Viden: Attacker Identification on In-Vehicle Networks

Kyong-Tak Cho and Kang G. Shin
University of Michigan
Ann Arbor, MI, 48109-2121
{ktcho,kgshin}@umich.edu

ABSTRACT

Various defense schemes — which determine the presence of an attack on the in-vehicle network — have recently been proposed. However, they fail to identify *which* Electronic Control Unit (ECU) actually mounted the attack. Clearly, pinpointing the attacker ECU is essential for fast/efficient forensic, isolation, security patch, etc. To meet this need, we propose a novel scheme, called Viden (Voltage-based attacker identification), which can identify the attacker ECU by measuring and utilizing voltages on the in-vehicle network. The first phase of Viden, called *ACK learning*, determines whether or not the measured voltage signals really originate from the genuine message transmitter. Viden then exploits the voltage measurements to construct and update the transmitter ECUs' voltage profiles as their fingerprints. It finally uses the voltage profiles to identify the attacker ECU. Since Viden adapts its profiles to changes inside/outside of the vehicle, it can pinpoint the attacker ECU under various conditions. Moreover, its efficiency and design-compliance with modern in-vehicle network implementations make Viden practical and easily deployable. Our extensive experimental evaluations on both a CAN bus prototype and two real vehicles have shown that Viden can accurately fingerprint ECUs based solely on voltage measurements and thus identify the attacker ECU with a low false identification rate of 0.2%.

CCS CONCEPTS

• **Security and privacy → Embedded systems security**;

KEYWORDS

Automotive Security; CAN bus; Attacker Identification

1 INTRODUCTION

Remote and/or driverless control of a car is no longer science fiction. In fact, demonstration and deployment of such a vehicle control have become prevalent, triggering significant R&D efforts and investments from industry, governments, and academia. Despite their numerous benefits, these technological developments have created serious safety/security concerns.

CCS'17, , Oct. 30–Nov. 3, 2017, Dallas, TX, USA.

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/http://dx.doi.org/10.1145/3133956.3134001

These concerns are genuine and real. For example, researchers evaluated various remote access points on vehicles and demonstrated that an attacker can exploit them to remotely compromise Electronic Control Units (ECUs) [11]. By exploiting the compromised ECUs, researchers have shown that it is feasible to remotely control or even shut down a vehicle [9–11, 13, 16, 21, 23].

Numerous schemes have been proposed to detect and/or prevent various vehicle cyber attacks [12, 19, 20, 28, 29, 31, 33]. Although these countermeasures are capable of determining whether or not there is an intrusion in the in-vehicle network, they cannot determine *which* ECU is actually mounting the attack, i.e., incapable of *attacker identification*. This is because in-vehicle networks are mostly configured as broadcast buses and their messages lack information on the transmitters. An accurate attacker identification, however, is imperative as it provides a swift pathway for forensic, isolation, security patch, etc. No matter how well an Intrusion Detection System (IDS) detects the presence of an intrusion in a vehicle, if we still do not know which ECU is mounting the attack and hence which ECU to isolate/patch, the vehicle remains insecure and unsafe. It is much better and more economical to isolate/patch the attacker ECU, than blindly treating *all* ECUs as (possible) attackers.

To meet this essential need for attacker identification — that existing solutions have not yet been able to satisfactorily meet — we propose a novel scheme, called Viden (Voltage-based attacker identification), which fingerprints message transmitter ECUs on Controller Area Network (CAN) via voltage measurements and thus facilitates attacker identification. Of the various in-vehicle network protocols, we focus on CAN as it is the *de facto* standard for in-vehicle networks and its adoption has been mandated in all cars manufactured since 2008 [5]. The rationale behind using voltage for fingerprinting ECUs is the existence of small inherent discrepancies in different ECUs' voltage outputs when they inject messages. To capture this and then use it to fingerprint the transmitter ECUs, Viden first monitors the output voltages from the two dedicated wires on the CAN bus: CAN-High (CANH) and CAN-Low (CANL). All ECUs' transceivers are connected to, and use these for their message transmissions and receptions. Through the acquired voltage measurements for each message ID, Viden first learns the *ACK threshold*, the key information Viden uses to discard the measurements of voltages outputted by ECUs while acknowledging the receipt of, but not transmitting, the message. Viden utilizes the thus-derived ACK threshold to learn the voltage output behavior of each in-vehicle ECU by constructing new features called *voltage instances*. Then, it transforms those instances to the transmitter ECU's voltage profile (i.e., *fingerprint*) via Recursive Least Square (RLS) algorithm, an adaptive signal processing technique. As a result, Viden utilizes the derived voltage profiles for an accurate attacker identification. Through experimental evaluations on a CAN bus prototype and on two real vehicles, we show that

the constructed voltage profiles are distinct for different ECUs, thus validating Viden's capability of identifying the attacker ECU.

While there have been proposals to fingerprint ECUs with timing [12] or voltage (like Viden) measurements [14, 27], their practicality and efficiency in identifying the attacker ECU remain limited to only certain attack scenarios, mainly because they were designed for intrusion detection, not attacker identification. In other words, there are many scenarios in which existence of an attack is detected but the attacker cannot be identified correctly. Thus, we design, implement, and evaluate Viden by focusing on attacker identification via a distinct way of fingerprinting ECUs from the existing schemes. As a result, Viden is efficient and easy to deploy on any ECU, thanks to its *adaptability* and *practicality*.

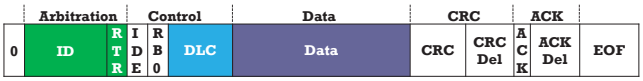
Adaptability. Existing voltage-based fingerprinting uses supervised batch learning that generates a norm model by learning from a pre-defined training data set [14, 27]. So, until the training data set and hence models/fingerprints are updated again, the norm models remain unchanged. Such an approach, however, cannot adapt the norm models to unexpected changes (e.g., changes in temperature) inside/outside the vehicle. More importantly, adversaries who intentionally generate changes can evade these existing fingerprinting schemes. Viden takes a very different approach from them in that it models and updates the voltage-based fingerprints by applying adaptive signal processing (i.e., *online* (not batch) learning) to its new set of features: voltage instances. This enables Viden to correctly modify the fingerprints and hence adapt to inevitable but unpredictable changes in vehicles that can either occur naturally (due to the mother nature) or be intentionally triggered by an intelligent adversary. Such adaptability is essential for vehicle security.

Practicality. Unlike the existing voltage-based fingerprinting schemes, the unique approach taken by Viden eliminates the requirement/assumption of using a specific CAN message type or CAN bus speed, thus facilitating its deployment. Moreover, it does not require any knowledge of which message fields the voltages are measured on, i.e., *message-field-agnostic*. This enables Viden to achieve its goal even with a low voltage sampling rate, thus lowering cost. Furthermore, even though it is message-field-agnostic, since Viden filters out undesired samples using its derived ACK thresholds, there is no need to impose restrictions on which fields of the message should be sampled to run Viden. All of these salient features enable Viden to run without re-designing current CAN controllers and make Viden very practical and cost-efficient, which is very important for the cost-conscious automotive industry.

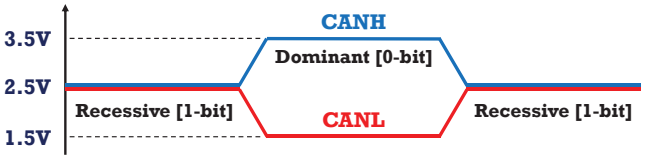
We have implemented and evaluated Viden on a CAN bus prototype and on two real vehicles. Our evaluation results show that Viden can identify the attacker ECU with a low false identification rate of 0.2%, thanks to its unique fingerprinting that makes it adaptive to handle various attack scenarios.

This paper makes the following main contributions:

(1) Proposal of a new scheme which retains only the voltage measurements output by the transmitter ECU (Section 3.4);
(2) Design of Viden which constructs voltage profiles, i.e., fingerprints, by modeling the norm voltage output behaviors of in-vehicle ECUs and exploits them for accurate identification of the attacker ECU (Sections 3.5–3.8);



(a) Format of a standard CAN data frame.



(b) CAN output voltages when sending a message.

Figure 1: Message transmission via outputting voltages.

(3) Implementation and demonstration of Viden on a CAN bus prototype and on two real vehicles (Section 4).

2 BACKGROUND

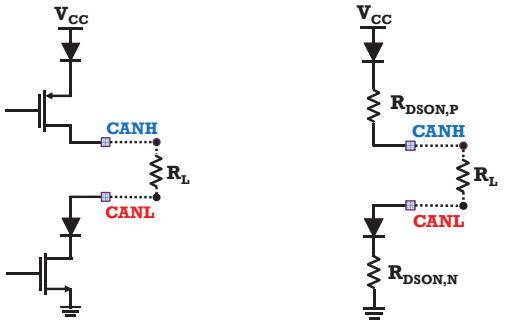
2.1 CAN Message Transmission

In-vehicle ECUs broadcast their retrieved sensor data via a CAN frame/message. Instead of carrying the address of the transmitter/receiver, as shown in Fig. 1a, it contains a unique identifier (ID), which represents its priority. Starting from a 0-bit followed by a sequence of dominant (0) or recessive (1) bits, all fields within the CAN frame are sent on the bus by the "transmitter ECU" except for the Acknowledgment (ACK) slot. The ACK slot is, in fact, used by *all* ECUs *at the same time* — except for the transmitter ECU — that have correctly received the preceded fields of the ACK slot, regardless of whether they are interested in their content or not. If correctly received, those ECUs send a 0-bit in the ACK slot. Thus, multiple ECUs acknowledge the message simultaneously, even before the transmitter finishes sending its message on the bus.

To send either a 0- or 1-bit, CAN transceivers (are agreed to) output certain voltage levels on the two dedicated CAN wires: CANH and CANL. As shown in Fig. 1b, to issue a 0-bit on the CAN bus, CAN transceivers (are agreed to) output approximately 3.5V on CANH and 1.5V on CANL so that the differential voltage becomes approximately 2V. On the other hand, when sending a 1-bit, the transceivers output approximately 2.5V on both CANH and CANL, yielding a differential voltage of approximately 0V [2, 8]. So, by measuring the differential voltage of CANH and CANL, receiver ECUs read the streams of 0 and 1 bits, and thus receive the message. From this perspective, CAN is a *differential bus*.

CAN transceivers output the intended voltages by simultaneously switching on/off their transistors. Fig. 2a shows an equivalent schematic of a CAN transceiver [6, 7]. Note that CAN transceivers of multiple ECUs are connected to the CAN bus in parallel, thus sharing the same load resistance R_L, which is normally set to 60Ω [3]. The high- and low-side output circuits consist of a series diode and a P- and N-channel transistor, respectively.

For the transceiver to send a 1-bit, both the high and low side transistors are switched *off* and are thus in a high impedance state. This results in negligible current flowing from V_{CC} to ground, yielding negligible differential voltage on CANH and CANL. On the other hand, when sending a 0-bit, both transistors are turned *on* and are thus in a low impedance state. When the transistors are on, they can



(a) Transceiver schematic. (b) When sending a 0-bit.

Figure 2: Output schematics of a CAN transceiver.

be equivalently described as resistors with drain-to-source on-state resistance R_{DSON} as shown in Fig. 2b, where current flows from V_{CC} to ground through R_L and thus creates a differential voltage of (approximately) 2V between CANH and CANL. This way, the CAN transceivers are capable of outputting either 0 or 2V of differential voltage on the two CAN wires.

2.2 Related Work

Researchers have attempted to fingerprint ECUs in various ways, mostly for the purpose of intrusion detection.

A clock-based intrusion detection system (CIDS) was proposed in [12] to detect intrusions by fingerprinting ECUs on CAN. CIDS derived the fingerprints by extracting the ECUs' *clock skews* from message arrival times. While the main objective of CIDS was to detect intrusions, the authors of [12] mentioned that the thus-derived fingerprints may also be used for attacker identification, but only when attack messages are injected *periodically*. In other words, if the attacker transmits messages *aperiodically*, then CIDS cannot identify the attacker ECU, i.e., the adversary can evade CIDS as far as attacker identification is concerned. `Viden` takes an entirely different approach: looking at attack messages from the perspective of ECUs' *output voltages* on CAN. This allows `Viden` to accurately identify the attacker ECU irrespective of how and when the attacker injects its messages, which is crucial for attacker identification.

Instead of fingerprinting ECUs based on message timings, as in `Viden`, some researchers also proposed to fingerprint them via voltage measurements. The authors of [27] used the Mean Squared Error (MSE) of voltage measurements as fingerprints of ECUs. However, they were shown to be valid only for the voltages measured during the transmission of CAN message IDs, and more importantly when voltages were measured on a low-speed (10Kbps) CAN bus; this is far from contemporary vehicles that usually operate on a 500Kbps CAN bus.

To overcome these difficulties, researchers proposed to extract other time and frequency domain features of voltage measurements (e.g., RMS amplitude) and use them as inputs for classification; more specifically, supervised (batch) learning algorithms (e.g., SVM) [14]. This way, they were able to fingerprint ECUs with enhanced accuracy and was successful on high-speed CAN buses. However, this solution was neither practical nor attractive for attacker identification for the following reasons. First, it required not only a high sampling rate (2.5 GSamples/sec), but also the use of the *extended* CAN frame format with 29-bit IDs, which is seldom used (due to its bandwidth waste) in contemporary vehicles; most vehicles use the *standard* format with 11-bit IDs. Moreover, since the modeling was done via batch learning, unpredictable changes in the CAN bus (e.g., temperature, battery level) and adversary's behaviors can lead to false identifications. These will be detailed later when we discuss the details of `Viden`.

In contrast, `Viden` fingerprints ECUs very differently and hence achieves effective attacker identification (1) through online update of fingerprints via adaptive signal processing to provide adaptability; (2) at a low sampling rate (50 KSamples/sec); and more importantly, (3) without imposing restrictions on the type of CAN message or the speed of CAN bus to be used. As a result, the deployment of `Viden` in legacy and new vehicles will be much easier.

3 VIDEN

Attacker identification is essential for expedited forensic, isolation, and security patches, all of which are the key requirements for vehicle safety. To meet this need, we propose a novel fingerprinting scheme, `Viden`, that exploits small inherent discrepancies in different ECUs' voltage outputs. Before delving into the inner workings of `Viden`, we first describe the system and threat models.

3.1 System and Threat Models

3.1.1 System Model. The vehicle's CAN bus under consideration is assumed to have been equipped with an IDS as well as a timing- (e.g., CIDS [12]) and voltage-based (e.g., schemes in [14, 27] or `Viden`) fingerprinting device; the latter complements the former via attacker identification. We discern a fingerprinting device from an IDS based on the fact that the IDS detects the *presence* of an attack whereas the fingerprinting device identifies the *source* of the (detected) attack. An attack can be mounted by the adversary who has control of a physically/remotely compromised ECU. In our system model, however, we consider such an ECU to have been *remotely* compromised and thus controlled by the adversary as in [11, 25]. We do not consider a compromised device which was attached to the in-vehicle network (e.g., device plugged in the OBD-II port), as it requires physical access and its identification has been addressed elsewhere [14, 32]. So, the compromised ECU we consider is one of those originally installed on the vehicle's CAN bus.

3.1.2 Threat Model. By injecting fabricated attack messages through his compromised ECU, the attacker can control the vehicle maneuver. We consider the attacker to be smarter than this: beyond just controlling the vehicle, the attacker's goal is to also hide the identity of the ECU injecting the attack messages. That is, while the deployed IDS may detect the presence of an attack, the adversary tries to *evade* the fingerprinting device, i.e., prevent it from determining the source of the attack. For evasion, the adversary can perform two different impersonations when injecting his attack messages:

- *Arbitrary impersonation*: The attacker misleads the fingerprinting device to think that some *arbitrary* ECU other than himself is the attacker.
- *Targeted impersonation*: The attacker acts smarter by impersonating a *targeted* ECU for evasion, i.e., make the fingerprinting device believe that the targeted ECU is the attacker.

Depending on the adversary's capabilities and knowledge of different defense schemes (available in the market or in literature) as well as their operation, his approach to evading a fingerprinting device would be different. Specifically, based on whether the adversary is aware of the fact that an in-vehicle ECU can be fingerprinted via timing and/or voltage measurements, his best effort in achieving his goal would be different. Thus, we consider three different types of adversaries: *naive*, *timing-aware*, and *timing-voltage-aware* adversaries.

While all attackers are capable of injecting and sniffing messages on the CAN bus, a *naive* adversary does not have any knowledge of how ECUs can be fingerprinted (either via timing or voltage), due possibly to lack of his technical expertise or curiosity. Thus, the naive adversary injects his attack messages imprudently at arbitrary times with forged message IDs (for impersonation).

An intelligent adversary, however, might know how ECUs can be fingerprinted via timing analysis. Thus, the adversary uses his knowledge to evade any (possibly-installed) fingerprinting scheme as much as possible as follows. The adversary logs CAN traffic, learns the timing behavior of other ECUs, and exploits the learned information in injecting attack messages at the appropriate (learned) times so as to imitate other ECUs' timing behavior. This way, the adversary can perform arbitrary/targeted impersonation and thus attempt to evade the fingerprinting device. We refer to this adversary as a *timing-aware* adversary.

The adversary might also have knowledge of how ECUs can be fingerprinted via voltage and timing measurements. Hence, when injecting attack messages, such an adversary may try to exploit his knowledge in impersonating other ECU(s) and thus evade any fingerprinting device as much as possible. We call this adversary a *timing-voltage-aware* adversary. We consider such an adversary to be capable of changing his voltage outputs via running battery draining processes, changing the supply voltage level, or by heating up or cooling down the ECU. Although he can change them to a certain level, we consider him to be incapable of *precisely* controlling their instantaneous values. This is reasonable as precise control of voltages would require, for example, control of even the ambient temperature. By changing his ECU's voltage outputs to a certain level in which the targeted ECU is outputting, a timing-voltage-aware adversary can perform a targeted impersonation. Similarly, he can arbitrarily change the output levels for arbitrary impersonation. Since changing his voltage levels (either before or during message injections) does not necessarily imply that he is attacking the CAN bus, in this paper, we differentiate "impersonation" from an actual attack of message injections. In addition, the adversary might even know when the voltage-based fingerprints are updated (if not updated in real time) and thus use that as a reference in determining when to perform arbitrary/targeted impersonation. Note, however, that he must "play" within the setting boundaries of the given CAN bus. For example, the attacker cannot control/tune

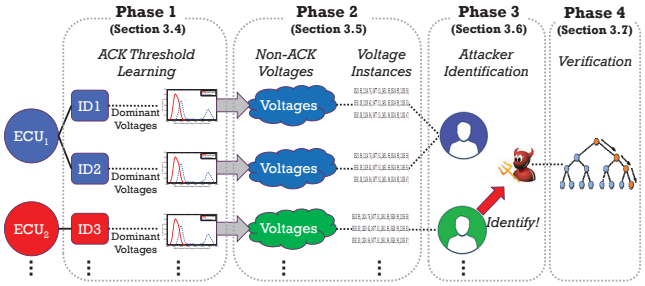


Figure 3: An overview of Viden.

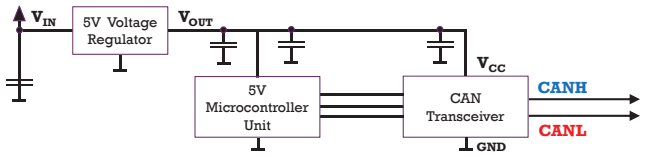


Figure 4: CAN typical application schematic.

the values of resistors within the CAN bus in order to control the voltage levels, as this requires physical access.

3.2 High-Level Overview of Viden

As shown in Fig. 3, Viden fingerprints ECUs via voltage measurements and achieves attacker identification in four phases.

Phase 1: Viden measures the CANH & CANL voltages and maps the recently acquired values to the ID of the message it has just received through the ECU's receive buffer. Then, for that message ID, Viden learns its ACK threshold. This threshold helps Viden determine whether or not the measured voltage originates from the actual message transmitter. Phase 1 is run in the initialization step of Viden and when an update is necessary.

Phase 2: Exploiting the learned ACK threshold, Viden selects voltages that are outputted solely by the message transmitter. Then, Viden uses them to derive a *voltage instance*, which is a set of features that reflect the transmitter ECU's voltage output behavior. Phase 2 and onwards are run iteratively.

Phase 3: Viden uses every newly derived voltage instance to update the voltage profile of the message transmitter. When an attack is detected by the IDS, Viden constructs a voltage profile for the attack messages and maps that profile to one of those Viden has, thus identifying the attacker ECU.

Phase 4: The results from Phase 3 are verified further via multi-class classification, only when necessary.

For a given message ID, only one ECU is assigned for its transmission in most cases. Thus, for now we consider the relationship between the numbers of ECUs, IDs, and voltage profiles to be 1, $N(\geq 1)$, and 1, respectively. We will discuss further in Section 5 on how Viden deals with cases where the relationship between the numbers of ECUs and IDs might be N and 1, respectively.

3.3 CANH and CANL Voltage Outputs

Before presenting the details of Viden, we first discuss which voltage characteristics of ECUs it exploits for attacker identification.

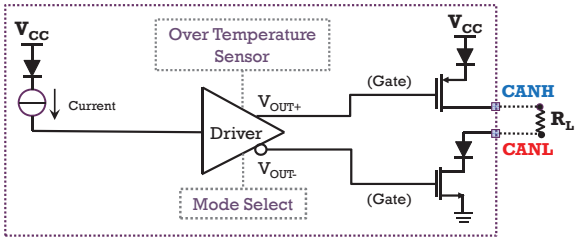


Figure 5: Transistors' gate voltages are fed by the driver.

Variations in supply and ground voltages. Fig. 4 shows a typical ECU connection to CAN [6, 7]. In order to output the desired voltage levels on CANH and CANL, transceivers are powered with the nominal supply voltage (V_{CC}) of 5V, which is provided and maintained by a voltage regulator. The input of the regulator, V_{IN}, comes from a power supply, i.e., a 12V/24V battery powering all the ECUs [22]. Not only the voltage regulator but also the connected bypass capacitors help stabilize the V_{CC} level. However, the output voltage of an ECU's regulator varies *independently* and *differently* from other ECUs' regulators, as their supply characteristics are different (e.g., different regulators' common-mode rejection ratios). Thus, there are inherent, small but non-negligible differences in ECUs' V_{CC}. There exist variations in not only V_{CC} but also in the ground voltage since there does not exist a perfect ground [3].

For these reasons, CAN transceivers are built to operate over a range of voltages (e.g., TI TCAN10xx devices are designed to handle 10% supply variations [7]). This guarantees transceivers with different V_{CC} and/or ground to communicate messages correctly.

Variations in on-state resistance. When transceivers send a 0-bit, their two transistors are turned on so that the flowing current generates the required differential voltage between CANH and CANL. In such a case, transistors in the transceivers are considered as resistors $R_{DSON,P/N}$ (see Fig. 2b). Although transceivers are designed to have the same $R_{DSON,P/N}$ values, process/manufacturing variations/imperfections cause transistors' $R_{DSON,P/N}$ values to be slightly different from each other [26].

Fig. 5 shows a typical circuit diagram of a CAN transceiver. Transistors' R_{DSON} values are inversely related to their gate voltages, which are supplied by a driver, i.e., a fully differential amplifier [4, 17]. Interestingly, since the driver input is affected by V_{CC}, which also varies with ECU, the transistors' gate voltages are also affected by V_{CC}. Therefore, variations in V_{CC} lead to variations in transistors' actual R_{DSON} values. In summary,

$\mathbb{V}1$. *There exist differences/variations in CAN transceivers' nominal supply voltage, ground voltage, and $R_{DSON,P/N}$ values, especially during the transmission of a 0-bit.*

When transmitting a 1-bit, the two transistors are simply turned off and thus there is little voltage variation between nodes. Hence, we do not consider any voltage measurements when the transmitter was sending a 1-bit. Instead, we only consider those measured when it was sending a 0-bit, and refer to those as *dominant voltages*.

Variations in dominant voltages. From Fig. 2b, when transceiver i is transmitting a 0-bit, the current, $I_{(i)}$ flowing from its $V_{CC(i)}$ to its ground can be derived as $I_{(i)} = \frac{V_{CC(i)} - V_{G(i)} - 2V_D}{R_{DSON,P(i)} + R_{DSON,N(i)} + R_L}$, where

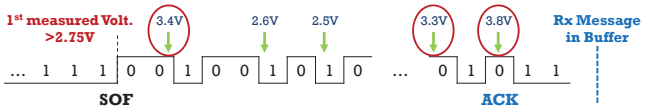


Figure 6: `Viden` measuring CANH voltages.

$V_{G(i)}$ denotes its ground voltage, and V_D the diodes' forward bias (assuming they are equivalent). To simplify the analysis, we omit other factors such as leakage current or variations in diodes. We can thus derive the CANH and CANL dominant voltages, $V_{CANH(i)}$ and $V_{CANL(i)}$, from transceiver i as:

$$\begin{aligned} V_{CANH(i)} &= V_{CC(i)} - V_D - I_{(i)} R_{DSON,P(i)}, \\ V_{CANL(i)} &= V_{G(i)} + V_D + I_{(i)} R_{DSON,N(i)}. \end{aligned} \quad (1)$$

From Eq. (1), one can see that

$\mathbb{V}2$. *Variations in V_{CC}, ground, and $R_{DSON,P/N}$ result in different ECUs with different CANH and CANL dominant voltages.*

For this reason, the ISO11898-2 specifies that a compliant transceiver must accommodate dominant voltages of CANH=2.75~4.5V and CANL=0.5~2.25V [8]. Hence, we refer to any voltage values meeting this requirement as *dominant voltages*.

Transient changes in on-state resistances. In Fig. 5, when V_{OUT+} of the driver increases, V_{OUT-} concurrently decreases as they are differential outputs. So, for both transistors, the absolute differences between their gate and source voltages simultaneously decrease. This results in both $R_{DSON,P}$ and $R_{DSON,N}$ to increase, i.e., change in the *same* direction [4, 17]. Even when a change in the ECU temperature affects $R_{DSON,P}$ & $R_{DSON,N}$, they change in the same direction. So, for a given V_{CC} and ground voltage, the opposite signs of $I_{(i)}R_{DSON,P/N(i)}$ in (1) indicate that

$\mathbb{V}3$. *Transient changes in the ECU temperature and driver's input/output affect $R_{DSON,P/N}$, and thus make V_{CANH} and V_{CANL} temporarily deviate in the "opposite" direction.*

Since regulated V_{CC} and ground voltage remain constant, and are not affected by transient changes in $R_{DSON,P/N}$,

$\mathbb{V}4$. *Transient changes in V_{CC} and ground are significantly smaller than those in V_{CANH} and V_{CANL}, i.e., their values remain relatively constant.*

$\mathbb{V}1$–$\mathbb{V}4$ indicate that CANH and CANL dominant voltages of each ECU are different from each other. `Viden` exploits this fact in constructing different voltage profiles for (fingerprinting) ECUs.

3.4 Phase 1: ACK Threshold Learning

`Viden` is designed to run with a low voltage sampling rate so that it can be easily installed as a low-cost software application, which requires no changes in the CAN protocol; the high rate of voltage sampling would only be required for the CAN protocol to receive messages as it is designed to be. Such a feature, however, renders `Viden` incapable of determining at which slot the voltage values were measured; all it knows is the value. Thus, `Viden` goes through a phase of learning the *ACK threshold*, which determines whether or not the measured voltage was outputted by the message transmitter.

Measuring dominant voltages. `Viden`'s measurement is triggered whenever a CANH voltage exceeds 2.75V after a certain idle period. This is because the first measured voltage exceeding 2.75V

represents the case of some transmitter transmitting a 0-bit on the bus [8]. Since Viden is only interested in dominant voltages, it discards any measurements that are lower than 2.75V on CANH and higher than 2.25V on CANL. The measurement continues until some message is shown to have been received into Viden's receive message buffer, i.e., an indication that the transmitter has finished sending a message. By reading the ID value of that received message, Viden knows which message ID the acquired dominant voltages correspond to.

Non-ACK voltages. Viden continues collection of more dominant voltages for the acquired ID (whenever the message is received) until it learns its CANH and CANL ACK thresholds. When collecting and exploiting voltage measurements, one needs to be cautious of the fact "During the ACK slot of a transmitted message, if received, all other nodes but its transmitter output a 0-bit on the CAN bus" [1]. Thus, even though Viden samples at least a few dominant voltages while receiving a certain message, *not all* represent the outputs from the actual message transmitter. Fig. 6 shows an example of Viden's five voltage measurements of {3.4V, 2.6V, 2.5V, 3.3V, 3.8V} from the CANH line during the reception of a message, where 3.8V was measured during the ACK slot. Of them, Viden discards measurements {2.6V, 2.5V} as they do not meet the criteria of dominant voltages. If Viden had considered the remaining 3 measurements as if they were output by the message transmitter, it would have been incorrect since 3.8V was from all ECUs but the message transmitter in the ACK slot. Therefore, to accurately fingerprint the transmitter ECU, Viden derives the ACK threshold which distinguishes a non-ACK voltage measurement from an ACK voltage measurement. We refer to *non-ACK voltages* as dominant voltages measured from slots other than the ACK slot, and *ACK voltages* as those measured from the ACK slot. The threshold is derived by exploiting the following two facts of the ACK voltage.

- $\mathbb{K}1$. *Low probability*: Since ACK is only 1 bit long, when measuring dominant voltages during a message reception, most of them would be outputted from the message transmitter.
- $\mathbb{K}2$. *Different voltage level for ACK*: During an ACK slot, all ECUs but the transmitter acknowledge their message reception. Since those responders are connected in parallel and turned on concurrently, when receiving the ACK, the measured voltages are much higher on CANH and much lower on CANL than those when receiving non-ACK bits.

Viden exploits these facts to collect M dominant voltages from both CANH and CANL for N rounds for a given message ID. So, based on $\mathbb{K}1$, the *most frequently* measured voltage value (of the M values) will most likely represent the non-ACK voltage. During the N rounds, we refer to the set of N most frequently measured values as the *most frequent set*, S_{freq}. On the other hand, if we were to determine the *maximum* and the *minimum* of the M values from CANH and CANL, respectively, then they would represent ACK as well as non-ACK voltages. This is because even a single dominant voltage value collected (without awareness) from the ACK slot would become the maximum/minimum of the M values due to $\mathbb{K}2$. Here, the set of N maximum/minimum values measured from CANH/CANL is defined as the *maximum/minimum set*, denoted as $S_{max/min}$. For each message ID, Viden exploits sets S_{freq} and $S_{max/min}$ to derive the ACK threshold that differentiates a non-ACK voltage from an ACK voltage.

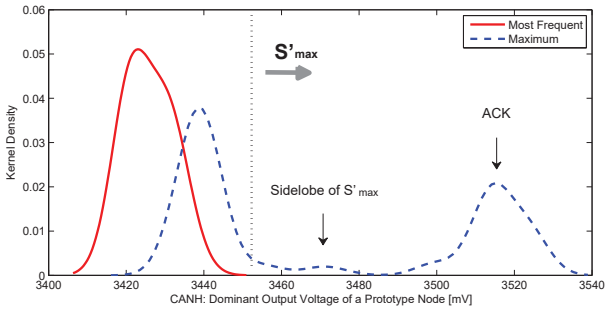


Figure 7: ACK threshold in a CAN bus prototype.

Derivation of ACK threshold. Fig. 7 shows the kernel density plots of the most frequent and the maximum sets of the measured dominant voltages from the CANH line. The measurements were made while running Viden on our CAN bus prototype, which will be detailed in Section 4. One can see that only for the maximum set, there exists a side lobe, whereas the most frequent set resembles a Gaussian distribution. Note that during the N rounds of M measurements each, the most frequent and the maximum values can be different. Thus, from the maximum set, Viden first discards values lower than $\max(S_{freq}) + B\sigma_{S_{freq}}$, where $\sigma_{S_{freq}}$ is the standard deviation of set S_{freq}, and B a design parameter determining how aggressive one wants to be in discarding ACK voltages. Note that such a value also represents the rightmost end-point of the most frequent set's kernel density (e.g., dotted vertical line in Fig. 7). Then, the usual side lobe of the maximum set (S_{max}) becomes the main lobe of a refined maximum set, S'_{max}. From S'_{max}, Viden determines $\Gamma_1 = median(S'_{max}) - 3MAD(S'_{max})$ and $\Gamma_2 = \mu_{S'_{max}} - 3\sigma_{S'_{max}}$, where $MAD(x)$ denotes the median absolute deviation of x, and μ_x its mean. The CANH ACK threshold of the given message ID (or its transmitter), Γ^H_{ACK}, is then derived to be $\max(\Gamma_1, \Gamma_2)$. We take the maximum of the two to be conservative in discarding any non-ACK voltages. Moreover, not only the lower 3σ limit but also the lower 3-MAD limit is used since the refined maximum set S'_{max} may still contain its own (new) side lobe as shown in Fig. 7, i.e., an outlier for S'_{max}. Using these processes, the ACK threshold of the example in Fig. 7 is determined to be $\Gamma^H_{ACK} = 3.499$V — a point where the two lobes in the maximum set are separated. Depending on the transmitter ECU, the ACK threshold can be different as the set of responders is different. Thus, the ACK learning is performed for all message IDs of interest.

When deriving the CANL ACK threshold, Γ^L_{ACK}, the minimum (instead of the maximum) and the upper (instead of the lower) limits are used. In Appendix A, we show that the proposed scheme can correctly determine the ACK thresholds even in real vehicles.

3.5 Phase 2: Deriving a Voltage Instance

Once ACK thresholds, Γ^H_{ACK} and Γ^L_{ACK}, of the given message ID are learned, from that point and on, Viden continuously collects dominant voltages, but discards those from CANH that are lower than 2.75V or higher than Γ^H_{ACK}, and those from CANL that are

Algorithm 1 Dispersion Update

```
1: function UpdateDispersion(V, Λ, P*)
2:     return Λ ← Λ + α(P* − #(V<Λ)/#V)^3        ▷ Adjust tracking position
3: end function
4: if #measured CANH and CANL voltages both ≥ κ then
5:     V_H, V_L ← {past κR CANH, CANL measurements}
6:     F_3 ← UpdateDispersion(V_H, F_3, 0.75)
7:     F_4 ← UpdateDispersion(V_L, F_4, 0.25)
8:     F_5 ← UpdateDispersion(V_H, F_5, 0.9)
9:     F_6 ← UpdateDispersion(V_L, F_6, 0.1)
10: end if
```

higher than 2.25V or lower than Γ^L_{ACK}. This way, Viden selects and processes only *non-ACK voltages*. Whenever Viden obtains κ new measurements of CANH and CANL non-ACK voltages, Viden derives a new *voltage instance* which is defined as the set of 6 tracking points, $\mathbf{F_1}$–$\mathbf{F_6}$.

F_1–F_2: Most frequent values. Similarly to Phase 1, Viden determines the most frequently measured CANH and CANL voltages (from κ values), which are denoted as F_1 and F_2, respectively. Since Viden knows the ACK thresholds, the main differences from Phase 1 are that only non-ACK voltages as well as κ ($< M$) of them are used in deriving the most frequent values. This way, Viden keeps track of the *median* of the transmitter's dominant voltages.

F_3–F_6: Dispersions. Viden also keeps track of the *dispersions* of CANH and CANL dominant voltages. As the transmitter's voltage output behavior can change over time, Viden continuously updates 4 different *tracking points*, F_3–F_6, which reflect (1) F_3: 75th, (2) F_5: 90th percentile of the transmitter's CANH outputs, (3) F_4: 25th, and (4) F_6: 10th percentile of CANL outputs. By tracking the transmitter's voltage distribution, Viden understands its *momentary* voltage output behavior. Thus, voltage instances represent those momentary behaviors. Since even a single ACK voltage can significantly distort Viden's understanding of transmitters' behaviors, it is important to learn the ACK threshold. The reasons for Viden's tracking of different percentiles of CANH and CANL are that the low percentiles of CANH would contain voltages measured when the transmitter switches from sending a 1-bit to sending a 0-bit, and vice versa. The same applies for the high percentiles of CANL measurements. Although other percentiles can be tracked as well, to minimize Viden's overhead, we only track F_3–F_6.

Algorithm 1 describes how the tracked dispersions are updated whenever Viden acquires κ dominant voltages from each of CANH and CANL. Using the past κR measurements, as in line 2, Viden roughly estimates what percentile the current tracking point, Λ, represents. In Viden, we set $R = 10$. Then, to correct and thus move the tracking point Λ to the desired position — where it represents the P^* percentile — an adjustment is made as in line 2, where α is a design parameter determining the sensitivity to changes. With the adjustment function proportional to $(P^* - \frac{\#(V<\Lambda)}{\#V})^3$, the tracking points move faster if they are far away from their desired positions. As a result, the four tracking points move if the transmitter's voltage distribution (i.e., output behavior) shows changes, thus adapting to any changes on the CAN bus. Instead of tracking, it is also possible to directly derive the percentiles from the κR values. Viden, however, does not follow this since it is too sensitive to transient changes, especially when κR is small, i.e., insufficient samples in deriving the percentiles. Thus, in order to make Viden work under various circumstances, we *track* them instead.

3.6 Phase 3: Attacker Identification

A voltage instance (F_1–F_6) represents the momentary voltage output behavior of the message transmitter. So, to log its usual behavior, Viden exploits every newly derived voltage instance to construct/update the *voltage profile* of the message transmitter. Although the voltage instances are derived "per message ID", if messages originate from the same transmitter/ECU, their instances are near-equivalent, thus leading to construction of the same voltage profile. We will later show through evaluations that there exists only one voltage profile for a given transmitter/ECU, thus enabling its fingerprinting. By exploiting a newly derived voltage instance, Viden first updates the *cumulative voltage deviations* (CVDs) of features F_1–F_6. We define a CVD to represent how much the transmitter's dominant voltages deviated cumulatively from their ideal values. Thus, for feature F_x, the CVD at step n, $CVD_x[n]$, is updated as:

$$CVD_x[n] = CVD_x[n-1] + \Delta[n]\left(1 - v_x[n]/v_x^*\right), \qquad (2)$$

where $\Delta[n]$ is the elapsed time since step $n-1$, $v_x[n]$ the value of feature F_x at step n, and v_x^* the desired value of v_x. Ideally, the most frequently measured as well as any percentiles of the CANH and CANL dominant voltages should be equal to 3.5V and 1.5V, respectively, i.e., no variations in their output voltages. Therefore, for features $\{F_1, F_3, F_5\}$, which represent CANH values, we set $v^*_{\{1,3,5\}} = 3.5V$ and similarly we set $v^*_{\{2,4,6\}} = 1.5V$.

Suppressing transient changes. As ECUs have different V_{CC}, ground, and R_{DSON} values, they output different CANH and CANL dominant voltages. Their momentary voltage instances would, therefore, be different, and hence the trends in their CVD changes would also be different from each other. So, for every obtained CVD of features F_1–F_6, Viden derives $\Psi[n] = \sum_{x=1}^{6} CVD_x[n]$. The reason for Viden's summing of all the CVDs is to exploit $\mathbb{V}3$. Recall from Section 3.3 that $\mathbb{V}3$ gives us transient deviations in CANH and CANL output voltages are opposite in direction. So, via CVD summation, Viden *suppresses* any transient deviations that have occurred (due to changes in driver, temperature, etc.) when constructing and/or updating the voltage profiles. Note that since CAN is a differential bus, F_2, F_4, F_6 suppress F_1, F_3, F_5, respectively.

Voltage profile. Suppression of transient changes yields a value, Ψ, that (mostly) represents the *consistent* factors in the voltage instances: V_{CC}, ground voltages, and the usual voltage drops across the transistors. As stated in $\mathbb{V}4$, since these values are rather constant, the accumulated sum of Ψ, $\Psi_{accum}[n] = \sum_{k=1}^{n} \Psi[k]$ becomes linear in time. Moreover, from $\mathbb{V}1$–$\mathbb{V}2$, as Ψ values are distinct for different ECUs, the *trends* in how Ψ_{accum} changes also become different, i.e., the slopes in a Ψ_{accum}–time graph are different. Therefore, Viden formulates a linear parameter identification problem as $\Psi_{accum}[n] = \Upsilon[n]t[n] + e[n]$, where at step n, $\Upsilon[n]$ is the regression parameter, $t[n]$ the elapsed time, and $e[n]$ the identification error. As the regression parameter Υ represents the slope of the linear model and varies with the transmitter, we define this as the *voltage profile*. This way of formulating the problem and constructing the profiles facilitates Viden's online update of fingerprints, which is key to Viden's adaptability. To determine the voltage profile Υ, i.e.,

fingerprint ECUs, we use an adaptive signal processing technique, the Recursive Least Squares (RLS) [18], which is an online approach in learning the regression parameter. Note, however, that the choice of algorithm does not affect Viden's performance. In RLS, we use kiloseconds (=10^3 secs) as the unit for t. Due to space limitation, we omit details of RLS, and refer the readers to [18] for its details. We will later show, via experimental evaluations, that the thus-derived profile Υ is constant over time and also distinct for different ECUs, thus allowing Viden to correctly fingerprint them.

Identifying the attacker. When an adversary mounts an attack, the underlying IDS can determine whether the message is malicious or not, so Viden can filter out the voltage outputs obtained only from the (detected) attack messages and build a voltage profile from only those. We refer to such a voltage profile as an *intrusion voltage profile*. Viden then looks up the voltage profiles it had built until the detection of the attack and searches for the one that is similar to the intrusion voltage profile.[1] This way, Viden identifies the attacker ECU.

The performance of Viden will, of course, depend on how well the IDS detects the intrusion; this dependency needs to be investigated when an IDS and Viden are integrated as a whole system. Note, however, that the mostly periodic nature of in-vehicle messages makes correct detection of intrusions not as difficult as pinpointing the attacker ECU. Researchers and car-makers are now well aware of how to detect intrusions, but not how to accurately identify the attacker ECU.

The only case where the identified ECU would have an unknown/unlearned profile is when it was physically attached to the vehicle by an adversary. However, since this requires physical access and its identification has been addressed elsewhere [14, 32], we do not discuss its detection any further in this paper.

3.7 Phase 4: Verification

By the birthday paradox, two different ECUs may naturally have near-equivalent voltage profiles, i.e., voltage profile collision, thus confusing Viden in identifying the attacker ECU. Note, however, that Viden has at least narrowed its search scope significantly. An adversary may also attempt to mimic some other ECU's voltage output behavior, i.e., targeted impersonation. In such a case where further verification besides the voltage profiles is required, in Phase 4 of Viden, machine classifiers are run with the (momentary) *voltage instances* as their inputs, i.e., F_1–F_6 as their features. This way, an analysis of attacks from a different vantage point — not only its trend (Phase 3) but also its momentary behavior — is performed, thus resolving ambiguities in attacker identification. We, however, stress that while the adaptability achieved from Phase 3 is an essential attribute for an accurate attacker identification, Phase 4 cannot totally replace it, i.e., only complements Phase 3. We will later show through evaluations that by using voltage instances as machine classifiers' input, Viden can resolve issues such as voltage profile collision and an adversary's targeted impersonation.

[1]The initial set of "ground truth" voltage profiles can be verified via timing-based fingerprinting schemes [12, 30].

3.8 Voltage Profile Adjustment

For attacker identification, it is important to not only have the correct fingerprint of an ECU but also that fingerprint to be still valid when examining a voltage measurement obtained during the (detected) attack. If it was updated much earlier than when the attack was detected, any changes occurred between those two time instants would not be reflected in the latest model, thus leading to false identifications. We refer to this as a *model-exam discrepancy*. Since Viden continuously updates the voltage profiles in real time, such a model-exam discrepancy is minimized/nullified. Since attacker identification is performed *upon* detection of an intrusion, as long as Viden keeps the fingerprints up-to-date until an intrusion is detected (by an IDS), Viden can locate the source of the attack. Even when there are abrupt changes in the temperature of an ECU, Viden suppresses those transient changes, and adapts its model accordingly for an accurate attacker identification.

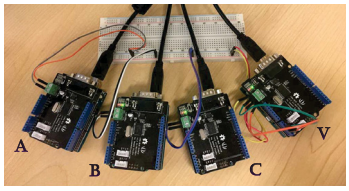
One corner case in which the performance of Viden might suffer from the model-exam discrepancy would be when the vehicle has not been turned on for a long time. During that period, various features (e.g., power supply level, ambient temperature) which affect the output voltages might have changed. In such a case, since the old voltage profiles may not correctly reflect the current status, Viden may have to reconstruct (instead of update) them. In fact, a timing-voltage-aware adversary may attempt to exploit such a fact and attack the CAN bus as soon as the vehicle is turned on, making Viden incapable of handling the attacks. However, even in such a case, as ECUs use the same power source, i.e., battery, and thus all voltage profiles change in the same direction and with the same magnitude, Viden re-adjusts and reuses the old ones as a starting point for voltage profile *update* rather than reconstructing it from scratch when the vehicle is turned on. Specifically, Viden first determines how much of *common* changes occurred in ECUs' V_{CC} by deriving the differences between the previous and current mean values of $(F_3+F_4+F_5+F_6)/2$ — an estimated value of V_{CC} based on Eq. (1). Viden then adds the thus-derived differences to v^* (in Eq. (2)) based on the fact that if common changes in V_{CC} incur, CANH and CANL output values increase simultaneously [6, 7]. This way, Viden correctly adjusts/updates its voltage profile(s) and thus identifies such a type of timing-voltage-aware adversary; we will later evaluate this via real vehicle experiments. Note, however, that if voltage-based fingerprinting was done solely via batch learning (as in [14, 27]), it cannot make such an adjustment, suffer from high model-exam discrepancy, thus allowing a timing-voltage-aware adversary to evade it.

3.9 Security of Viden

Once an intrusion is detected, via voltage measurements, Viden can identify the attacker ECU.

A *naive* adversary would be capable of controlling the vehicle via continuous message injections. However, since he has no knowledge of how ECUs might be fingerprinted, he would inject them imprudently. In such a case, he cannot evade Viden.

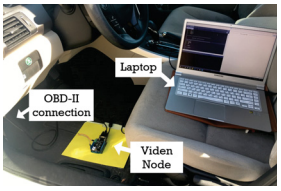
A *timing-aware* adversary who knows that ECUs can be fingerprinted via timing analysis, will attempt to exploit this knowledge in not only controlling the vehicle but also evading the fingerprinting device. For example, the adversary may know that CIDS [12]



(a) CAN bus prototype.

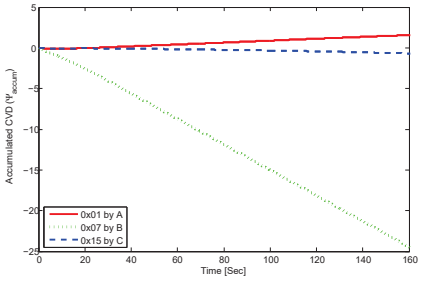
(b) 2013 Honda Accord.

(c) 2015 Chevrolet Trax.

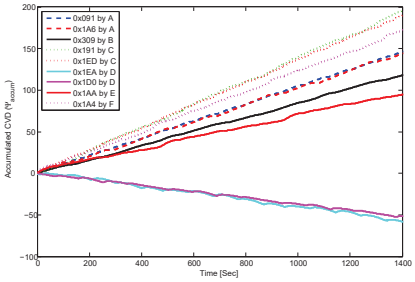


(d) Connection to the vehicle.

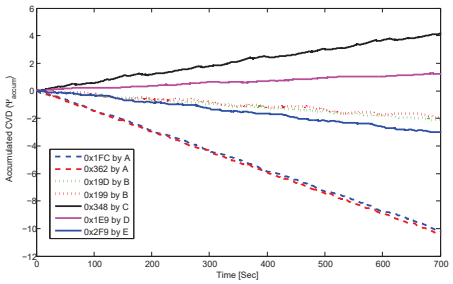
Figure 8: Experiments were conducted on a CAN bus prototype and on two real vehicles.



(a) CAN bus prototype.



(b) 2013 Honda Accord.



(c) 2015 Chevrolet Trax.

Figure 9: Voltage profiles obtained from the CAN bus prototype and the two real vehicles.

can identify the attacker ECU only if the attack messages were injected periodically. Hence, he may perform an arbitrary impersonation by injecting messages *aperiodically*, thus fooling CIDS. Note, however, that CIDS would still detect the presence of the attack. In addition, based on his knowledge that CIDS's fingerprints are basically clock skews, he may attempt to imitate the targeted ECU's clock behavior, i.e., targeted impersonation. However, with Viden also installed in the vehicle, since it identifies the attacker ECU via voltage measurements, i.e., irrespective of message timings, a timing-aware adversary can evade CIDS, but not Viden.

A *timing-voltage-aware* adversary may also try to evade Viden using his knowledge of how voltage-based fingerprinting devices run. In order to achieve this, when or before the adversary injects the attack messages, he may attempt to change the voltage output levels by changing the supply voltage (e.g., run processes which drain battery) or by heating up or cooling down the ECUs so that the transistors' internal resistance values change. He could even attempt to start attacking the CAN bus only when the vehicle is turned on after staying off for a long time as discussed in Section 3.8. However, since Viden performs an online update of voltage-based fingerprints and also adjusts them if necessary, thus minimizing/nullifying model-exam discrepancy, it would be difficult for the timing-voltage-aware adversary to evade Viden. Moreover, since Viden analyzes voltage outputs from two different perspectives — momentary behavior (Phase 4) and its trend (Phase 3) — a timing-voltage-aware adversary incapable of precisely controlling the instantaneous voltage outputs cannot evade Viden.

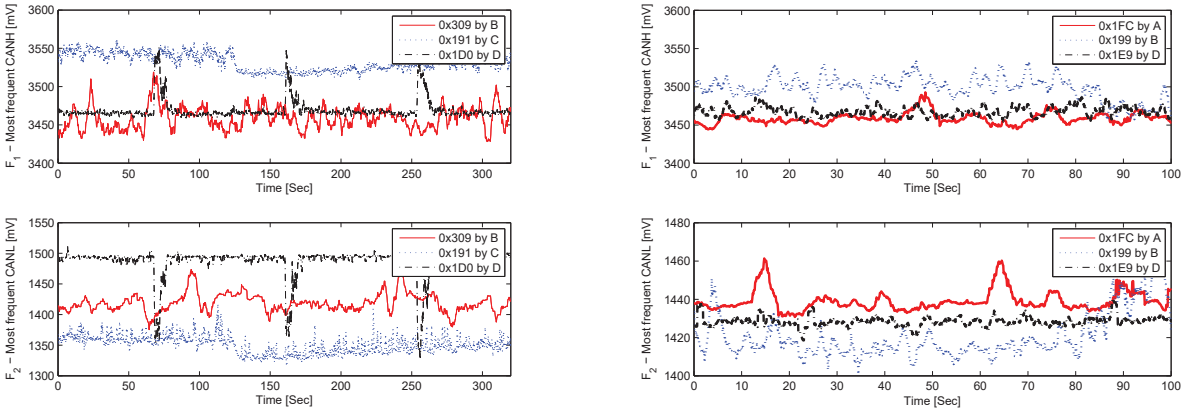
4 EVALUATION

We now evaluate the practicability and efficiency of Viden in achieving an effective and accurate attacker identification on a CAN bus prototype and two *real* vehicles. When running Viden for both evaluation settings, in Phase 1, $M = 30$ dominant voltages were obtained for each message ID for $N = 50$ rounds. From Phase 2, voltage instances were outputted whenever $\kappa = 15$ non-ACK voltages from both CANH and CANL were acquired.

4.1 Evaluation Setups

CAN bus prototype. As shown in Fig. 8a, we configured a CAN prototype in which all four nodes were connected to each other. Each node consists of an Arduino UNO board and a SeeedStudio CAN shield. The CAN bus shield consists of a Microchip MCP2515 CAN controller and a MCP2551 CAN transceiver to provide CAN bus communication capabilities. Only two nodes were configured to have a 120Ω terminal resistor so as to match $R_L = 60\Omega$.

The three prototype nodes $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$ were programmed to inject messages 0x01, 0x07, and 0x15 at random message intervals within [20ms, 200ms]. The fourth node $\mathbb{V}$ was programmed to run Viden and construct voltage profiles for messages 0x01, 0x07, and 0x15 (i.e., transmitters $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$), respectively. The reason for injecting the messages *aperiodically* is to show that even in such cases, Viden is capable of fingerprinting the transmitters. For node $\mathbb{V}$ that runs Viden, its CANH and CANL lines were not only connected to the bus but also to the microcontroller's Analog-to-Digital Converter (ADC), which had 10-bit resolution and was configured to sample voltages at its maximum rate of 50 KSamples/sec. This way, $\mathbb{V}$ acquired measurements of dominant voltages on the bus when nodes $\mathbb{A}$–$\mathbb{C}$ were sending their messages. The CAN bus prototype was set up to operate at 500Kbps, which is typical for in-vehicle high-speed CAN buses. In such settings, Viden required only 2–3 messages to output a voltage instance and update the profiles.



(a) 2013 Honda Accord. (b) 2015 Chevrolet Trax.

Figure 10: Features F_1 and F_2 of Viden in the two real vehicles.

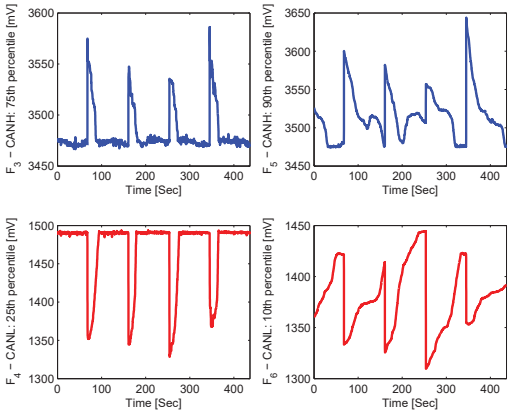


Figure 11: Changes of message 0x1D0 in the Honda Accord.

Real vehicles. Two cars, 2013 Honda Accord (Fig. 8b) and a 2015 Chevrolet Trax (Fig. 8c), were also used for our experimental evaluation of Viden. Through the OBD-II port, the Viden node ($\mathbb{V}$) was connected to the in-vehicle CAN bus, both running at 500Kbps. From a laptop and through the Viden node, as shown in Fig. 8d, we were able to read messages from the 2013 Honda Accord's and the 2015 Chevrolet Trax's CAN buses. While Viden was receiving messages from the two vehicles, it sampled their CANH and CANL voltages and then derived their ECUs' voltage instances and profiles.

4.2 Voltage Profiles as Fingerprints

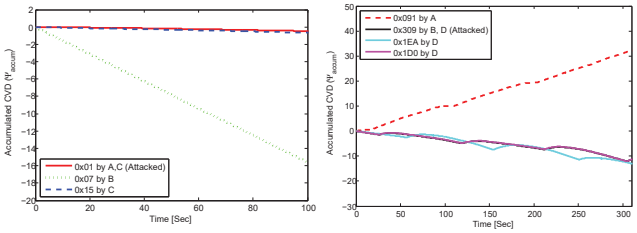
We first evaluate the accuracy and validity of using voltage profiles to fingerprint the transmitter ECUs.

CAN bus prototype. Fig. 9a shows the voltage profiles of all the three messages sent on the prototype bus. Although the three CAN prototypes nodes were built with the same hardware, the corresponding message IDs showed different trends in how their Ψ_{accum} changed over time, since the three ECUs differ in their supply and transistor characteristics. Based on the RLS implemented in Viden, we were able to find that nodes $\mathbb{A}$, $\mathbb{B}$, and $\mathbb{C}$ had different voltage profiles (Υ) being equal to 10.1, -154.3, and -4.9, respectively. In other words, voltage profiles of 0x01, 0x07, and 0x15 were shown to be different from each other as they were sent by different ECUs, thus verifying the feasibility and accuracy of Viden.

Real vehicles. In the CAN prototype, we knew which ECU is sending which message(s), but it is difficult to know this in a real vehicle. In order to obtain the ground truth on the message source(s), we exploit the schemes in [12, 30], which analyzed timing patterns in CAN for fingerprinting the ECUs. Note, however, that these are used only for obtaining the ground truth, since those cannot identify the attacker ECU if messages are injected at random times.

Through the connected Viden node, we not only logged the CAN traffic of the 2013 Honda Accord but also measured the dominant voltages from its CAN bus. The measurements were made on a stationary vehicle, but while continuously changing their operations (e.g., pressing brake pedal, turning the steering wheel) to generate some transient changes. In Appendix B, we show that outputs in Viden is *not* affected by whether the car is being driven or stationary. By logging the CAN traffic and exploiting the schemes in [12, 30], we were able to verify that messages {0x091, 0x1A6} were sent from some ECU $\mathbb{A}$, {0x309} from $\mathbb{B}$, {0x191, 0x1ED} from $\mathbb{C}$, {0x1EA, 0x1D0} from $\mathbb{D}$, {0x1AA} from $\mathbb{E}$, and {0x1A4} from $\mathbb{F}$. Fig. 9b shows the messages' voltage profiles. The profiles (Υ) derived by Viden are shown to be equivalent *only* for those messages sent from the same ECU; ECU $\mathbb{A}$ sending {0x091, 0x1A6} had $\Upsilon_{\mathbb{A}} = 102.6$, $\mathbb{B}$ sending {0x309} had $\Upsilon_{\mathbb{B}} = 85.0$, $\mathbb{C}$ sending {0x191, 0x1ED} had $\Upsilon_{\mathbb{C}} = 137.0$, $\mathbb{D}$ sending {0x1EA, 0x1D0} had $\Upsilon_{\mathbb{D}} = -39.2$, $\mathbb{E}$ sending {0x1AA} had $\Upsilon_{\mathbb{E}} = 67.5$, while $\mathbb{F}$ sending {0x1A4} had $\Upsilon_{\mathbb{F}} = 120.8$. This result again shows that voltage profiles for *different* ECUs are different and can thus be used as their fingerprints.

To further verify that Viden's capability of fingerprinting is not restricted to a specific vehicle model, Viden was also run on a 2015 Chevrolet Trax. Again, by exploiting the schemes in [12, 30], we obtained the ground truths of messages {0x1FC, 0x362} sent from some ECU $\mathbb{A}$, {0x19D, 0x199} from $\mathbb{B}$, {0x348} from $\mathbb{C}$, {0x1E9} from $\mathbb{D}$, and {0x2F9} from $\mathbb{E}$. Fig. 9c shows the result of Viden determining that {0x1FC, 0x362} have a voltage profile of $\Upsilon_{\mathbb{A}} =$



(a) CAN bus prototype. (b) Real vehicle.

Figure 12: Viden identifying a timing-aware adversary.

-14.7, {0x19D, 0x199} have $\Upsilon_{\mathbb{B}} = -2.8$, {0x348} has $\Upsilon_{\mathbb{C}} = 5.9$, {0x1E9} has $\Upsilon_{\mathbb{D}} = 1.8$, and {0x2F9} has $\Upsilon_{\mathbb{E}} = -4.4$. Thus, using voltage measurements, Viden correctly fingerprinted their transmitters. This again confirms the diversity of voltage profiles (of different ECUs), thus facilitating Viden's fingerprinting of in-vehicle ECUs. Moreover, these results show that Viden's fingerprinting is not limited to a specific vehicle model, and can thus be applied to other vehicle models.

4.3 Voltage Outputs in Real Vehicles

We provided 4 characteristics, $\mathbb{V}1$–$\mathbb{V}4$, which were imperative for Viden in fingerprinting ECUs. We evaluate whether $\mathbb{V}1$–$\mathbb{V}3$ actually hold in real vehicles. Note that Fig. 9 verifies $\mathbb{V}4$, corroborating that the voltage profiles of ECUs were constant over time, i.e., linear.

Different outputs. According to $\mathbb{V}1$–$\mathbb{V}2$, ECUs output different dominant voltages. Fig. 10a plots features F_1–F_2 (i.e., the most frequently measured CANH and CANL values) outputted by Viden for messages 0x309 (sent by $\mathbb{B}$), 0x191 (sent by $\mathbb{C}$), and 0x1D0 (sent by $\mathbb{D}$) in the Honda Accord. Although the transceivers of all those messages are to output the agreed-on CANH=3.5V and CANL=1.5V when sending a 0-bit, they outputted values deviating from them. More importantly, their output levels were clearly discriminable. Even though ECU $\mathbb{B}$, which sent 0x309, was shown to output similar CANH dominant voltages to ECU $\mathbb{D}$, it outputted totally different voltages on CANL. Similarly, Fig. 10b plots F_1–F_2 values of 0x1FC (sent from $\mathbb{A}$), 0x199 (sent from $\mathbb{B}$), and 0x1E9 (sent from $\mathbb{D}$) outputted by Viden in the 2015 Chevrolet Trax. Again, we can see that the transmitters of those messages did not output the desired levels, but outputted discernible levels. These results confirm that $\mathbb{V}1$–$\mathbb{V}2$ hold even in real vehicles, thus facilitating Viden's fingerprinting.

Transient changes. $\mathbb{V}3$ states that transient changes in CANH and CANL voltages are opposite in direction. Fig. 11 shows the 4 tracked percentiles, F_3–F_6, of message 0x1D0 in the 2013 Honda Accord. F_3–F_6 values are shown to temporarily deviate from and later return to their usual values. Since F_3 and F_5 are inverses of F_4 and F_6, respectively, vertically reversed shapes of the former resemble those of the latter. Thus, summing them suppressed their transient deviations when deriving the voltage profiles. Note, however, that since the tracked values in Viden depend on the time of sampling and its accuracy, the summation did not completely remove the deviations, but it sufficed for fingerprinting.

4.4 Against a Timing-Aware Adversary

We evaluated Viden's performance of attacker identification in the CAN bus prototype and in a real vehicle against a timing-aware adversary. We did not evaluate its performance against a naive adversary since the timing-aware adversary subsumes his capabilities.

CAN bus prototype. In the CAN bus prototype, we further programmed node $\mathbb{C}$ to be the timing-aware adversary who injects not only 0x15 but also attack messages with ID=0x01 at a random interval of 10–20ms; injecting messages aperiodically to perform arbitrary impersonation and thus evade timing-based fingerprinting devices. Note that 0x01 is also being sent from the legitimate node $\mathbb{A}$ at a random interval of 20–200ms. Fig. 12a shows the determined voltage profiles for all three messages during the mounted attack. Even though the voltage profile for 0x01 now reflects both the voltage outputs from $\mathbb{A}$ and $\mathbb{C}$, since the injection frequency from the attacker $\mathbb{C}$ was much higher, the voltage profile for 0x01 changed to a profile equivalent to the one shown in 0x15 (sent by $\mathbb{C}$). As a result, Viden determined that the transmitters of 0x01 and 0x15 are the same, thus identifying the source of the attack to be ECU $\mathbb{C}$. Note that even when the injection frequency is lower, the attacker ECU can be identified by observing the intrusion voltage profile.

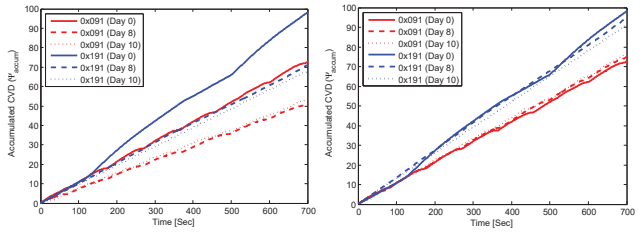
Real vehicle. We also evaluated Viden's performance against a timing-aware adversary in a real vehicle setting. We focus on the results obtained from the 2013 Honda Accord for the purpose of more in-depth discussion. We consider a scenario in which a timing-aware adversary controlling the Honda Accord ECU $\mathbb{D}$ attacks ECU $\mathbb{B}$ and also impersonates ECU $\mathbb{A}$, i.e., targeted impersonation. Thus, from the vehicle, Viden acquired voltage instances and profiles of the monitored messages: 0x091 sent from $\mathbb{A}$, 0x309 from $\mathbb{B}$, and {0x1EA, 0x1D0} from $\mathbb{D}$. To generate the scenario of $\mathbb{D}$ impersonating $\mathbb{A}$ (while attacking $\mathbb{B}$), $\mathbb{V}$ was further programmed to record only every 4-th message of 0x091 (sent by $\mathbb{A}$ every 15ms), and every 3rd message of 0x1D0 (sent by $\mathbb{D}$ every 20ms) as its ID to be 0x309. This was to emulate a scenario where the attacker $\mathbb{D}$ injects its attack messages with forged ID=0x309 at a *similar* frequency to $\mathbb{A}$, thus attempting to imitate its timing behavior for impersonation.

Fig. 12b plots the voltage profiles of {0x091, 0x1EA, 0x1D0} and the intrusion voltage profile of 0x309. Although the adversary attempted to impersonate ECU *A*, one can see that since Viden fingerprints the transmitter regardless of message timings, the intrusion voltage profile of 0x309 matched the profiles of {0x1EA, 0x1D0}. As a result, Viden concluded the attacker to be $\mathbb{D}$.

4.5 Against a Timing-Voltage-Aware Adversary

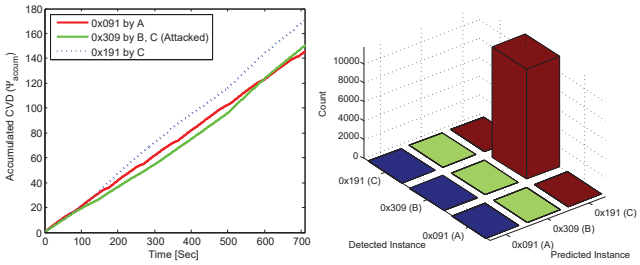
Based on his knowledge of voltage-based fingerprinting devices, a timing-voltage-aware adversary could attempt to evade Viden in two ways. First, the adversary might perform arbitrary impersonation by attacking the vehicle only when voltage-based fingerprints have not been updated for a long period of time, i.e., a high model-exam discrepancy. Next, the adversary might also perform targeted impersonation by changing its voltage output levels so as to imitate some specific ECUs' voltage output behavior.

4.5.1 Arbitrary impersonation. In most cases of a timing-voltage-aware adversary performing arbitrary impersonation, Viden accordingly/adaptively updates the voltage profiles and can thus correctly identify the attacker. One corner case, however, in detecting



(a) Before adjustment. (b) After adjustment.

Figure 13: Adjusting voltage profiles of {0x091, 0x191}.



(a) Targeted impersonation. (b) Predicting the attacker ECU.

Figure 14: Efficacy of Viden's Phase 4 execution.

the adversary would be when he performs arbitrary impersonation by attacking the vehicle only after a long idle period. To verify Viden's reaction to such an adversary, we evaluated the following scenario. We first obtained the voltage profiles of 0x091 and 0x191 from the 2013 Honda Accord while driving the vehicle for approximately 10 mins. After 8 and 10 days had elapsed, we again obtained their profiles; the average temperatures during the three days were 14.4℃, 7.7℃, and 12.2℃, respectively. In between the three update dates, the vehicle was driven 700 miles and 40 miles to generate (on purpose) the considered scenario where the voltage profiles might be outdated, thus becoming a chance for the timing-voltage-aware adversary to perform arbitrary impersonation.

Fig. 13a shows the acquired voltage profiles corresponding to messages 0x091 and 0x191 on the three different dates. The initial profiles obtained were found different from those obtained on the 8-th and 10-th elapsed days, whereas the latter two were equivalent. One interesting observation, however, was that the voltage profiles of both message IDs were decreased by the *same* amount. The changes we observed were due to a slight shift in all ECUs' V_{CC} — most probably due to the change in the battery state after the long 700 miles driving. In such a case, as we discussed in Section 3.8, Viden adjusts its voltage profiles. Once such an adjustment was made, we obtained the results shown in Fig. 13b, where all voltage profiles were properly aligned. This result shows that Viden is capable of handling cases where a timing-voltage-aware adversary performs arbitrary impersonation right after a vehicle's long idle period.

4.5.2 Targeted impersonation. Under scenarios where a timing-aware adversary performs arbitrary/targeted impersonation or where a timing-voltage-aware adversary performs an arbitrary impersonation, Viden can correctly identify him solely based on voltage profiles, i.e., within Phase 3. It could be much more challenging for Viden (requiring to run Phase 4) when a timing-voltage-aware adversary performs a *targeted* impersonation, i.e., trying to imitate some specific ECU's voltage outputs. Since the adversary creates a situation of at least two ECUs having similar voltage profiles (i.e., not unique), targeted impersonation would be more difficult for Viden to handle than arbitrary impersonation. We evaluated how Viden performs against such an adversary via (1) vehicle experiments and (2) simulations based on vehicle data.

Experiment-based evaluation. In the real vehicle setting, to generate a case which reflects a timing-voltage-aware adversary performing targeted impersonation, we considered the following scenario in the Honda Accord: adversary's ECU $\mathbb{C}$, which usually sends 0x191, injects attack messages with ID=0x309, thus attacking its original sender $\mathbb{B}$ and at the same time imitating $\mathbb{A}$'s voltage output behavior. In generating such a scenario, evaluation settings were similar to the previous ones, except that Viden recorded every $10n$-th message of 0x191 as its ID to be 0x309. This was to generate the voltage profile of 0x309 to be similar to $\mathbb{A}$'s as in Fig. 14a; $\mathbb{C}$ impersonates $\mathbb{A}$. To introduce ambiguity in the decision, we do not use the intrusion voltage profile in this evaluation. In such a case, if only voltage profiles are exploited, Viden might consider ECU $\mathbb{A}$ to be the attacker. However, Viden deals with this in Phase 4 by using voltage instances as machine classifier's input. In this evaluation, we used a 200-tree Random Forest classifier with 50% of the acquired data until detecting the intrusion as its training set.

Fig. 14b shows the number of *misclassified* voltage instances by the Random Forest classifier. It shows that Viden misclassified a large number of 0x309's voltage instances as those of 0x191. That is, even when the voltage profiles of 0x091 and 0x309 were similar, since Viden observed the measurements in a momentary manner and the adversary was incapable of precisely matching them, the attack source was correctly identified as $\mathbb{C}$, i.e., transmitter of 0x191, not $\mathbb{A}$. This validates that by using voltage instances as machine classifiers' inputs, Viden can prevent targeted impersonation by a timing-voltage-aware adversary.

By the Birthday paradox, at least two ECUs may *naturally* have similar voltage profiles. However, since Viden was feasible to distinguish them via machine classifiers, profile collision can be mitigated.

Simulation-based evaluation. In addition to the scenario shown in Fig. 14a, which we evaluated via real vehicle experiments, there could be different ways in which a timing-voltage-aware adversary might perform targeted impersonation. For example, the adversary might heat up or cool down his ECU to match some other ECUs' voltage profiles, even before he starts injecting attack messages. Thus, we conducted a more in-depth evaluation as follows. Based on the 35-min data of voltage instances output by the Honda Accord's 6 ECUs and those output by the Chevrolet Trax's 5 ECUs, two attack datasets were constructed to each contain 1000 different "targeted impersonation" attempts by a timing-voltage-aware adversary. We refer to Honda Accord's ECUs as $\mathbb{A}$–$\mathbb{F}$ and Chevrolet Trax's ECUs as $\mathbb{G}$–$\mathbb{K}$. The first dataset was based on only voltage instances of $\mathbb{A}$–$\mathbb{F}$ whereas the second was based on data from both vehicles, assuming that $\mathbb{A}$–$\mathbb{K}$ lie in the same vehicle. Such an assumption was made to evaluate how Viden performs when the number of ECUs increases. Each impersonation attempt was constructed by

	A	B	C	D	E	F
A	100	0	0	0	0	0
B	0	99.3	0	0	0.7	0
C	0	0	100	0	0	0
D	0	0	0	100	0	0
E	0	0	0	0	100	0
F	0	0	0	0	0	100

(a) "Honda Accord" attack dataset.

	A	B	C	D	E	F	G	H	I	J	K
A	100	0	0	0	0	0	0	0	0	0	0
B	1.5	98.5	0	0	0	0	0	0	0	0	0
C	0	0	100	0	0	0	0	0	0	0	0
D	0	0	0	100	0	0	0	0	0	0	0
E	0	0	0	0	100	0	0	0	0	0	0
F	0	0	0	0	0	100	0	0	0	0	0
G	0	0	0	0	0	0	100	0	0	0	0
H	0	0	0	0	0	0	0	100	0	0	0
I	0	0	0	0	0	0	0	0	100	0	0
J	0	0	0	0	0	0	0	0	0	100	0
K	0	0	0	0	0	0	0	0	3.2	0	96.8

(b) "Honda Accord + Chevrolet Trax" attack dataset.

Table 1: Confusion matrix of Viden [Unit: %].

(1) randomly choosing one ECU to be the adversary and another to be the victim, then (2) randomly choosing the times when the adversary starts to change his voltage outputs and (later) when to start attacking the victim, and finally (3) steadily shifting the adversary's voltage instance values (when it starts impersonation) so that his voltage profile matches the victim's, i.e., profile collision, before mounting an attack. Note, however, that such a shift does not make their instantaneous instances to be equivalent. As we discussed in Section 3.1, although an adversary may match the target's profile, it would be very difficult for him to precisely follow the target's instantaneous behaviors (e.g., transient changes due to temperature). This way, we were able to emulate a scenario where the adversary first imitates some specific ECU's voltage output behavior and then injects attack messages.

Table 1a shows the confusion matrix of Viden when identifying the attacker of the 1000 targeted impersonation attempts in the first attack dataset. For identification, Viden not only used voltage profiles but also a 200-tree Random Forest with voltage instances as its input. Again, half of the data until an attack was detected was used as the training set. Thanks to Viden's analysis of the adversary's impersonation attempts from two different viewpoints — ECU's usual voltage output behavior via voltage profiles and its momentary behavior via voltage instances — Viden was able to identify the attacker with only a 0.2% false identification rate. Even when Viden was evaluated based on our second attack dataset, which had 11 ECUs, Viden identified the attacker with a 0.3% false identification rate where the confusion matrix is shown in Table 1b. Albeit the increased number of ECUs, Viden's false identification rate increased only by 0.1%, thus corroborating its effectiveness. Note that such false rates reflect Viden's capability and robustness against the most skillful adversary who is aware of timing and voltage, i.e., the timing-voltage-aware adversary. Thus, Viden's false rate against *all* types of the considered adversaries — including the naive and timing-aware adversaries — would be much lower.

One can also interpret such good performance of Viden equivalent to its effectiveness in mitigating (naturally occurred) profile collision.

5 DISCUSSION

Number of ECUs on CAN. As of 2017, the average vehicle is reported to have approximately 25 ECUs, while luxury cars have approximately 50 [34], but *not all* of them on CAN; some are installed on LIN, MOST, etc. Moreover, to accommodate a large (increasing) number of ECUs on bandwidth-limited CAN, each vehicle is equipped with *multiple* CAN buses [15]. Accordingly, network architectures of various modern vehicles (Audi A8, Honda Accord, Jeep Cherokee, Infiniti Q50, etc.) are shown to have 3∼20 ECUs *per* CAN bus [24]; a similar figure to which we considered in our evaluations. Hence, if Viden was installed on each CAN bus in a vehicle, profile collision within that bus is much less likely to occur than the case when all ECUs are (considered to be) installed on one single CAN bus. Even in such a case with profile collisions, Viden can still handle it via the execution of its Phase 4.

Multiple ECUs per ID. Viden may underperform when multiple ECUs are assigned to send messages with the same ID, albeit unusual/rare. For example, although message ID=0x040 is scheduled to be sent, in turn, by ECUs $\mathbb{A}$–$\mathbb{D}$, Viden would construct only one voltage profile for 0x040. However, if such scheduling information is known in advance (e.g., every $4n$-th message of 0x040 is sent by $\mathbb{D}$), which is in fact defined by the car-makers, then Viden could construct voltage profiles accordingly, thus solving the problem.

Intrusion Detection. Timing-based IDSs exploit the *periodic* nature of CAN messages and thus suffice to detect attacks on periodic messages, but fail to detect attacks on *aperiodic* ones. Since Viden determines the transmitter ECU based on voltages, similarly to [14, 27], it can complement those IDSs in detecting intrusions. However, since most in-vehicle messages are periodic [12] and thus most intrusions are detectable, Viden's potential is maximized when it is used for attacker identification.

Attacker from Another In-vehicle Network. If the attack originates from a different in-vehicle network (e.g., FlexRay, MOST, LIN) inside the vehicle other than CAN, the corresponding gateway ECU will be the one that injects attack messages into CAN. Viden will, therefore, identify that gateway ECU as the attacker, since Viden is designed just for CAN. In such a case, the best both Viden and the gateway ECU can do is to look up the message routing table (describing which messages/signals to forward to/from), and identify the "compromised network". Handling such a scenario is important in integrating Viden in real vehicles.

Limitations. For Viden to identify the attacker ECU, it requires at least one voltage profile to use. For the example shown in Fig. 12b, Viden referred to the voltage profiles of {0x1EA, 0x1D0} to determine that the attacker ECU was $\mathbb{D}$. Since most ECUs are designated to transmit at least one message ID, one can identify the attacker ECU with Viden. However, if the compromised ECU does not send any messages, Viden's attacker identification can be inaccurate. In such a case, the best Viden can do would be obtaining the voltage profile of those ECUs during the manufacturing stage and updating them via voltage profile adjustments.

6 CONCLUSION

State-of-the-art vehicle security solutions lack a key feature of identifying the attacker ECU on the in-vehicle network, which is essential for efficient forensic, isolation, security patching, etc.

To meet this need, we have proposed Viden, which fingerprints ECUs based on voltage measurements. Via the ACK learning phase, Viden obtained correct measurements of voltages only from the message transmitters, and exploited them for constructing and updating correct voltage profiles/fingerprints. Using these profiles, we showed via evaluations on a CAN bus prototype and two real vehicles that Viden can identify the attacker ECU with a low false identification rate of 0.2%. Considering the fact that vehicles are safety-critical, Viden is an important first step toward securing the vehicles and protecting drivers and passengers.

ACKNOWLEDGMENTS

The work reported in this paper was supported in part by NSF under Grants CNS-1505785 and CNS-1646130. Assistance from Manoj Sastry and Zhao Li of Intel is also gratefully acknowledged.

REFERENCES

[1] 1991. CAN Specification Version 2.0. *Robert Bosch GmbH* (1991).
[2] 2002. Microchip AN228 - CAN Physical Layer Discussion [Online] Available: http://ww1.microchip.com/downloads/en/AppNotes/. (2002).
[3] 2003. Vector: Common High Speed Physical Layer Problems [Online] Available: http://vector.com. (2003).
[4] 2005. Optimizing MOSFET Characteristics by Adjusting Gate Drive Amplitude [Online] Available: http://www.ti.com/lit/an/slua341/slua341.pdf. (2005).
[5] 2010. CAN/CANbus and CAN Protocol Licensing [Online] Available: http://soc.microsemi.com/ipdocs/. (2010).
[6] 2016. SN65HVD1040-Q1 EMC-Optimized Can Transceiver Datasheet. [Online] Available: http://www.ti.com/lit/ds/symlink/sn65hvd1040-q1.pdf. (2016).
[7] 2016. TCAN1051 Fault Protected CAN Transceiver with CAN FD Datasheet. [Online] Available: http://www.ti.com/lit/ds/sllses8c/sllses8c.pdf. (2016).
[8] Dec. 2003. ISO 11898-2. Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit. *ISO Standard-11898, International Standards Organisation (ISO)* (Dec. 2003).
[9] Jul. 2015. Hackers Remotely Kill a Jeep on the Highway - With Me in It. [Online] http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. *WIRED* (Jul. 2015).
[10] Sep. 2016. Tesla Responds to Chinese Hack With a Major Security Upgrade. [Online] https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/. *WIRED* (Sep. 2016).
[11] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Aug. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. *Proceedings of USENIX Security* (Aug. 2011).
[12] Kyong-Tak Cho and Kang G. Shin. Aug. 2016. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. *Proc. of the 25th USENIX Security Symposium* (Aug. 2016).
[13] Kyong-Tak Cho and Kang G. Shin. Oct. 2016. Error Handling of In-vehicle Networks Makes Them Vulnerable. *Proc. of the 23rd ACM Conference on Computer and Communications Security (CCS)* (Oct. 2016).
[14] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. Jun. 2016. Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks. *arXiv preprint arXiv:1607.00497* (Jun. 2016).
[15] Ian Foster and Karl Koscher. 2015. Exploring controller area networks. In *USENIX ;Login: magazine.*
[16] I Foster, A Prudhomme, K Koscher, and S Savage. 2015. Fast and Vulnerable: A Story of Telematic Failures, In WOOT. *Proceedings of USENIX Security* (2015).
[17] Carlos Galup-Montoro and Marcio Cherem Schneider. 2007. MOFSET modeling for circuit analysis and design, In International series on advances in solid state electronics and technology. *Proceedings of the 5th Workshop on Embedded Systems Security* (2007).
[18] S. Haykin. 1991. Adaptive Filter Theory, In 2nd ed. Prentice-Hall. *Proceedings of the 5th Workshop on Embedded Systems Security* (1991).
[19] A. Herrewege, D. Singelee, and I. Verbauwhede. 2011. CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus, In ECRYPT Workshop on Lightweight Cryptography. *ECRYPT Workshop on Lightweight Cryptography* (2011).
[20] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Jan. 2011. Security threats to automotive CAN networks - Practical examples and selected short-term countermeasures, In Reliability Engineering and System Safety. *IEEE Symposium on Security and Privacy* (Jan. 2011).
[21] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. May 2010. Experimental security analysis of a modern automobile, In IEEE Symposium on Security and Privacy. *IEEE Symposium on Security and Privacy* (May 2010).
[22] J. Lepkowski and B. Wolfe. 2005. EMI/ESD Protection Solutions for the CAN Bus. *iCC* (2005).
[23] C. Miller and C. Valasek. 2013. Adventures in Automotive Networks and Control Units. *Defcon 21* (2013).
[24] C. Miller and C. Valasek. 2014. A Survey of Remote Automotive Attack Surfaces. *Black Hat USA* (2014).
[25] C. Miller and C. Valasek. 2015. Remote Exploitation of an Unaltered Passenger Vehicle. *Black Hat USA* (2015).
[26] Sparsh Mittal. May 2016. A Survey of Architectural Techniques for Managing Process Variation, In ACM Computing Surveys (CSUR) Journal, Volume 48, Issue 4. *IEEE Symposium on Security and Privacy* (May 2016).
[27] P. Murvay and B. Groza. Apr. 2014. Source identification using signal characteristics in controller area networks, In IEEE Signal Processing Letters, vol. 21, no. 4, pp. 395-399. *IEEE Symposium on Security and Privacy* (Apr. 2014).
[28] M. Muter and N. Asaj. 2011. Entropy-Based Anomaly Detection for In-Vehicle Networks. *IEEE Intelligent Vehicles Symposium* (2011).
[29] M. Muter, A. Groll, , and F. C. Freiling. 2010. A structured approach to anomaly detection for in-vehicle networks, In Information Assurance and Security (IAS), Sixth International Conference. *Proceedings of USENIX Security* (2010).
[30] Marco Di Natale, H. Zeng, P. Giusto, and A. Ghosal. 2012. Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice, In Springer Science & Business Media - Technology & Engineering. *Proceedings of the 5th Workshop on Embedded Systems Security* (2012).
[31] D. Nilsson, D. Larson, and E. Jonsson. 2008. Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes, In VTC-Fall. *Proceedings of USENIX Security* (2008).
[32] Andrea Palanca, Eric Evenchick, Federico Maggi, and Stefano Zanero. Sep. 2016. A Stealth, Selective, Link-layer Denial-of-Service Attack Against Automotive Networks. *Ph.D Thesis* (Sep. 2016).
[33] C. Szilagyi and P. Koopman. 2010. Low Cost Multicast Network Authentication for Embedded Control Systems, In Proceedings of the 5th Workshop on Embedded Systems Security. *Proceedings of the 5th Workshop on Embedded Systems Security* (2010).
[34] AMPG Body Electronics Systems Engineering Team. 2017. Future Advances in Body Electronics. *NXP White paper* (2017).

APPENDICES

A ACK threshold learning in a real vehicle

Viden first learns the thresholds which determine whether the measured voltages are from the ACK slot or not, before outputting voltage instances and profiles. This was achieved by determining most frequent and maximum/minimum sets, and exploiting the side lobe which only exists in the latter. In other words, the existence of such a side lobe (as shown in Fig. 7 in a CAN prototype), which represents the distribution of ACK voltages, is critical in learning the ACK threshold. Thus, to show that the proposed ACK learning is feasible even in real vehicles, through Viden, we obtained both most frequent and maximum/minimum sets for message ID=0x091, which was sent every 10ms by some ECU in the 2013 Honda Accord.

Fig. 15 (upper) shows the kernel density plots of the most frequent and maximum sets of the CANH dominant voltages while receiving ID=0x091, and Fig. 15 (lower) the kernel density of those obtained from the CANL line. One can see that as in the CAN prototype result (Fig. 7), side lobes exist in both the CANH and CANL lines. Thus, the proposed ACK learning mechanism in Viden derived the refined maximum and minimum sets, S'_{max} and S'_{min}, correctly and thus derived the ACK thresholds of message 0x091 to be 3.844V for CANH outputs and 1.114V for CANL outputs.

One interesting observation is how high and low CANH and CANL ACK voltage levels are. In our evaluation of Viden on the CAN prototype, since we had only 3 nodes acknowledging to the message, the median of the CANH ACK voltages was 3.514V as

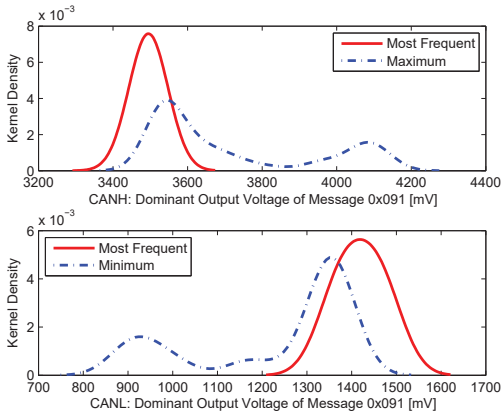


Figure 15: Deriving the ACK thresholds for message 0x091 in the 2013 Honda Accord.

shown in Fig. 7. On the other hand, in our experiment on the 2013 Honda Accord, the median of the CANH ACK voltage showed a high voltage level, 4.049V — much higher than the one obtained from the CAN prototype. For the CANL ACK voltage, the median was 0.953V. Such a result is due to the fact that there were much more ECUs (compared to 3 in the prototype) inside the vehicle which ACKed message 0x091.

B Voltage profiles while driving

We further validate that `Viden`'s derived voltage profiles do not depend on whether and how the vehicle is driven. We first obtained the voltage instances of 0x191 from the 2013 Honda Accord's CAN bus. At this time of measurement, the vehicle was stationary. Later on that day, instances of 0x191 was once again obtained, but this time while driving the vehicle for approximately 10 mins; the same data which we used in Section 4.5.1.

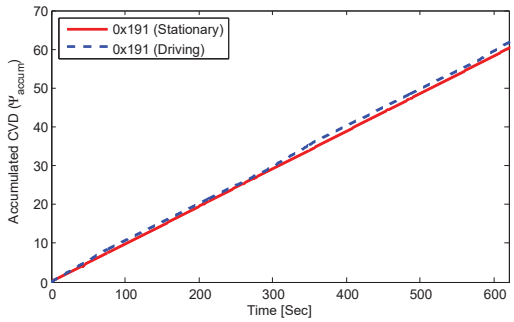


Figure 16: Voltage profiles of message 0x191 when the vehicle was stationary and driven.

Fig. 16 shows the voltage profiles of 0x191 obtained while the vehicle was stationary and driven. One can see that the two voltage profiles are equivalent, even though they were measured under a different condition. These are due to the fact that the voltage outputs are much more dependent on hardware components' characteristics than their momentary conditions. Moreover, transient deviations incurred from changes in momentary conditions would have been suppressed thanks to how `Viden` derives its voltage profiles; summing CVDs of CANH and CANL.