Fingerprinting Electronic Control Units for Vehicle Intrusion Detection

Kyong-Tak Cho and Kang G. Shin, ***University of Michigan***

https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho

This paper is included in the Proceedings of the 25th USENIX Security Symposium

August 10–12, 2016 • Austin, TX

ISBN 978-1-931971-32-4

Open access to the Proceedings of the 25th USENIX Security Symposium is sponsored by USENIX

Fingerprinting Electronic Control Units for Vehicle Intrusion Detection

Kyong-Tak Cho and Kang G. Shin
The University of Michigan
{*ktcho, kgshin*}*@umich.edu*

Abstract

As more software modules and external interfaces are getting added on vehicles, new attacks and vulnerabilities are emerging. Researchers have demonstrated how to compromise in-vehicle Electronic Control Units (ECUs) and control the vehicle maneuver. To counter these vulnerabilities, various types of defense mechanisms have been proposed, but they have not been able to meet the need of strong protection for safety-critical ECUs against in-vehicle network attacks. To mitigate this deficiency, we propose an anomaly-based intrusion detection system (IDS), called *Clock-based IDS* (CIDS). It measures and then exploits the intervals of periodic in-vehicle messages for fingerprinting ECUs. The thus-derived fingerprints are then used for constructing a baseline of ECUs' clock behaviors with the Recursive Least Squares (RLS) algorithm. Based on this baseline, CIDS uses Cumulative Sum (CUSUM) to detect any abnormal shifts in the identification errors — a clear sign of intrusion. This allows quick identification of in-vehicle network intrusions with a low false-positive rate of 0.055%. Unlike state-of-the-art IDSs, if an attack is detected, CIDS's fingerprinting of ECUs also facilitates a root-cause analysis; identifying which ECU mounted the attack. Our experiments on a CAN bus prototype and on real vehicles have shown CIDS to be able to detect a wide range of in-vehicle network attacks.

1 Introduction

Security has now become an important and real concern to connected and/or automated vehicles. The authors of [9] systematically analyzed different attack vectors in vehicles (e.g., Bluetooth, Cellular), and showed that in-vehicle Electronic Control Units (ECUs) can be compromised for remote attacks. Through a compromised ECU, the adversary can control the vehicle by injecting packets in the in-vehicle network [20, 23]. Researchers have also been able to compromise and remotely stop a Jeep Cherokee running on a highway [7, 25], which triggered a recall of 1.4 million vehicles. Such a reality of vehicle attacks has made automotive security one of the most critical issues.

As a countermeasure against such attacks on in-vehicle networks, two main lines of defense have been pursued: *message authentication* and *intrusion detection*. Although message authentication provides a certain level of security and is shown to be efficient for Internet security, its adoption in in-vehicle networks is hindered by (i) the limited space available for appending a Message Authentication Code (MAC) in in-vehicle messages and (ii) its requirements of real-time processing and communication.

Various types of Intrusion Detection Systems (IDS) have been proposed [16, 23, 30, 31]. The essence of state-of-the-art IDSs is to monitor the contents and the periodicity of in-vehicle messages and verify whether there are any significant changes in them. Since they are either constant or predictable in in-vehicle networks, such approaches can be feasible in most circumstances. However, there still remain critical attacks which existing IDSs can neither detect nor prevent, for two main reasons: 1) in-vehicle messages do not carry information on their transmitters, and thus one cannot tell whether they originate from genuine transmitters; and 2) lack of the transmitters' information makes it very difficult or impossible for state-of-the-art IDSs to identify which ECU has mounted an attack.

To overcome these limitations and defend against various vehicle attacks, we propose a new anomaly-based IDS, called *Clock-based IDS* (CIDS). The need of CIDS for vehicles is motivated through an analysis of three representative in-vehicle network attacks — fabrication, suspension, and masquerade attacks. Our analysis shows that state-of-the-art IDSs are insufficient, especially in detecting the masquerade attack due to the absence of the transmitters' information in messages. CIDS over-

comes these limitations of existing IDSs by fingerprinting in-vehicle ECUs. Researchers have proposed various schemes for fingerprinting network devices by estimating their clock skews through the timestamps carried in their control packet headers [17, 19, 34, 42]. However, since such embedded timestamps are not available for in-vehicle networks making them inapplicable, CIDS fingerprints in-vehicle ECUs in a very different way.

CIDS monitors the intervals of (commonly seen) periodic in-vehicle messages, and then exploits them to estimate the clock skews of their transmitters which are then used to fingerprint the transmitters. That is, instead of assuming or requiring timestamps to be carried in messages for fingerprinting, CIDS exploits the periodic feature (seen at receivers) of in-vehicle network messages for fingerprinting transmitter ECUs. This makes CIDS invulnerable to attackers who use faked timestamps and thus clock skews — a problem that timestamp-based fingerprinting schemes cannot handle. Based on the thus-obtained fingerprints, CIDS constructs a norm model of ECUs' clock behaviors using the Recursive Least Squares (RLS) algorithm and detects intrusions with a Cumulative Sum (CUSUM) analysis. This enables CIDS to detect not only attacks that have already been demonstrated or discussed in literature, but also those that are more acute and cannot be detected by state-of-the-art IDSs. Our experimental evaluations show that CIDS detects various types of in-vehicle network intrusions with a low false-positive rate of 0.055%. Unlike state-of-the-art IDSs, if an intrusion is detected in CIDS, its fingerprinting capability facilitates identification of the (compromised) ECU that mounted the attack. We validate these capabilities of CIDS through experimental evaluations on a CAN bus prototype and on real vehicles.

We focus on building CIDS for Control Area Network (CAN), which is the *de facto* standard in-vehicle network. Its applicability to other in-vehicle network protocols is also discussed in Section 6. Considering the ubiquity of CAN and its direct relationship with the drivers/passengers' safety, it is critically important to build as capable a CAN bus IDS as possible.

This paper makes the following contributions:

- Development of a novel scheme of fingerprinting ECUs by exploiting message periodicity;
- Proposal of CIDS, which models the norm behavior of in-vehicle ECUs' clocks based on fingerprints and then detects in-vehicle network intrusions;
- Implementation and validation of CIDS on a CAN bus prototype as well as on 3 real vehicles.

The rest of the paper is organized as follows. Section 2 provides the necessary background of CAN and IDS-related work, and Section 3 details the attack model we consider. Section 4 details the design of CIDS, which is evaluated in Section 5 on a CAN bus prototype as well as on three real vehicles. Section 6 discusses CIDS further, such as its overhead and extension to emerging in-vehicle networks. Finally, we conclude the paper in Section 7.

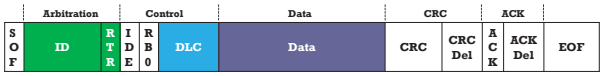


Figure 1: Format of a CAN data frame.

2 Background

For completeness, we first provide necessary background on the CAN protocol, and then discuss the related work on security solutions for in-vehicle networks.

2.1 Primer on CAN Protocol

CAN frame. CAN is the most widely deployed in-vehicle communication protocol, which interconnects ECUs/nodes through a multi-master, message broadcast bus system [4]. To maintain data consistency and make control decisions, data is exchanged between ECUs via CAN frames, the format of which is shown in Fig. 1. A CAN frame contains fields such as ID, Data Length Code (DLC), Data, and CRC. Since CAN is message-oriented, instead of containing the transmitter/receiver address, a CAN frame contains a unique ID which represents its priority and meaning. For example, a frame with ID=0x20 may contain wheel speed values whereas a frame with ID=0x55 may contain temperature values.

Arbitration. Once the CAN bus is detected idle, nodes with buffered messages to transmit, attempt to access the bus. Multiple nodes could attempt to access the bus simultaneously, i.e., contention occurs for access. Such a contention is resolved via bus arbitration as follows. Each node first transmits the ID value of its CAN frame one bit at a time, starting with the most significant bit. Since CAN is designed to logically behave as a wired-AND gate, some contending nodes see an output of 0 from the bus, although they had transmitted 1. Such nodes withdraw from bus contention and switch to the receive mode. As a result, among the contending nodes, the ECU sending the message with the lowest ID value wins arbitration, and gains exclusive access for message transmission. Those which have lost arbitration re-attempt to transmit once the bus becomes idle again.

Synchronization. For proper bitwise message transmission and reception, hard and soft *bit* synchronizations are achieved, respectively, by using the Start-of-Frame (SOF) signal and bit stuffing in CAN frames [4]. Al-

though these provide alignment of bit edges for message exchange, they do not synchronize the *clocks* of ECUs, i.e., CAN lacks clock synchronization. Thus, since time instants for ECUs are provided by their own quartz crystal clocks, these clocks, in reality, run at different frequencies, resulting in random drifting of clocks: a drift of 2400*ms* over a period of 24 hours is possible [27].

2.2 Related Work

To defend against various types of vehicle cyber attacks, there have been two main streams of security solutions: *message authentication* and *intrusion detection*.

Message authentication. In the area of Internet security, cryptographic message authentication provides strong protection against forgery. Thus, researchers have attempted to borrow such approaches from the domain of Internet security to address in-vehicle network security problems. However, since the maximum payload length allowed in the CAN data field is only 8 bytes, the available space for appending a cryptographically secure Message Authentication Code (MAC) is very limited, i.e., the protocol specification limits its maximum cryptographic strength. To overcome this difficulty, rather than appending a MAC in one CAN frame's data field, the authors of [38] proposed to truncate it across multiple frames. Instead of the data field, the authors of [33] proposed to use multiple CRC fields to include 64 bits of CBC-MAC. The authors of [15] suggested to exploit an out-of-band channel for message authentication.

Although such preventive measures provide some degree of security, they alone cannot guarantee complete security due to their inability to handle certain critical attacks, e.g., Denial-of-Service (DoS). Moreover, their operations not only require a significant amount of processing power but also increase message latencies and bus utilization. Since in-vehicle networks must operate in real time and ECUs are resource-limited for cost reasons, unlike in the Internet, these "costs" of preventive measures hinder their adoption [30]. More importantly, when an adversary has full access to any data stored in RAM and/or FLASH, including data used for implementing security mechanisms (e.g., shared secret keys), some cryptographic solutions become incapable [24].

Intrusion detection. To overcome such limitations of preventive measures, different Intrusion Detection Systems (IDSs) have been proposed. Some state-of-the-art IDSs exploit the fact that most CAN messages are periodic, i.e., sent at fixed time intervals. The authors of [30] proposed an IDS which monitors the intervals of periodic messages, measures their entropies, and exploits them for intrusion detection. Similarly, a method of modeling the distribution of message intervals, and utilizing it for intrusion detection was proposed in [23]. In addition to message frequency, researchers also proposed to verify the message contents. The authors of [31] exploited in-vehicle sensors to verify message range, correlation, etc. Abnormal measurements on brake-related sensors were detected by using the tire-friction model [10].

Although existing IDSs are capable of detecting most attacks through the above approaches, they fail to cover some critical attacks which are more acute, and thus are not sufficient to provide security. We will elaborate on such shortcomings of state-of-the-art IDSs while analyzing the attack scenarios under consideration in Section 3.3.

3 Attack Model

We first discuss the adversary model under consideration, and then the three representative attack scenarios.

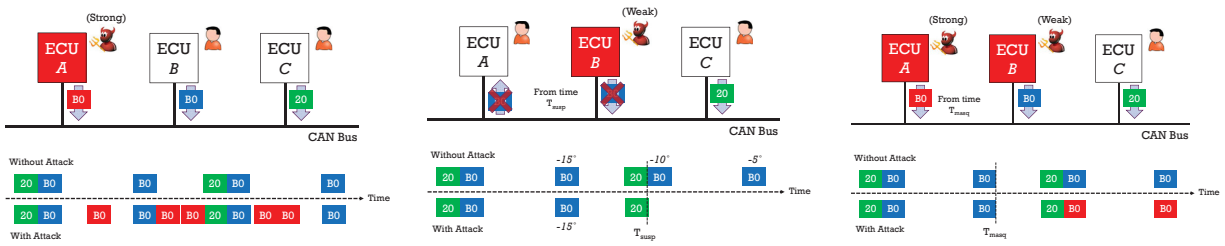
3.1 Adversary Model

Adversaries can physically/remotely compromise more than one in-vehicle ECU via numerous attack surfaces and means [9]. We consider an adversary who wants to manipulate or impair in-vehicle functions. The adversary can achieve this by either injecting arbitrary messages with a spoofed ID into the in-vehicle network, which we refer to as *attack messages*, or by stopping/suspending message transmissions of the compromised ECU.

Strong and weak attackers. Depending on their hardware, software, and attack surfaces, ECUs of different vehicles have different degrees of vulnerabilities, thus providing attackers different capabilities. So, we consider two different types of compromised ECUs: *fully* and *weakly* compromised ECUs.

Through a *weakly* compromised ECU, the attacker is assumed to be able to stop/suspend the ECU from transmitting certain messages or keep the ECU in listen-only mode, but cannot inject any fabricated messages. We call such an attacker with limited capabilities a *weak attacker*, and will use this term interchangeably with "weakly compromised ECU".

In contrast, with a *fully* compromised ECU, the attacker is assumed to have full control of it and access to memory data. Thus, in addition to what a weak attacker can do, the attacker controlling a fully compromised ECU can mount attacks by injecting arbitrary attack messages. We call such an attacker with more attack capabilities a *strong attacker*, and will use this term interchangeably with a "fully compromised ECU". Even when preventive security mechanisms (e.g., MAC) are built into the ECUs, since the strong attacker has full access to any data stored in their memory, including data used for implementing security mechanisms (e.g., shared secret keys), it can disable them [24]. On the other hand,



(a) Fabrication attack. (b) Suspension attack. (c) Masquerade attack.

Figure 2: Three representative attack scenarios on in-vehicle networks.

a weak attacker can only stop, or listen to message transmissions, but cannot start a new one.

Foster *et al.* [13] have recently proved the possible existence of these two types of attackers in in-vehicle networks. They have shown that the firmware versions of telematics units can affect/limit the attacker's capabilities in injecting and monitoring in-vehicle network messages. Specifically, for a certain firmware version of the telematics unit, an attacker having control of that ECU was shown to be able to receive CAN messages but unable to send the messages. On the other hand, for some other firmware versions, the attacker was capable of both sending and receiving CAN messages to and from the in-vehicle network. In other words, the firmware version of an ECU determines which type of an attacker — strong or weak — it can become, if compromised.

To further comprehend how and why these two different types of attackers can exist, let's consider one of the most common CAN controllers, Microchip MCP2515 [1]. For ECUs with such a controller, various operation modes like configuration, normal, and listen-only can be selected by user instructions through the Serial Peripheral Interface (SPI). Thus, user-level features for configuring the CAN controller allow attackers to easily enter different modes (e.g., listen-only mode for a weak attacker). In contrast, there are no such features allowing attackers to easily inject forged messages. In other words, the specification of the ECU hardware/software, if compromised, can *restrict* the adversary to become a weak attacker only. Note that the required functionalities of a strong attacker subsume those of a weak attacker. It is thus easier for an adversary to become a weak attacker than a strong attacker, let alone researchers have already demonstrated how to create such a strong attacker [9, 20, 23, 24].

3.2 Attack Scenarios

Based on the adversary model discussed so far, we consider the following attack scenarios that can severely impair in-vehicle functions: *fabrication*, *suspension*, and *masquerade*.[1]

Fabrication attack. Through an in-vehicle ECU compromised to be a strong attacker, the adversary fabricates and injects messages with forged ID, DLC, and data. The objective of this attack is to override any periodic messages sent by a legitimate safety-critical ECU so that their receiver ECUs get distracted or become inoperable. For example, as shown in Fig. 2(a), the strong attacker $\mathbb{A}$ injects several attack messages with ID=0xB0, which is usually sent by a legitimate ECU $\mathbb{B}$, at a high frequency. Thus, other nodes which normally receive message 0xB0 are forced to receive the fabricated attack messages more often than the legitimate ones. We refer to such a case as $\mathbb{A}$ mounting a fabrication attack on message 0xB0 or its genuine transmitter $\mathbb{B}$. Demonstrated attacks such as controlling vehicle maneuver [20] and monopolizing the CAN bus with highest priority messages [16] exemplify a fabrication attack.

Suspension attack. To mount a suspension attack, the adversary needs only one weakly compromised ECU, i.e., become a weak attacker. As one type of Denial-of-Service (DoS) attack, the objective of this attack is to stop/suspend the weakly compromised ECU's message transmissions, thus preventing the delivery/propagation of information it acquired, to other ECUs. For some ECUs, they must receive certain information from other ECUs to function properly. Therefore, the suspension attack can harm not only the (weakly) compromised ECU itself but also other receiver ECUs. An example of this attack is shown in Fig. 2(b) where the weak attacker having control of the Electric Power Steering ECU $\mathbb{B}$ stops transmitting its measured steering wheel angle value. So, the Electronic Stability Control (ESC) ECU $\mathbb{A}$, which requires the steering wheel angle value from $\mathbb{B}$ for detecting and reducing the loss of traction, no longer receives its updates and thus malfunctions.

Masquerade attack. To mount a masquerade attack, the adversary needs to compromise two ECUs, one as a strong attacker and the other as a weak attacker. The

[1] In this paper, we focus on only these three attack scenarios and do not consider others as they may be less feasible or harmful, or be detectable by existing IDSs.

objective of this attack is to manipulate an ECU, while shielding the fact that an ECU is compromised. Fig. 2(c) shows an example where the adversary controls a strong attacker $\mathbb{A}$ and a weak attacker $\mathbb{B}$. Until time T_{masq}, the adversary monitors and learns which messages are sent at what frequency by its weaker attacker, e.g., $\mathbb{B}$ sends message 0xB0 every 20*ms*. Since most in-vehicle network messages are periodic and broadcast over CAN, it is easy to learn their IDs and intervals. Once it has learned the ID and frequency of a message, at time T_{masq}, the adversary stops the transmission of its weak attacker and utilizes its strong attacker $\mathbb{A}$ to fabricate and inject attack messages with ID=0xB0. Stopping $\mathbb{B}$'s transmission and exploiting $\mathbb{A}$ for transmission of attack messages are to overcome the weak attacker's inability of injecting messages. After T_{masq}, the original transmitter of 0xB0, $\mathbb{B}$, does not send that message any longer, whereas $\mathbb{A}$ sends it *instead* at its original frequency. So, when the CAN bus traffic is observed, the frequency of message 0xB0 remains the same, whereas its transmitter has changed. We refer to such a case as $\mathbb{A}$ mounting a masquerade attack on message 0xB0 or its original transmitter $\mathbb{B}$.

In fact, in order to attack and remotely stop a Jeep Cherokee running on a highway, Miller *et al.* [25] had to control its ABS collision prevention system by mounting a *masquerade* (not fabrication) attack. In contrast to other vehicles which they had previously examined (e.g., Toyota Prius), the Jeep Cherokee's brake was not controllable via the fabrication attack as its ABS collision prevention system, which was the attack vector for engaging brakes, was switched off when the fabrication attack was mounted. On the other hand, when mounting the masquerade attack, the system was not switched off, thus allowing them to control the Jeep Cherokee's braking maneuver.

Using masquerade attacks, the adversary can not only inject attack messages from the compromised/impersonating ECU but also cause other severe problems, significantly degrading the in-vehicle network performance. The impersonating ECU sending a message instead of another ECU implies that it would generate more messages to periodically transmit than before, making its transmit buffer more likely overloaded. This may, in turn, lead to severe consequences, such as non-abortable transmission requests [12], deadline violation [18], and significant priority inversion [32]. Moreover, the original sequence of messages may also change, thus failing to meet the requirement of some in-vehicle messages to be sent sequentially in a correct order for proper vehicle operations. These network problems from a masquerade attack occur while not deviating much from the norm network behavior (e.g., message frequency remains the same). This is in sharp contrast to the cases of mounting a fabrication attack or a suspension attack, which may also incur similar problems. Such problems have been identified to be critical since they degrade the real-time performance of CAN significantly, and thus undermine vehicle safety [12, 18, 32]. The masquerade attack can thus cause more problems to the in-vehicle network than just injecting attack messages.

3.3 Defense Against the Attacks

When the fabrication or suspension attack is mounted, the frequency of certain messages significantly and abnormally increases or decreases, respectively. Thus, if state-of-the-art IDSs [16, 23, 30, 31], which monitor the message frequencies, were to be used, the attacks can be detected.

When mounting the masquerade attack, however, the adversary does not change the original frequency of messages. Thus, the adversary may use this attack to evade state-of-the-art IDSs. Moreover, if the adversary does not change the content of messages as well, it can behave like a legitimate ECU. However, the adversary may later mount other types of attacks (e.g., a fabrication attack) through the impersonating ECU. Hence, defending against the masquerade attack implies not only detecting the attack reactively, but also preventing other attacks *proactively*.

4 Clock-Based Detection

Although state-of-the-art IDSs are capable of detecting some basic attacks such as fabrication attack and suspension attack, they fail to detect more sophisticated ones such as the masquerade attack for the following reasons.

- *No authenticity* — CAN messages lack information on their transmitters. So, existing IDSs do not know whether or not the messages on the CAN bus were sent by the genuine transmitter, and hence cannot detect any changes of the message transmitter.
- *Inability of identifying a compromised ECU* — Lack of the transmitter's information makes it very difficult or impossible for state-of-the-art IDSs to identify which of compromised ECUs mounted an attack.

If CAN frames do not carry any information on their transmitters, how could an IDS identify them and detect intrusions such as the masquerade attacks? Which behavior of CAN should the IDS model for detection of such intrusions? We answer these questions by developing a novel IDS, `CIDS`, which exploits message frequency to fingerprint the transmitter ECUs, and models a norm behavior based on their fingerprints for intrusion

detection. We focus on detecting intrusions in *periodic* messages as most in-vehicle messages are sent periodically [32, 36].

4.1 Fingerprinting ECUs

For each in-vehicle ECU in CAN, the time instants of periodic message transmissions are determined by its quartz crystal clock [27]. We follow the nomenclature of clocks of the NTP specification [26] and Paxson [35]. Let $\mathbb{C}_{true}$ be a "true" clock which reports the true time at any moment and $\mathbb{C}_i$ be some other non-true clock. We define the terms "clock offset, frequency, and skew" as follows.

- **offset:** difference in the time reported by clock $\mathbb{C}_i$ and the true clock $\mathbb{C}_{true}$. We define *relative offset* as the offset between two non-true clocks.
- **frequency:** the rate at which clock $\mathbb{C}_i$ advances. Thus, the frequency at time t is $\mathbb{C}_i'(t) \equiv d\mathbb{C}_i(t)/dt$.
- **skew:** difference between the frequencies of clock $\mathbb{C}_i$ and the true clock $\mathbb{C}_{true}$. We define *relative skew* as the difference in skews of two non-true clocks.

If two clocks have relative offset and skew of 0, then they are said to be *synchronized*. Otherwise, we consider they are *unsynchronized*. Since CAN lacks clock synchronization, it is considered to be unsynchronized.

Clock skew as a fingerprint. The clock offsets and skews of unsynchronized nodes depend solely on their local clocks, thus being distinct from others. As others have also concluded [17, 19, 42], clock skews and offsets can therefore be considered as fingerprints of nodes. Various studies have exploited this fact to fingerprint physical devices [17, 19, 34, 42]. However, they are not applicable to our problem as they exclusively rely on the timestamps carried in the packet headers, which are, as discussed before, not available in in-vehicle networks. Kohno *et al.* [19] considered an alternative to embedded timestamps: using Fourier Transform for clock skew estimation. However, as their approach relies on the unique characteristics of the Internet (e.g., multi-hop delays, large network topology), it cannot be directly applied to in-vehicle networks.

To build an effective IDS, which can detect various types of attack including the masquerade attack, it should be capable of verifying the transmitter of each message. However, since such information is not present in CAN messages, we must fingerprint ECUs with other "leaked" information. Unlike the existing approaches that exploit embedded timestamps, we exploit *message periodicity* to extract and estimate the transmitters' clock skews, which are then used to fingerprint the transmitter ECUs.

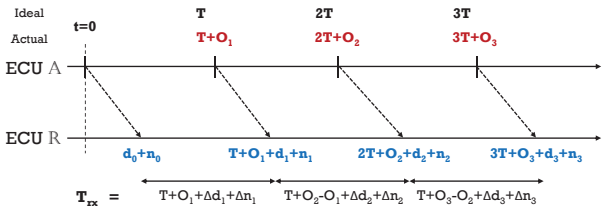


Figure 3: Timing analysis of message arrivals.

Clock skew estimation. Consider an ECU $\mathbb{A}$ which broadcasts a message every T *ms* and an ECU $\mathbb{R}$ which periodically receives that message. From the perspective of $\mathbb{R}$, since only its timestamp is available, we consider its clock as the true clock. As shown in Fig. 3, due to the clock skew, periodic messages are sent at times with small offsets from the ideal values (e.g., T, $2T$, $3T$, $\cdots$). Let $t = 0$ be the time when the first message was sent from $\mathbb{A}$, and O_i be the clock offset of $\mathbb{A}$ when it sends the i-th message since $t = 0$. Then, after a network delay of d_i, ECU $\mathbb{R}$ would receive that message and put an arrival timestamp of $iT + O_i + d_i + n_i$, where n_i denotes the noise in $\mathbb{R}$'s timestamp quantization [42]. Thus, the intervals between each arrival timestamp, $T_{rx,i} = T + \Delta O_i + \Delta d_i + \Delta n_i$, where ΔX_i denotes the difference of X between step i and $i-1$, and $O_0 = 0$. Since the change in O_i within one time step is negligible and n_i is a zero-mean Gaussian noise term [2], the expected value of the timestamp intervals, $\mu_{T_{rx}} = \mathrm{E}[T_{rx,i}]$, can be expressed as:

$$\begin{aligned} \mu_{T_{rx}} &= \mathrm{E}[T + \Delta O_i + \Delta d_i + \Delta n_i] \\ &= T + \mathrm{E}[\Delta O_i + \Delta d_i + \Delta n_i] \\ &\approx T, \end{aligned} \tag{1}$$

where the second equality holds since T is a predetermined constant. Since the data lengths of CAN periodic messages, i.e., DLCs, are constant over time, for now, we consider $\mathrm{E}[\Delta d_i] = 0$. Later in Section 4.4, we will discuss the case when d_i is not constant, and how it may affect the performance of CIDS.

Based on the arrival timestamp of the first message, $d_0 + n_0$, and the average of timestamp intervals, $\mu_{T_{rx}}$, we extrapolate and determine the *estimated* arrival time of the i-th message as $i\mu_{T_{rx}} + d_0 + n_0$, whereas the actual *measured* arrival time is $iT + O_i + d_i + n_i$. As we are estimating subsequent arrival times, $\mu_{T_{rx}}$ is determined by past measurements. Since T is constant over time and thus again $\mu_{T_{rx}} \approx T$, the average difference between the estimated and measured times is:

$$\mathrm{E}[\mathbb{D}] = \mathrm{E}[i(T - \mu_{T_{rx}}) + O_i + \Delta d + \Delta n] \approx \mathrm{E}[O_i]. \tag{2}$$

That is, from message periodicity, we can estimate the *average clock offset*, $\mathrm{E}[O_i]$, which will indeed be distinct for different transmitters. Since clock offset is

Algorithm 1 Clock skew estimation with RLS

```
1:  Initialize: S[0] = 0, P[0] = δI
2:  function SkewUpdate(t, e)                          ▷ RLS algorithm
3:      G[k] ← λ^{-1}P[k-1]t[k] / (1+λ^{-1}t^2[k]P[k-1])
4:      P[k] ← λ^{-1}(P[k-1] − G[k]t[k]P[k-1])
5:      return S[k] ← S[k-1] + G[k]e[k]
6:  end function
7:  for k^th step do
8:      a_0 ← arrival timestamp of most recently rxed message
9:      n ← 1
10:     while n ≤ N do
11:         if current time ≫ a_{n-1} then
12:             /* No longer receives the message */
13:             a_n, ··· , a_N ← significantly high values
14:             T_n, ··· , T_N ← significantly high values
15:             break
16:         else
17:             a_n ← arrival timestamp of n^th message
18:             T_n ← a_n − a_{n-1}                    ▷ Timestamp interval
19:             n ← n + 1
20:         end if
21:     end while
22:     μ_T[k] ← (1/N) Σ_{i=1}^{N} T_i                 ▷ Avg. timestamp interval
23:     O[k] ← (1/(N-1)) Σ_{i=2}^{N} a_i − (a_1 + (i-1)μ_T[k-1])
24:     O_acc[k] ← O_acc[k-1] + |O[k]|                 ▷ Accumulated offset
25:     e[k] ← O_acc[k] − S[k-1]t[k]                   ▷ Identification error
26:     S[k] ← SkewUpdate(t, e)                        ▷ Clock skew
27: end for
```

slowly varying and non-zero [17, 42], $E[O_i] \neq 0$, whereas $E[\Delta O_i] = 0$.

If ECU $\mathbb{R}$ were to determine the average clock offset for every N received messages, since it is derived in reference to the first message (of N messages), it represents only the average of *newly* incurred offsets. Thus, to obtain the total amount of incurred offset, which we call the *accumulated clock offset*, the absolute values of the average clock offsets have to be summed up. By definition, the slope of the accumulated clock offset would thus represent the clock skew, which is constant as we will show and as others have also concluded [19, 29, 35, 40]. This enables CIDS to estimate the clock skew from arrival timestamps and thus fingerprint the message transmitter for intrusion detection. We will later show, via experimental evaluations on a CAN bus prototype and on 3 real vehicles, that the thus-derived clock skew is indeed a fingerprint of an in-vehicle ECU.

4.2 CIDS — Per-message Detection

By determining the clock skew from observation of message intervals, transmitter ECUs can be fingerprinted. We exploit this in designing CIDS, a clock-based IDS for in-vehicle networks which detects intrusions in two different ways: *per-message* detection and *message-pairwise* detection, where the latter supplements the former in reducing false positive/negative results. Next, we first discuss per-message detection and then pairwise detection.

Modeling. For a given message ID, CIDS derives the accumulated clock offset inherent in the arrival timestamps. Since clock skew is constant, the accumulated clock offset is linear in time, and hence CIDS describes it as a linear regression model. A linear parameter identification problem is thus formulated as:

$$O_{acc}[k] = S[k] \cdot t[k] + e[k], \tag{3}$$

where at step k, $O_{acc}[k]$ is the accumulated clock offset, $S[k]$ the regression parameter, $t[k]$ the elapsed time, and $e[k]$ the identification error. The regression parameter S represents the slope of the linear model and thus the *estimated clock skew*. The identification error, e, represents the residual which is not explained by the model. In CIDS, O_{acc}, S, t, and e are updated every N messages, i.e., kN messages are examined up to step k.

To determine the unknown parameter S, we use the Recursive Least Squares (RLS) algorithm [14], which uses the residual as an objective function to minimize the sum of squares of the modeling errors. Hence, in RLS, the identification error skews towards 0, i.e., has 0 mean. We will discuss the computational overhead of RLS as well as other possible solutions in Section 6.

Algorithm 1 describes how the clock skew is estimated using RLS. First, CIDS measures the arrival times and their intervals of N messages for a given ID. If the intended message has not been received for a long time — possibly due to suspension attack — as in line 13–14, CIDS sets the remaining timestamp and interval values significantly higher. Once N values are measured, CIDS determines the accumulated clock offset and accordingly, the identification error. Based on the thus-derived value, the gain, G, and the covariance, P, are updated with RLS for identifying the regression parameter S, i.e., estimate clock skew. This procedure of clock skew estimation continues iteratively during the operation of CIDS and, if uncompromised, outputs an identification error skewed towards 0 and a constant clock skew. This way, the *norm clock behavior* of the transmitter can be described as a linear model with the clock skew being the slope. In RLS, a forgetting factor, λ, is used to give exponentially less weights to older samples and thus provide freshness. In CIDS, we set λ=0.9995.

Detection. For a given message ID, CIDS runs RLS for clock skew estimation, constructs a norm model on clock behavior, and verifies whether there are any abnormal measurements deviating from it, i.e., intrusions.

Consider a fabrication attack in which the adversary injects an attack message with ID=0x01, which is originally sent every 10ms by some ECU. The fabrication

attack significantly increases the absolute average difference between the estimated and measured arrival times of 0x01. As a result, due to a sudden increase in the rate at which the accumulated clock offset changes, a high identification error results. Similarly, when the suspension attack is mounted, the absolute average difference also increases and thus a high error is also incurred. When a masquerade attack is mounted, since the adversary sends the message through a different ECU than its original one, the increase rate of accumulated clock offset, i.e., clock skew, suddenly changes and also results in a high identification error. In summary, unlike when the mean of identification error should usually skew towards 0, which is the norm clock behavior, its mean suddenly shifts towards a high non-zero value when there is an intrusion.

CIDS exploits the Cumulative Sum (CUSUM) method, which derives the cumulative sums of the deviations from a target value to detect sudden shifts. Since it is cumulative, even minor drifting from the target value leads to steadily increasing or decreasing cumulative values. It is therefore optimal in detecting small persistent changes and is widely used for change-point detection [8]. CIDS detects intrusions via CUSUM as follows. At each step of clock skew estimation, CIDS updates the mean and variance of the identification errors (e), μ_e and σ_e^2, respectively. In CIDS, these values represent the CUSUM target values of e (i.e., norm clock behavior), and thus require proper tracking. Hence, as a precaution of abnormal values incurring from an attack to be reflected into the target values, μ_e and σ_e^2 are updated only if $|\frac{e-\mu_e}{\sigma_e}| < 3$. Then, per derived identification error e, the upper and lower control limits of CUSUM, L^+ and L^- are updated as [41]:

$$\begin{aligned} L^+ &\leftarrow \max\left[0, L^+ + (e-\mu_e)/\sigma_e - \kappa\right] \\ L^- &\leftarrow \max\left[0, L^- - (e-\mu_e)/\sigma_e - \kappa\right] \end{aligned} \tag{4}$$

where κ is a parameter reflecting the number of standard deviations CIDS intends to detect. Note that κ can be learned offline, or by monitoring normal in-vehicle traffic. If either of the control limits, L^+ or L^-, exceeds a threshold Γ_L, a sudden positive or negative shift in value has been detected, respectively, and thus CIDS declares it as an intrusion. As the general rule of thumb for CUSUM is to have a threshold of 4 or 5 [28], we set $\Gamma_L = 5$.

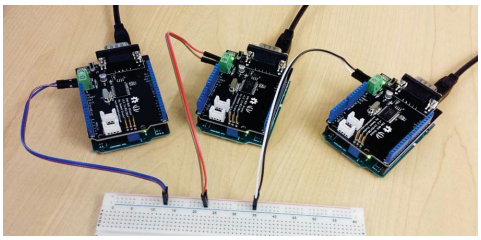
4.3 CIDS — Message-pairwise Detection

In addition to per-message detection, CIDS also alarms intrusions via message-pairwise detection, which examines the *correlation* between the average clock offsets in two periodic messages. Consider two messages M_1 and M_2 periodically sent by an ECU $\mathbb{A}$. Since these messages originate from the same transmitter, their instantaneous average clock offsets are likely equivalent. Thus, the correlation coefficient, ρ, between their average clock offsets (derived per step) would show a high value close to 1, i.e., correlated. On the other hand, if the two messages were sent by different ECUs, $\rho \simeq 0$, i.e., uncorrelated.

Modeling and detection. If clock offsets in two messages are highly correlated ($\rho > 0.8$), their relationship can be linear. So, CIDS describes them as a linear regression model: $O_{M_2}[k] = \alpha O_{M_1}[k] + e_{corr}[k]$, where O_{M_i} denotes the average clock offset of message M_i at step k, α the regression parameter, and $e_{corr}[k]$ the identification error. As per-message detection, message-pairwise detection is also based on a linear model. Thus, we apply the same detection method, CUSUM. Since message-pairwise detection seeks intrusions from a different perspective than per-message detection, it reduces false positive/negative results. Note, however, that message-pairwise detection is only applicable when two messages' clock offsets are highly correlated, whereas per-message detection is applicable to any periodic message. Moreover, albeit effective, it requires pairwise computations. Therefore, we use message-pairwise detection as an *optional* feature of CIDS. We will later show via experimental evaluations how message-pairwise detection further improves the performance of CIDS.

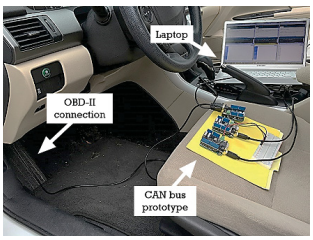
4.4 Verification

To reduce possible false positives/negatives, CIDS also performs a verification process. Suppose that a possible intrusion was alarmed due to a high identification error when verifying message V_i, the i-th message of V. Although such a high error can be due to an intrusion, it can also be due to an incorrect computation of average clock offset. In Section 4.1, we considered $\mathrm{E}[\Delta d_i] = 0$ and could thus extract and determine the average clock offset. Although this is true in most cases, occasionally $\mathrm{E}[\Delta d_i] \neq 0$, which affects the accuracy of deriving the true clock offset and thus the detection result. In CAN, $\mathrm{E}[\Delta d_i] \neq 0$ only occurs if the transmission of V_i was delayed due to the bus being busy or its transmitter losing arbitration when attempting to send V_i. Note that the latter also results in the bus being busy before the transmission/reception of V_i. Thus, CIDS also checks if the possibility of $\mathrm{E}[\Delta d_i] \neq 0$ is the main cause of a (possibly false) alarm of intrusion by verifying whether the CAN bus was busy right before receiving V_i. This way, CIDS enhances its detection accuracy. However, as discussed before, usually $\mathrm{E}[\Delta d_i] = 0$ in an actual CAN bus due to its high speed, its messages having short lengths, and low bus load. In other words, the nature of CAN bus communication helps CIDS reduce false positives/negatives.



(a) CAN bus prototype.

(b) The vehicle used for experiment.



(c) Connection to the vehicle.

Figure 4: Different evaluation settings: (a) CAN bus prototype; (b) Honda Accord 2013 used for experiments on real vehicle; and (c) three prototype nodes communicating with real ECUs through the OBD-II port.

4.5 Root-cause Analysis

When an intrusion is detected for some message ID, CIDS can also identify which compromised ECU mounted the attack. It can extract the clock skew for that attacked message ID, compare it with other clock skew values extracted from other message IDs, and exploit the comparison result in determining whether they originated from the same transmitter. This way, CIDS can at least reduce the scope of ECUs which may (or may not) have mounted the attack, thus facilitating a root-cause analysis.

5 Evaluation

We now validate that clock skews can be used as fingerprints of transmitter ECUs, and evaluate the performance of CIDS on a CAN bus prototype and real vehicles.

CAN bus prototype: As shown in Fig. 4(a), we built a prototype with 3 CAN nodes, each of which consists of an Arduino UNO board and a SeeedStudio CAN shield. The CAN bus shield consists of a Microchip MCP2515 CAN controller, MCP2551 CAN transceiver, and a 120Ω terminal resistor to provide CAN bus communication capabilities. This prototype was set up to operate at a 500Kbps bus speed as in typical CAN buses. The first node $\mathbb{A}$ was programmed to send messages 0x11 and 0x13 every 50ms, and the second node $\mathbb{B}$ to send message 0x55 at the same frequency. The third node $\mathbb{R}$ was programmed to run CIDS.

Real vehicle: A 2013 Honda Accord (Fig. 4(b)) is also used for our experiments in an isolated and controlled (for safety) environment. As shown in Fig. 4(c), via the On-Board Diagnostic (OBD-II) system port [3], we connected our CAN bus prototype nodes — which function as an adversary or CIDS — to the in-vehicle network. Through the OBD-II port, the three nodes were able to communicate with real ECUs.

CAN log data: To further validate that CIDS's fingerprinting is applicable to other vehicles, we also refer to CAN traffic data logged from a Toyota Camry 2010 by Ruth *et al.* [36] and data logged from a Dodge Ram Pickup 2010 by Daily [11]. Both data were logged through a Gryphon S3 and Hercules software. In the Toyota Camry 2010, there were 42 distinct messages transmitted on the CAN bus: 39 of them sent periodically at intervals ranging from 10ms to 5 seconds, and 3 of them sent sporadically. In the Dodge Ram Pickup 2010, there were 55 distinct messages which were all sent periodically on the CAN bus.

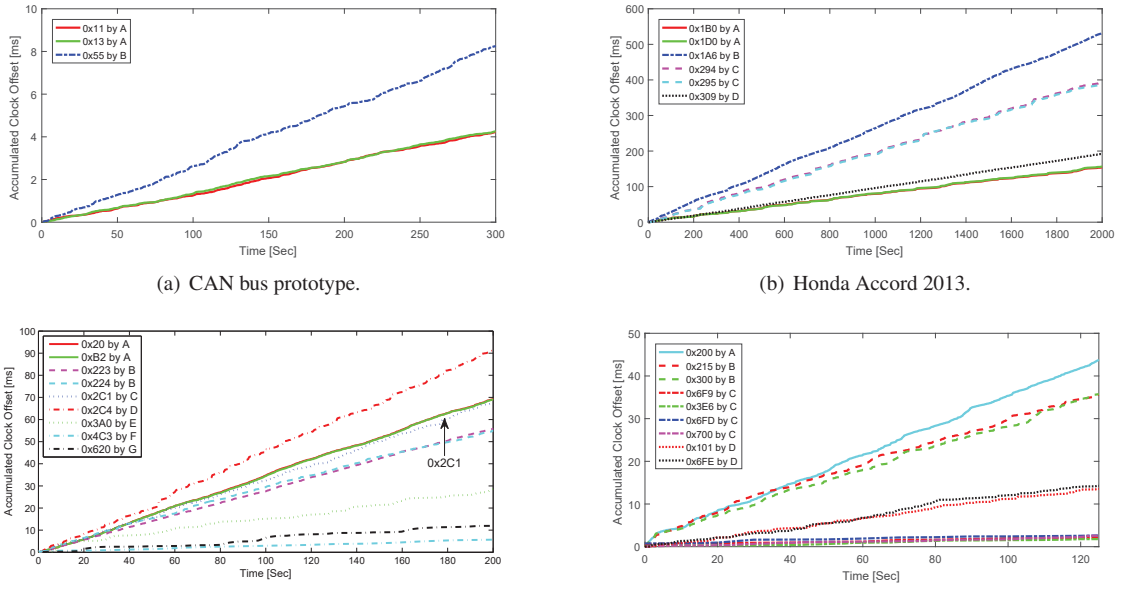
In order to identify which messages originate from the same real ECU and thus exploit it as a ground truth, we used the naive method discussed in [32]. The messages, which originate from the same ECU and have the same preset message interval, were shown to have the same number of transmissions on the bus, when traced for at least a few minutes. Such a method can be an alternative to fingerprinting, but it requires pairwise comparisons and cannot be completed in real time as required in the design of CIDS, which is essential for intrusion detection in real in-vehicle networks.

While running CIDS, we determined offsets and skews for every 20 received samples, i.e., $N = 20$, and set $\kappa = 5$.

5.1 Clock Skew as a Fingerprint

We first evaluate the validity of CIDS's fingerprinting of the transmitter ECUs based on the estimated clock skews. We evaluate skew estimates in microseconds per second ($\mu s/s$) or parts per million (ppm).

CAN bus prototype. Fig. 5(a) plots our evaluation results of CIDS's fingerprinting on the CAN bus prototype: accumulated clock offsets of messages 0x11, 0x13, and 0x55. Note that the slopes in this figure represent the estimated clock skews. All the derived accumulated clock offsets were found to be linear in time, i.e., constant estimated skews. Messages 0x11 and 0x13, both of which were sent from node $\mathbb{A}$, exhibited the same constant clock skew of 13.4ppm. On the other hand, the message 0x55 sent from a different node $\mathbb{B}$ showed a different clock skew of 27.2ppm. Thus, the clock skews derived by CIDS can be used to differentiate ECUs.



(a) CAN bus prototype.

(b) Honda Accord 2013.

(c) Toyota Camry 2010.

(d) Dodge Ram Pickup 2010.

Figure 5: Accumulated clock offsets derived by CIDS in different evaluation settings.

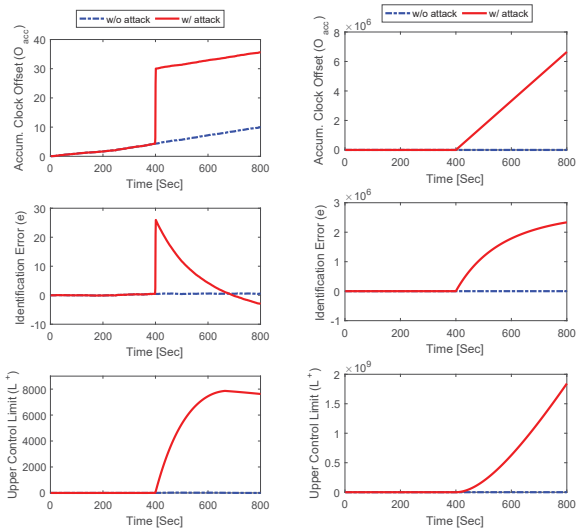
Honda Accord 2013. For CIDS's evaluation on a real vehicle, the CAN prototype nodes logged the in-vehicle CAN traffic of the Honda Accord 2013, and ran CIDS on messages 0x1B0, 0x1D0, 0x1A6, 0x294, 0x295, and 0x309. The approach in [32] was adopted to verify that messages {0x1B0, 0x1D0} were sent from the same ECU, {0x294, 0x295} both sent from another ECU, whereas others were sent from different ECUs. Utilizing these facts, one can conclude from Fig. 5(b) that the clock offsets and the skews derived in CIDS are equivalent *only* for those messages sent from the same ECU; 0x1B0 and 0x1D0 showed a skew of 78.4ppm, 0x294 and 0x295 showed a skew of 199.8ppm, while messages 0x1A6 and 0x309 showed very different skews of 265.7ppm and 95.78ppm, respectively. This result again shows that clock skews between *different* ECUs are distinct and can thus be used as the fingerprints of the corresponding ECUs.

Toyota Camry 2010. To show that the applicability of CIDS's fingerprinting is not limited to the specific vehicle model used, we also conducted experiments on a different vehicle: running CIDS's fingerprinting on the Toyota Camry logged data. Similarly to the real vehicle evaluation in Section 5.1, the approach in [32] was used as the ground truth. It was verified that messages {0x20, 0xB2} within the CAN log data were all sent from some ECU $\mathbb{A}$. Also, {0x223, 0x224} were both sent from some ECU $\mathbb{B}$, whereas 0x2C1, 0x2C4, 0x3A0, 0x4C3, and 0x620 were each sent from a different ECU. As shown in Fig. 5(c), messages 0x20 and 0xB2 both showed a clock skew of approximately 345.3ppm, whereas 0x223 and 0x224 showed a different clock skew of 276.5ppm. 0x2C4, 0x3A0, 0x4C3, and 0x620 showed very different clock skews of 460.1ppm, 142.5ppm, 26.1ppm and 58.7 ppm, respectively.

We made an interesting observation on message 0x2C1, showing a clock skew of 334.1ppm, which was different from the skews of messages {0x20, 0xB2} only by 3%, despite the fact that it was sent by a different ECU. This may confuse CIDS in determining whether they were sent by the same ECU or not. However, in such a case, CIDS can further examine the correlation between clock offsets and can thus fingerprint with a higher accuracy, which we will discuss and evaluate further in Section 5.4.

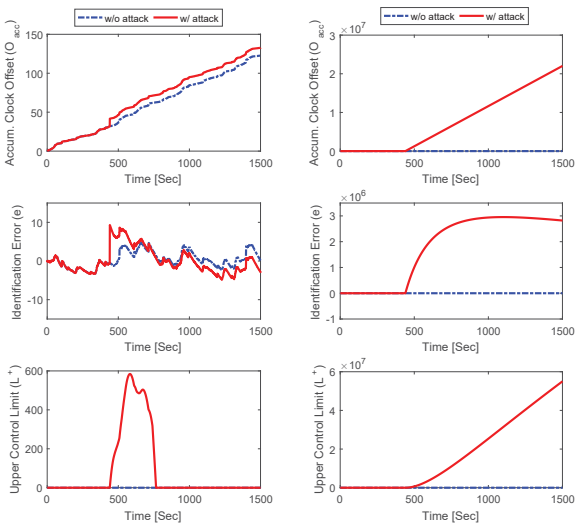
Dodge Ram Pickup 2010. We also ran CIDS's fingerprinting on the CAN log data of a Dodge Ram Pickup 2010. For this vehicle, it was verified that message 0x200 was sent from some ECU $\mathbb{A}$, {0x215, 0x300} sent from $\mathbb{B}$, {0x6F9, 0x3E6, 0x6FD, 0x700} sent from $\mathbb{C}$, and {0x101, 0x6FE} sent from $\mathbb{D}$. Fig. 5(d) shows that CIDS determined that 0x200 has a clock skew of 351.7ppm, {0x215, 0x300} to have approximately 295.3ppm, {0x6F9, 0x3E6, 0x6FD, 0x700} to have 24.5ppm, and {0x101, 0x6FE} to have 110.3ppm, thus correctly fingerprinting their transmitters.

These results of a Toyota Camry and a Dodge Ram Pickup CAN log data again affirm the fact that the clock skews derived by CIDS are diverse and can indeed be used as fingerprints of in-vehicle ECUs. Moreover, they



(a) Fabrication attack. (b) Suspension attack.

Figure 6: CIDS defending fabrication attack (left) and suspension attack (right) in a CAN bus prototype.



(a) Fabrication attack. (b) Suspension attack.

Figure 7: CIDS defending fabrication attack (left) and suspension attack (right) in a Honda Accord 2013.

show that CIDS's fingerprinting is not limited to a specific vehicle model, and can thus be applied to other vehicle models.

5.2 Defending Against Fabrication and Suspension Attacks

On both the CAN bus prototype and the real vehicle setting (Honda Accord 2013), we launch the fabrication and suspension attacks, and evaluate CIDS's effectiveness in detecting them.[2] To this end, we consider CIDS to only perform per-message detection, and will later evaluate CIDS with message-pairwise detection.

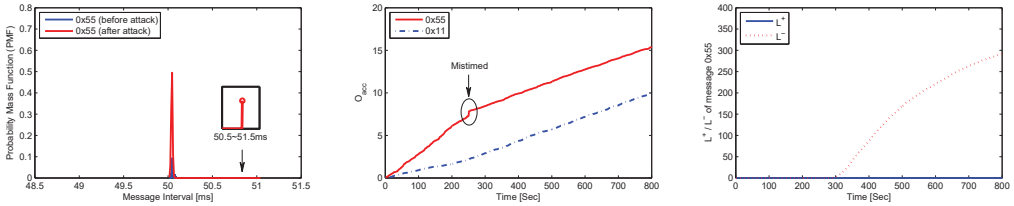
CAN bus prototype. For evaluation of CIDS defending against fabrication attack on the CAN bus prototype, $\mathbb{B}$ was programmed to inject a fabricated message at $t = 400$ secs with ID=0x11, which is a periodic message usually sent by $\mathbb{A}$, i.e., $\mathbb{B}$ launches a fabrication attack on $\mathbb{A}$. ECU $\mathbb{R}$ was running CIDS on message 0x11 and derived accumulated clock offset (O_{acc}), identification error (e), and control limits (L^+, L^-). For the suspension attack, $\mathbb{A}$ was instead programmed to stop transmitting 0x11 at $t = 400$ secs.

Fig. 6(a) shows how such values changed for message 0x11 in the presence and absence of a fabrication attack. As soon as $\mathbb{B}$ mounted a fabrication attack, as discussed in Section 4.2, there was a sudden positive shift in the accumulated clock offset, thus yielding a high identification error. Due to such a shift, the upper control limit, L^+, of CUSUM suddenly increased and exceeded its threshold Γ_L =5, i.e., detecting an intrusion. Similarly, Fig. 6(b) shows that since the suspension attack also shifted the accumulated clock offset significantly, CIDS was able to detect the attack.

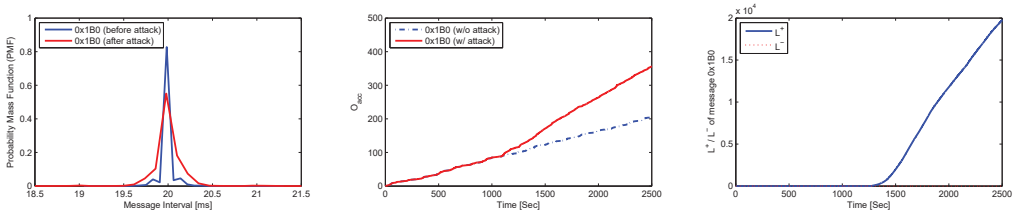
Real vehicle. To evaluate CIDS against the fabrication attack under the real vehicle setting, one CAN prototype node $\mathbb{R}$ was programmed to run CIDS, and another node $\mathbb{A}$ as an adversary mounting the attack on a real ECU. The attack was mounted by injecting a fabricated attack message with ID=0x1B0, which was sent every 20ms by some real in-vehicle ECU, i.e., $\mathbb{A}$ mounted the fabrication attack on a Honda Accord ECU sending 0x1B0. For the suspension attack, the message filter of $\mathbb{R}$ was reset at $t = 420$ secs so as to no longer receive 0x1B0, thus emulating the suspension attack.

Fig. 7(a) shows how accumulated clock offsets (O_{acc}), identification errors (e), and upper control limits (L^+) changed for both cases of with and without a fabrication attack. Again, the attack message injected at around $t = 420$ secs caused a sudden increase in O_{acc} and e, thus increasing L^+ to exceed Γ_L =5. As a result, CIDS declares the detection of an attack. After the attack, since 0x1B0 was still periodically sent by the real in-vehicle ECU, the clock skew — i.e., the slope of O_{acc} graph — remains unchanged. Similarly, as shown in Fig. 7(b), the suspension attack increases the offset values, thus causing L^+ to exceed the threshold, i.e., the suspension attack was detected by CIDS.

[2] As the attacks cannot be emulated using the CAN log data, we do not consider their use for evaluating CIDS against the attacks.



(a) CAN bus prototype.



(b) Real vehicle.

Figure 8: Masquerade attack — Probability mass function of message intervals (left), changes in accumulated clock offsets (middle), and control limits (right) derived in CIDS.

5.3 Defending Against Masquerade Attack

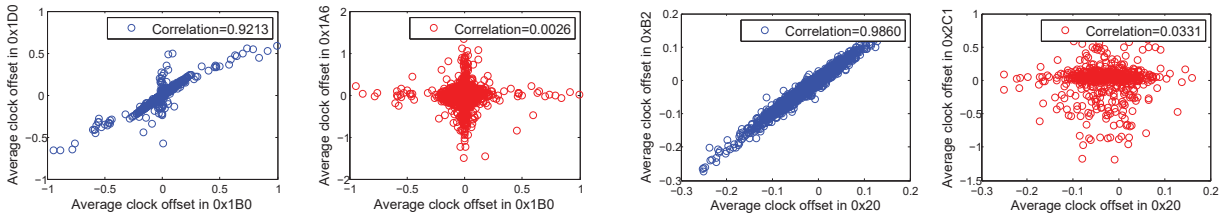
We now evaluate the performance of CIDS in detecting a masquerade attack.

CAN bus prototype. To evaluate CIDS's defense against the masquerade attack in the CAN bus prototype, nodes $\mathbb{A}$ and $\mathbb{B}$ were considered to have been compromised as strong and weak attackers as in Fig. 2(c), respectively. $\mathbb{A}$ was programmed to mount a masquerade attack on $\mathbb{B}$, i.e., stop $\mathbb{B}$ transmitting message 0x55 and instead send it through $\mathbb{A}$ onwards, once $T_{masq} = 250$ secs had elapsed. As usual, messages 0x11 and 0x13 were periodically sent by $\mathbb{A}$, and CIDS was run by $\mathbb{R}$.

Fig. 8(a) (left) shows the Probability Mass Function (PMF) of the intervals of message 0x55: before and after the attack was mounted. In contrast to the fabrication attack, since the attacker sent the attack message at its original frequency after masquerading, the distribution did not deviate much from that before the attack. However, at T_{masq}, since there was some delay when the transmitter was switched from one node to another, the first masquerade attack message was sent 51.04*ms* after its previous transmission, whereas it should have been approximately 50*ms* which is the preset message interval of 0x55. Due to such a slightly *mistimed* masquerade attack, the PMF graph shows a message interval with an abnormal deviation from the mean. We will later evaluate the perfectly *timed* masquerade attack — a much more severe case than a mistimed attack — on a real vehicle, and show the efficacy of CIDS in detecting it.

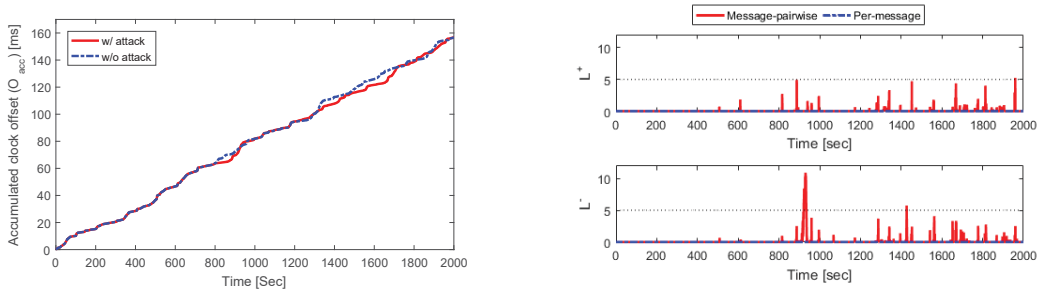
The resulting changes in O_{acc}, L^+, and L^- at $\mathbb{R}$ are also shown in Fig. 8(a) (middle and right). The change in the ECU transmitting message 0x55 caused the slope (i.e., clock skew) in O_{acc} graph to change after the attack was mounted. Since the measurements of O_{acc} after T_{masq} significantly deviated from their expected values, which is determined by the estimated clock skew of $t < T_{masq}$, the CUSUM lower control limit, L^-, in CIDS exceeded the threshold, thus declaring detection of an intrusion. Since the transmitter of 0x55 was changed (to ECU $\mathbb{A}$), its clock skew after $t = T_{masq}$ was equivalent to the clock skew in 0x11. Accordingly, via root-cause analysis, CIDS identifies the compromised ECU to be ECU $\mathbb{A}$. Unlike the previous results, since the change in slope was negative, persistent identification error with high negative values caused L^- to exceed the threshold.

Real vehicle. To evaluate CIDS's defense against the masquerade attack in a real vehicle, we consider a scenario in which real in-vehicle ECUs $\mathbb{V}_1$ and $\mathbb{V}_2$ transmitting 0x1A6 and 0x1B0 are compromised as a strong and a weak attacker, respectively. Of the three CAN prototype nodes ($\mathbb{A}$, $\mathbb{B}$, and $\mathbb{R}$), which were connected to the real in-vehicle network via OBD-II, we programmed node $\mathbb{R}$ to run CIDS on in-vehicle message 0x1B0 and another node $\mathbb{B}$ to simply log the CAN traffic. To generate a scenario of real ECU $\mathbb{V}_1$ mounting a masquerade attack on real ECU $\mathbb{V}_2$, $\mathbb{R}$ was programmed further to receive message 0x1A6 instead of 0x1B0, but still record the received messages' ID to be 0x1B0, once $T_{masq} = 1100$ seconds had elapsed. That is, we let $\mathbb{R}$ interpret 0x1A6 as 0x1B0 for $t > T_{masq}$, i.e., the transmitter of 0x1B0 changes from $\mathbb{V}_2$ to $\mathbb{V}_1$. Such a change in interpretation was achieved by programming $\mathbb{R}$ to modify its message acceptance filter from only accepting 0x1B0 to only accepting 0x1A6. Since 0x1B0 and 0x1A6 were observed to be always transmitted nearly at the same time, such a



(a) Honda Accord 2013. (b) Toyota Camry 2010.

Figure 9: Correlated and uncorrelated clock offsets.



(a) Accumulated clock offset. (b) Control limits.

Figure 10: Defense against the worst-case masquerade attack via message-pairwise detection.

setting replicates the *timed* masquerade attack. During such a process, $\mathbb{B}$ continuously logged 0x1B0 so that we can obtain a reference for circumstances when no attacks are mounted.

Fig. 8(b) (left) shows the PMF of the message intervals of 0x1B0 before and after the attack. Since the message periodicity remained the same, the distribution of the messages intervals did not change. Moreover, since we considered a timed masquerade attack, in contrast to the result in Fig. 8(a), there were no such abnormal message intervals. Such a result indicates that state-of-the-art IDSs, which try to find abnormal message frequencies, cannot detect such an attack. Although the distribution of message intervals remained unchanged, due to the change in ECU transmitting 0x1B0 ($\mathbb{V}_2 \rightarrow \mathbb{V}_1$), the accumulated clock offset suddenly exhibited a different trend in its change, i.e., a different clock skew after the attack. Here, the original trend in offset changes was determined by the data obtained from $\mathbb{B}$. So, as shown in Fig. 8(b) (right), `CIDS` was able to detect a sudden shift in its identification error and thus outputted a high level of CUSUM upper control limit, i.e., an intrusion detection. `CIDS`'s capability of detecting various types of masquerade attack is evaluated further in Section 5.5.

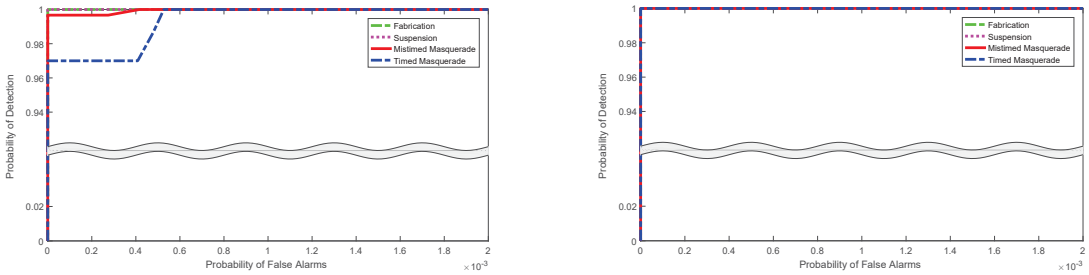
In conclusion, through its modeling and detection processes, `CIDS` can detect not only the fabrication attack but also the masquerade attack, i.e., is capable of doing not only what existing solutions can do, but also more.

5.4 Message-pairwise Detection

We evaluate the feasibility and efficiency of message-pairwise detection in `CIDS`. To validate its practicability in the real-world, we first examine whether there exists pairs of messages inside real vehicles with correlated clock offsets — the condition for `CIDS` to run message-pairwise detection.

Fig. 9(a) shows two cases of correlated and uncorrelated clock offsets of in-vehicle messages collected from the Honda Accord 2013. Fig. 9(a) (left) shows that the average clock offsets of messages 0x1B0 and 0x1D0, which were determined to have been sent from the same ECU, showed a high correlation of 0.9213, i.e., linear relationship. In contrast, as shown in Fig. 9(a) (right), average clock offsets of messages 0x1B0 and 0x1A6, which were sent every 20*ms* from different ECUs, showed a near 0 correlation.

By the Birthday paradox, some ECUs in the vehicle may probably have near-equivalent clock skews — as it was for messages 0x20 and 0x2C1 in the examined Toyota Camry 2010 (see Fig. 5(c)). Although clock skews may be near-equivalent, instantaneous clock offsets of two different ECUs cannot be near-equivalent and are thus uncorrelated as they run different processes. The results in Fig. 9(b) corroborate such a fact by showing that clock offsets of messages 0x20 and 0xB2, which were sent by the same ECU, had a high correlation of 0.9860, whereas offsets of messages 0x20 and 0x2C1 —



(a) Per-message detection.

(b) Per-message + Message-pairwise detection.

Figure 11: ROC curves of CIDS in the real vehicle.

sent by different ECUs with similar clock skews — had a low correlation of 0.0331. Thus, for messages with near-equivalent clock skews, CIDS can further examine the correlation between their clock offsets, and correctly determine their transmitters.[3] These facts and observations indicate the feasibility and efficiency of message-pairwise detection in CIDS.

To show that message-pairwise detection can support per-message detection in decreasing false positives/negatives by examining offset correlations, we consider a scenario in which an attacker $\mathbb{V}_1$ has mounted a masquerade attack on a Honda Accord ECU $\mathbb{V}_2$ at $t_{masq} = 800$ secs. We refer to $\mathbb{V}_2$ as the ECU which originally transmits message 0x1B0. To consider the *worst case* in detecting the masquerade attack, we assume that the clock skews of $\mathbb{V}_1$ and $\mathbb{V}_2$ are nearly equivalent, similarly to messages 0x20 and 0x2C1 in the Toyota Camry. We replicated such a worst-case scenario by randomly permuting the acquired offset values of 0x1B0 for $t > t_{masq}$, and considering the permuted values to be output from $\mathbb{V}_1$. As shown in Fig. 10(a), this leads to a situation where the clock skew does not change even though the message transmitter has been changed from one ECU to another. Although the clock skew remained equivalent at $t = t_{masq}$, the correlation between offsets of 0x1B0 and 0x1D0 suddenly dropped from 0.9533 to 0.1201, i.e., a linear to non-linear relationship. As a result, as shown in Fig. 10(b), the control limits in CIDS's message-pairwise detection exceeded the threshold $\Gamma_L = 5$. On the other hand, since the clock skews before and after the attack were equivalent, per-message detection was not able to detect the intrusion.

5.5 False Alarm Rate

We also examined the false alarm rate of CIDS under the real vehicle setting. The results obtained from the CAN bus prototype are omitted due to their insignificance, i.e., not many false alarms occurred due to its less complex bus traffic. Based on data recorded for 30 minutes from the Honda Accord 2013 — approximately 2.25 million messages on the CAN bus — four attack datasets were constructed to each contain 300 different intrusions. The intrusions either had different injection timings, suspension timings, or changes in clock skews: each in the form of fabrication attack, suspension attack, mistimed masquerade attack, and timed masquerade attack. For each dataset, we varied the κ parameter of CIDS to acquire one false positive rate (false-alarm rate) and one false negative rate (1−detection rate).

Fig. 11(a) shows the Receiver Operating Characteristic (ROC) curve of CIDS, which represents its trade-off between false alarm and detection, executing *only* per-message detection on the attack datasets. Clearly, CIDS is shown to be able to detect fabrication, suspension, and masquerade attacks with a high probability. Since the timed masquerade attack is the most difficult to detect, it showed the highest false positive rate among all the attack scenarios considered: a false positive rate of 0.055% while not missing any anomalies (100% true positives). Even for false positives $< 0.055\%$, 97% of the anomalies were detected by CIDS. However, these false positives can be of great concern for in-vehicle networks. Therefore, to eliminate such false positives, CIDS can additionally run message-pairwise detection. Fig. 11(b) shows the ROC curve of CIDS executing not only per-message detection but also message-pairwise detection for further verification. Accordingly, CIDS was able to detect all types of attacks considered without having any false positives, which is in contrast to CIDS with only per-message detection, i.e., all false positives were eliminated via message-pairwise detection.

6 Discussion

Discussed below are the overhead, deployment, limitations, and applications of CIDS.

[3] If the two ECUs' clock behaviors are still not distinguishable, CIDS can be set up to exclude them for examination so that the risk of false positives significantly decreases. However, this may impact CIDS's capability of detecting attacks mounted through those ECUs.

Identification algorithm. To estimate clock skew, one can also use other algorithms than RLS, such as Total Least Squares (TLS) and Damped Least Squares (DLS), which perform orthogonal linear and non-linear regression, respectively. Although they might identify the clock skew with a higher accuracy than RLS, their gains are offset by the accompanying high complexity. TLS requires Singular Value Decomposition (SVD), which is computationally expensive, and DLS requires a large number of iterations for curve fitting. RLS is known to have a computation complexity of $\mathbb{O}(N^2)$ per iteration, where N is the size of the data matrix. However, in CIDS, only a scalar clock offset is exploited for identification, and thus the computational complexity is relatively low.

Defeating CIDS. There may be several ways the adversary may attempt to defeat CIDS. First, the adversary may try to compromise the ECU running CIDS and disable it. However, if *cross-validation* for CIDS was to be exploited, such an attempt can be nullified. For the detection of intrusions, CIDS only requires an ECU to record the timestamps of message arrivals. Such a low overhead makes it feasible for CIDS to be installed distributively across several in-vehicle ECUs for cross-validation. Suppose using CIDS, ECU $\mathbb{A}$ monitors attacks on messages $\{M_1, M_2\}$, ECU $\mathbb{B}$ monitors $\{M_2, M_3\}$, and ECU $\mathbb{C}$ monitors $\{M_1, M_3\}$. Since CIDS regards the *receiver*'s time clock as the true clock, cross-validation provides multiple perspectives of clock behaviors for each message ID, e.g., two different perspectives of M_2 from $\mathbb{A}$ and $\mathbb{B}$. Thus, even when an ECU running CIDS gets compromised, cross-validation via CIDS can handle such a problem.

Another way the adversary may try to defeat CIDS is to adapt to how its algorithm is running and thus deceive it. The adversary may figure out the clock skew of the target ECU and then heat up or cool down the compromised ECU so that its clock skew changes to match that of the target. In such a case, the clock skew can be matched and thus may bypass CIDS's per-message detection. However, as discussed in Section 5.4, unless the adversary also matches the instantaneous clock offset, which is affected by the ECU's *momentary* workload and temperature, CIDS can detect the intrusion via message-pairwise detection.

Upon intrusion detection. False alarms for intrusion detection systems, especially in in-vehicle networks, are critical. Thus, CIDS should also deal with them as accurately as possible. To meet this requirement, if an intrusion has been determined, even after going through the verification process, CIDS can follow the following steps for further examination:

1. If an intrusion was detected while using only per-message detection, examine it further via message-pairwise detection.
2. If still alarmed as an intrusion and the attacked ECU is a safety-critical ECU, go straight to step 4.
3. If not, communicate with other ECUs for cross-validation as they would provide different perspectives of the clock skew results. If communicating with other ECUs incurs too much overhead (in terms of bus load, processing overhead, etc.), send traffic data for a remote diagnosis.
4. Request re-patching of firmware and advise the driver to stop the vehicle.

Limitation of CIDS. CIDS is shown to be effective in detecting various types of in-vehicle network intrusions. One limitation of CIDS might be that since it can only extract clock skews from periodic messages, it would be difficult to fingerprint ECUs which are sending aperiodic messages. That is, if the attacker injects messages aperiodically, although CIDS can still detect the intrusion, it would not be able to pinpoint where the attack message came from, i.e., finding the root-cause of attacks launched with or on aperiodic messages. Recall that CIDS can achieve this only for periodic messages. In future, we would like to find new features other than clock skew, which can fingerprint ECUs, regardless of whether they send messages periodically or aperiodically.

Applicability to other in-vehicle networks. Although most modern in-vehicle networks are based on CAN, some may be equipped with other protocols, such as CAN-FD, TTCAN and FlexRay, for more complex operations. CAN-FD is an enhanced version of CAN, providing flexible and higher data rates [5]. Since its basic components conform to CAN and thus also lacks synchronization, CIDS can be applied to CAN-FD. For protocols such as TTCAN [21] and FlexRay [22], nodes are periodically synchronized for determinative timing of message exchanges. The interval between two consecutive synchronizations depends on how each protocol is deployed [32]. For TTCAN, it can be up to $2^{16} = 65536$ bits long, i.e., $131ms$ in a 500Kbps bus [37]. This lets some messages be sent multiple times between consecutive synchronizations. So, if the time interval is long, CIDS would still be able to extract clock skews from messages which are sent multiple times, whereas, if the period is short, CIDS may not be feasible. However, the fact that TTCAN and FlexRay have high implementation cost, whereas for CAN-FD it is minimal, makes CAN-FD a favorite candidate for next-generation in-vehicle networks [6, 39]. This means that CIDS can be applicable to not only current but also future in-vehicle networks.

7 Conclusion

New security breaches in vehicles have made vehicle security one of the most critical issues. To defend against

vehicle attacks, several security mechanisms have been proposed in the literature. They can cope with some attacks but cannot cover other safety-critical attacks, such as the masquerade attack. To remedy this problem, we have proposed a new IDS called CIDS, which extracts clock skews from message intervals, fingerprints the transmitter ECUs, and models their clock behaviors using RLS. Then, based on the thus-constructed model, CIDS detects intrusions via CUSUM analysis. Based on our experiments on a CAN bus prototype and on real vehicles, CIDS is shown to be capable of detecting various types of in-vehicle network intrusions. CIDS can address all attacks that existing IDSs can and cannot handle as well as facilitates root-cause analysis. Thus, it has potential for significantly enhancing vehicle security and safety.

8 Acknowledgments

We would like to thank the anonymous reviewers and the shepherd, Tadayoshi Kohno, for constructive suggestions. The work reported in this paper was supported in part by the NSF/Intel Grant CNS-1505785 and the DGIST Global Research Laboratory Program through NRF funded by MSIP of Korea (2013K1A1A2A02078326).

References

[1] Microchip MCP2515 Datasheet. [Online] Available: www.microchip.com/MCP2515.

[2] Microchip TB078, PLL Jitter and Its Effects in the CAN Protocol.

[3] On-Board Diagnostic System [Online] http://www.obdii.com.

[4] CAN Specification v2.0. *Robert Bosch GmbH* (1991).

[5] CAN with Flexible Data-Rate Specification Version 1.0. *Robert Bosch GmbH* (2012).

[6] Infineon: CAN FD Success Goes at Expense of FlexRay. [Online] http://www.eetimes.com/. *EETimes* (Feb. 2015).

[7] Hackers Remotely Kill a Jeep on the Highway - With Me in It. [Online] http://www.wired.com.

[8] Basseville, M., and Nikiforov, I. Detection of abrupt changes: Theory and application. In *Prentice Hall information and system sciences series* (1993).

[9] Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security* (2011).

[10] Cho, K. T., Park, T., and Shin, K. G. CPS Approach to Checking Norm Operation of a Brake-by-Wire System. In *ICCPS* (2015).

[11] Daily, J. Analysis of critical speed yaw scuffs using spiral curves. In *SAE Technical Paper 2012-01-0606* (2012).

[12] Davis, R., Kollmann, S., Pollex, V., and Slomka, F. Controller area network (can) schedulability analysis with fifo queues. *ECRTS* (2011).

[13] Foster, I., Prudhomme, A., Koscher, K., and Savage, S. Fast and Vulnerable: A Story of Telematic Failures. In *WOOT* (2015).

[14] Haykin, S. Adaptive filter theory. In *2nd ed. Prentice-Hall* (1991).

[15] Herrewege, A., Singelee, D., and Verbauwhede, I. Canauth - a simple, backward compatible broadcast authentication protocol for can bus. In *ECRYPT Workshop on Lightweight Cryptography* (2011).

[16] Hoppe, T., Kiltz, S., and Dittmann, J. Security threats to automotive can networks - practical examples and selected short-term countermeasures. In *Reliability Engineering and System Safety* (Jan. 2011).

[17] Jana, S., and Kasera, S. K. On fast and accurate detection of unauthorized wireless access points using clock skews. In *ACM MobiCom* (2008).

[18] Khan, D., Bril, R., and Navet, N. Integrating hardware limitations in can scbedulability analysis. In *WFCS* (May. 2010).

[19] Kohno, T., Broido, A., and Claffy, K. Remote physical device fingerprinting. In *IEEE Symposium on Security and Privacy* (2005).

[20] Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., and Savage, S. Experimental security analysis of a modern automobile. In *IEEE Security and Privacy* (2010).

[21] Leen, G., and Heffernan, D. Ttcan: a new time-triggered controller area network. In *Elsevier Microprocessors and Microsystems* (2002).

[22] Milbredt, P., Horauer, M., and Steininger, A. An investigation of the clique problem in flexray. *SIES* (2008).

[23] Miller, C., and Valasek, C. Adventures in automotive networks and control units. *Defcon 21* (2013).

[24] Miller, C., and Valasek, C. A survey of remote automotive attack surfaces. *Black Hat USA* (2014).

[25] Miller, C., and Valasek, C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* (2015).

[26] Mills, D. L. Network time protocol: Specification, implemenation, and analysis. RFC 1305.

[27] Mohalik, S., Rajeev, A. C., Dixit, M. G., Ramesh, S., Suman, P. V., Pandya, P. K., and Jiang, S. Model checking based analysis of end-to-end latency in embedded, real-time systems with clock drifts. In *DAC* (2008).

[28] Montgomery, D. Introduction to statistical quality control. In *4th edition, Wiley* (2000).

[29] Moon, S. B., Skelly, P., and Towsley, D. Estimation and removal of clock skew from network delay measurements. In *INFOCOM* (1999).

[30] Muter, M., and Asaj, N. Entropy-based anomaly detection for in-vehicle networks. *IEEE IVS* (2011).

[31] Muter, M., Groll, A., , and Freiling, F. C. A structured approach to anomaly detection for in-vehicle networks. In *Information Assurance and Security (IAS), Sixth International Conference* (2010).

[32] Natale, M. D., Zeng, H., Giusto, P., and Ghosal, A. Understanding and using the controller area network communication protocol: Theory and practice. In *Springer Science & Business Media* (2012).

[33] Nilsson, D., Larson, D., and Jonsson, E. Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes. In *VTC-Fall* (2008).

[34] PASZTOR, A., AND VEITCH, D. Pc based precision timing without gps. In *ACM SIGMETRICS* (2002).

[35] PAXSON, V. On calibrating measurements of packet transit times. In *SIGMETRICS* (1998).

[36] RUTH, R., BARTLETT, W., AND DAILY, J. Accuracy of event data in the 2010 and 2011 Toyota camry during steady state and braking conditions. In *SAE International Journal on Passenger Cars* (2012).

[37] RYANA, C., HEFFERNANB, D., AND LEENA, G. Clock synchronisation on multiple ttcan network channels. In *Elsevier Microprocessors and Microsystems* (2004).

[38] SZILAGYI, C., AND KOOPMAN, P. Low cost multicast network authentication for embedded control systems. In *Proceedings of the 5th Workshop on Embedded Systems Security* (2010).

[39] TALBOT, S. C., AND REN, S. Comparison of fieldbus systems, can, ttcan, flexray and lin in passenger vehicles. In *ICDCSW* (2009).

[40] VEITCH, D., BABU, S., AND PASZTOR, A. Robust synchronization of software cclock across the internet. In *IMC* (2004).

[41] WOODALL, W. H., AND ADAMS, B. The statistical design of cusum charts. In *Quality Engineering, 5(4),* (1993).

[42] ZANDER, S., AND MURDOCH, S. An improved clock-skew measurement technique for revealing hidden services. In *USENIX Security* (2008).