DETECTING ANOMALIES IN CONTROLLER AREA NETWORK FOR AUTOMOBILES

A Thesis

by

DAKSH KUMAR VASISTHA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	A. L. Narasimha Reddy
Co-Chair of Committee,	Riccardo Bettati
Committee Members,	Srinivas Shakkottai
	Alireza Talebpour
Head of Department,	Miroslav M. Begovic

August 2017

Major Subject: Computer Engineering

ABSTRACT

Availability of interfaces such as WI-FI, Bluetooth and Cellular networks, software components to control a vehicle's functionality, and lack of security mechanisms in the Controller Area Network (CAN) bus protocol make vehicles vulnerable to attacks. In the recent past, researchers used internal and external attacks on automobiles to demonstrate that it is feasible to compromise the vehicle through the transmission of malicious messages on the vehicle's CAN bus.

To defend against such attacks, we propose three detection techniques. First, cross correlating and validating sensor values across multiple sensors can improve the data integrity of CAN bus messages. Second, the order of the messages from a single Electronic Control Unit (ECU) can be used to detect anomalies. CAN messages from the ECU should always be seen in a specific order as they are transmitted one after the other based on the priorities of messages. Fabrication and suspension attacks can be detected using such schemes. Third, a timing based detector is proposed to observe and detect changes in the timing behavior through deterministic and statistical techniques. An anomaly detection is possible after one malicious message if the CAN bus utilization is less than 50% or after at most three malicious messages if the CAN bus utilization is greater than 50% using a deterministic detection technique. The detection of an attack is possible with good accuracy and low false positive rates using a statistical detection technique but at the cost of longer detection latency.

ACKNOWLEDGMENTS

I would like to thank Dr. A. L. Narasimha Reddy for his continuous support throughout the research. It was a great learning experience under him. He was open to discuss problems related to the thesis and very helpful in the thesis. I would also like to thank Dr. Riccardo Bettati for spending time with me and enabling me understand concepts. The solutions proposed in the thesis were a direct benefit from my discussions with him. He was very helpful throughout the research.

I want to thank Dr. Srinivas Shakkottai and Dr. Alireza Talebpour for serving as my committee members and being always reachable for help.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. A. L. Narasimha Reddy and Dr. Srinivas Shakkottai of the Department of Electrical and Computer Engineering and Dr. Riccardo Bettati of the Department of Computer Science and Engineering and Dr. Alireza Talebpour of the Department of Civil Engineering. All work for the thesis was completed by the student, under the advisement of Dr. A. L. Narasimha Reddy of the Department of Electrical and Computer Engineering and Dr. Riccardo Bettati of the Department of Computer Science and Engineering.

Funding Sources

Graduate study was supported by a scholarship from the Department of Electrical and Computer Engineering.

NOMENCLATURE

CAN	Controller Area Network
ECU	Electronic Control Unit
DLC	Data Length Code
CRC	Cyclic Redundancy Check
ACK	Acknowledgment
EOF	End Of Frame
RPM	Rotation Per Minute
OBD	On Board Diagnostics
WI-FI	Wireless Fidelity
SOF	Start Of Frame
RTR	Remote Transmission Request
IDE	Identifier Extension
DOS	Denial Of Service

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The security in automobiles is of paramount importance as it directly impacts property and well-being of drivers, passengers and pedestrians. To control various features of an automobile, many software components are added, and exposure to external interfaces like Bluetooth [1], WI-FI [2] and Cellular networks [3] for vehicle-to-vehicle or vehicle-to-infrastructure communication have made vehicles vulnerable to external and internal attacks. Modern automobiles are equipped with dozens of Electronic Control Units (ECU), which are interconnected via in-vehicle networks such as Controller Area Network (CAN) bus [4], FlexRay [5] and Local Interconnect Network(LIN) [6]. ECUs are responsible for the overall behavior of the automobile such as Lane Assist, Park Assist, Steering, Anti-lock brake system, cruise control etc. The security of the automobile is dependent on the real-time communication between these ECUs, which take data from sensors and accordingly control actuators. The overall security of the automobile can be compromised if an attacker sends fake messages with or without compromising any of the ECU's behavior in the automobile. A CAN data frame does not contain the identifier of the sender and hence the messages cannot be easily authenticated. This allows injection of fake packets either by directly connecting to the CAN bus or remotely through wireless interfaces. These messages can behave like normal messages and perform actions as desired by the attacker. The attacker can compromise an ECU and use it to send fake messages on the CAN bus or suspend existing messages.

Techniques such as anomaly detection can be used to detect attacks. We consider the following criteria to build an anomaly detector:

- The anomaly detector should be fully capable to be integrated in legacy systems, either as a software update or as a separate device that can be inserted in the On-

Board Diagnostics (OBD)-II port.

- The anomaly detector should be simple enough to be integrated within the CAN subsystem.
- The anomaly detector should have a high accuracy and very low false alarm rates. False alarms may cause user discomfort or panic and hence should be reduced to a minimum.

To defend against attacks, there are two possible approaches: *Authentication* and *Anomaly detection*. Due to the limited space in the CAN messsage, authentication is not easily possible. Researchers explored anomaly detection techniques to detect attacks. Muter *et al.* [7] introduce anomaly detection sensors, which allow attacks detection. This approach requires extra hardware to be introduced in the automobiles. Hence, installing this system into existing cars would be difficult. Artificial neural networks can help us detect attacks, but the detection efficiency of such methods may not be high. Theissler [8] proposes use of machine learning techniques such as a one-class support vector machine to detect anomalous behavior, but this approach has false negatives. Hence, may not raise an alarm when needed. Narayanan *et al.* [9] try to solve anomaly detection as a data analytic problem and applied machine learning techniques such as hidden Markov models to predict if a car is in normal or abnormal mode. This approach is dependent on the model of the car to differentiate between normal and abnormal modes. Clock-skew based detection [10] can be used to fingerprint an ECU based on the periodicity of the in-vehicle messages. This method has low false positive rates and hence is effective in detecting attacks, but it is complex to implement.

We propose several techniques for detecting anomalies on the CAN bus. We propose to cross-correlate and validate sensor values from multiple sensors. For example, wheel speeds from the four wheels can be correlated with each other to detect anomalies. Simi-

larly, other indicators such as steering angle, acceleration and Rotation Per Minute (RPM) can be cross-correlated with each other. We need to be aware of the semantics of the protocol to decode these values to feed into such a detector. Other cross-correlations such as the time when a door or a trunk can be opened with respect to speed of the car are also considered to flag anomalous behavior.

We propose a timing based detector to observe and detect changes in the timing behavior of messages through deterministic and statistical techniques. Attacks can be detected either by comparing the change in the delay distribution of messages because of insertion of attack or anomalous messages. The detection techniques can work by observing the timing behavior of a single message or through the effect on other messages. This detector is independent of the semantics of the protocol as it only requires the arrival time of messages instead of the decoded values of various fields to flag anomalous behavior.

Additionally, we can detect attacks considering the order of messages from a single ECU. CAN messages from the same ECU should always be seen in a specific order as they will be transmitted one after the other based on the priorities of messages, and any deviation from this order can be flagged. Message identifiers can be used to detect such changes in the order. This detector is simple and fast to detect attacks as any change in the order is immediately flagged. Observation of two successive messages from the same ECU can enable the detection of a fake message.

The rest of the thesis is organized as follows: Section 2 provides an introduction to the Controller Area Network protocol, and Section 3 provides the attack model we consider for this thesis. Section 4 provides solution criteria we consider while developing an anomaly detection module. Section 5 details the anomaly detection based on cross-correlation, Section 6 details the anomaly detection based on the order of messages, and Section 7 details the anomaly detection based on the timing behavior. Section 8 concludes the thesis.

2. CONTROLLER AREA NETWORK PROTOCOL

The Controller Area Network (CAN) protocol is a carrier sense multiple access protocol with collision detection. All CAN transmissions are broadcast in nature, which means nodes receive all the transmissions on the CAN bus. Bus arbitration, i.e. the resolution of which message gets transmitted next if a collision occurs, is decided by the message priority. Priority is decided based on the so-called identifier field in the message header. A lower identifier corresponds to a higher priority, which means messages with lower identifiers will be transmitted earlier than messages with higher identifiers if both are ready to be transmitted in the transmission buffer or if arbitration is triggered by collision during the transmission.

The structure of a CAN message is illustrated in Figure 2.1. It has the following important fields:

- **Identifier**: It represents the priority of a message. A lower-value identifier indicates higher priority on the CAN bus.
- **Data Length Code(DLC)**: It represents the length of the data field in a CAN message.
- **Data**: Data is the actual payload of the message. A total of 64 bits of data can be transmitted in the message.
- **Cyclic Redundancy Check(CRC)**: This field contains the checksum of the data field for the error detection.
- **Acknowledgment(ACK)**: The ACK bit is used to check the integrity of data. The receiving node overwrites the recessive bit to the dominant bit for an error-free mes-

sage. In the case of any error, the recessive bit is not overwritten, and the sending node retransmits the message.

Figure 2.1: Standard CAN Packet Format

2.1 CAN Arbitration

A CAN message transmission is non-preemptive, which means that a message currently being transmitted cannot be preempted by other messages. When messages are broadcast simultaneously on the bus, collisions occur. A priority driven arbitration protocol ensures that the highest priority message continues transmitting while the lower priority message transmission backs off, and its message gets transmitted later. Specifically, if two nodes try to transmit at the same time during an idle state, both nodes need to compete to get access to the bus. This conflict is resolved through the bit-wise arbitration as follows: the dominant bit always overwrites the recessive bit on the CAN bus, and the node transmitting 0 retains control of the bus, while the node transmitting 1 stops transmitting the message. Nodes transmitting a bit read the same bit on the bus, believing they win the arbitration, and nodes reading a different bit switch to listening mode. In other words, a lower identifier node continues its transmission, and all the higher identifier nodes switch to listening mode. These higher identifier nodes will contend for the CAN bus in the next available slot. In the absence of two similar identifier message competing to occupy the CAN bus, only one message can occupy the bus.

Figure 2.2 shows CAN arbitration; two nodes compete to send messages on the CAN

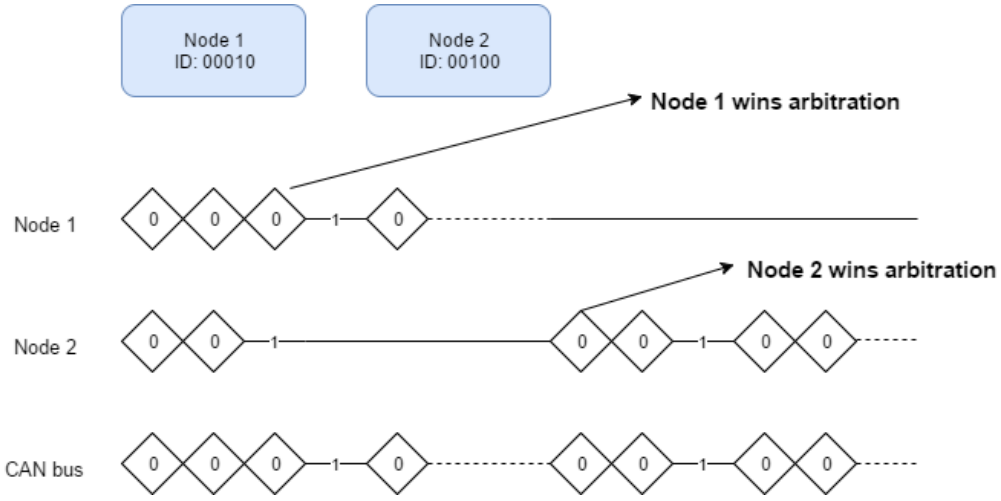


Figure 2.2: CAN Bus Arbitration

bus. Node 1 and Node 2 send messages at the same time, and contention will start by the SOF bit. Node 1 and Node 2 send bit 0 as a part of the arbitration; as a result the CAN bus state will be shown as 0. Both nodes don't see any difference between the CAN bus state and the last transmitted bit, so they continue to send arbitration bits until any one of them find differences between the transmitted bit and the current state of the bus. Node 1 transmits the third arbitration bit as 0, but Node 2 transmits bit as 1. When the bus state is 0, Node 2 figures out that it lost arbitration and switches to listening mode. Node 1 will continue transmitting the CAN message. Node 2 will try again once Node 1 is done transmitting the message, and this time as there is no node contending for the bus, it will win the arbitration and send the whole message on the CAN bit by bit.

2.2 Message Transmission and Reception

The CAN controller is the hardware component responsible for physical access to the CAN bus and has one or more registers responsible to store, transmit and receive messages. These registers are called TxObjects and RxObjects respectively. When CAN transmissions are broadcasted, all nodes will receive all the messages, but few messages go up in

the software stack. Nodes need a mechanism to select few relevant messages. Filtering can be applied to each register available to receive messages based on the identifier field of a message.

The CAN driver is the software component which initializes the CAN controller and perform transmission and reception of messages. It stores transmission messages in the queue when the size of the hardware buffer is less than the total number of messages queued for transmission. To receive new messages, the CAN driver can use polling to look for the content of the hardware buffer and store the new message when it arrives. Additionally, the CAN controller can send an interrupt signal to the CAN driver when it receives a new message.

Lower identifier messages are selected out of all the messages queued for transmission in the transmission buffer. The CAN controller can have the transmission buffer to accommodate all messages queued to be transmitted. In this case, we can have a priority queue to arrange the messages in decreasing order of priority and choose the top element for transmission when the bus is idle. When the transmission buffer is smaller than the total number of messages available for transmission, extra messages can be accommodated in the software queue, and when the transmission buffer is free, messages from the software stack can be copied to the hardware buffer. When a new message arrives, and its priority is higher than any message present in the hardware buffer [4], lowest priority element is evicted from the hardware buffer, copied to the software queue, and the new message is enqueued in its location. During such eviction process and when the hardware buffer size is small, a low priority packet from another node can occupy the CAN bus while the higher priority message is being copied to the transmission buffer of the CAN controller from the software stack.

2.3 System Model

We consider a flow m_i with period P_i, priority τ_i, length S_i and $f_{i,j}$ be its jth instance. When $f_{i,j}$ is en-queued to be transmitted on the CAN bus, it can be delayed [4] due to the blocking delay due to the transmission of lower priority flows or the interference delay due to higher priority flows.

3. ATTACK MODEL

ECUs can be compromised in a way that they are now allowed to send messages or send fake messages. Fake messages can also be inserted physically or remotely on the CAN bus. Recent studies by Miller and Valasek [11, 12] on Jeep Cherokee, Ford Escape, Toyota Prius and researchers from Keen Security Lab on Tesla demonstrated that it is possible to control brake, steering, speed, trunk, door, acceleration etc. physically or remotely. [13] showed vulnerabilities in Telematics control unit (TCU) and showed that these devices can be compromised by an attacker. [14, 15] show that ECUs can be compromised using various interfaces. We need anomaly detection techniques to detect attacks, and let passengers know about the intrusion by ringing an alarm. The attacker can target many cars by remote attacks, and hence we need some measure to overcome such attacks. We consider the following attacks in this thesis.

- **Fabrication attack**: The attacker can inject fake messages of different flows physically or remotely in the vehicle's CAN bus
- **Suspension attack**: The attacker can compromise one or multiple ECUs, and restrict them from sending few or all messages.
- **Denial Of Service (DOS) attack**: The attacker can flood the CAN bus so that nodes cannot transmit valid messages.

An adversary can attack the CAN bus in the following ways:

- *Compromising an ECU.*
- *Addition of new ECU* on the CAN bus.
- *Remote attacks* through interfaces such as WI-FI and Bluetooth.

4. SOLUTION CRITERIA

To build an anomaly detector, the following criteria are considered:

- We need some way to tackle anomalies in existing cars. The change of hardware or addition of new hardware requires extra out-of-pocket cost.
- When possible, we will look for software solutions.
- The detector should be fast or have a small latency in detecting an attack. Detecting anomalies after a long time is equivalent to not detecting at all. We need to warn users quickly to limit the effect caused by the attack.
- We also need to use less resources to incorporate the detector to reduce cost and overall load on the CAN bus.

4.1 Location of Anomaly Detection

The CAN protocol is broadcast in nature which means the CAN controller receives all the messages, and based on the filtering decision at the CAN controller, few messages will go up in the stack for further processing. An anomaly detector scheme in the CAN subsystem can be located at various levels of the protocol stack as shown in Figure 4.1. For instance, the anomaly detector can be located close enough to the physical layer of the CAN bus or in the software stack of an ECU. Depending on the location, the anomaly detector may or may not receive all the messages. We consider the following two deployment scenarios:

4.1.1 Hardware Detector

In this deployment scenario, a detector is located close enough to the physical layer of the CAN bus so that all broadcast messages can be monitored. The detector has the

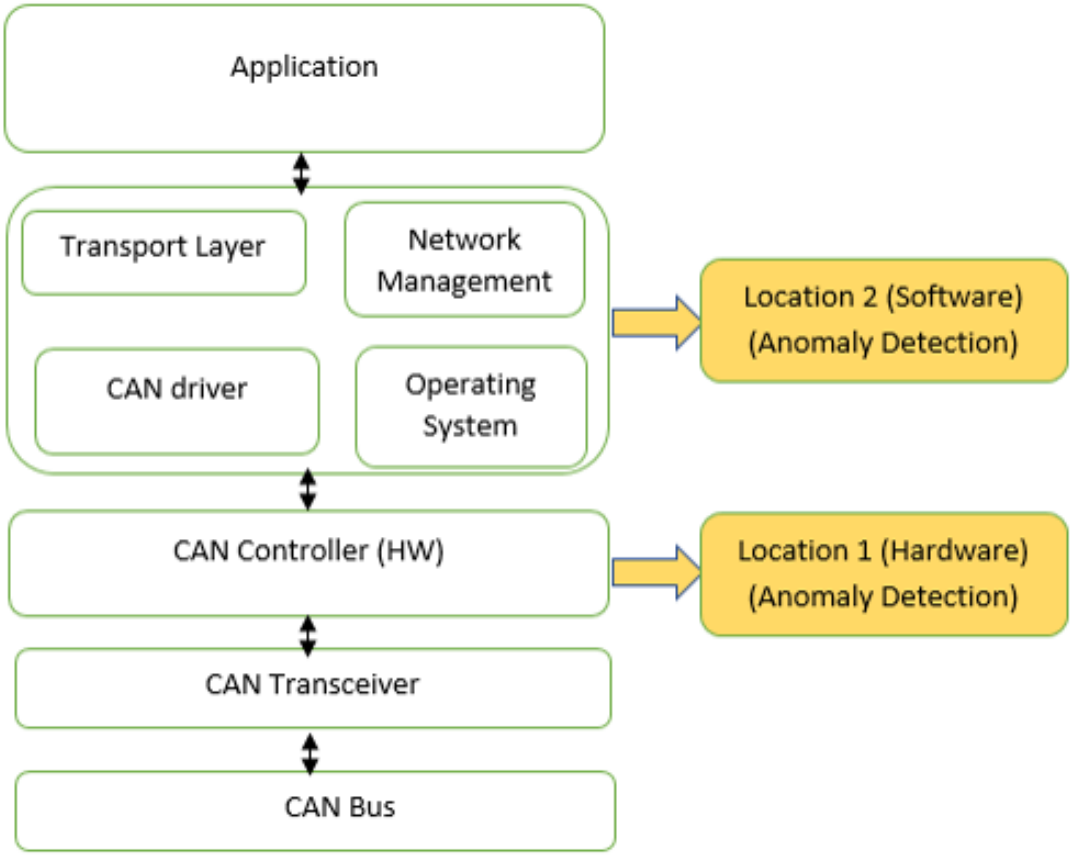


Figure 4.1: Location of Anomaly Detection Module in CAN Subsystem

knowledge of all the messages transmitted on the CAN bus and can look for the behavior, such as inter-arrival time, payload content etc. Any deviation from the normal behavior can be flagged as an anomaly.

4.1.2 Software Detector

Alternatively, one way to locate a detector is close to the software stack of an ECU. The detector has the knowledge of only few messages meant to be received by the ECU and hence can only detect anomalies observed in those messages.

5. ANOMALY DETECTION BASED ON CROSS-CORRELATION

A cross-correlation based anomaly detection approach can be applied to determine a drastic change in the value of a field at a given time with respect to the value of another field. For instance, all wheel speeds are correlated, which means any change in the value of one wheel speed should result in a similar amount of change in the value of other wheel speed. An anomaly can be flagged when one value at a given time changed too much, while the other value didn't change much. The wheel speeds can be correlated with the steering angle. If the car is being steered straight, the wheel speeds should match. When the steering angle indicates a turn, the wheel speeds can differ, but the differences can be correlated against the steering angle.

The drastic change in a value at a given time can be affected by sending fake messages, such an attack needs to be detected to prevent abnormal behavior. A strong correlation between two values at a given point of time can allow us to design a mechanism for the attack detection. To find fields that are correlated, CAN bus message trace is analyzed, and we make the following observations:

- Wheel 1 & 2 speed are strongly correlated as shown in Figure 5.1 and Table 5.1 probably because two wheels are connected to the same axle.

- Wheel 3 & 4 speed and Wheel 1 & 3 speed are strongly correlated as shown in Figure 5.2 and Figure 5.3 respectively. Similarly, any other combination of wheel speeds are also correlated. Table 5.1 confirms such strong correlation. Attacks compromising few but not all wheel speeds can easily be detected as all wheel speeds are highly correlated.

- RPM & wheel speeds are strongly correlated until a brake is not applied, but not

strongly correlated after the brake is applied. Figure 5.4a and Figure 5.4b show scatter plots for both the scenarios. The cross-correlation between RPM & wheel speed 1 for both the above scenarios are shown in Table 5.1. It is noted that this correlation is dependent on the current gear and the driving behavior.

- RPM & acceleration are also correlated as shown in Table 5.1.

Table 5.1: Cross-Correlation between Different Fields

Fields	Cross-Correlation
Wheel 1 and 2 speed	1.0
Wheel 3 and 4 speed	0.994
Wheel 1 and 3 speed	0.998
Wheel 1 speed and RPM (before brake)	0.98
Wheel 1 speed and RPM (Overall)	0.84
Acceleration and RPM	.786

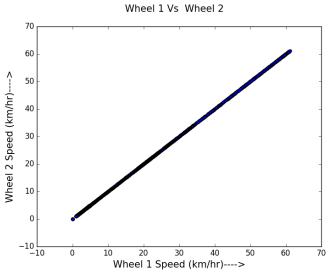


Figure 5.1: Scatter Plot between Wheel 1 and 2 Speed

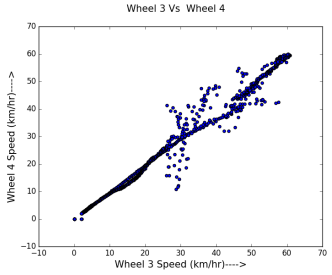


Figure 5.2: Scatter Plot between Wheel 3 and 4 Speed

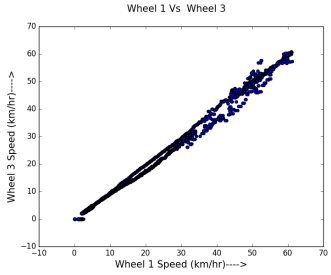
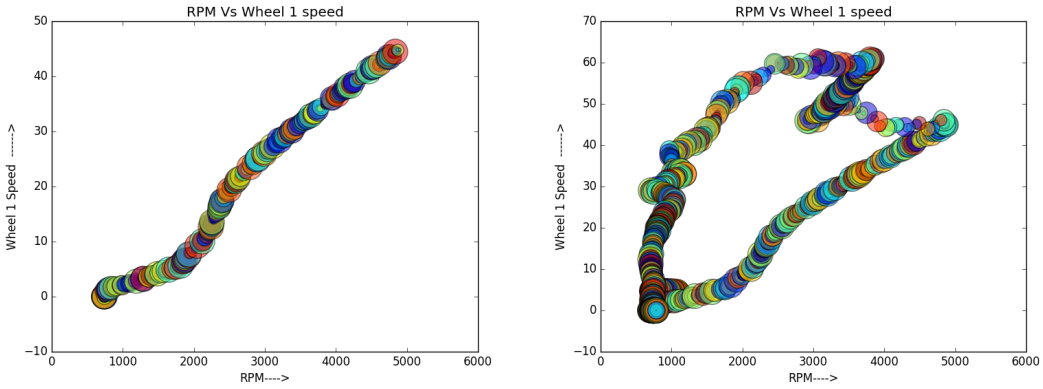


Figure 5.3: Scatter Plot between Wheel 1 and 3 Speed



(a) Before Applying Brake (b) During the Whole Trip

Figure 5.4: Scatter Plot between RPM and Speed

Several other cross validations can be applied to detect anomalies in a car based on the normal behavior such as:

- The trunk should not be opened when speed of the car is greater than 0. The desired behavior is closed trunk when the car is moving. Sudden opening of the trunk while the car is moving should be flagged as an anomaly.
- The door should not be opened when speed of the car is greater than 0. The desired behavior is closed door when the car is moving. Sudden opening of the door while the car is moving should be flagged as an anomaly.
- Speed of the car should not be greater than 5 km/hr. during Park Assist as described in [11] as it doesn't make sense to have high speed while parking the car.

The drastic change in one value can be determined if other value is not changed for a given time. Our detector works for both periodic and non-periodic traffic. All the fields are decoded in vendor specific format, so we need to know the semantics of the messages to decode the values. Such cross correlations or invariants across different sensors can provide validation of data integrity.

6. ANOMALY DETECTION BASED ON ORDER OF MESSAGES

An anomaly can be detected based on the order of messages transmitted by a single ECU. CAN messages from the single ECU should always be seen in a specific order as they are transmitted one after the other based on the priorities of messages. Any deviation from this order should be flagged. This could enable the detection of insertion of fake messages or suppression of valid messages.

6.1 Fabrication Attack

A fabrication attack can be detected as insertion of malicious messages change the order of message identifiers. The ECU can be compromised to send fake messages.

In Figure 6.1, ECU-1 sends messages with identifier 0x301, 0x302, 0x303 and 0x304, ECU-2 sends messages with identifier 0x13c, 0x158, 0x17c and 0x188, and ECU-3 is compromised to send fake messages with identifier 0x302 and 0x17c, which are normally sent by ECU-1 and ECU-2. In the normal case, messages transmitted from ECU-1 and ECU-2 are always in order. When the fabrication attack is staged through ECU-3, the order of messages may not be preserved enabling the detection of the attack. These attacks can be detected immediately when the order is violated.

A detector considering the order of messages to flag anomalous behavior can look for predefined patterns of message identifiers. Figure 6.2 shows the message identifiers transmitted on the CAN bus at any given index. The deviation from the normal behavior can be seen when a fake message with identifier 0x301, and a message with identifier 0x302 are inserted at index 9 and 10 after a bonafide message with identifier 0x304 is transmitted. Ideally 0x301 precedes 0x304, but due to the fake insertion of 0x301 and 0x302 after 0x304, the detector fires an alarm. Figure 6.3 shows an anomaly when a fake message with identifier 0x301 is inserted after a bonafide message with identifier 0x304.

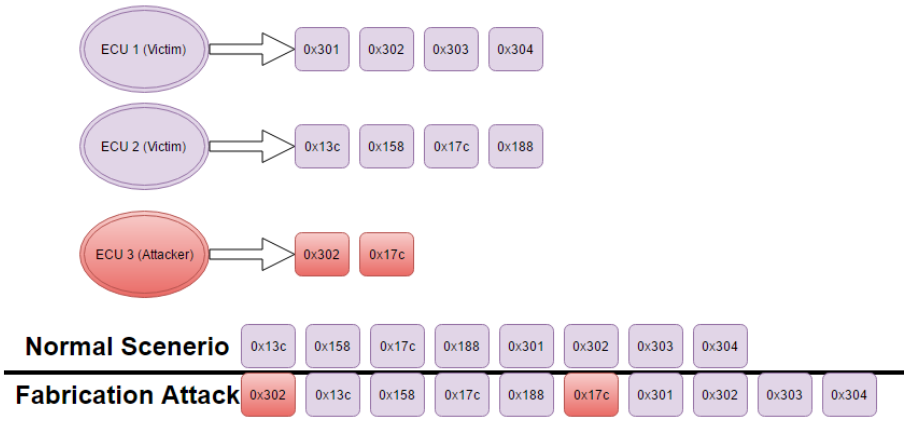


Figure 6.1: Detecting Fabrication Attacks Based on Order of Messages

Two 0x301 messages at index 9 and 10 can be seen. This will raise the alarm as another message identifier like 0x302, 0x303 and 0x304 should be transmitted by an ECU after transmitting 0x301.

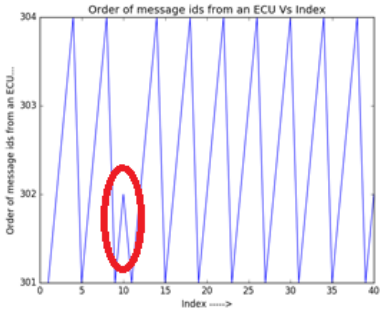


Figure 6.2: Inserting Fake Messages with Identifier 0x301 and 0x302

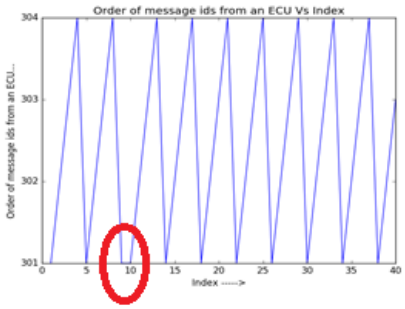


Figure 6.3: Inserting Fake Message with Identifier 0x301 at index 9

6.2 Suspension Attack

Figure 6.4 shows ECU-1 sends messages with identifier 0x301, 0x302, 0x303 and 0x304, and a compromised version of ECU-1 does not send a message identifier 0x301,

but sends messages with identifier 0x302, 0x303 and 0x304. A flag can be raised when a message with identifier 0x302 comes after a message identifier 0x304 instead of a message identifier 0x301. This happened because message identifier 0x301 is suspended in the compromised ECU-1.

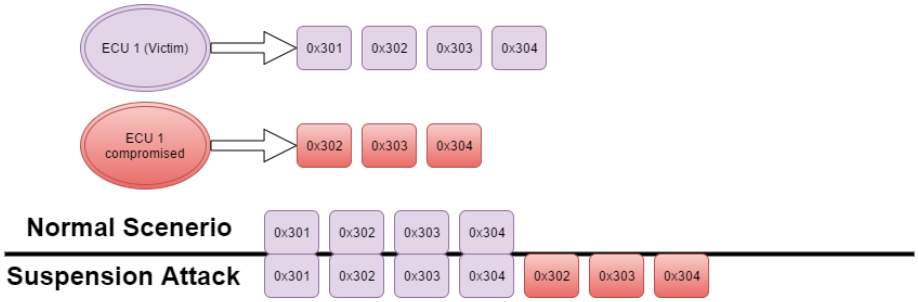


Figure 6.4: Detecting Suspension Attacks Based on Order of Messages

Few messages are not allowed to be transmitted by an attacker after compromising an ECU. Figure 6.5 shows suspension of 0x301 at index 13. Ideally, 0x302 should be preceded by 0x301, but 0x301 is suspended. Figure 6.6 shows the deviation from the normal behavior when the attacker suspends certain messages entirely.

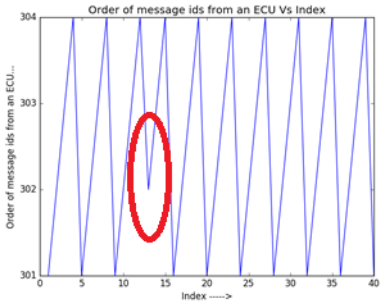


Figure 6.5: Suspension of 0x301 at Index 13

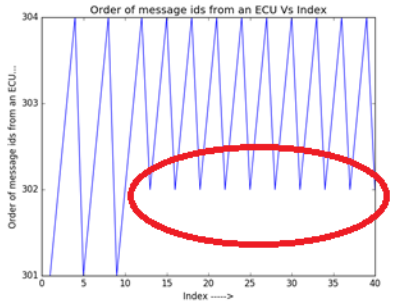


Figure 6.6: Suspension of 0x301 Messages after Index 13

This detector is simple and fast as it requires keeping track of the order of messages and raises an anomaly when the arrival pattern does not match the expected pattern. To detect an attack using such a detector, each ECU should transmit at least two messages to define the order.

An attacker could modify its behavior knowing the presence of such an anomaly detector. However, this requires that the attacker has to carefully monitor the arrival of messages on the CAN bus to insert fake messages of another ECU. Second, a compromised ECU has to continue sending all the packets of that ECU, that is suppression is not feasible.

7. ANOMALY DETECTION BASED ON TIMING BEHAVIOR

The timing behavior of periodic CAN messages can be leveraged to detect anomalies. Several approaches are possible to detect anomalies based on the timing behavior. They differ in two main aspects:

1. Based on the types of anomaly bounds, we consider the two approaches *Deterministic and Statistical Detection*;

2. Based on the types of messages used in the detection, messages of *a same or different identifier* than attacked messages can be used to detect an attack.

7.1 Analysis of Timing Behavior

To detect attacks on the vehicle's CAN bus, a mechanism such as anomaly detection is needed to find intrusions. For the anomaly detection based on the timing of messages, the timing behavior of messages in the CAN subsystem need to be determined, i.e. whether messages are periodic or not. In case messages are periodic, we need to find whether the period remains constant. We need these answers to go forward with the timing analysis.

We analyzed the traces of Honda Civic and Toyota Camry from [16] to see the timing behavior of messages. The traces contain information regarding CAN frames such as the identifier, data and the arrival time of a message. After a detailed analysis of the data, we make the following observations:

- Most of the CAN messages are periodic, which can be confirmed by observing the auto-correlation in Figure 7.1 and data in Table 7.1;

- The fluctuation in the inter-arrival times of packets occur as shown in Figure 7.2, which can be explained by the delay in transmitting a message by an ECU caused

by higher priority packets or the non-preemptive nature of the CAN bus.

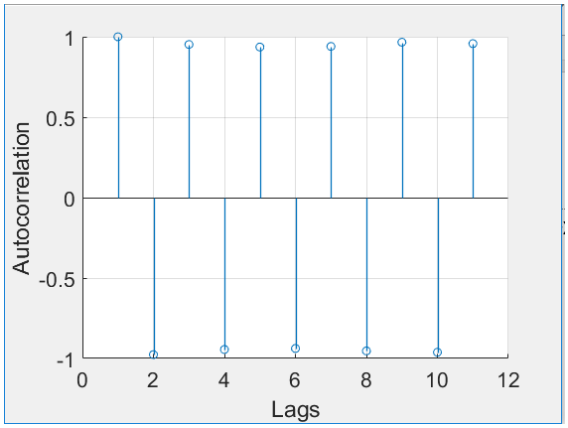


Figure 7.1: Auto-Correlation of Message with Identifier 0x17c

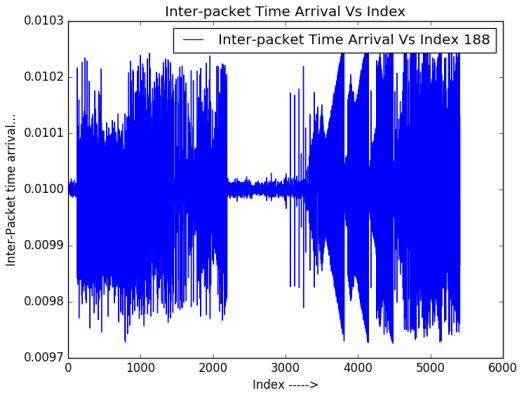


Figure 7.2: Inter-Arrival Times vs Message Index for Message with Identifier 0x188

Table 7.1: Number of Periodic and Non-Periodic Messages in Honda Civic and Toyota Camry

Model	Periodic messages	Non-Periodic messages
Honda Civic	100%	0%
Toyota Camry	93.1%	6.9%

7.2 Deterministic vs Statistical Detector

A workload on the CAN bus can be known at design time. In such cases, the worst-case workload can be bounded. Any deviation in the bounds can be detected by a deterministic anomaly detector. The insertion of a fake message changes the delay distribution of messages transmitted after the insertion of that fake message. The impact of such changes can be measured over time by a statistical anomaly detector.

7.3 Detection through Same vs Different Message Identifier

We consider how attacks can be detected through the timing behavior of different messages. In the first case, we consider attacks on the same message that is being analyzed or observed. In the second case, we look at the possibility of detecting attacks on other messages through the observation and analysis of a single message identifier. The inter-arrival time of a message changes when the same identifier message is attacked; such changes can be detected by a deterministic detector. The insertion of a fake message changes the delay distribution of messages with the same or different identifier than attacked messages over time; such changes can be detected by a statistical detector.

7.4 Deterministic Detector

It is common in automotive applications that the communication workload on the CAN bus is often known at design time. In such cases, the worst-case workload at any time can be bounded. When such a bound is exceeded, this indicates a malfunction of a component (example, a node may have turned into "babbling idiot") or the presence of additional workload. The latter can be bonafide or malicious. This information can be leveraged to design a deterministic anomaly detector. The bounded workload expresses itself as a set of transmission delays for the messages on the CAN bus. The detection rationale is that the additional workload naturally increases the transmission delay of messages or decrease in the workload effects the inter-arrival times of messages. We use the real time delay analysis to determine the worst-case delay to be experienced by the messages in the system.

The queuing delay WC_{fi} of a message is defined as the time interval from when the message is en-queued until it is transmitted on the bus. The delay WC_{fi} [4] consists of two factors:

- *Blocking delay* due to the transmission of a lower priority message when a new mes-

sage is en-queued for the transmission. Since these messages cannot be preempted, the newly en-queued message must wait;

- *Interference delay* caused by the transmission of higher priority messages.

We can calculate the worst-case response time WC_{fi} to find out the maximum possible time taken by $f_{i,j}$ once it is en-queued to be transmitted on the CAN bus.

In Figure 7.3, we denote Δ_1, the inter-arrival time between f_j and f_{j+1} when f_j instance is delayed and f_{j+1} instance is not delayed. We call Δ_2, the inter-arrival time between between f_{j+1} and f_{j+2} when neither instance is delayed. We call Δ_3, the inter-arrival between f_{j+2} and f_{j+3} when f_{j+2} instance is not delayed, but f_{j+3} instance is delayed. We call Δ_4, the inter-arrival time between f_{j+3} and f_{j+4} when both instances are delayed. We call $\Delta_{i,j}$ as the inter-arrival time between any consecutive instances of the flow m_i.

Equations 7.1, 7.2, 7.3 and 7.4 represent Δ_1, Δ_2, Δ_3 and Δ_4 respectively.

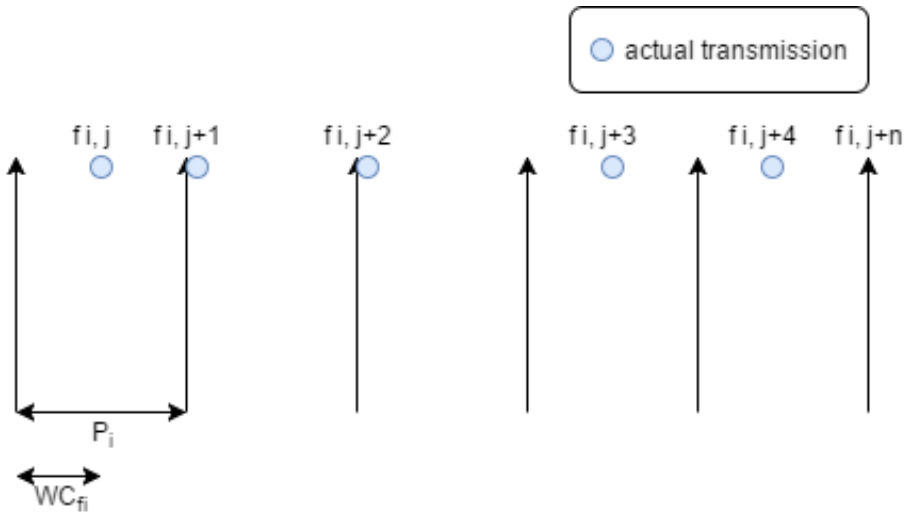


Figure 7.3: Inter-Arrival Times between Consecutive Message Instances

$$\Delta_1 \geqslant P - WC_{fi} \tag{7.1}$$

$$\Delta_2 = P \tag{7.2}$$

$$\Delta_3 \leqslant P + WC_{fi} \tag{7.3}$$

$$P - WC_{fi} < \Delta_4 < P + WC_{fi} \tag{7.4}$$

$$P - WC_{fi} \leqslant \Delta_{i,j} \leqslant P + WC_{fi} \tag{7.5}$$

We observe that the inter-arrival time between any two consecutive instances of the flow m_i should lie between $P - WC_{fi}$ and $P + WC_{fi}$ as shown in Equation 7.5. In case the inter-arrival time of the consecutive instances exceeds these bounds, an anomaly detector should raise an alarm.

7.4.1 Deterministic Same Message Identifier Detector

A deterministic detector can be used to detect an attack using a message with the same identifier as the attacked message. The above bounds may be exceeded when the message is inserted between normal messages. When the bounds are not exceeded, the timing behavior of the next message enables the detection of the attack. Attacks such as fabrication and suspension can be detected using this detector.

7.4.1.1 Detection of Fabrication Attack

Detecting an attack in a timely manner is important to raise an alarm as soon as something wrong happens in an automobile. Assuming we have at least one transmission of

m_i, and an attacker capable of inserting a fake message of same type as m_i physically or remotely at any given time. The fake message injection is typically detected as soon as the fake message is transmitted on the CAN bus and the minimum inter-message bound $P - WC_{fi}$ is exceeded. In case the bounds are not exceeded, we consider the detection scenarios where an alarm can be triggered after one fake message or at-most three fake messages depending on whether the bus utilization is below or above 50%:

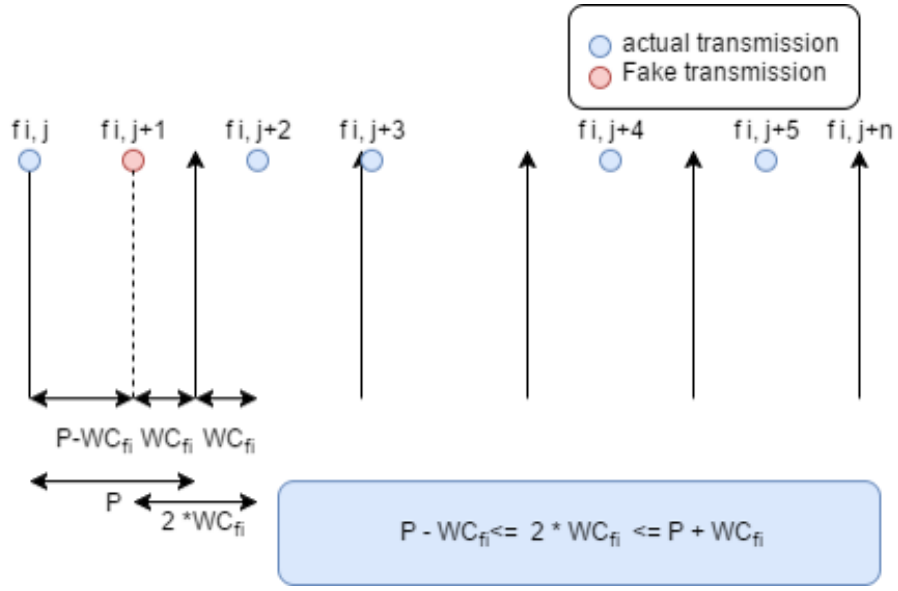


Figure 7.4: Fake Message Inserted between Message Instances

- **Detection with $P - WC_{fi}$ and $P + WC_{fi}$ bounds**: The bounds in Equation 7.5 can be used to detect an attack, but there are chances when the bounds are not exceeded. We simulated the attack on a Honda Civic by constructing a trace of malicious messages. This trace contains information such as the arrival time of a malicious message, the identifier etc. We merge this trace with the normal trace based on the arrival time of messages and the priority, i.e. a higher priority message will be transmitted

before a lower priority message. Attacks can be detected considering the bounds in Equation 7.5. To calculate the effect on the detection rate with an increase in the utilization, we change the transmission time of messages to 2 and 2.5 times the original transmission time. Figure 7.5 shows a decrease in the detection rate with the increase in the utilization. Such a behavior can be explained by the fact that the increase in the utilization will increase the worst-case response time. The change in the bounds in Equation 7.5 make attack hard to be detected by considering all inter-arrival times and the bounds. This behavior is also explained later when the utilization is low, the attack can be detected after a fake message insertion, but the increase in the utilization makes the attack hard to be detected, and more malicious messages can be inserted before the detection. We will also see later that when we say the attack detection can be done after a fake message with low utilization, the detection is based on $\Delta_{i,j}$ between a fake transmission and the next bonafide transmission but currently we are considering all inter-arrival times including $\Delta_{i,j}$ between a previous bonafide message and a fake message transmission which make our detection rate less than 100%.

- **Detection with smart adversary when $WC_{fi} < P/2$ (Utilization is < 50%)**: An adversary capable of sniffing the CAN bus physically or remotely and aware of the bounds described in Equation 7.5 tries to make sure fake messages go undetected. Consider an attack scenario as depicted in Figure 7.4. The adversary inserts a fake message of type m_i $P-WC_{fi}$ time units after the transmission of a bonafide instance of m_i. We assume the first bonafide message is not delayed as it would make the attack more likely to be detected, the adversary will have sufficient time to be within the bounds after inserting the fake message. We also assume WC_{fi} delay in sending the next bonafide message $f_{i,j+1}$ to make sure our detection method works in the

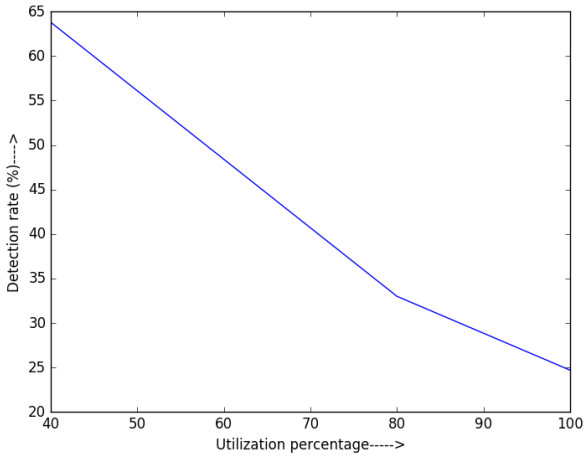


Figure 7.5: Detection Rate vs Utilization

worst-case as well. To detect the attack based on the bounds in equation 7.5, the following condition should be met as depicted in Figure 7.4.

$$P - WC_{fi} \leqslant 2 * WC_{fi} \leqslant P + WC_{fi} \tag{7.6}$$

We observe that two subsequent inter-arrival times cannot be equal to $P - WC_{fi}$ because the only way to get the inter-arrival bound as $P - WC_{fi}$ is when an instance is delayed by the worst-case delay and the next instance is not delayed. So, the next inter-arrival time should be between P and $P + WC_{fi}$ as the last instance is not delayed and the next instance can be delayed by the worst-case finish time or not delayed as shown in Figure 7.4. In other words, $P - WC_{fi}$ in Equation 7.6 can be replaced by P, and the new equation becomes:

$$P \leqslant 2 * WC_{fi} \leqslant P + WC_{fi} \tag{7.7}$$

From Equation 7.7, we can infer that our anomaly detector can raise an alarm if

$$WC_{fi} < P/2 \tag{7.8}$$

We can also infer the condition when our anomaly detector will not be able to raise the alarm:

$$P/2 \leqslant WC_{fi} \leqslant P \tag{7.9}$$

At design time, we can set the worst-case response time for m_i to be less than its period P to make sure our detector can detect the anomaly even if only one fake message is inserted.

- **Detection with smart Adversary when $P/2 \leqslant WC_{fi} \leqslant P$ (Utilization is > 50%)**: When WC_{fi} is less than P/2, the insertion of fake messages can still be detected if we are willing to tolerate N fake messages to be transmitted before the alarm. We are now interested in finding the maximum number of fake messages N that the adversary can insert before our anomaly detector triggers.

  Lemma 7.4.1. *A deterministic anomaly detector can detect fake messages injection after at most three fake messages.*

  Proof. To find N, we consider U_x to be the total number of bonafide instances of the flow m_i transmitted at any given point in time by an ECU, and U_y to be the total number of instances (bonafide + fake) of the flow m_i transmitted on the CAN bus at any given point in time. Let T be the current local clock time at our detector, which starts ticking after the first message instance of the flow m_i is received. Based on Figure 7.6, T is bounded by:

$$(U_x - 1)P - WC_{fi} \leqslant T \leqslant (U_x - 1)P + WC_{fi} \tag{7.10}$$

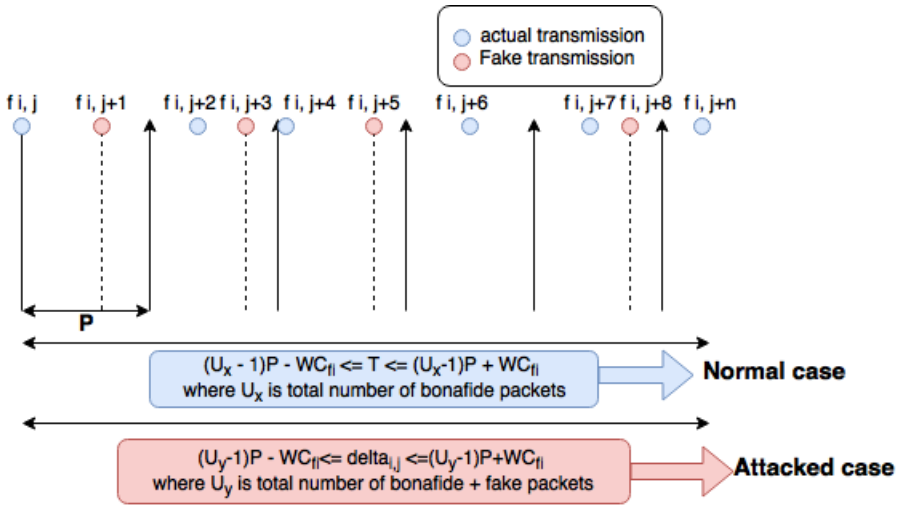


Figure 7.6: Legitimate and Fake Transmission of Instances

In an attack scenario, let the inter-arrival time between the first message instance and any other message instance of the flow m_i at any given point in time be $\delta_{i,j}$. Based on Figure 7.6, $\delta_{i,j}$ is bounded by:

$$(U_y - 1)P - WC_{fi} \leqslant \delta_{i,j} \leqslant (U_y - 1)P + WC_{fi} \tag{7.11}$$

From Equation 7.10 and 7.11, we can infer that the detector would raise an alarm if:

$$(U_y - 1)P - WC_{fi} > (U_x - 1)P + WC_{fi} \tag{7.12}$$

Combining Equation 7.9 and 7.12 with some rearranging, we can infer the following relationship:

$$U_y > U_x + 2 \tag{7.13}$$

From Equation 7.13, we infer that the attack can be detected with 100 % confidence after at-most three fake messages. □

The inter-arrival time of a message instance of the flow m_i changes due to the insertion of a fake message of the same flow. Considering the start time of the first instance of a message and all the subsequent arrivals of fake and normal instances of that message, we can detect the attack.

7.4.1.2 Detection of Suspension Attack

The effectiveness of a deterministic anomaly detection against suspension attacks follows as a corollary from the discussion of fabrication attacks.

Corollary 7.4.1.1. *The deterministic anomaly detection approach detects a suspension attack within $P + WC_{fi}$ time units.*

Proof. We assume that a compromised ECU will not send any message of the flow m_i after the suspension attack. Due to the suspension attack, the ECU is compromised and no longer allowed to send messages of the flow m_i. In such a case, the bounds are exceeded when $\Delta_{i,j} \geqslant P + WC_{fi}$. Our detector fires an alarm within $P + WC_{fi}$ time units as described in Equation 7.5. □

Figure 7.7 shows the suspension attack where $f_{i,j+3}$ is not transmitted from the ECU. The attack is detected after $P + WC_{fi}$ time units.

7.4.2 Deterministic Different Message Identifier Detector

When the anomaly detection module is located at the software layer, it only sees messages headed to and from the local ECU. The anomaly detection module thus does not see all traffic, and the techniques described in section 7.3.1 fail to work. In this section, we describe a deterministic anomaly detector that leverages local inter-message time information.

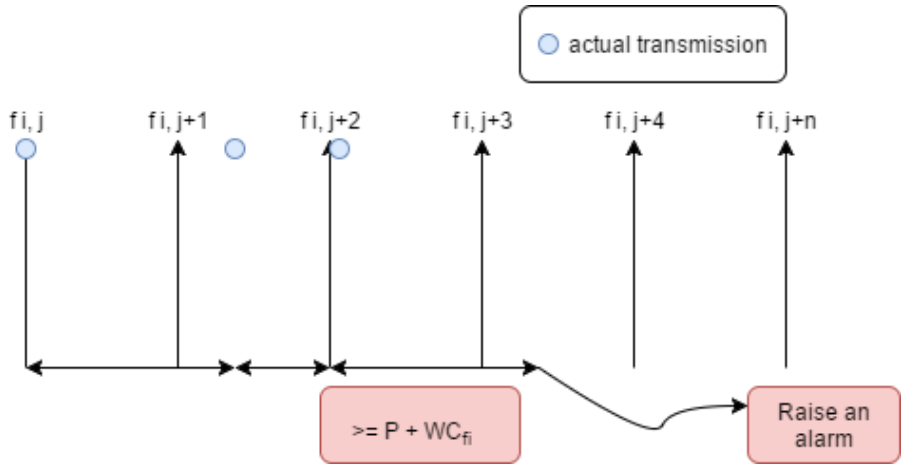


Figure 7.7: Detection Latency for Suspension Attacks

7.4.2.1 Detection of Fabrication Attack

A message instance of the flow m_i can be delayed due to the fake insertion of messages of flows($\neq m_i$). Each message instance of the flow m_i is shifted to accommodate the fake insertion of another message instance, especially if the fake messages have higher priority than m_i. In Figure 7.8, transmission of a message can be delayed by the transmission of one or more instances. To detect an attack, Equation 7.5 can be used for the bounds, but the detection is only possible in exceptional cases when the message instance of the flow m_i is delayed by WC_{fi} due to the insertion of instances of other flows.

7.5 Statistical Detector

Attacks can be detected through a statistical analysis of delay distributions. Changes in delay distributions can result through the insertion or deletion of messages. These changes can be detected by other messages even if they are not subject to the attacks.

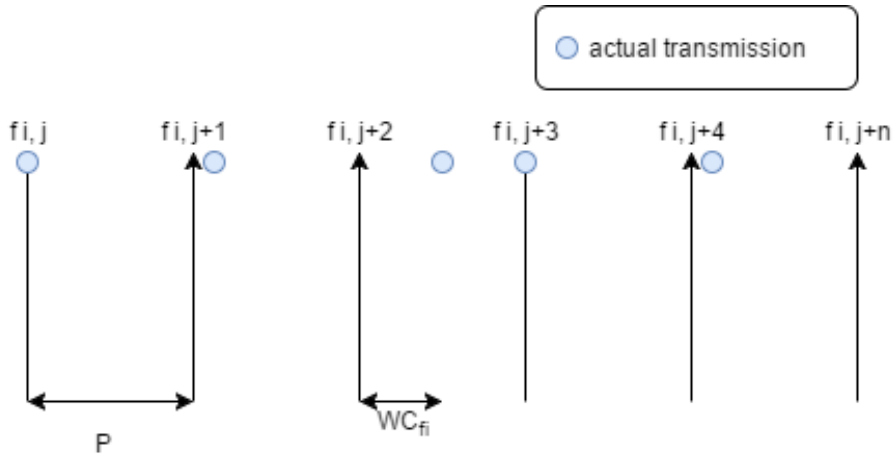


Figure 7.8: Detection of Fabrication Attacks with Different Message Identifier

7.5.1 Statistical Anomaly Detector

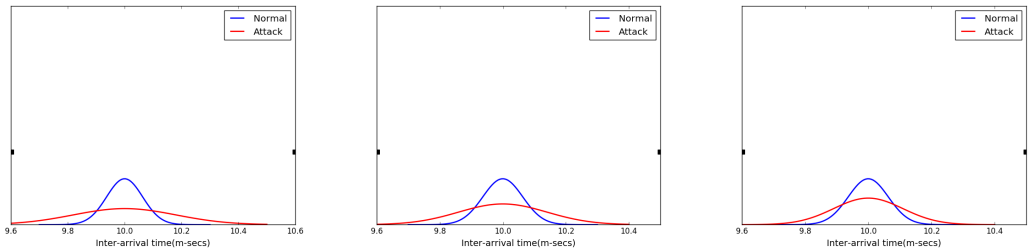
This detector can detect changes in the delay distribution of local traffic due to the insertion of instances of other traffic on the CAN bus. To evaluate the performance of such a detector, attacks were manually simulated on Honda Civic and KIA traces collected through the OBD-II port.

We simulated an attack by inserting one or two fake messages to determine if the detection is possible. All the instances or few instances of a message can be delayed due to the insertion of a message of a different identifier. Fake messages were added after every X instance of a message that is targeted to be delayed due to such insertions.

7.5.1.1 Attack on Honda Civic

To simulate an attack on a Honda Civic, a malicious message of identifier 0x91 with a transmission time of 200 μsec is inserted before a bonafide message of identifier 0x158. The impact of such insertion is measured by comparing the delay distribution of 0x158 messages for both the normal and attacked cases as shown in Figure7.9.

To calculate the accuracy for our detector, Honda Civic traces are divided into 20



(a) Delaying 33.33% Messages (b) Delaying 20% Messages (c) Delaying 10% Messages

Figure 7.9: Normal Distribution of Inter-Arrival Times for Normal vs Attacked Cases for Message with Identifier 0x158

buckets of 200 messages each. We consider a hard classifier, which takes the threshold as the maximum of 3σ in the training set to classify the testing set as anomalous or not. The accuracy for insertion of one or two malicious messages is measured using the 10-fold cross-validation as shown in Table 7.2 and Table 7.3 respectively.

Table 7.2: Efficiency Comparison Due to Insertion of One Fake Message with Identifier 0x91 before Message with Identifier 0x158

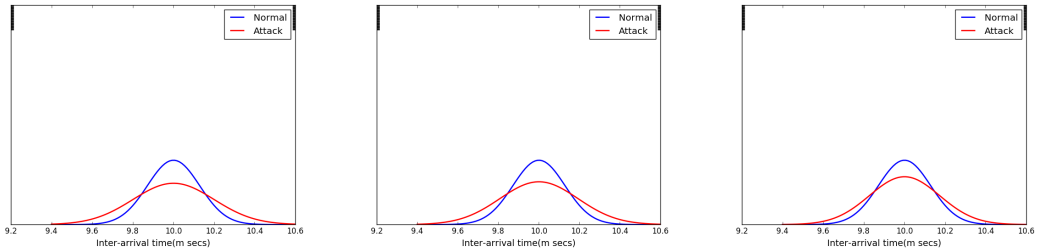
Number of Buckets	Bucket size	Instances effected	Efficiency
20	200	33.33%	100
20	200	20%	65
20	200	10%	25

7.5.1.2 Attack on KIA-1

To simulate an attack on a KIA, a malicious message of identifier 0x161 with a transmission time of 200 μsec is inserted before a bonafide message of identifier 0x357. The impact of such insertion is measured by comparing the delay distributions of 0x357 message for both the normal and attacked cases as shown in Figure7.10.

Table 7.3: Efficiency Comparison Due to Insertion of Two Fake Messages with Identifier 0x91 before Message with Identifier 0x158

Number of Buckets	Bucket size	Instances effected	Efficiency
20	200	33.33%	100
20	200	20%	100
20	200	10%	100
20	200	1%	10



(a) Delaying 25% Messages (b) Delaying 20% Messages (c) Delaying 10% Messages

Figure 7.10: Normal Distribution of Inter-Arrival Times for Normal vs Attacked Cases for Message with Identifier 0x357 for KIA-1

To calculate the accuracy for our detector, KIA traces are divided into 10 buckets of 200 messages each. We consider a hard classifier, which takes the threshold as the maximum of 3σ in the training set to classify the testing set as anomalous or not. The accuracy for insertion of one or two malicious messages is measured using the 10-fold cross-validation as shown in Table 7.4 and Table 7.5 respectively.

7.5.1.3 Attack on KIA-2

We simulated another attack on KIA traces collected at least 1 month after KIA-1 to see the effect of the clock synchronization on the detector performance. Like in KIA-1, Figure 7.11 shows the comparison of the delay distribution of normal and attack cases.

Table 7.4: Efficiency Comparison Due to Insertion of One Fake Message with Identifier 0x161 before Message with Identifier 0x357 for KIA-1

Number of Buckets	Bucket size	Instances effected	Efficiency
10	200	25%	100
10	200	20%	100
10	200	10%	60

Table 7.5: Efficiency Comparison Due to Insertion of Two Fake Messages with Identifier 0x161 before Message with Identifier 0x357 for KIA-1

Number of Buckets	Bucket size	Instances effected	Efficiency
10	200	25%	100
10	200	20%	100
20	200	10%	100
20	200	1%	10

Table 7.6 and Table 7.7 show the detection accuracy after inserting one or two malicious messages.

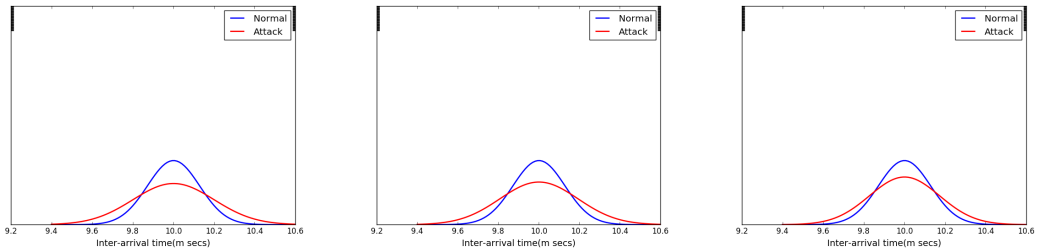
The threshold calculated using KIA-1 is used to classify the attacked KIA-2 as anomalous or not as shown in Table 7.8

7.5.1.4 Detection Latency and False Positive Rates for KIA-1 And KIA-2

We measured the detection latency, which is defined as the time taken by the detector to detect attacks on KIA-1 and KIA-2 due to the insertion of a malicious message of identifier 0x161 before a bonafide message of identifier 0x357 as shown in Table 7.9.

We calculated the false positive rates for KIA-1 and KIA-2 as shown in Table 7.10, which means the detector is necessary but not sufficient to detect attacks.

We compared the false positive rates as shown in Figure 7.12 based on the training size



(a) Delaying 25% Messages (b) Delaying 20% Messages (c) Delaying 10% Messages

Figure 7.11: Normal Distribution of Inter-Arrival Times for Normal vs Attacked Cases for Message with Identifier 0x357 for KIA-2

Table 7.6: Efficiency Comparison Due to Insertion of One Fake Message with Identifier 0x161 before Message with Identifier 0x357 for KIA-2

Number of Buckets	Bucket size	Instances effected	Efficiency
10	200	25%	100
10	200	20%	100
10	200	10%	80

and found that the increase in the training data reduces the false positive rate. It also shows that the increase in the threshold value for the classification can reduce the false positive rate to 0.

7.5.2 Self Statistical Anomaly Detector

This detector can detect the change in the delay distribution of traffic due to a malicious insertion of a message of the same traffic. To evaluate the performance of such a detector, we manually simulated an attack on a Honda Civic.

7.5.2.1 Attack on Honda Civic

The delay distribution of messages of the flow m_i will change drastically due to malicious insertion of a message of the same flow. Figure 7.13 shows the normal distribution

Table 7.7: Efficiency Comparison Due to Insertion of Two Fake Messages with Identifier 0x161 before Message with Identifier 0x357 for KIA-2

Number of Buckets	Bucket size	Instances effected	Efficiency
20	200	25%	100
20	200	20%	100
20	200	10%	100
20	200	1%	10

Table 7.8: Efficiency Comparison Due to Insertion of One Fake Message with Identifier 0x161 before Message with Identifier 0x357 Using Threshold Calculated from KIA-1

Number of Buckets	Bucket size	Instances effected	Efficiency
10	200	25%	100
10	200	20%	100
10	200	10%	60

of messages with identifier 0x91 after a fake insertion of a message with identifier 0x91.

To calculate the accuracy for our detector, we divide CAN traces into 10 buckets of 10 messages each and see the value of $\mu \pm \sigma$ for both the normal and attacked cases. Attacks can easily be detected as there is a considerable difference in $\mu \pm \sigma$ value between normal and attacked cases based on the data in Table 7.11.

The semantics of the protocol are not needed to detect an attack using deterministic and statistical detectors, but such a detection scheme does not work for non-periodic traffic. The fake insertion can be detected after one message or after at most three fake messages depending on the bus utilization for the deterministic detector. The detection of the attack is possible with good accuracy and low false positive rates using the statistical detection but with higher detection latency.

Table 7.9: Detection Latency for KIA-1 and KIA-2

Model	Instances effected	Detection latency
KIA-1	25%	2 secs
KIA-1	20%	2 secs
KIA-2	25%	2 secs
KIA-2	20%	2 secs

Table 7.10: False Positive for KIA-1 and KIA-2

Model	False Positive
KIA-1	.3%
KIA-2	1.8%
KIA-1 classifier on KIA-2	0%
KIA-2 classifier on KIA-1	3.5%

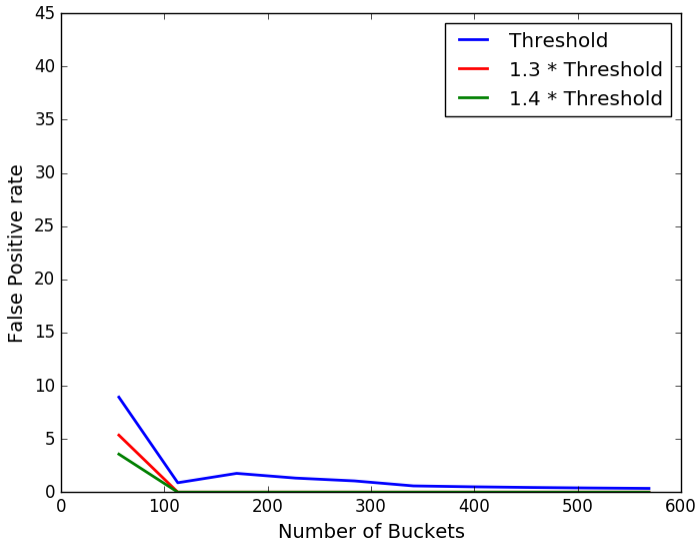


Figure 7.12: False Positive Rate vs Training Samples for KIA-1

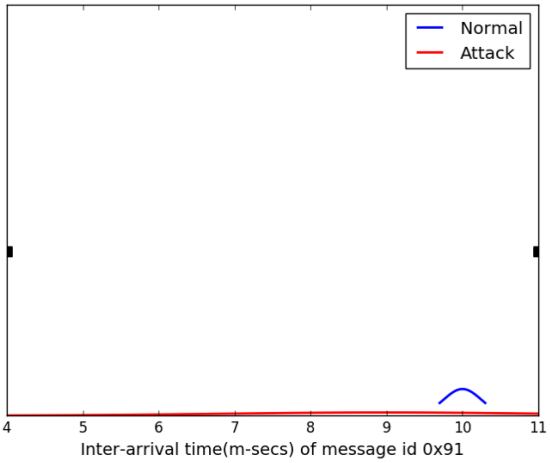


Figure 7.13: Normal Distribution of Inter-Arrival Times for Normal vs Attacked Cases for Message with Identifier 0x91

Table 7.11: Comparison of Mean and Sigma for Normal and Attacked Cases

Bucket Number	Bucket size	Instances effected	$\mu \pm \sigma$ (Normal)	$\mu \pm \sigma$ (Attack)
1	10	10%	10.0 ± 0.24	8.97 ± 2.08
2	10	10%	10.0 ± 0.22	9.02 ± 1.98
3	10	10%	9.99 ± 0.24	8.97 ± 1.96
4	10	10%	10.0 ± 0.22	9.02 ± 1.98
5	10	10%	9.99 ± 0.24	8.97 ± 1.96
6	10	10%	10.0 ± 0.24	9.02 ± 1.99
7	10	10%	10.0 ± 0.22	8.97 ± 1.96
8	10	10%	9.99 ± 0.23	9.02 ± 1.98
9	10	10%	9.99 ± 0.24	8.97 ± 1.96
10	10	10%	10.0 ± 0.23	9.02 ± 1.99

8. CONCLUSIONS

Three anomaly detection techniques are discussed to detect fabrication and suspension attacks in a CAN. The cross-correlation of fields like wheel speeds, vehicle speed, acceleration and RPM at a given time can be used to detect an anomaly, but not all the fields are strongly correlated. This technique works for the periodic and non-periodic traffic.

The order of messages transmitted from an ECU can be used for the anomaly detection. CAN messages from the ECU should always be seen in a specific order as they will be transmitted one after the other based on the priorities of messages, and any deviation from this order can be flagged. This detector is simple and fast to detect attacks as any change in the order is immediately flagged. The insertion of one malicious message and the ECU transmitting at least two messages are needed to detect any intrusion.

A timing-based detector can be used to determine the impact of attacks on messages of a same or different identifier than attacked messages. Detection on the same message identifier can be possible after one malicious message if $WC_{fi} < P/2$ or after at most three malicious messages if $P/2 \leqslant WC_{fi} \leqslant P$ using a deterministic detection. Statistical detection techniques can detect the impact on the delay distribution over time on messages of a same or different identifier than attacked messages. The detection of an attack is possible with good accuracy and low false positive rates using the statistical detection but with a compromise in the detection latency. This scheme works for the periodic traffic, which includes most of the messages on the CAN bus.

REFERENCES

[1] J. Haartsen, M. Naghshineh, J. Inouye, O. J. Joeressen, and W. Allen, "Bluetooth: Vision, goals, and architecture," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 2, pp. 38–45, Oct. 1998.

[2] M. S. Gast, *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, Inc., 2005.

[3] G. Miao, J. Zander, K. W. Sung, and S. Ben Slimane, *Fundamentals of Mobile Data Networks*. Cambridge University Press, 2016.

[4] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer Publishing Company, Incorporated, 2012.

[5] S. H. Seo, J. H. Kim, S. Hwang, K. H. Kwon, and J. W. Jeon, "An evaluation of the flexray-can gateway-embedded system in the hev test bench," in *2009 IEEE International Symposium on Industrial Electronics*, pp. 664–669, July 2009.

[6] M. Ruff, "Evolution of local interconnect network (lin) solutions," *Vehicular Technology Conference*, vol. 5, pp. 3382 – 3389, Oct. 2003.

[7] M. Muter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*, pp. 92–98, Aug 2010.

[8] A. Theissle, "Anomaly detection in recordings from in-vehicle networks," First International Workshop, BIGDAP, 2014.

[9] S. N. Narayanan, S. Mittal, and A. Joshi, "Using data analytics to detect anomalous states in vehicles," *CoRR*, vol. abs/1512.08048, 2015.

[10] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 911–927, USENIX Association, 2016.

[11] C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units," *DEFCON*, vol. 21, pp. 260–264, 2013.

[12] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," 2015.

[13] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, (Washington, D.C.), USENIX Association, 2015.

[14] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, SEC'11, (Berkeley, CA, USA), pp. 6–6, USENIX Association, 2011.

[15] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, pp. 447–462, May 2010.

[16] TU-CRRC, "Raw data for the 2012 civic tests," 2012.