case for rule database to detect known attacks
  - 72 and 457
    - Example rules reference one parameter per message





Organization

1. Introduction
2. Background
3. Denial-of-Service Attacks on the SAE J1939 Transport Layer
4. Behavioral Anomaly-based Detection
5. **Rule-based Detection and Prevention**
    1. Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks
    2. Rule Enforcement
    3. Performance Analysis
    4. **Real-World Demonstration and Discussion**
    5. Summary
6. Conclusion and Future work

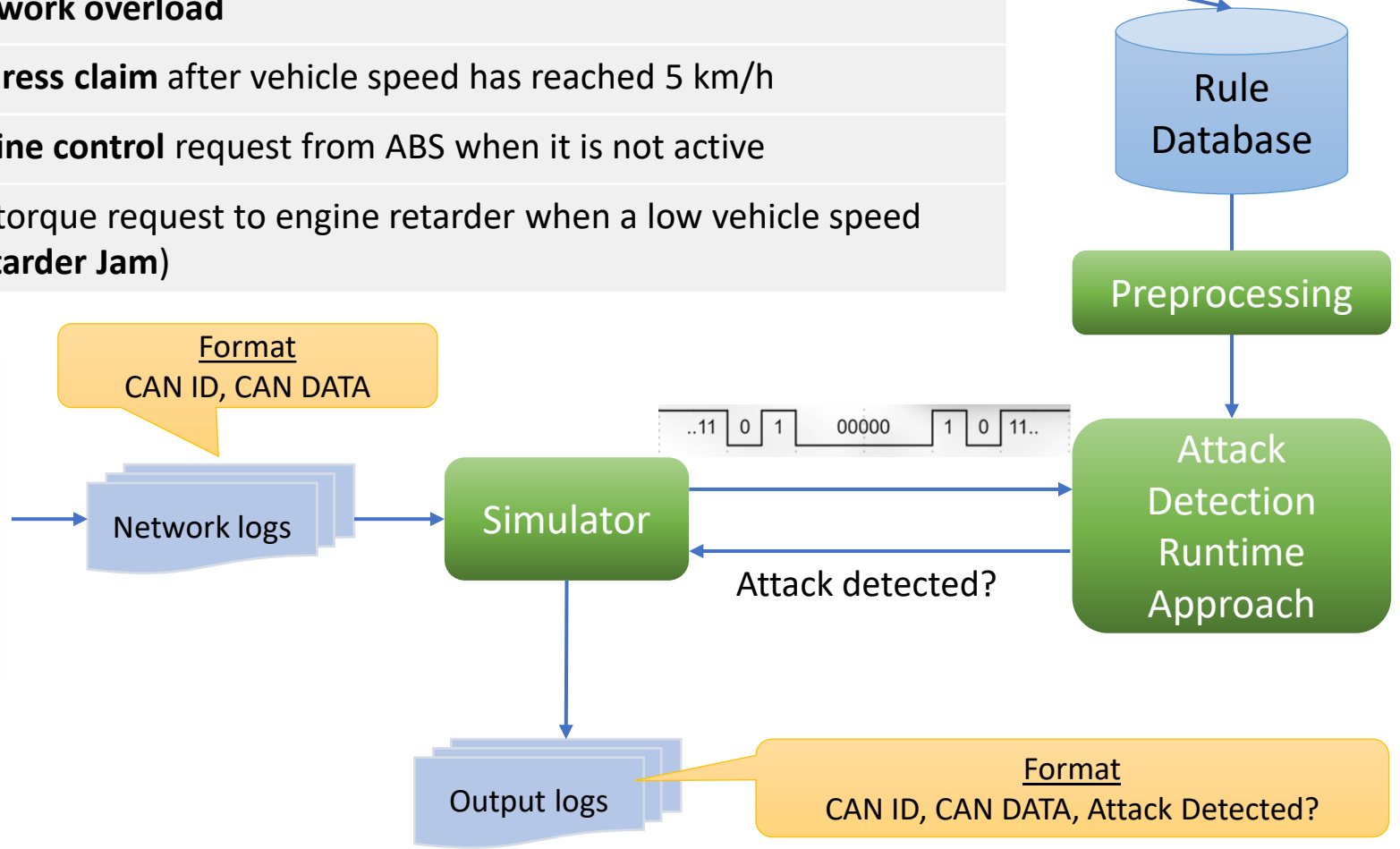
Experiment Method (Attack Detection)

Attacks
Engine control request from body controller
Very low torque request from engine controller (**Pedal Jam**)
Request overload on engine controller
Connection exhaustion from a diagnostic tool on engine controller
Network overload
Address claim after vehicle speed has reached 5 km/h
Engine control request from ABS when it is not active
0% torque request to engine retarder when a low vehicle speed (**Retarder Jam**)

Execute attacks on the research truck

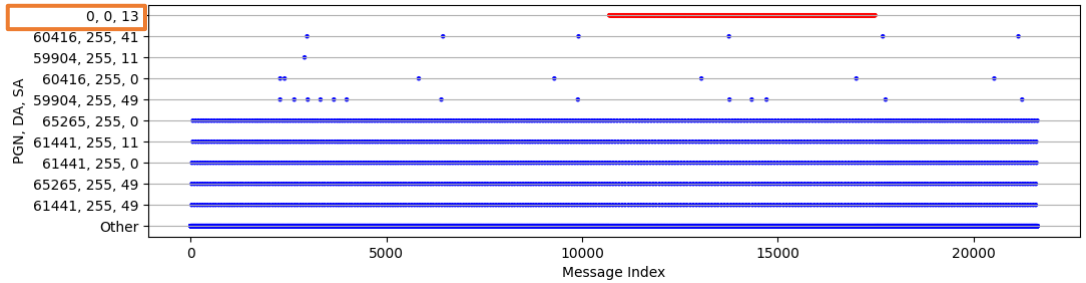
Kenworth T270 driven on an airstrip

Compose rules to detect the attacks

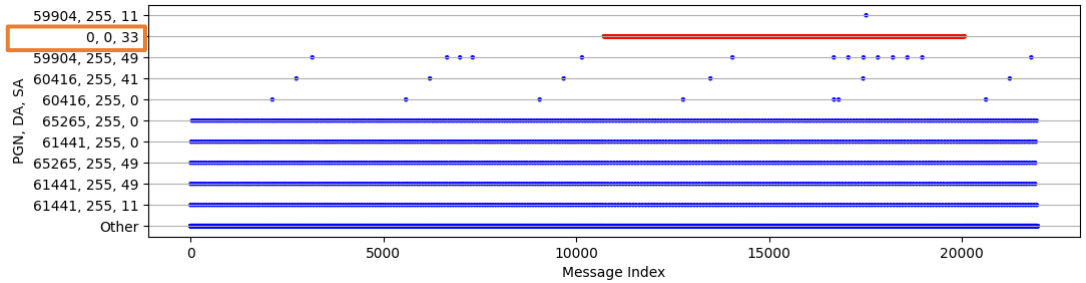


Experiment Results (Request Overload Attack Detection)

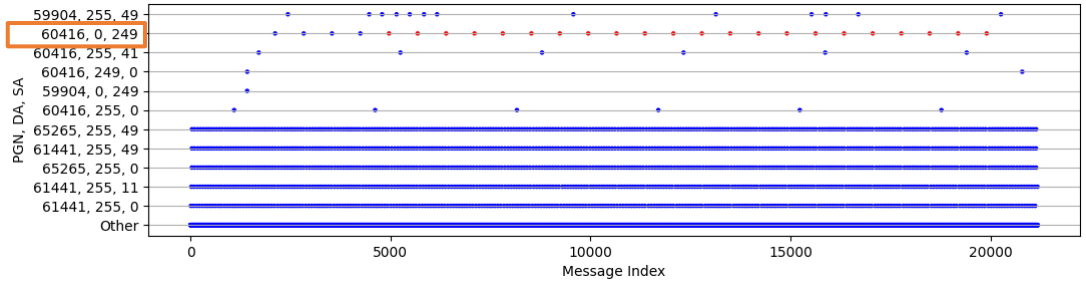
Experiment Results (Attack Detection)



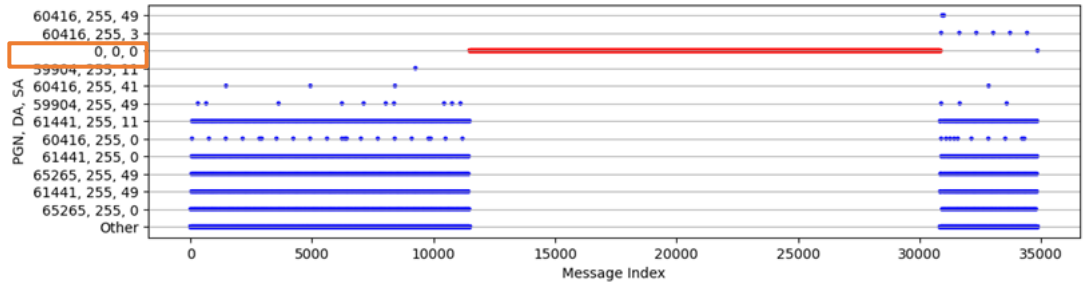
Engine control attack from body controller



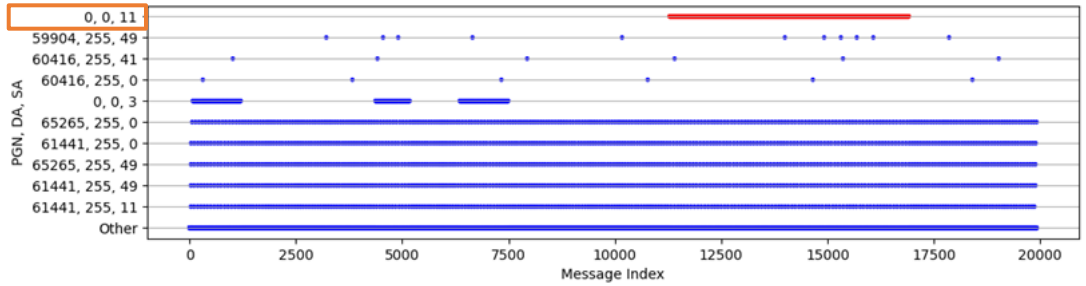
Very low torque request to engine controller (Pedal Jam Attack)



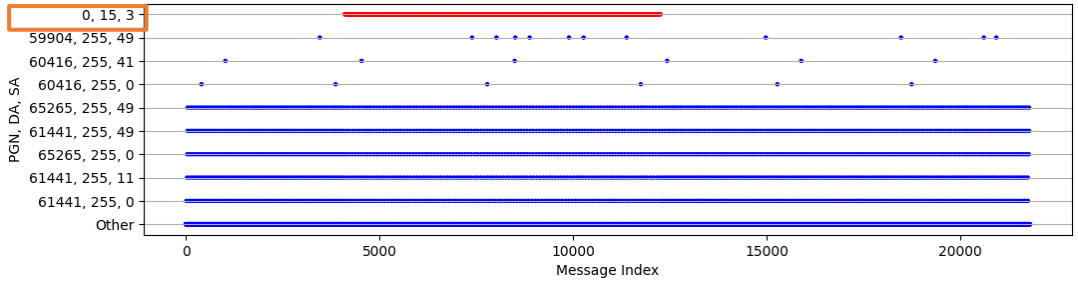
Connection exhaustion attack from a diagnostic tool on engine controller



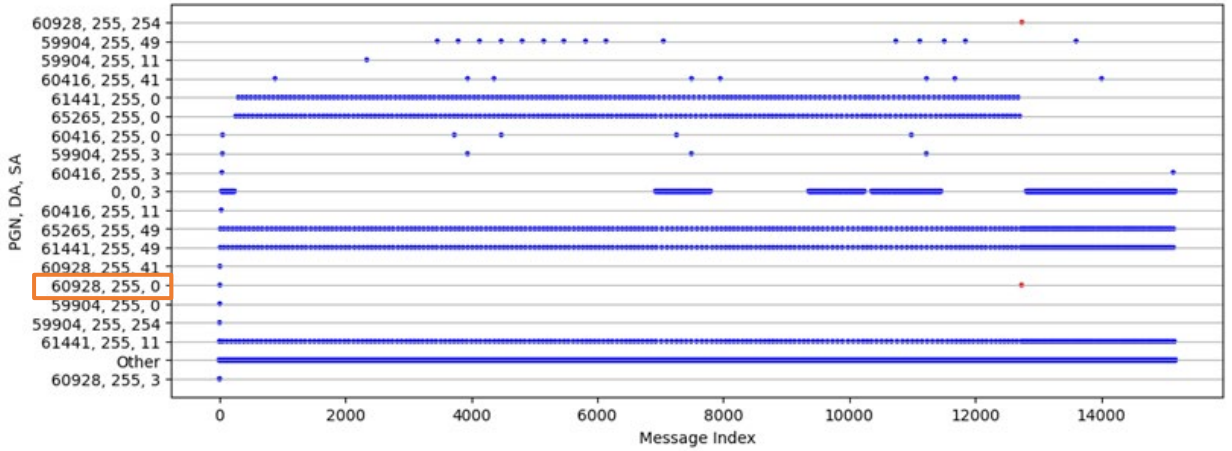
Network overload attack



Engine control attack from ABS controller



0 % torque request to the engine retarder when the vehicle speed <= 30km/



Address Claim attack on engine controller

Attack message

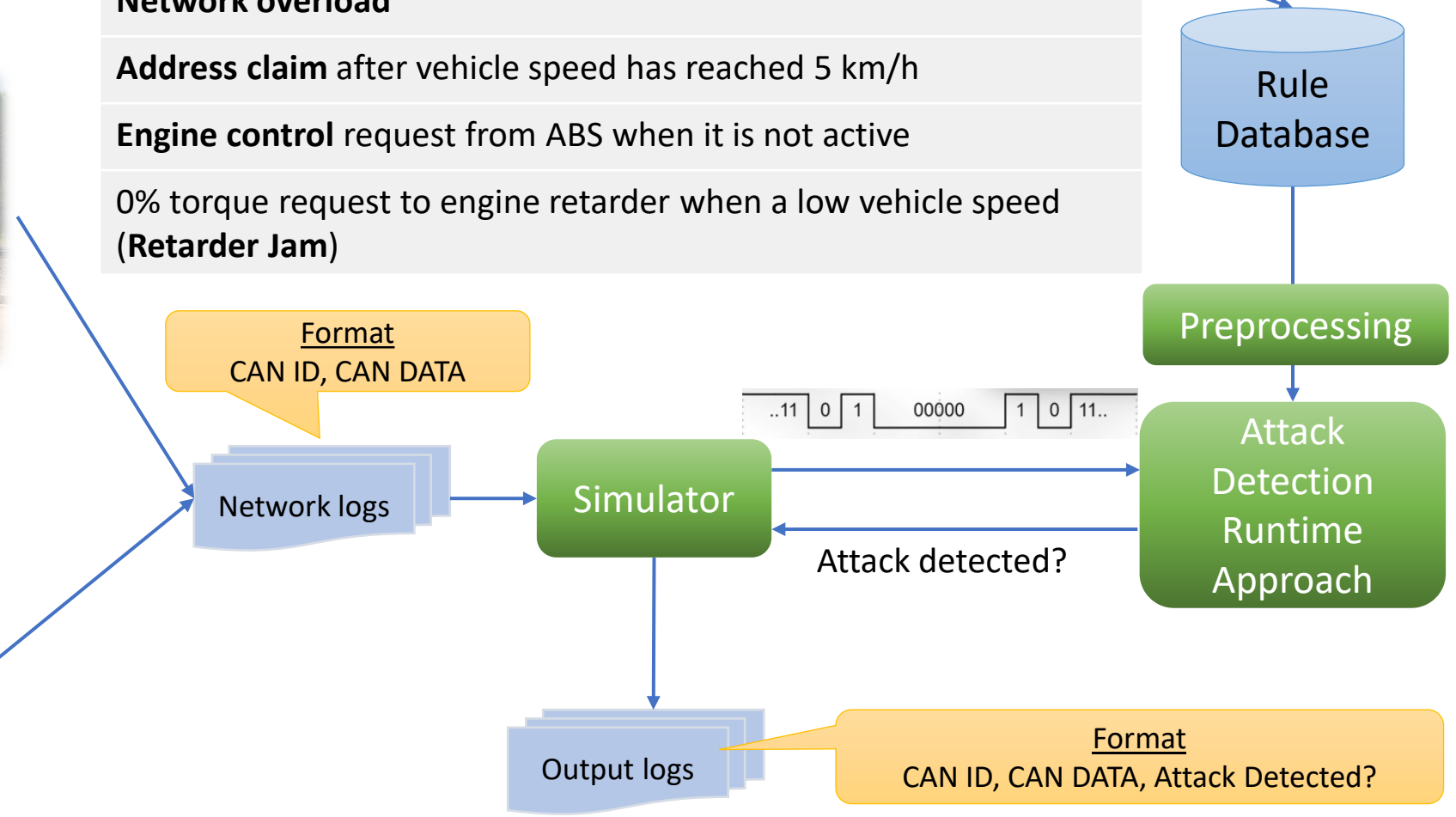
Experiment Method (False Alarm Generation)

Kenworth T270 driven on an airstrip

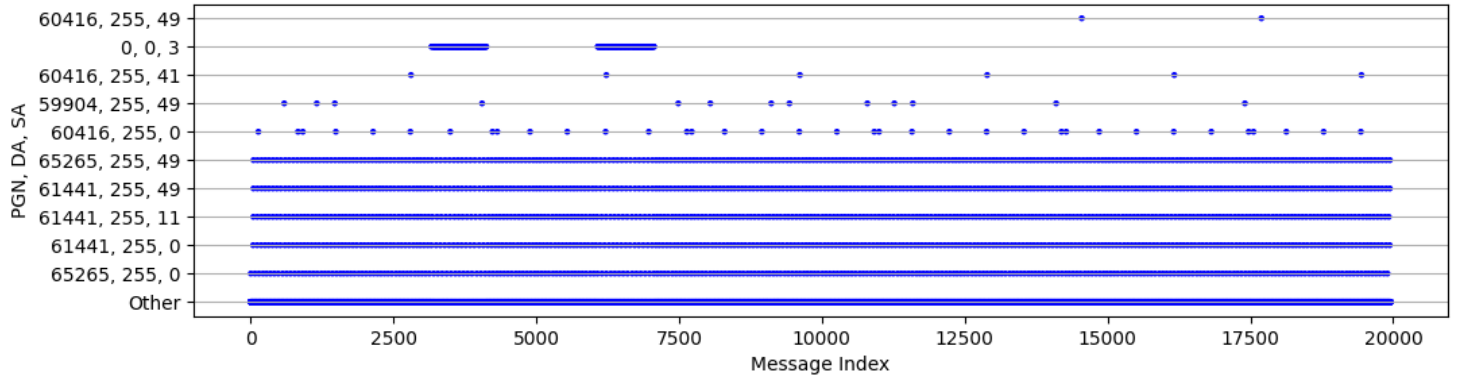
Kenworth T660 driven around an industrial block

Attacks
Engine control request from body controller
Very low torque request from engine controller (**Pedal Jam**)
Request overload on engine controller
Connection exhaustion from a diagnostic tool on engine controller
Network overload
Address claim after vehicle speed has reached 5 km/h
Engine control request from ABS when it is not active
0% torque request to engine retarder when a low vehicle speed (**Retarder Jam**)

Compose rules to detect the attacks



Experiment Results (False Alarm Generation)

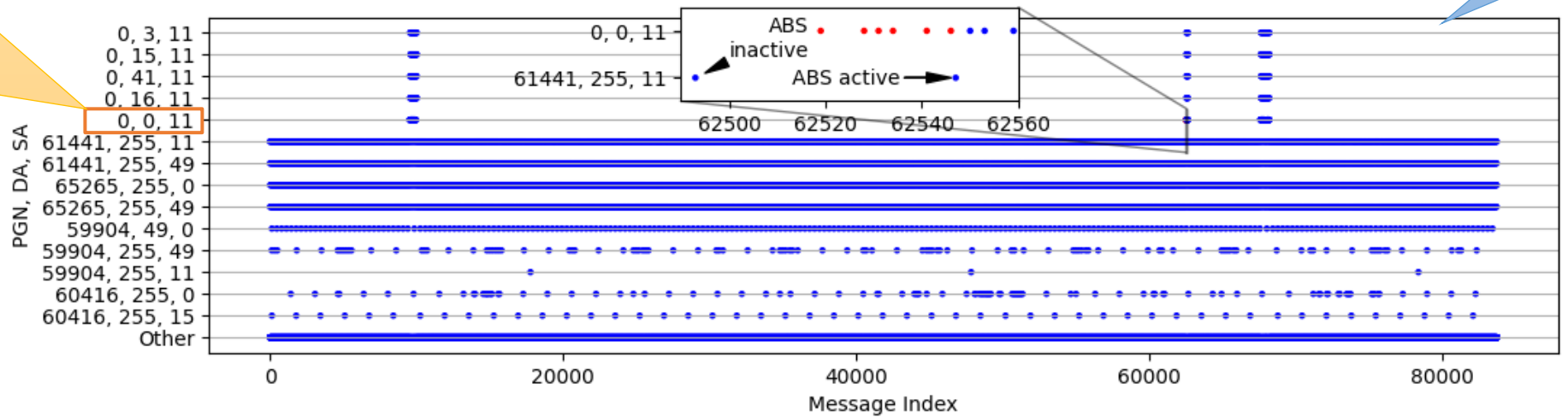


False Alarms on the Kenworth T270 Log

False alarms due to delayed context updates may be handled by

- Increasing the rule threshold

Flag all **engine control requests from ABS when it is not active**

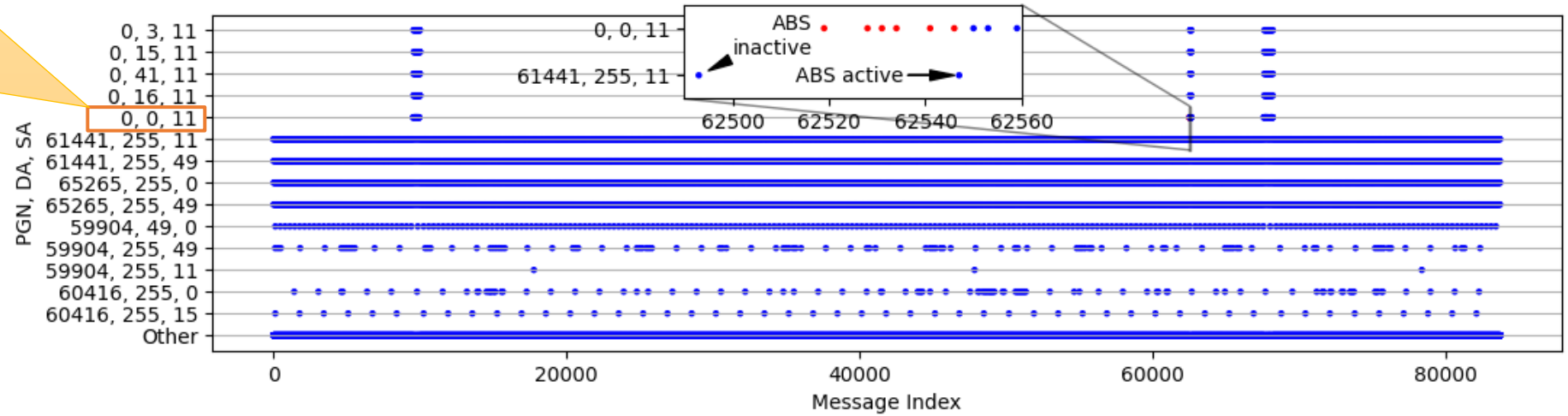


Flagged message

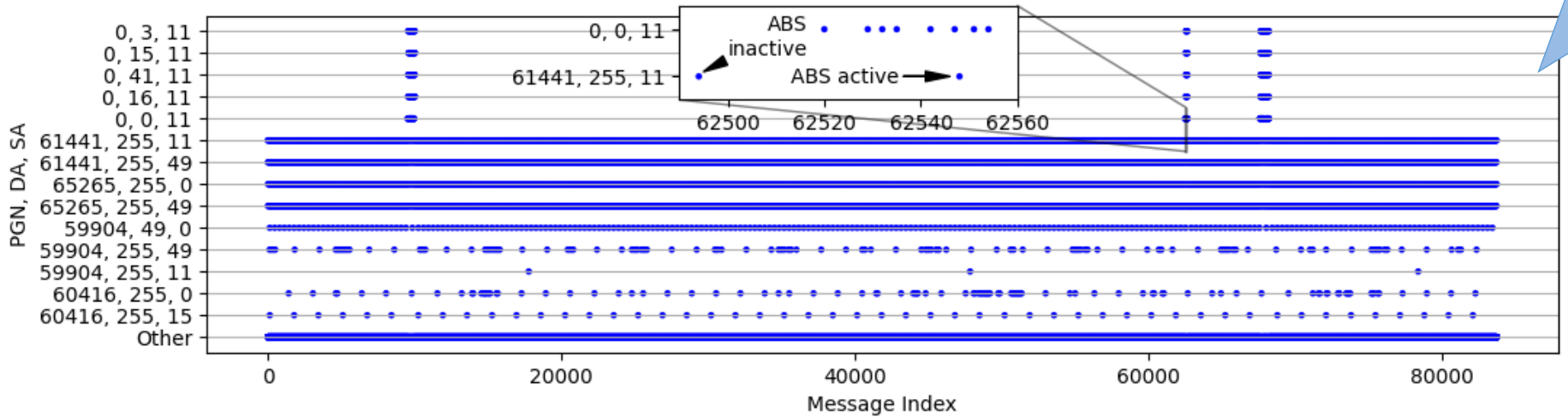
False Alarms on the Kenworth T660 Log

Experiment Results (False Alarm Elimination)

Flag all **engine control requests from ABS when it is not active**



Flagged message



False alarms due to delayed context updates handled by increasing the rule threshold to 10

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Real-time rule-based intrusion detection and prevention system

Towards Rule-based Identification of in-MHD Vehicle Cyber Attacks

Rule Enforcement

Performance Analysis

Real-World Demonstration and Discussion

Summary

Summary

Research Question

Can a rule-based system be designed to detect threatening SAE J1939 messages as they are being transmitted and mitigate their effect based on features other than message content only?

Contribution

A rule-based intrusion detection and prevention system that

- Allows identifying malicious messages based on features other than message content only
- Is real-time for a certain number of rules and can be used to disrupt malicious messages in transmission

Organization

Introduction

Background

Contribution 1

Contribution 2

Contribution 3

Conclusion and Future Work

Conclusion

Topic

- SAE J1939 specific cyber security for medium and heavy-duty vehicles

Research questions

- Can weaknesses in the data-link layer specifications of SAE J1939 be exploited to attack in-MHD vehicle ECUs?
- Can a system be designed to detect network anomalies on an SAE J1939 network in an online manner?
- Can a rule-based system be designed to detect threatening SAE J1939 messages as they are being transmitted and mitigate their effect based on features other than message content only?

Contributions

- Three denial-of-service attacks on the data-link layer specifications
- A online anomaly-based intrusion detection system
- A real-time rule-based intrusion detection and prevention systems that can identify messages that cannot be flagged based on message content only

Learning

- SAE J1939 specifications can be leveraged to design both offensive and defensive security solutions for medium and heavy-duty vehicles

Future Works

Extending to Other Areas of Interest

- Protocol specifications of other types of networks can be exploited
- Defensive solutions are high-level: may be deployable on other types of networks

Remote Testbenches and Generation of Research Data

- Outsourcing on research equipment's
- Reconfigurable testbench
- Network data dissemination for research and rule-generation

Hybrid Detection and Prevention Systems

- Detect unknown (0-day) attacks as well as known attacks
- Leverage context to detect and suppress false alarms

Accomplishments

Publications

- S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, “Practical DoS Attacks on Embedded Networks in Commercial Vehicles,” in *International Conference on Information Systems Security*, Jaipur, Rajasthan, India, 2016, pp. 23--42. doi: 10.1007/978-3-319-49806-5_2.
- S. Mukherjee, J. Walkery, I. Rayz, and J. Daily, “A Precedence Graph-Based Approach to Detect Message Injection Attacks in J1939 Based Networks,” in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, Aug. 2017, pp. 67–6709. doi: 10.1109/PST.2017.00018.
- S. Mukherjee, J. C. Van Etten, N. R. Samyukta, J. Walker, I. Ray, and I. Ray, “TruckSTM: Runtime Realization of Operational State Transitions for Medium and Heavy Duty Vehicles,” ACM Trans. Cyber-Phys. Syst., vol. 4, no. 1, pp. 1–25, Jan. 2020, doi: 10.1145/3300183.
- M. T. Campo, S. Mukherjee, and J. Daily, “Real-Time Network Defense of SAE J1939 Address Claim Attacks,” SAE Int. J. Commer. Veh., vol. 14, no. 3, Aug. 2021, doi: 10.4271/02-14-03-0026.
- Mukherjee, S. and Daily, J. (2021), Towards a Software Defined Truck. INCOSE International Symposium, 31: 1019-1034. https://doi.org/10.1002/j.2334-5837.2021.00884.x

Awards

- Recipient of the platinum award in CSU Ventures: Drivers of Innovation category at the CSU graduate showcase

Thank you for attending my presentation

Questions Please

State-of-the-art in In-Vehicle Security

	Drawbacks
Sender authentication-based	• Sender agnostic message processing • Compromise of legitimate sender
Behavioral Anomaly-Based IDPS	• Current solutions offline trained
Specification-Based IDPS	• Attacks can be specification abiding
Rule-Based IDPS	• Current solutions use rules based on message content only

** IDPS: Intrusion Detection and Prevention Systems

Review of In-Vehicle Security Solutions: Cryptography-based

General Method

Authenticate messages through digital signatures created using pre-shared keys

Pros

- No false positives

Published Works

- CAN specific
  - Generate digital-signature from CAN ID and Data [Groz17, Kura24]
- SAE J1939 specific
  - Generate digital-signature from SE J1939 message fields [Jich22, Murv18]

Cons

- Resource intensive
- Unable to detect attacks from legitimate but compromised senders
- Introduces communication overhead
- Key management can be challenging

Review of In-Vehicle Security Solutions: Anomaly-based

General Method

- Learn normal behavior from offline collected data
- Flag abnormal deviations from normal as attack

Pros

- Can detect unknown (0-day) attacks if it causes significant deviations from normal behavior

Published Work

- Voltage-based
  - Flag abnormal voltage usage [Cho17, Choi18]
- Periodicity-based
  - Flag aperiodic transmissions [Tayl15, Moor17,Song16, Mill14,Cho16]
- Parameter-based
  - Predict parameter values using contextual information and compare actual value with predicted [Nara16]
  - SAE J1939 specific [Shir20, Shir22]

Cons

- Does not account for normal behavior that is not encountered during the training phase [Stac19]

Review of In-Vehicle Security Solutions: Specification-based

General Method

- Build reference model for normal behavior using manufacturer specifications
- Flag deviations from normal as attack

Pros

- Can detect unknown (0-day) attacks if they violate specifications

Published Work

- Logical Expression-based
  - Convert manufacturer specifications to logical expressions and evaluate them on a host ECU [Stud15]
- Finite automation-based
  - Convert manufacturer specifications to finite automaton and flag abnormal transitions [Lars08]
- No SAE J1939 specific-method

Cons

- Unable to detect attacks that obey specifications

Review of In-Vehicle Security Solutions: Rule-based

General Method

- Create a database of attack patterns based on CAN ID and data
- Flag frames as malicious if matching patterns are found in the database

Pros

- Low false positives

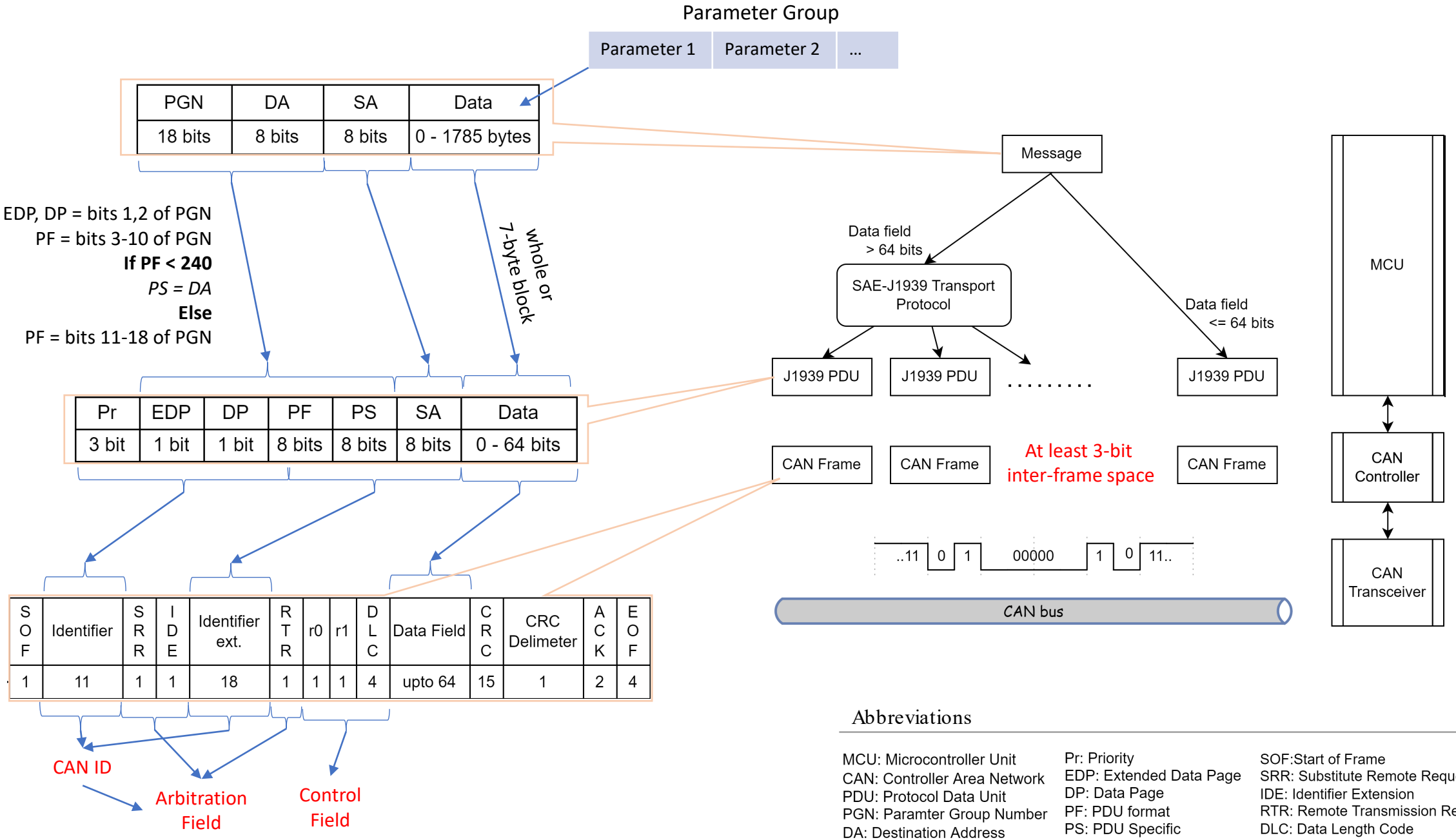
Published Work

- CAN ID-based
  - Compare CAN ID with pre-defined whitelist or blacklist [Boud16, Daga17, Ansa17, Mats12, Abbo21]
- CAN data-based
  - Inspect CAN data for specific byte patterns [Ujji16, Lena21]
- No SAE J1939 specific-method

Cons

- Not all malicious CAN frames can be identified based on their ID and data

SAE J1939 Message Processing



Abbreviations

MCU: Microcontroller Unit
CAN: Controller Area Network
PDU: Protocol Data Unit
PGN: Paramter Group Number
DA: Destination Address
SA: Source Address

Pr: Priority
EDP: Extended Data Page
DP: Data Page
PF: PDU format
PS: PDU Specific
SA: Source Address

SOF:Start of Frame
SRR: Substitute Remote Request
IDE: Identifier Extension
RTR: Remote Transmission Request
DLC: Data Length Code
CRC: Cyclic Redundancy Check
ACK: Acknowledgement
EOF: End of Frame

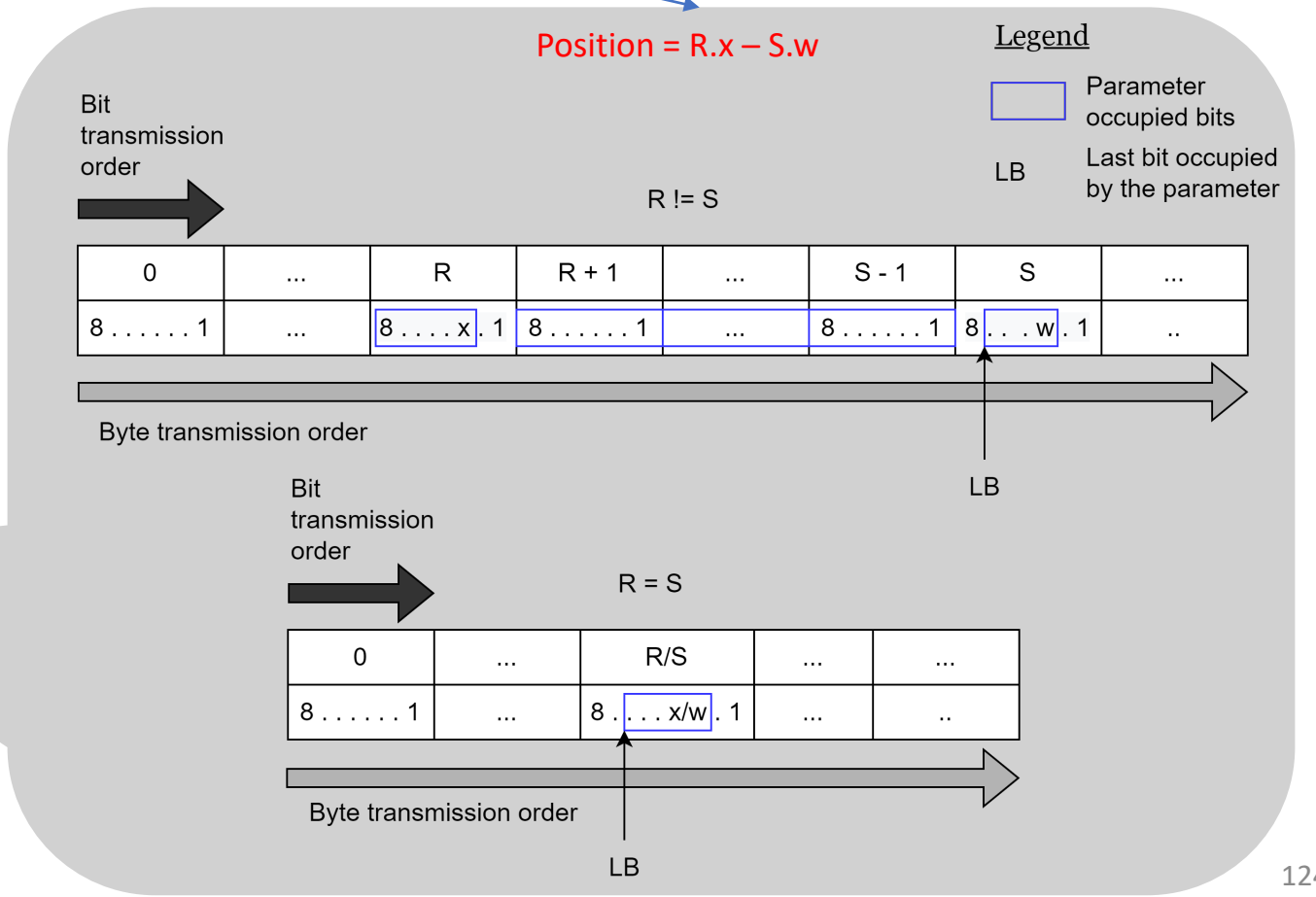
Parameter Placement

Suspect Parameter Number (SPN) uniquely identifies a parameter

SAE J1939 Digital Annex

SPN	Resolution	Offset	Position	Length

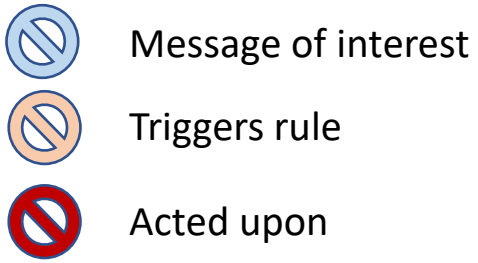
Parameter value → value = (value – offset)/ resolution → Message Data



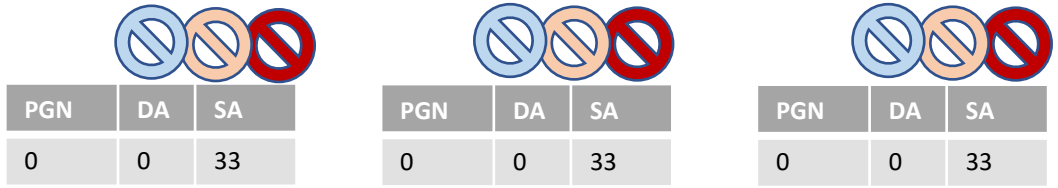
Advances in Testing on Local Testbench

Unanswered questions from previous experiment	New experiment methods
Was the drop in count was because of a request overload or messages losing arbitration to higher priority request messages?	• Rapidly send three different IDs and observe the effect: 00000000_{16}, $1C000000_{16}$, $1CEA0000_{16}$ • Drop in ECM traffic due to the last message but not on the second last, implies request overload is successful
Did the rate of injection of the request messages had any relation with the effect of the attack?	• Vary the rate of injection between 0.1 to 1 milliseconds and observe the effect
Was the sensor simulator was dropping any messages while forwarding traffic to and from the engine controller?	• Local testbench does not include the sensor simulator forwarding device
Does requesting a parameter group that is not present with the engine controller have any effect on the output traffic?	• Send requests for valid and invalid parameter groups and observe the effect

Example Rules and Their Enforcement

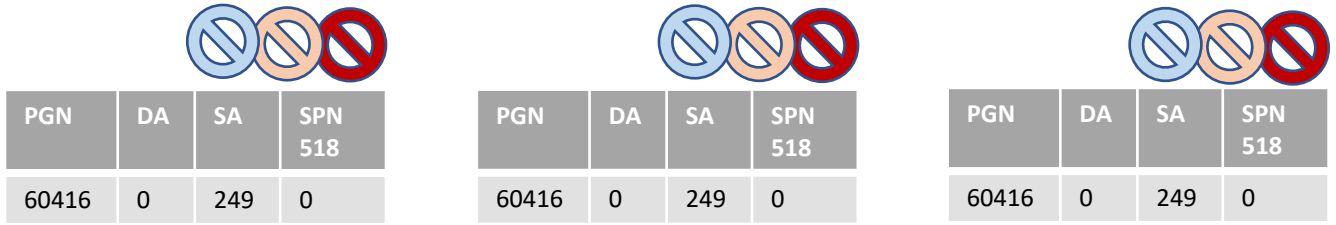


R1



PGN	DA	SA
0	0	33

R2



PGN	DA	SA	SPN 518
60416	0	249	0

	Type	Description	Threshold	Interval	MOI/NetPFilter					
					Relation	PGN	DA	SA	PFilter	
									SPN	Value
R1	Rule	Engine control request from body controller	1	N/A	moi	0	0	33		
R2	Rule	Very low torque request	1	N/A	moi	0	0		518	[-125,-125]

Example Rules and Their Enforcement

	Type	Description	Threshold	Interval	MOI/NetPFilter					
					Relation	PGN	DA	SA	PFilter	
									SPN	Value
R1	IRule	Request overload on engine controller	1	5	moi	59904	0			
R2	IRule	Connection exhaustion from a diagnostic tool on engine controller	5	1250	moi	60416	0	249	2556	[17,17]
R3	IRule	Network overload	1	5	moi	0	0	0		

Example Rule Enforcement

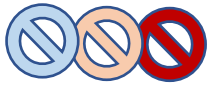
R1

PGN	DA	SA
60928	*	*

PGN	DA	SA	SPN 84
65265	255	0	2560

PGN	DA	SA
60928	*	*

PGN	DA	SA	SPN 84
65265	255	0	2560

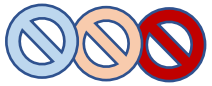


PGN	DA	SA
60928	*	*

R2

PGN	DA	SA
0	0	11

PGN	DA	SA	SPN 563
61441	255	11	0



PGN	DA	SA
0	0	11

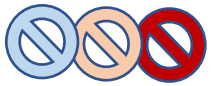
PGN	DA	SA	SPN 563
61441	255	11	1

PGN	DA	SA
0	0	11

R3

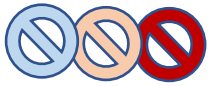
PGN	DA	SA	SPN 84
65265	0	249	125

PGN	DA	SA	SPN 84
65265	0	249	1280



PGN	DA	SA	SPN 84
65265	0	249	125

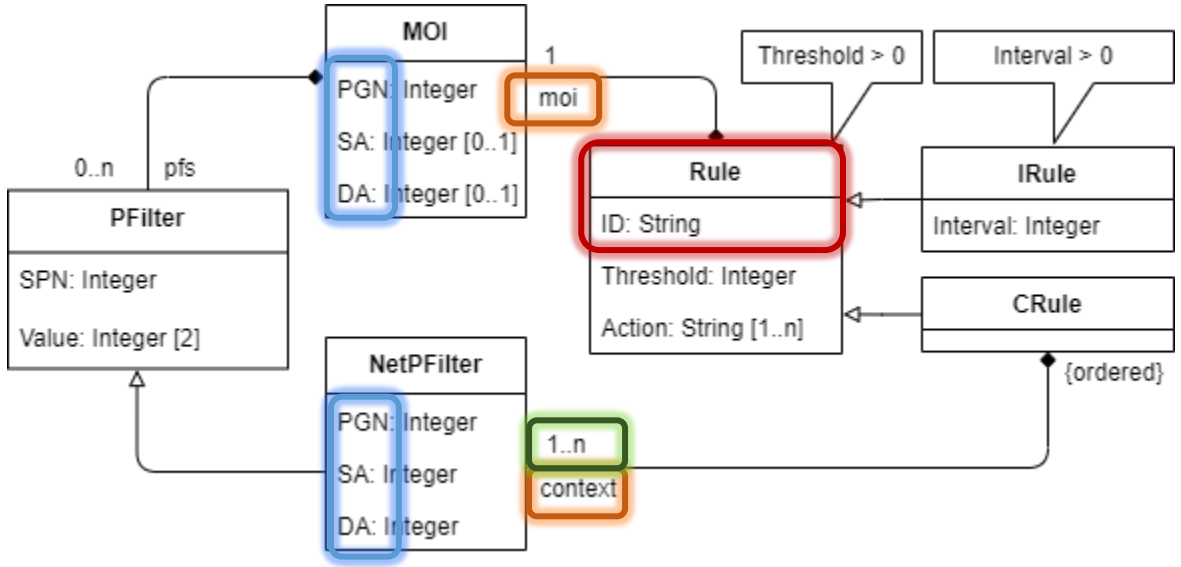
PGN	DA	SA	SPN 84
65265	0	249	125

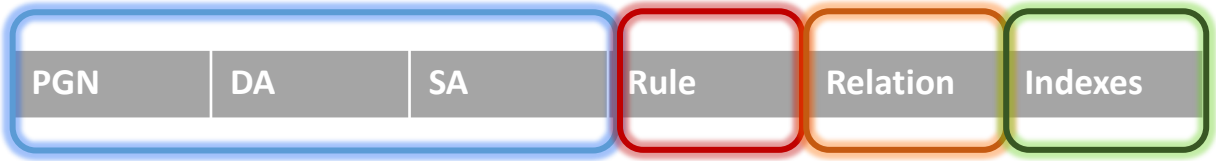


PGN	DA	SA	SPN 518
65265	0	249	125

	Type	Description	Threshold	Interval	MOI/NetPFilter					
					Relation	PGN	DA	SA	PFilter SPN	PFilter Value
R1	CRule	Address claim after vehicle speed has reached 5 km/h	1	N/A	moi	60928				
					context	65265	255	0	84	[5,300]
R2	CRule	Engine control request from ABS when it is not active	1	N/A	moi	0	0	11		
					context	61441	255	11	563	[0,0]
R3	CRule	0% torque request to engine retarder when a low vehicle speed	1	N/A	moi	0	15		518	[0,0]
					context	65265	255	0	84	[0,30]

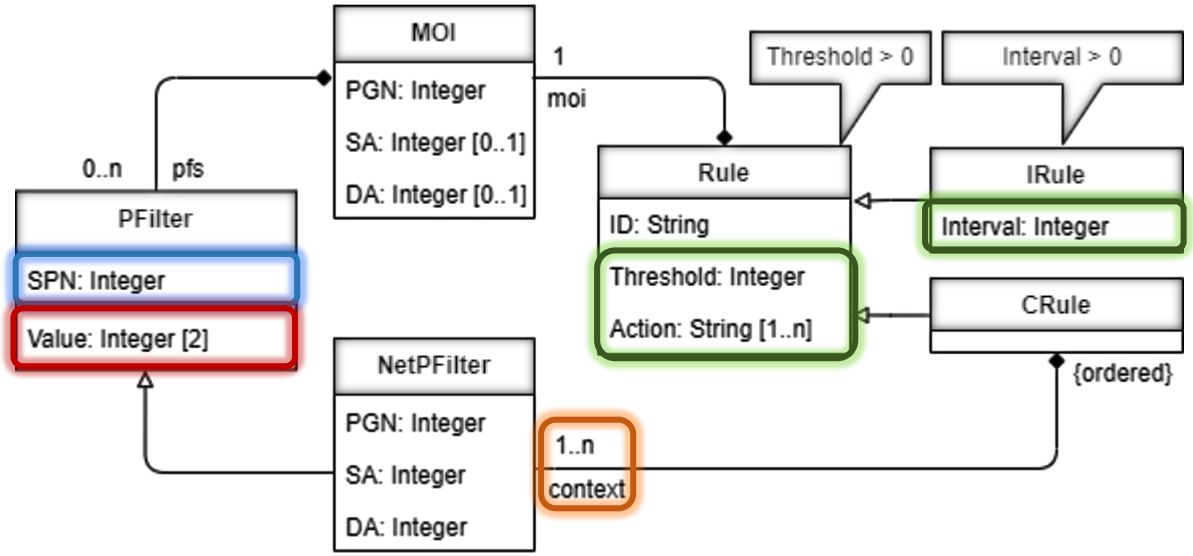
Temporary Lookup Table Generation

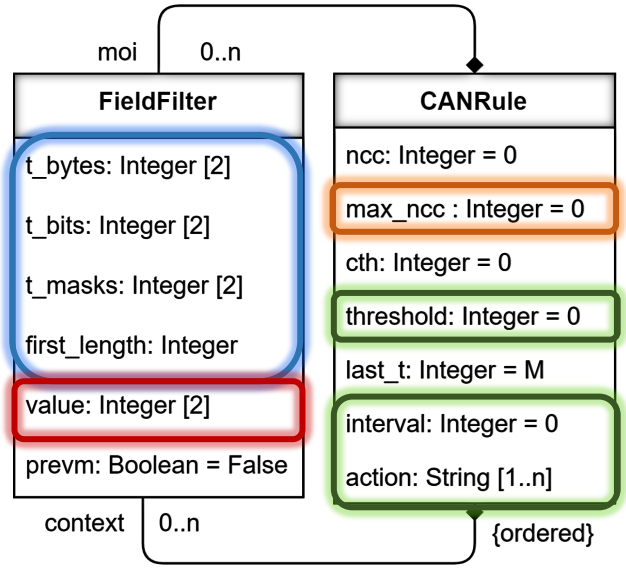




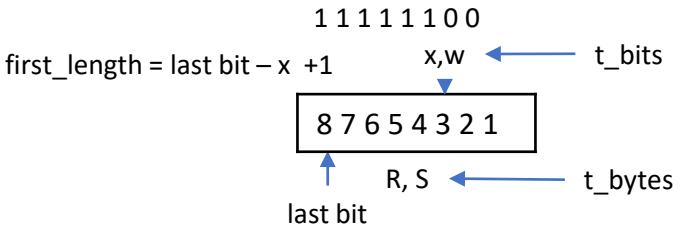
- Temporary Lookup Table:
  PGN × DA × SA → Rule × {moi, context} × Indexes
- ∀x ∈Indexes[c], c is a record in the temporary lookup table
  - Empty if Relation[c] = *moi*
  - Otherwise,
    - x ∈ z^+ and
    - Rule[c].context[x].PGN = PGN[c],
    - Rule[c].context[x].DA = DA[c],
    - Rule[c].context[x].SA = SA[c]

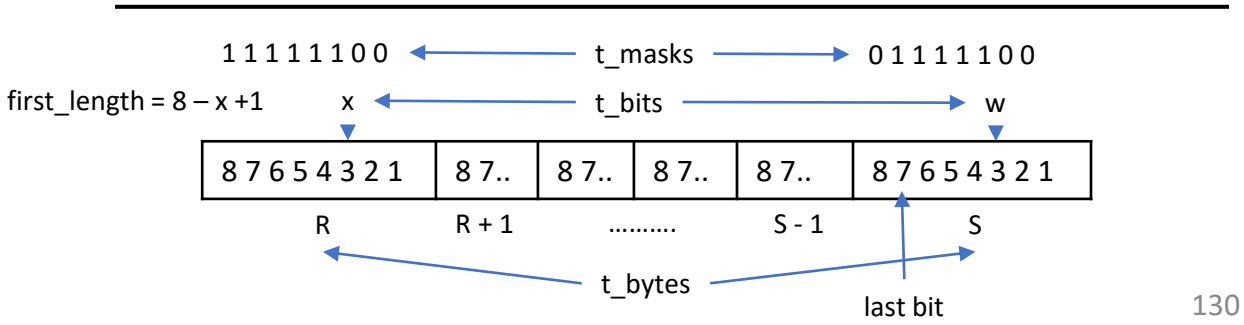
CAN Rule Generation





- `threshold`, `interval` and `action` are copied from the parent rule
- `max_ncc` = cardinality of the context relation
- `value` = [(Value[*i*]-*offset*)/*resolution* for *i* in 0..1]
- `t_bytes`, `t_bits`, `t_masks`, `first_length` derived from the parameter placement notation *R.x – S.w*
- *R.x – S.w*, *offset* and *resolution* obtained by querying the SAE J1939 digital annex using the SPN





Arbitration Field String Generation

Entries in the SAE J1939 Digital Annex

SPN	Position	Length	Resolution	Offset
2540	1.1 - 3.1	3 bytes	1	0
597	4.5 - 4.5	2 bits	1	0
898	2.1 - 3.1	2 bytes	0.125	0

Rules

Type	ID	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	R1	2	9	moi	00000_{16}	00_{16}			
Rule	R2	1	N/A	moi	00000_{16}	00_{16}	31_{16}		
Rule	R3	1	N/A	moi	00000_{16}	00_{16}	$0F_{16}$	898	[50,100]
CRule	R4	5	N/A	moi	00000_{16}	00_{16}	$0F_{16}$		
				context	$0FEE1_{16}$	FF_{16}	00_{16}	597	[1,1]
IRule	R5	1	5	moi	$0EA00_{16}$	$0F_{16}$	$0B_{16}$	2540	[65259, 65259]

L2Rules

ID	ncc, max_ncc	cth, threshold	last_t, interval	FieldFilter						
				Relation	t_bytes	t_bits	t_masks	first_length	value	prevm
R1	0,0	0,2	M,9							
R2	0,0	0,1	M,0							
R3	0,0	0,1	M,0	moi	2,3	1,1	ff_{16}, ff_{16}	8	[400, 800]	False
R4	0,1	0,5	M, 0	moi						
				context	4,4	5,5	30_{16}, 30_{16}	2	[1,1]	False
R5	0,0	0,1	M, 5	moi	1,3	1,1	ff_{16}, ff_{16}	8	[65259, 65259]	False

** M = Maximum integer value supported by the deployment device architecture

Temporary Lookup Table

PGN	DA	SA	Rule	Relation	Indexes
00000_{16}	00_{16}		R1	moi	[]
00000_{16}	00_{16}	31_{16}	R2	moi	[]
00000_{16}	00_{16}	$0F_{16}$	R3	moi	[]
			R4	moi	[]
$0FEE1_{16}$	FF_{16}	00_{16}	R4	context	[0]
$0EA00_{16}$	$0F_{16}$	$0B_{16}$	R5	moi	[]

Key Value

Arbitration Field Strings

$00_2 00_2 0_2 C00_{16}$
$00_2 00_2 0_2 C00_{16} 00_2 11_2 1_{16}$
$00_2 00_2 0_2 C00_{16} 00_2 00_2 F_{16}$
$00_2 111_2 010_2 E0F0B_{16}$
$00_2 111_2 111_2 EE100_{16}$

PGN = 00000_{16} DA = 00_{16}

Priority	EDP	DP	PF	PS	SA
***	0	0	00000000	00000000	********

SAE J1939 PDU

* = Don't care

Identifier	SRR	IDE	Identifier ext.	RTR
***00000000	1	1	0000000000********	*
$00_2 00_2 0_{16}$	$C00_{16}$			

CAN Arbitration Field

Radix Tree Generation

Entries in the SAE J1939 Digital Annex

SPN	Position	Length	Resolution	Offset
2540	1.1 - 3.1	3 bytes	1	0
597	4.5 - 4.5	2 bits	1	0
898	2.1 - 3.1	2 bytes	0.125	0

Rules

Type	ID	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	R1	2	9	moi	00000_{16}	00_{16}			
Rule	R2	1	N/A	moi	00000_{16}	00_{16}	31_{16}		
Rule	R3	1	N/A	moi	00000_{16}	00_{16}	$0F_{16}$	898	[50,100]
CRule	R4	5	N/A	moi	00000_{16}	00_{16}	$0F_{16}$		
				context	$0FEE1_{16}$	FF_{16}	00_{16}	597	[1,1]
IRule	R5	1	5	moi	$0EA00_{16}$	$0F_{16}$	$0B_{16}$	2540	[65259, 65259]

L2Rules

ID	ncc, max_ncc	cth, threshold	last_t, interval	FieldFilter						
				Relation	t_bytes	t_bits	t_masks	first_length	value	prevm
R1	0,0	0,2	M,9							
R2	0,0	0,1	M,0							
R3	0,0	0,1	M,0	moi	2,3	1,1	ff_{16}, ff_{16}	8	[400, 800]	False
R4	0,1	0,5	M, 0	moi						
				context	4,4	5,5	30_{16}, 30_{16}	2	[1,1]	False
R5	0,0	0,1	M, 5	moi	1,3	1,1	ff_{16}, ff_{16}	8	[65259, 65259]	False

** M = Maximum integer value supported by the deployment device architecture

Temporary Lookup Table

PGN	DA	SA	Rule	Relation	Indexes
00000_{16}	00_{16}		R1	moi	[]
00000_{16}	00_{16}	31_{16}	R2	moi	[]
00000_{16}	00_{16}	$0F_{16}$	R3	moi	[]
			R4	moi	[]
$0FEE1_{16}$	FF_{16}	00_{16}	R4	context	[0]
$0EA00_{16}$	$0F_{16}$	$0B_{16}$	R5	moi	[]

Key Value

Arbitration Field Strings

$00_200_20C00_{16}$
$00_200_20C00_{16}00_211_21_{16}$
$00_200_20C00_{16}00_200_2F_{16}$
$00_2111_2010_2E0F0B_{16}$
$00_2111_2111_2EE100_{16}$

RadixTreeNode

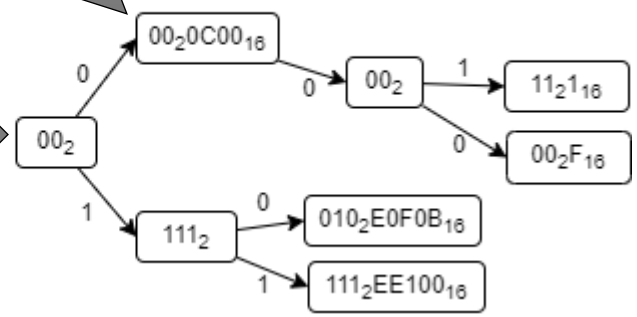
length: Integer

value: Integer

0child: RadixTreeNode [0..1]

1child: RadixTreeNode [0..1]

Radix Tree



Target Generation

Entries in the SAE J1939 Digital Annex

SPN	Position	Length	Resolution	Offset
2540	1.1 - 3.1	3 bytes	1	0
597	4.5 - 4.5	2 bits	1	0
898	2.1 - 3.1	2 bytes	0.125	0

Rules

Type	ID	Threshold	Interval	MOI/NetPFilter					
				Relation	PGN	DA	SA	PFilter	
								SPN	Value
IRule	R1	2	9	moi	00000_{16}	00_{16}			
Rule	R2	1	N/A	moi	00000_{16}	00_{16}	31_{16}		
Rule	R3	1	N/A	moi	00000_{16}	00_{16}	$0F_{16}$	898	[50,100]
CRule	R4	5	N/A	moi	00000_{16}	00_{16}	$0F_{16}$		
				context	$0FEE1_{16}$	FF_{16}	00_{16}	597	[1,1]
IRule	R5	1	5	moi	$0EA00_{16}$	$0F_{16}$	$0B_{16}$	2540	[65259, 65259]

L2Rules

ID	ncc, max_ncc	cth, threshold	last_t, interval	FieldFilter						
				Relation	t_bytes	t_bits	t_masks	first_length	value	prevm
R1	0,0	0,2	M,9							
R2	0,0	0,1	M,0							
R3	0,0	0,1	M,0	moi	2,3	1,1	ff_{16}, ff_{16}	8	[400, 800]	False
R4	0,1	0,5	M, 0	moi						
				context	4,4	5,5	30_{16}, 30_{16}	2	[1,1]	False
R5	0,0	0,1	M, 5	moi	1,3	1,1	ff_{16}, ff_{16}	8	[65259, 65259]	False

** M = Maximum integer value supported by the deployment device architecture

Temporary Lookup Table

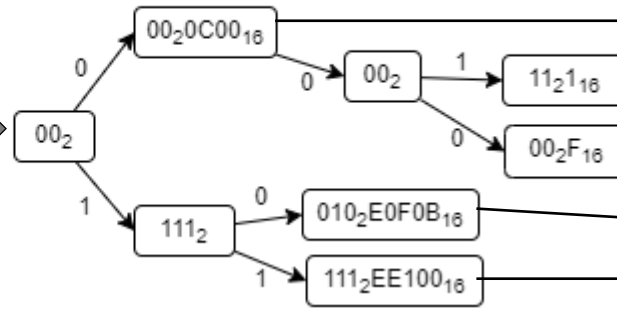
PGN	DA	SA	Rule	Relation	Indexes
00000_{16}	00_{16}		R1	moi	[]
00000_{16}	00_{16}	31_{16}	R2	moi	[]
00000_{16}	00_{16}	$0F_{16}$	R3	moi	[]
			R4	moi	[]
$0FEE1_{16}$	FF_{16}	00_{16}	R4	context	[0]
$0EA00_{16}$	$0F_{16}$	$0B_{16}$	R5	moi	[]

Key Value

Arbitration Field Strings

$00_2 00_2 0_2 C00_{16}$
$00_2 00_2 0_2 C00_{16} 00_2 11_2 1_{16}$
$00_2 00_2 0_2 C00_{16} 00_2 00_2 F_{16}$
$00_2 111_2 010_2 E0F0B_{16}$
$00_2 111_2 111_2 EE100_{16}$

Radix Tree



Last data byte after which rules can be processed

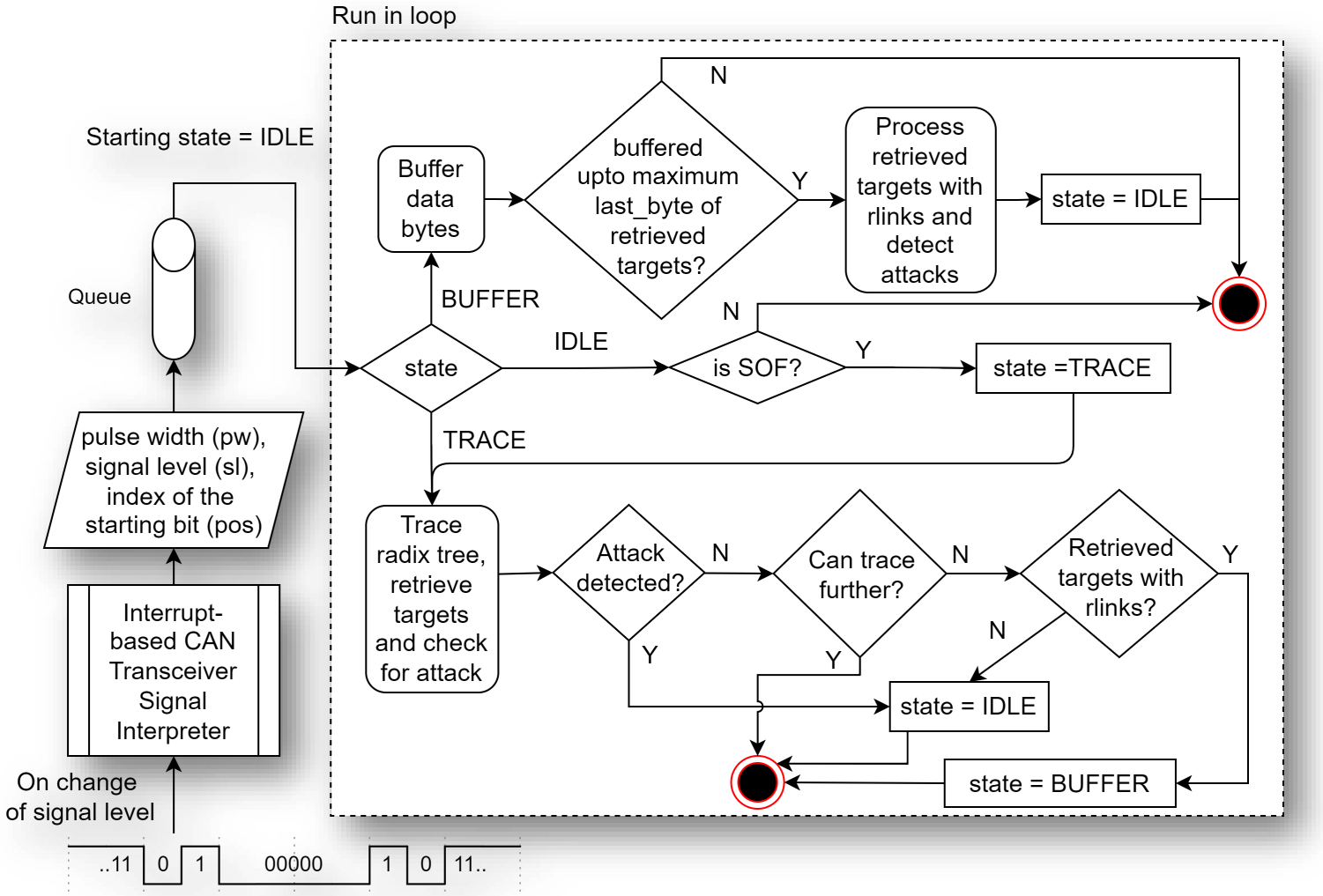
Rules without PFilters

Rules with PFilters

Targets

last_byte	np_l2rules	rlinks		
		l2rule	relation	indexes
0	[R1]			
0	[R2]			
3		R3	moi	[]
		R4	moi	[]
3		R5	moi	[]
4		R4	context	[0]

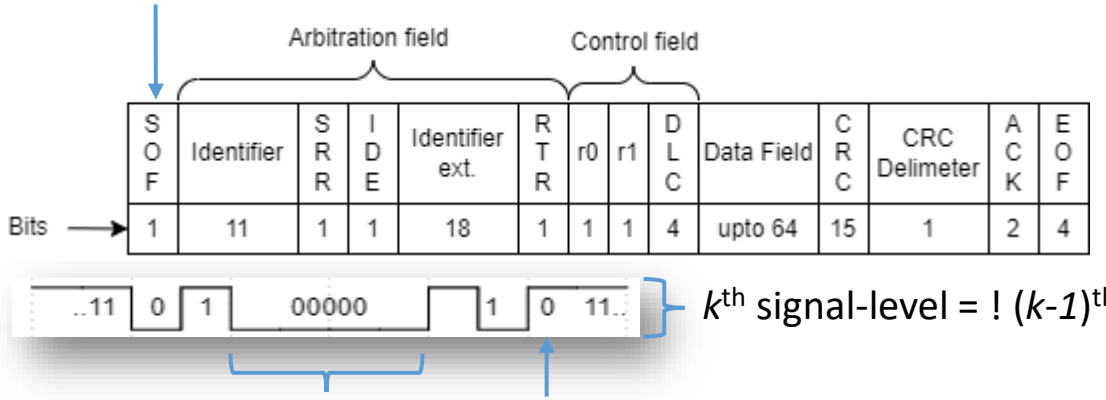
Runtime Approach Walkthrough



Interrupt-based CAN Controller Signal Interpreter [Camp21]

- Interrupts are set to be fired on change of signal-level
- Output
  - pulse width: number of bits in pulse
  - signal-level: 0 or 1
  - starting bit index in the CAN frame: 1 (if SOF) or greater
    - Stuff bits are ignored
    - Pulse with single stuff bit is not returned

1st pulse:
bit index= 1,
signal-level = 0



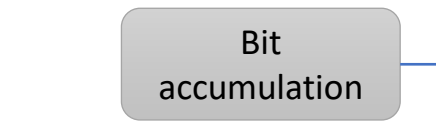
k^{th} signal-level = ! $(k\text{-}1)^{th}$ signal-level

pulse width =
difference in microseconds
between two consecutive interrupts/
bit-width in microseconds

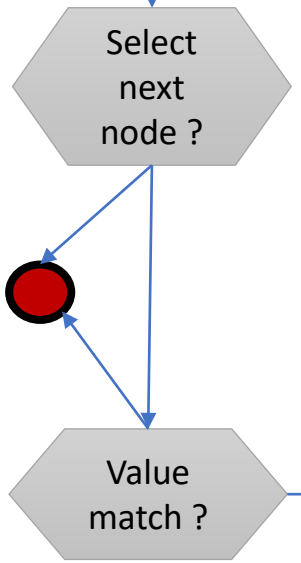
bit index of k^{th} ($k > 1$) pulse =
bit index of $(k - 1)^{th}$ pulse +
pulse width of $(k\text{-}1)^{th}$ pulse

Tree Tracing

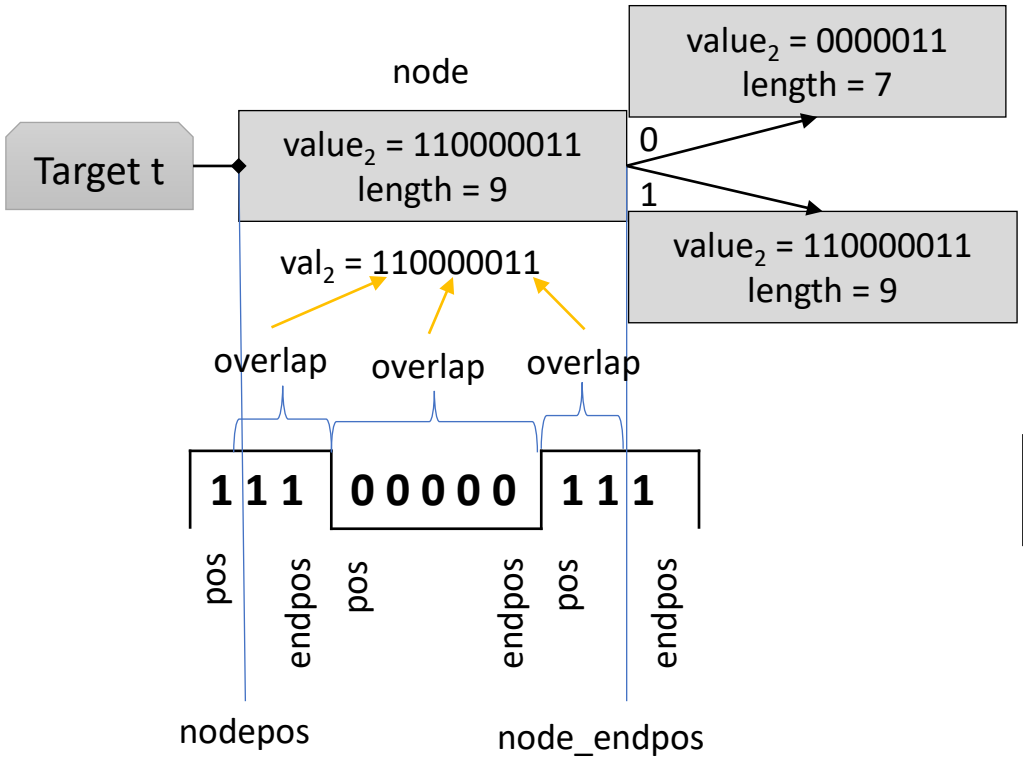
```
1  endpos ← pos + pw -1
2  node_endpos ← nodepos
     + node.length -1
3  overlap ← min(endpos,
     node_endpos) -
     max(pos,nodepos) + 1
4  if overlap > 0 then
5  |   val ← val << overlap
   |     | (2^overlap − 1) ∗ sl
6  end
```

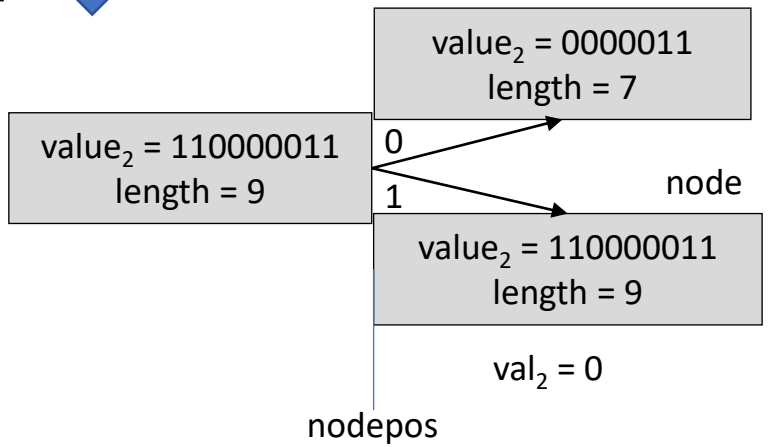



```
1  if endpos ≤
     node_endpos
     then
2  |   return
3  end
```




```
1  if node.value ≠
     val then
2  |   node= NULL
3  |   return
4  end
```





Target accumulation and attack detection

```
1  if node.target ≠ NULL then
2  |   if
   |     check_for_attack_during_ID(node.target)
   |     then
3  |   |   attack_detected ← True
4  |   end
5  |   if |node.target.rlinks| > 0 then
6  |   |   targets ← targets ∪ node.target
7  |   end
8  end
```

Node change

```
1  if sl = 0 then
2  |   node ← node.0child
3  end
4  else
5  |   node ← node.1child
6  end
7  nodepos ← nodepos + node.length
8  val ← 0
9  if node ≠ NULL then
10 |   trace_radix_tree(pw,sl,pos);
11 end
```

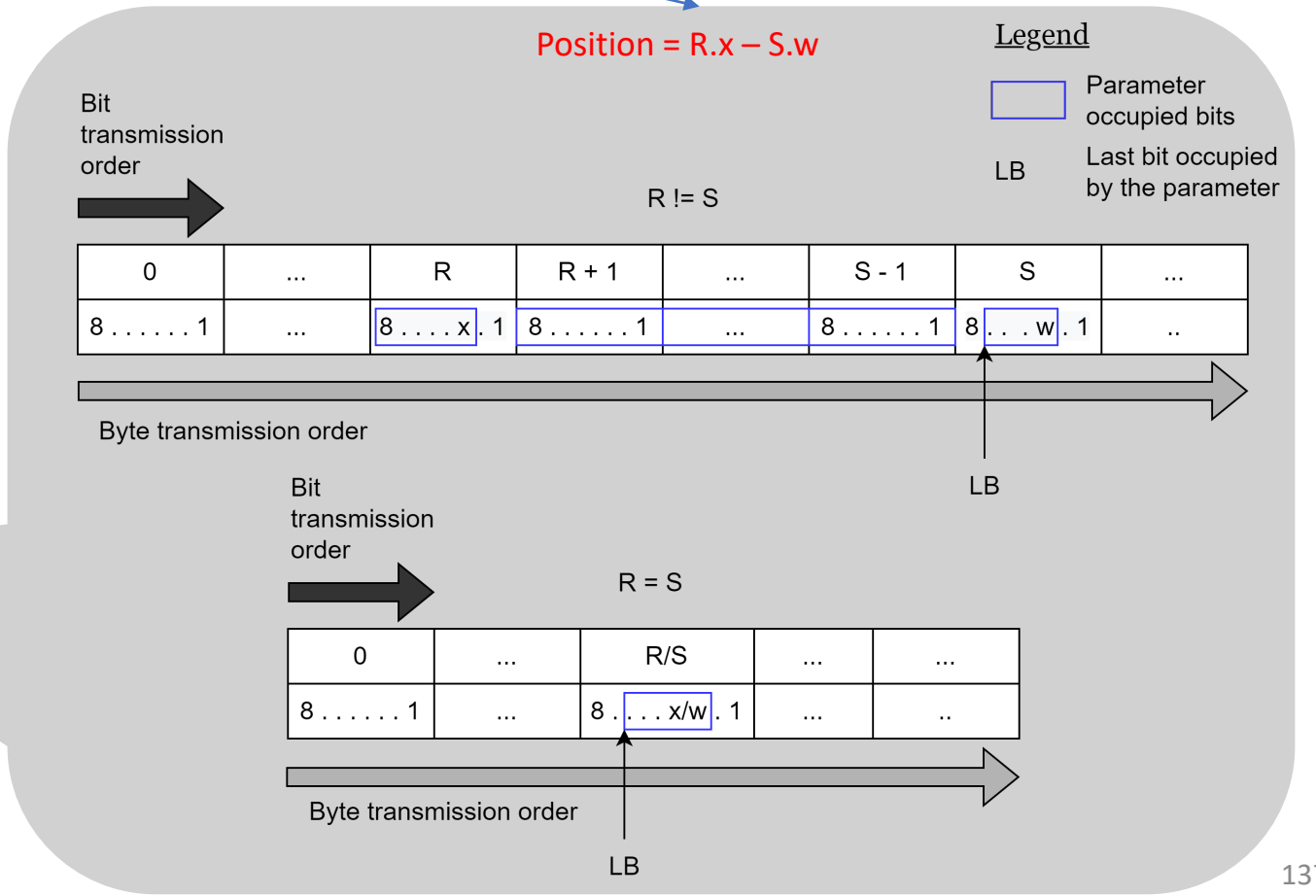
Parameter Placement

Suspect Parameter Number (SPN) uniquely identifies a parameter

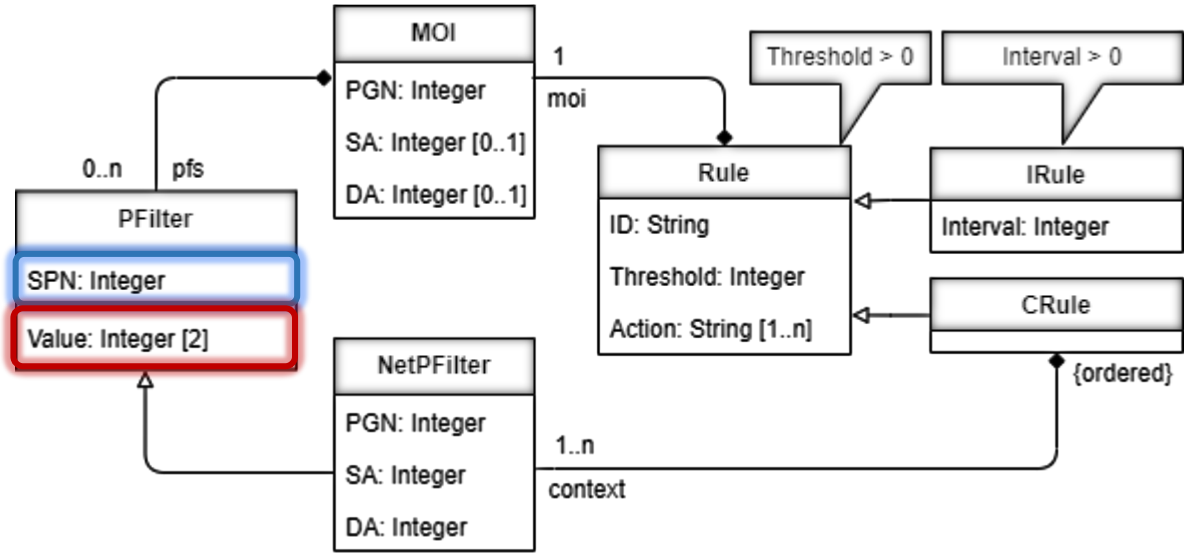
SAE J1939 Digital Annex

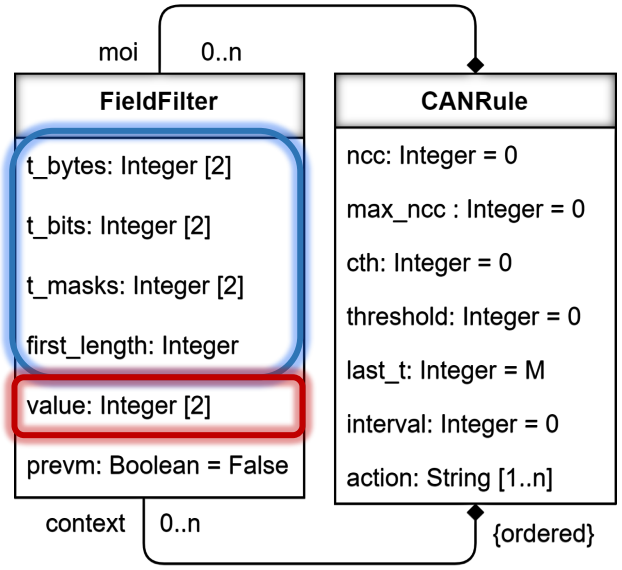
SPN	Resolution	Offset	Position	Length

Parameter value → value = (value – offset)/ resolution → Message Data

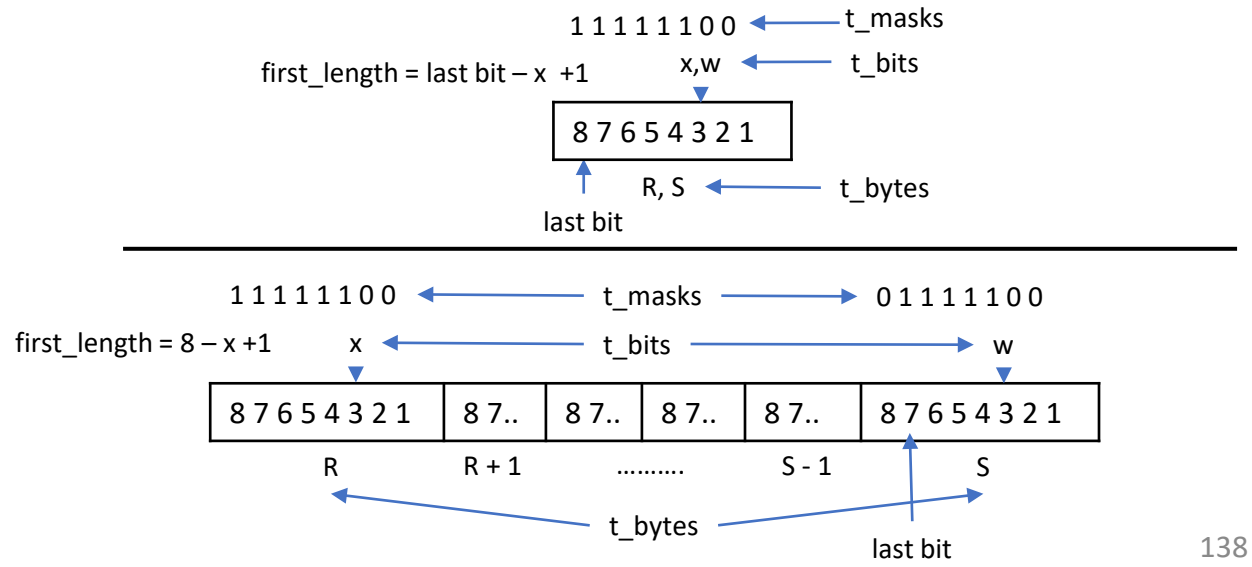


FieldFilter Placement-Related Field Derivation



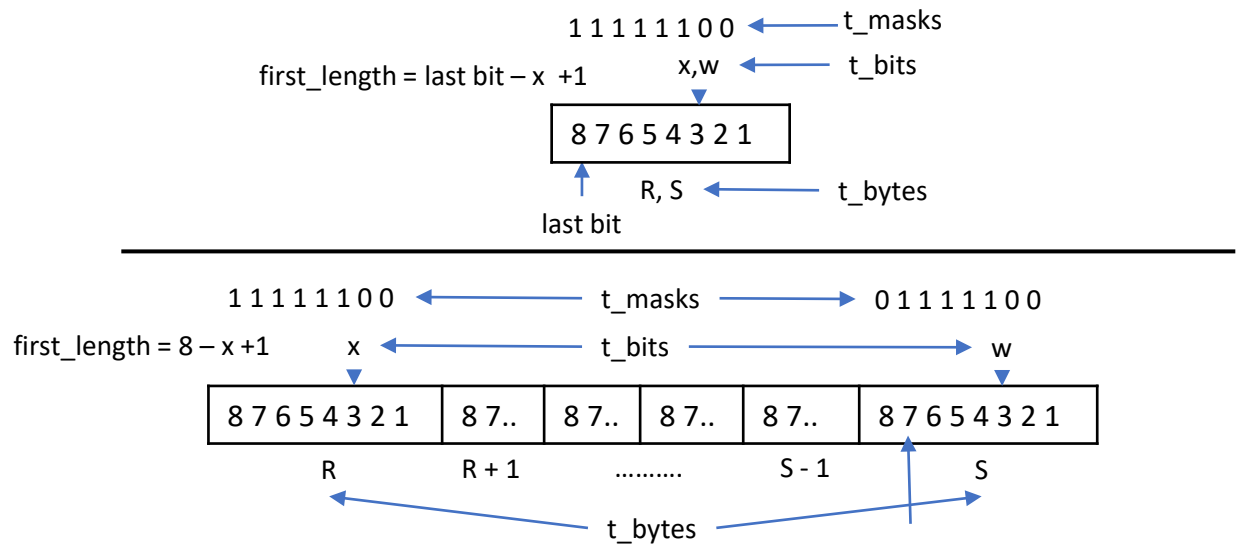


- `value` = [(`Value`[*i*]-*offset*)/*resolution* for *i* in 0..1]
- `t_bytes`, `t_bits`, `t_masks`, `first_length` derived from the parameter placement notation *R.x – S.w*
- *R.x – S.w*, *offset* and *resolution* obtained by querying the SAE J1939 digital annex using the SPN



Runtime get value(buffered data bytes, field filter)

- If R = S
  - Apply t_masks[0] to the t_bytes[0]th data byte and return it after right shifting by t_bits[0] -1
- Else
  - Apply t_masks[1] to the t_bytes[1]th (i.e. Sth) data byte and assign it to a temporary variable after right shifting by t_bits[1] -1.
  - Then append (using left shift and bitwise OR) the bits of bytes S -1 through R+1 to the temporary variable in that order.
  - Finally, we apply t_masks[0] to the t_bytes[0]th (i.e. Rth) data byte and append it to the temporary variable after right shifting it by t_bits[0] -1 and left shifting the temporary variable by first_length.



Bibliography

- [Stac19] Stephen Stachowski, Russ Bielawski, and André Weimerskirch, “Cybersecurity Research Considerations for Heavy Vehicles,” University of Michigan, Ann Arbor, Transportation Research Institute, 2019.
- [Stud13] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi, “Survey on security threats and protection mechanisms in embedded automotive networks,” in 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W), Budapest, Hungary, Jun. 2013, pp. 1–12.
- [Louk19] G. Loukas, E. Karapistoli, E. Panaousis, P. Sarigiannidis, A. Bezemskij, and T. Vuong, “A taxonomy and survey of cyber-physical intrusion detection approaches for vehicles,” Ad Hoc Networks, vol. 84, pp. 124–147, Mar. 2019
- [Babo05] F. Baboescu and G. Varghese, “Scalable packet classification,” IEEE/ACM Trans. Networking, vol. 13, no. 1, pp. 2–14, Feb. 2005
- [Brut00] J. D. Brutlag, “Aberrant Behavior Detection in Time Series for Network Monitoring,” in Proceedings of the 14th Systems Administration Conference (LISA 2000), New Orleans, Louisiana, USA, 2000, p. 9.

- [Boud16] A. Boudguiga, W. Klaudel, A. Boulanger, and P. Chiron, “A simple intrusion detection method for controller area network,” in 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, May 2016, pp. 1–7. doi: 10.1109/ICC.2016.7511098.
- [Daga16] T. Dagan and A. Wool, “Parrot, a software-only anti-spoofing defense system for the CAN bus,” in Embedded Security in Cars, EUROPE, Munich, Germany, 2016, p. 10.
- [Ansa17] M. R. Ansari, W. T. Miller, C. She, and Q. Yu, “A low-cost masquerade and replay attack detection method for CAN in automobiles,” in 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, May 2017, pp. 1–4. doi: 10.1109/ISCAS.2017.8050833.
- [Mats12] T. Matsumoto, M. Hata, M. Tanabe, K. Yoshioka, and K. Oishi, “A Method of Preventing Unauthorized Data Transmission in Controller Area Network,” in 2012 IEEE 75th Vehicular Technology Conference (VTC Spring), Yokohama, Japan, May 2012, pp. 1–5. doi: 10.1109/VETECS.2012.6240294.
- [Abbo16] S. Abbott-McCune and L. A. Shay, “Intrusion prevention system of automotive network CAN bus,” in 2016 IEEE International Carnahan Conference on Security Technology (ICCST), Orlando, FL, USA, Oct. 2016, pp. 1–8. doi: 10.1109/CCST.2016.7815711.
- [Ujii16] Y. Ujiie et al., “A Method for Disabling Malicious CAN Messages by Using a CMI-ECU,” presented at the SAE 2016 World Congress and Exhibition, Detriot, Michigan, USA, Apr. 2016. doi: 10.4271/2016-01-0068.
- [Gian17] H. Giannopoulos, A. M. Wyglinski, and J. Chapman, “Securing Vehicular Controller Area Networks: An Approach to Active Bus-Level Countermeasures,” IEEE Vehicular Technology Magazine, vol. 12, no. 4, pp. 60–68, Dec. 2017, doi: 10.1109/MVT.2017.2647814.

- [Lars08] U. E. Larson, D. K. Nilsson, and E. Jonsson, “An approach to specification-based attack detection for in-vehicle networks,” in 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands, Jun. 2008, pp. 220–225. doi: 10.1109/IVS.2008.4621263.
- [Stud18] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, and Y. Laarouchi, “A language-based intrusion detection approach for automotive embedded networks,” International Journal of Embedded Systems, vol. 10, no. 1, pp. 1--12, 2018.
- [Jin21] S. Jin, J.-G. Chung, and Y. Xu, “Signature-Based Intrusion Detection System (IDS) for In-Vehicle CAN Bus Network,” in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Online, May 2021, pp. 1–5. doi: 10.1109/ISCAS51556.2021.9401087.
- [Lars08A] U. E. Larson and D. K. Nilsson, “Securing vehicles against cyber attacks,” in Proceedings of the 4th annual workshop on Cyber security and informaiton intelligence research developing strategies to meet the cyber security and information intelligence challenges ahead - CSIIRW ’08, Oak Ridge, Tennessee, 2008, p. 1. doi: 10.1145/1413140.1413174.

  [Zhan17] M. Zhang, C. Chen, T. Wo, T. Xie, M. Z. A. Bhuiyan, and X. Lin, “SafeDrive: Online Driving Anomaly Detection From Large-Scale Vehicle Data,” IEEE Trans. Ind. Inf., vol. 13, no. 4, pp. 2087–2096, 2017, doi: 10.1109/TII.2017.2674661.
- [Murv18] P.-S. Murvay and B. Groza, “Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol,” IEEE Transactions on Vehicular Technology, vol. 67, no. 5, pp. 4325–4339, May 2018, doi: 10.1109/TVT.2018.2795384.
- [Bura16] Y. Burakova, B. Hass, L. Millar, and A. Weimerskirch, “Truck Hacking: An Experimental Analysis of the SAE J1939 Standard,” in Proceedings of the 10th USENIX Conference on Offensive Technologies, Austin, TX, USA, 2016, pp. 211–220.

- [Mill14] C. Miller and C. Valasek, “A Survey of Remote Automotive Attack Surfaces,” presented at the Blackhat USA, Las Vegas, NV, USA, 2014. [Online]. Available: http://illmatics.com/remote%20attack%20surfaces.pdf
- [Chec11] S. Checkoway *et al.*, “Comprehensive Experimental Analyses of Automotive Attack Surfaces,” in *USENIX Security Symposium*, San Francisco, CA, USA, 2011, vol. 4, pp. 447–462.
- [Mill13] C. Miller and C. Valasek, “Adventures in automotive networks and control units,” in Def Con 21, 2013, pp. 15--31. Accessed: Mar. 28, 2021. [Online]. Available: https://www.youtube.com/watch?v=n70hIu9lcYo&ab_channel=DEFCONConference
- [Mukh16] S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, “Practical DoS Attacks on Embedded Networks in Commercial Vehicles,” in *International Conference on Information Systems Security*, Jaipur, Rajasthan, India, 2016, pp. 23--42. doi: 10.1007/978-3-319-49806-5_2.
- [Mukh17] S. Mukherjee, J. Walkery, I. Rayz, and J. Daily, “A Precedence Graph-Based Approach to Detect Message Injection Attacks in J1939 Based Networks,” in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, Aug. 2017, pp. 67–6709. doi: 10.1109/PST.2017.00018.
- [Campo21] M. T. Campo, S. Mukherjee, and J. Daily, “Real-Time Network Defense of SAE J1939 Address Claim Attacks,” *SAE Int. J. Commer. Veh.*, vol. 14, no. 3, Aug. 2021, doi: 10.4271/02-14-03-0026.
- [Mukh20] S. Mukherjee, J. C. Van Etten, N. R. Samyukta, J. Walker, I. Ray, and I. Ray, “TruckSTM: Runtime Realization of Operational State Transitions for Medium and Heavy Duty Vehicles,” *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 1, pp. 1–25, Jan. 2020, doi: 10.1145/3300183.

- [Dail16] J. Daily et al., “Towards a Cyber Assurance Testbed for Heavy Vehicle Electronic Controls,” SAE Int. J. Commer. Veh., vol. 9, no. 2, pp. 339–349, Sep. 2016, doi: 10.4271/2016-01-8142.
- [Groz17] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede, “LiBrA-CAN: Lightweight Broadcast Authentication for Controller Area Networks,” ACM Trans. Embed. Comput. Syst., vol. 16, no. 3, pp. 1–28, Jul. 2017, doi: 10.1145/3056506.
- [Kura14] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, “CaCAN - Centralized Authentication System in CAN (Controller Area Network),” in 14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)., Munich, Germany, 2014, p. 10.
- [Choi18] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, “Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks,” IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pp. 4757–4770, Jun. 2018, doi: 10.1109/TVT.2018.2810232.
- [Cho17] K.-T. Cho and K. G. Shin, “Viden: Attacker Identification on In-Vehicle Networks,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas Texas USA, Oct. 2017, pp. 1109–1123. doi: 10.1145/3133956.3134001.
- [Tayl15] A. Taylor, N. Japkowicz, and S. Leblanc, “Frequency-based anomaly detection for the automotive CAN bus,” in 2015 World Congress on Industrial Control Systems Security (WCICSS), Dec. 2015, pp. 45–49. doi: 10.1109/WCICSS.2015.7420322.
- [Song16] H. M. Song, H. R. Kim, and H. K. Kim, “Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network,” in 2016 International Conference on Information Networking (ICOIN), Jan. 2016, pp. 63–68. doi: 10.1109/ICOIN.2016.7427089.

- [Moor17] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, “Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection,” in Proceedings of the 12th Annual Conference on Cyber and Information Security Research, Oak Ridge Tennessee USA, Apr. 2017, pp. 1–4. doi: 10.1145/3064814.3064816.
- [Cho16] K.-T. Cho and K. G. Shin, “Fingerprinting Electronic Control Units for Vehicle Intrusion Detection,” in Proceedings of the 25th USENIX Conference on Security Symposium, Austin, TX, USA, 2016, pp. 911--927.
- [Marc17] M. Marchetti and D. Stabili, “Anomaly detection of CAN bus messages through analysis of ID sequences,” in 2017 IEEE Intelligent Vehicles Symposium (IV), Jun. 2017, pp. 1577–1583. doi: 10.1109/IVS.2017.7995934.
- [Katr20] S. Katragadda, P. J. Darby, A. Roche, and R. Gottumukkala, “Detecting Low-Rate Replay-Based Injection Attacks on In-Vehicle Networks,” IEEE Access, vol. 8, pp. 54979–54993, 2020, doi: 10.1109/ACCESS.2020.2980523.
- [Marc16] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, “Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms,” in 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), Bologna, Italy, Sep. 2016, pp. 1--6. doi: 10.1109/RTSI.2016.7740627.
- [Mute11] M. Müter and N. Asaj, “Entropy-based anomaly detection for in-vehicle networks,” in 2011 IEEE Intelligent Vehicles Symposium (IV), Jun. 2011, pp. 1110–1115. doi: 10.1109/IVS.2011.5940552.
- [Tayl16] A. Taylor, S. Leblanc, and N. Japkowicz, “Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks,” in 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, Oct. 2016, pp. 130–139. doi: 10.1109/DSAA.2016.20.

- [Kang16] M.-J. Kang and J.-W. Kang, “A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security,” in 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, China, May 2016, pp. 1–5. doi: 10.1109/VTCSpring.2016.7504089.
- [Mart17] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone, “Car hacking identification through fuzzy logic algorithms,” in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, Italy, Jul. 2017, pp. 1–7. doi: 10.1109/FUZZ-IEEE.2017.8015464.
- [Nara16] S. N. Narayanan, S. Mittal, and A. Joshi, “OBD_SecureAlert: An Anomaly Detection System for Vehicles,” in Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St Louis, MO, USA, May 2016, pp. 1--6. doi: 10.1109/SMARTCOMP.2016.7501710.
- [Wasi17] A. R. Wasicek, M. D. Pese, and A. Weimerskirch, “Context-aware Intrusion Detection in Automotive Control Systems,” presented at the 5th ESCAR USA Conf, Ypsilanti, MI, 2017.
- [Nils09] D. K. Nilsson and U. E. Larson, “A Defense-in-Depth Approach to Securing the Wireless Vehicle Infrastructure,” Journal of Networks, vol. 4, no. 7, pp. 552--564, 2009.
- [Szpa90] W. Szpankowski, “Patricia tries again revisited,” J. ACM, vol. 37, no. 4, pp. 691–711, Oct. 1990, doi: 10.1145/96559.214080.
- [1] E. Aliwa, O. Rana, C. Perera, and P. Burnap, “Cyberattacks and Countermeasures for In-Vehicle Networks,” *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–37, Mar. 2021, doi: 10.1145/3431233.

Abbreviations

- MCU: Microcontroller Unit
- CAN: Controller Area Network
- PDU: Protocol Data Unit
- PGN: Parameter Group Number
- DA: Destination Address
- SA: Source Address
- RTS: Request to Send
- CTS: Clear to Send
- DT: Data Transfer
- EoMA: End of Message Acknowledgment
- Pr: Priority
- DP: Data Page
- PF: PDU Format
- PS: PDU Specific
- SOF: Start of Frame
- SRR: Substitute Remote Request
- IDE: Identifier Extension
- RTR: Remote Transmission Request
- DLC: Data Length Code
- CRC: Cyclic Redundancy Check
- ACK: Acknowledgment
- EOF: End of Frame
- EDP: Extended Data Page