EdgeTDC: On the Security of Time Difference of Arrival Measurements in CAN Bus Systems

Marc Roeschlin
ETH Zurich
marc.roeschlin@inf.ethz.ch

Giovanni Camurati
ETH Zurich
giovanni.camurati@inf.ethz.ch

Pascal Brunner
ETH Zurich
pascal.brunner@inf.ethz.ch

Mridula Singh
CISPA Helmholtz Center for Information Security
singh@cispa.de

Srdjan Capkun
ETH Zurich
srdjan.capkun@inf.ethz.ch

***Abstract*—A Controller Area Network (CAN bus) is a message-based protocol for intra-vehicle communication designed mainly with robustness and safety in mind. In real-world deployments, CAN bus does not offer common security features such as message authentication. Due to the fact that automotive suppliers need to guarantee interoperability, most manufacturers rely on a decade-old standard (ISO 11898) and changing the format by introducing MACs is impractical. Research has therefore suggested to address this lack of authentication with CAN bus Intrusion Detection Systems (IDSs) that augment the bus with separate modules. IDSs attribute messages to the respective sender by measuring physical-layer features of the transmitted frame. Those features are based on timings, voltage levels, transients—and, as of recently, Time Difference of Arrival (TDoA) measurements. In this work, we show that TDoA-based approaches presented in prior art are vulnerable to novel spoofing and poisoning attacks. We describe how those proposals can be fixed and present our own method called `EdgeTDC`. Unlike existing methods, `EdgeTDC` does not rely on Analog-to-digital converters (ADCs) with high sampling rate and high dynamic range to capture the signals at sample level granularity. Our method uses time-to-digital converters (TDCs) to detect the edges and measure their timings. Despite being inexpensive to implement, TDCs offer low latency, high location precision and the ability to measure every single edge (rising and falling) in a frame. Measuring each edge makes analog sampling redundant and allows the calculation of statistics that can even detect tampering with parts of a message. Through extensive experimentation, we show that `EdgeTDC` can successfully thwart masquerading attacks in the CAN system of modern vehicles.**

I. Introduction

Over the last two decades, cars have become increasingly connected both in intra-vehicle and external networks. Information exchange between sensors through a reliable internal network is essential in enabling modern safety and advanced driver assistance features, such as automatic emergency braking. Most modern cars use a Controller Area Network (CAN bus) for intra-vehicle communication. As far as external networks are concerned, many modern vehicles offer Bluetooth (and WiFi) connectivity to entertainment systems. Due to this interconnectivity of vehicles, the attack surface to abuse potential vulnerabilities has grown likewise. Numerous works have shown how vulnerabilities in Bluetooth and other (wireless) technologies [2], [8], [17], [29], [30], [35], [36] can be used to compromise an Electronic Control Unit (ECU) in a vehicle to mount a subsequent attack over the CAN bus [6], [9], [25], [49], sometimes even taking remote control of an entire vehicle [30].

Aside from remote compromise, physical layer attackers are becoming an increasing concern when studying the security of intra-vehicle communication [33], [46]. Modern cars often feature attach points to the internal CAN bus, for example, the tow-hitch or OBD port. With physical access, the attacker does not need to leverage a compromised ECU and can even connect its own device(s) to the bus, enabling more sophisticated attack strategies.

Irrespective of the attack vector, the main issue with CAN bus is the lack of authentication that facilitates so-called *masquerade attacks*: After having compromised a low-criticality device or gained physical access to the bus, an attacker "impersonates" a safety-critical ECU, such as the engine control unit, by transmitting on the bus using the ECU's identifier(s). To prevent such attacks, the most natural choice would be the introduction of MACs or encryption [7], [21], [47]. However, due to the CAN bus' fixed standard lacking payload size, implementing meaningful authentication or encryption is not feasible for many vehicles. As a result, none of those proposals have found adoption in the (automotive) industry. In fact, chip manufacturers have announced "secure" CAN transceivers *without* cryptography[1], see, for example, [37].

As an alternative, Intrusion Detection Systems (IDSs) have been proposed. IDSs extract characteristics from the transmitted messages on the CAN bus [10], [11], [19], [43] and use these attributes to determine the CAN node which disseminated them. With an IDS in place, masquerade attacks could be detected since a CAN message emitted by the compromised node does not exhibit the physical characteristics of the claimed node. Although many IDSs have been proposed, it is questionable if those based on physical features [11], [12], [24], [34] can operate reliably [4], especially over an extended period of time without (automatic) re-calibration. Voltage-based features [11], [12], [16], [23], [24], [26] as well as clock-skew [10] are volatile under temperature changes, which require constant re-training of the classification models. As a result, IDSs are vulnerable to poisoning attacks [39] where an adversary manages to continuously spoof the characteristics of a target node using compromised CAN transceivers [4].

[1]At time of writing documentation for those chips is available only under NDA.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24271
www.ndss-symposium.org

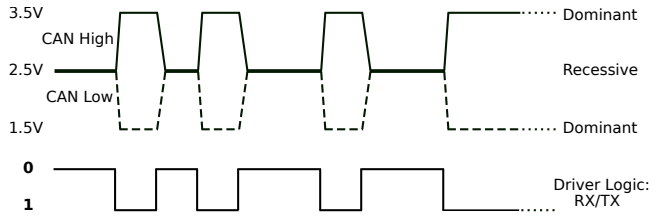


Fig. 1. To transmit a dominant bit 0, the transceiver drives CAN High to 3.5V and CAN Low to 1.5V. In idle state or when sending bit 1, both CAN low and CAN high are (passively) biased to 2.5V (according to ISO11898).

Recently, Time Difference of Arrival (TDoA) [5], [32], [33] was proposed as a robust feature for an IDS since, in theory, it solely depends on the location of the CAN transmitter and is resilient against the attacker's manipulation. We show that TDoA is vulnerable to spoofing and poisoning attacks if not carefully designed, mainly due to the fact that TDoA needs to be acquired over an entire message on falling and rising edges—single measurements are not sufficient. Additionally, if security of TDoA is analyzed under a stronger, physical layer attacker (as compared to the remote attacker), the security guarantees of TDoA do not necessarily persist. Our experimental set-up shows that `EdgeTDC` is secure unless the adversary has physical access and can interface with the bus at two or more locations.

II. Background

A. Controller Area Networks (CANs)

The Controller Area Network (CAN) is a multi-master serial bus protocol designed to be a robust vehicle bus, allowing the communication between various micro-controllers without needing a host computer/controller. CAN bus uses two dedicated wires for communication. The wires are called CAN High and CAN Low. The voltage differential of CAN High and CAN Low is changed to determine the logic state of the bus, see Figure 1. When the CAN bus is in idle mode or sending bit 1, both lines carry 2.5V. This state is weakly/passively driven by a resistor and is called **recessive**. When data bits 0 are being transmitted, the CAN high line goes to 3.5V, and the CAN low drops to 1.5V, thereby generating a 2V differential between the lines. This state is strongly/actively driven by a node connecting the lines to a voltage source and is called **dominant**. A rising edge at the logical level corresponds to a dominant-to-recessive edge, and a falling edge corresponds to a recessive-to-dominant edge at the electrical/physical level. As long as one node drives the CAN bus line to dominant (logic 0), the line is dominant.

Since communication relies on a voltage differential between the two bus lines, the bus is not sensitive to inductive spikes, electrical fields, or other noise. This makes CAN bus a reliable choice for networked communications on mobile equipment. Bit rates up to 5 Mbit/s are possible in short CAN networks ($\leq$ 40 m). Longer network distances reduce the available bit rate ($\leq$ 125 kbit/s at 500 m).

Figure 2 depicts the CAN bus standard message structure, it contains identifier-, control-, data-, checksum- and ack-bit(s). The message identifier (ID) describes the data content. CAN transceivers listen for activity on the bus and only transmit when the bus is idle. Due to the nodes on the CAN bus normally

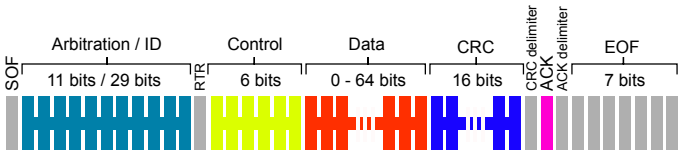


Fig. 2. Standard CAN frame structure.

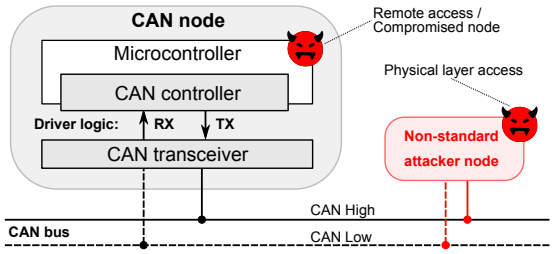


Fig. 3. A CAN node consists of a transceiver, a controller, and a micro-controller. The transceiver translates between logical and physical layer. The controller encapsulates the data link layer. It is part of the software stack of the micro-controller.

not being synchronized, there needs to be a mechanism to re-synchronize them to ensure consistent data transmission. The CAN protocol resolves this with an arbitration phase where only the transceiver sending a higher priority ID "wins" and the others back off.

Commercial CAN Transceivers. As depicted in Figure 3, a node on the CAN bus is a combination of three different types of components: transceiver, controller, and microcontroller. The CAN transceiver is responsible for the physical interaction. The interaction with the bus is achieved by pulling CAN Low and CAN High to the desired of the three voltage levels according to the CAN controller's clock-dependent input. Commonly used CAN transceivers operate at slew rates in the single digit range to tens of $\frac{V}{\mu s}$ [20], [44]. As a consequence of these physical limitations, commercial CAN transceivers have a physical upper bound at which they can operate. While the exact frequency depends on the model of the chip, operating frequency peaks at around 100 MHz.

B. Time Difference of Arrival (TDoA)

TDoA is the difference between the Time of Arrival (ToA) of a signal at multiple locations. Many positioning systems that are based on radio-frequency technology use TDoA measurements to calculate so-called pseudo-ranges that allow to perform multilateration and determine the exact location of a radio transmitter. Since the signals travel with a known velocity–often approximated with the speed of light–the true range or distance is directly calculated from the ToAs.

On a CAN bus, TDoA measurements can be used to determine the location of the transmitting node on the bus. For example, on a linear bus, the arrival time of the same message can be measured at two (or more) distinct locations, which reveals the exact position of the CAN node along the bus (see Figure 4). The exact time of arrival is captured by measuring the transition from a dominant to a recessive bit, or vice-versa.

	TCAN [5]	Biangulation [32]	TIDAL-CAN [33]	**EdgeTDC** (this work)
Sampling Frequency	NS	100MHz	$\geq$ 250MHz	220MHz
Granularity	Every recessive-to-dominant edge	Capture of entire frame	Capture of one edge per frame	Every falling and rising edge
Signal conversion	NS	Analog-to-digital (ADC)	Analog-to-digital (ADC)	Time-to-digital (TDC)
Detection method	Signal edge + echo edge	Edge inflection point	Thresholding	Propagation model for TDoA
Required hardware	Monitor (and repeater)	PCI-E Digitizer Card	USB Oscilloscope	FPGA
Bus structure	linear	linear	linear	linear
Precision	multiple of 15cm	$\leq$ 30cm	$\leq$ 40cm	$\leq$ 10cm
False reject rate	NS	NS	0.0	0.0 (after calibration)
False accept rate	NS	NS	0.0	0.0
Cost	NS	$>$ $3000	$1000	$185
Protected from remote attacker	✗	✗	✗	✓
Protected from physical layer attacker	✗	✗	✗	(✓) only if clock drift not present

NS = not specified, NA = not applicable

TABLE I. TDoA-BASED CAN BUS INTRUSION DETECTION SYSTEMS.

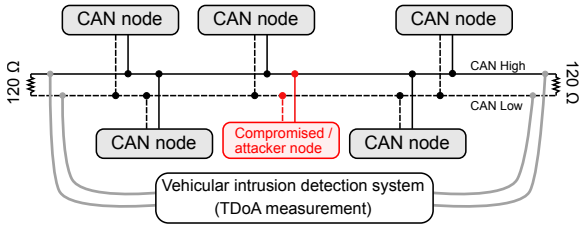


Fig. 4. Linear CAN bus with IDS attached to both ends of the bus.

TDoA-enabled IDSs are systems that run on top of the bus communication protocol [5], [32], [33] (see Table I). Such a system is fully transparent to the connected CAN devices. It constitutes an additional layer that should provide security guarantees. The CAN bus communication protocol does not have to be altered, and ECUs can receive and transmit normally. A TDoA system is its own entity in the form of a measurement unit with computational resources, such as a general-purpose CPU or MCU. The TDoA system is attached to the CAN bus through measurement "wires" that forward the CAN message to the measurement unit Figure 4. Its purpose is to determine the TDoA of every emitted message and verify the claimed sender (via the arbitration ID). Existing TDoA-enabled IDSs are capable of detecting sender location up to the precision of 30 cm. If the measured location and sender ID do not agree, the system triggers an alarm and possibly instructs the driver to stop operating the vehicle. Naturally, such a system needs to minimize false-positive events, i.e., when a false alarm is triggered. A false positive event is a severe intervention that can cause inconvenience and loss of time and money. Therefore, such a system has to effectively detect tampering with the CAN bus and identify compromised nodes while keeping the false reject rate at an absolute minimum.

III. THREAT MODEL AND ASSUMPTIONS

We consider an attacker model commonly found in literature on CAN bus security [4], [9], [10], [23], which we call **remote attacker**. This attacker gains control over one or more ECUs and/or any other CAN-bus connected devices/sensors in the vehicle, e.g., by exploiting a vulnerability in another communication technology, such as Bluetooth (see left side of Figure 3). Given this, we assume that such an attacker can eavesdrop on existing traffic and emit messages on the CAN bus, but is limited by the interface and the capabilities of the commercial CAN bus transceivers. We further assume that this adversary knows the CAN structure of the target vehicle [15], [28], [38]. This knowledge could be either obtained through public sources or by discovering the bus structure through one or more compromised CAN nodes.

In this work, we further consider a more powerful adversary that can attach their own proprietary devices to the CAN bus at different locations, such as the On-Board Diagnostics (OBD) port of the vehicle that exposes the CAN bus (see right side of Figure 3). We call this type of adversary **physical layer attacker**. Newer vehicles often provide CAN bus connectivity at multiple accessible locations, such at the tow hitch for trailers, and at the windshield for optional cameras (or LIDAR) for autopilot functionality [13]. A physical layer attacker therefore extends the attack surface from built-in devices and sensors to CAN bus attachment points. Direct access implies that the adversary can operate at the physical layer by measuring the voltages of the bus lines and injecting current.

Irrespective of the described capabilities (remote or physical layer access) and how many attacker nodes are present, we exclude attackers that change the physical layout of the CAN bus, e.g., by attaching or cutting wires. If a vehicle Intrusion Detection System (IDS) is installed, we assume that the attacker can not physically tamper with it or compromise it, i.e., the IDS firmware is not compromised. Furthermore, we do not consider DoS attacks, but focus on masquerade/impersonation attacks.

IV. ATTACKS AGAINST TDOA ON THE CAN BUS

We present novel attacks against TDoA measurements on the CAN bus. These attacks are relevant for IDS systems that use the TDoA of a node (measured at the extremities of the CAN bus) as a proof of its identity. For example, Biangulation [32] uses first edge of the start bit and TCAN and TIDAL-CAN [5], [33] use a recessive-to-dominant edge for TDoA estimation. These studies assume a logical layer attacker that cannot modify the TDoA of any node in the bus, as TDoA is supposedly an immutable physical-layer characteristic mainly determined by the node location. The possibility of colluding nodes is dismissed because synchronization is difficult and concurrent transmission alters the voltage profile [33]. However, recent work has shown that a logical-layer/remote attacker can easily synchronize nodes and bypass voltage fingerprints [4]. Prompted by this advance in attacker capabilities we conduct an in-depth study of the integrity of TDoA measurements on the

CAN bus, and we identify two novel spoofing and poisoning attacks.

A. Attack Vectors

To succeed at masquerading as a victim node, i.e., to emit a message using an ID allocated to a victim node without the IDS raising an alarm, the attacker can attempt to exploit two main attack vectors:

- **TDoA Spoofing**: The attacker directly alters the TDoA of the messages sent by its (compromised) node(s), so that the attacker's TDoA matches a desired value.
- **TDoA Poisoning**: The attacker alters the TDoA of messages sent by a victim node through interfering with (any of) the genuine messages on the bus. The encoded data is unchanged.

With spoofing, the adversary can launch a masquerading attack and also try to match the TDoA of the victim, forcing the IDS to resort to additional measures for detection. Poisoning can facilitate spoofing attacks by altering the TDoA of victim nodes while the IDS is (re)-calibrating.

As highlighted in [33], several physical layer factors (e.g., load) impact the propagation of CAN signals on the line. IDSs based on physical features generally require an initial calibration phase followed by continuous retraining to account for variations, e.g., due to temperature changes. Therefore, an attacker can influence the input for retraining the model, which ultimately affects classification. For example, if the model is poisoned to an extent where the attacker's own TDoA is accepted as the victim/targeted node, the attacker can successfully masquerade the victim node. We analyze how easily those attack vectors are accessible, considering the two attacker models described in Section III: a remote attacker, and a physical attacker. We first present the underlying concept of message overshadowing which is a building block for spoofing and poisoning and then we describe concrete attacks.

B. Message Overshadowing

A CAN frame that is transmitted by a single node on the bus exhibits a distinctive TDoA value that correlates with the position on the bus. The CAN bus can tolerate interference/cross-talk and environmental noise due to its balanced twisted pair design. Therefore, the TDoA value should be stable under normal operating conditions (we test the case of excessive noise experimentally). This implies that the adversary needs to be *active* on the bus in order to influence TDoA measurements. More specifically, the adversary can only alter the TDoA of any node (including any compromised nodes) if concurrent transmission can be enforced or specifically crafted interference is injected on the bus while the TDoA is acquired. We borrow the term *message overshadowing* for this concept. Overshadowing describes the action of transmitting on the bus while another message is underway with the goal of modifying the contents or any (physical) characteristic of the message, such as the the TDoA. By design, two or more CAN transceivers can transmit on the line simultaneously. If at least one of the transmitters sends a dominant symbol, the bus state is dominant. A standard CAN transceiver (and thus a remote attacker) can send a dominant symbol to override any recessive symbol, but not the other way around. The latter can be executed only by a physical layer attacker with a non-standard transceiver.

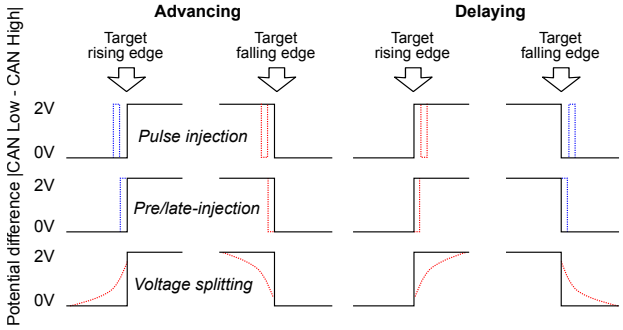


Fig. 5. When advancing, the edge(s) need to be anticipated. Overriding recessive state with dominant state (depicted in blue) can be executed with a standard CAN node. Any other attack, including voltage splitting requires a non-standard node (shown in red).

Injecting, Anticipating or Delaying Edges. Figure 5 shows different approaches to alter the arrival time of a symbol and subsequent edges using overshadowing. Both remote and physical layer attackers can:

- **Delay a dominant-to-recessive (0→1) edge** by transmitting a dominant bit longer than the currently transmitting node.
- **Anticipate a recessive-to-dominant (1→0) edge** by transmitting a dominant bit earlier than the currently transmitting node. The arrival time of the edge is advanced.
- **Inject a dominant (0) pulse** when another node is transmitting a recessive signal[2].

Only a physical layer attacker is able to:

- **Delay a recessive-to-dominant (1→0) edge** by overdriving the bus to recessive state while another node is still imposing a dominant state. Over-driving requires sinking/sourcing of current, leading to electrical states outside the specifications for standard CAN transceivers.
- **Anticipate a dominant-to-recessive (0→1) edge** by overdriving the bus to recessive state while the another node is still imposing dominant state.
- **Inject a recessive (1) pulse** by overdriving bus to recessive state while the another node is imposing dominant state.

C. TDoA Spoofing Attack

An adversary can use overshadowing to alter the TDoA of its own (compromised) nodes. To this end, the adversary needs to be in control of at least two nodes that transmit their messages at the same time. The degree of synchronization depends on the attacker. A physical layer attacker can run an arbitrary synchronization protocol to align the transmissions, whereas a remote attacker can use the CAN bus itself to synchronize two compromised nodes. It proceeds as follows: the first compromised node sends a sequence of immediately consecutive messages, to which the other nodes synchronize.

[2]Possible for a remote attacker since it requires setting a non-conforming configuration at the compromised note, e.g., incorrect baudrate.

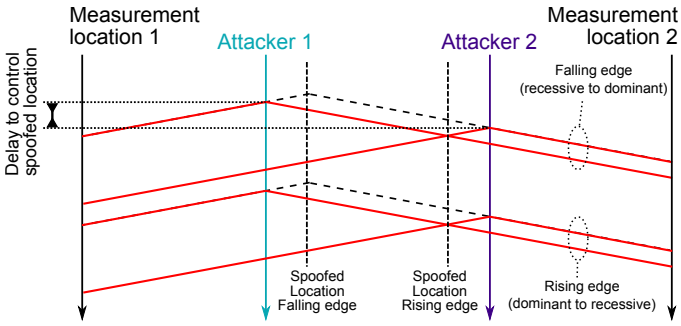


Fig. 6. Physical layer attacker with two synchronized nodes. The red lines depict an excerpt of the attacker transmission (2 edges). Both nodes transmit the same edges, but with a small delay in between. The delay is used to control the spoofed location(s). In theory, any location between the adversarial nodes can be realized.

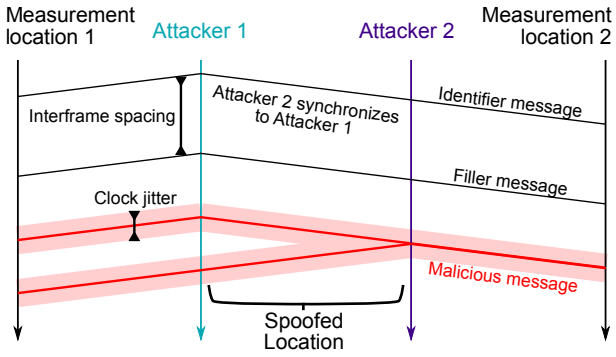


Fig. 7. Remote attacker synchronizes two compromised nodes for TDoA spoofing.

The first message contains an identifier, the second message is a filler message, and the third message is the actual (malicious) message which will exhibit an altered TDoA value. As soon as the second node receives the message with the identifier, it loads the same malicious message (the third message in the sequence of the first node) into the output buffer and tries to transmit it on the bus. However, the bus is currently occupied by the filler message. If the filler message has a higher priority ID than the third message, the node will back off and wait until the filler message has finished transmitting (see Figure 7). Immediately afterward, both compromised nodes will send a malicious message. Since both nodes transmit an identical message simultaneously, no bus error is caused. The attacker can choose arbitrary IDs for the malicious and filler messages. As long as the filler message has higher priority, the attacker can use bus arbitration to synchronize the emission of the malicious message. A similar method for synchronization is described in [4].

After synchronization, the attacker sends the same message from both nodes, but with a small delay. The delay can be used to control the spoofed location which can be realized at any position between the two adversarial nodes. If both nodes transmit at the same time the spoofed TDoA corresponds to the location exactly halfway. In Figure 6, we show an example where the first node (Attacker 1) transmits slightly earlier and therefore, the spoofed location is shifted closer to the Attacker 1 or 2, depending on edge polarity.

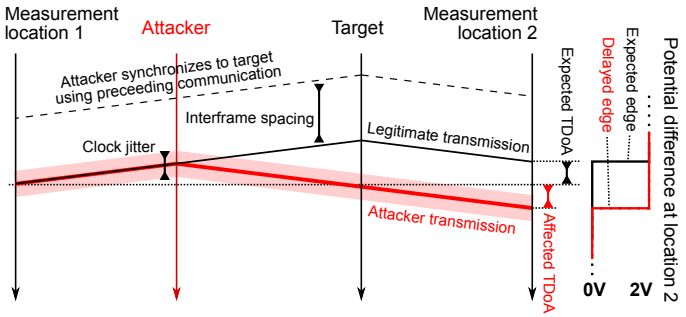


Fig. 8. Delay dominant-to-recessive edge. A compromised node introduces a falling edge on the line (i.e., rising edge in driver logic) which arrives later at location 2 than the legitimate edge. If TDoA is measured on this edge, the result reflects the location of the attacker.

Since the CAN bus acts like a wired OR on the physical layer, the polarity of the edge plays a role when spoofing TDoA. Dominant state overrides recessive state and thus a rising/dominant-to-recessive (0→1) edge does not have any effect if another node is still in dominant state. For recessive-to-dominant (1→0) edge, the transmission that arrives first is used for TDoA calculation. In the case of a dominant-to-recessive (0→1) edge, the transmission that arrives last is used in the calculation. As a result, there can be different spoofed locations for both edge polarities. If the attacker injects a (short) isolated pulse, a TDoA measurement corresponding to the location of the transmitting node is the outcome. If the pulse is not isolated, but falls next to an edge belonging to the legitimate transmission (see Figure 5), the result depends on how edges in quick succession are sampled, e.g., the IDS might only register the first or last edge. We perform various TDoA spoofing attacks and report attack success rates of our test set-up in Section VII.

D. TDoA Poisoning Attack

The main difference compared to spoofing is that poisoning attacks are possible only if the content of the victim's packet is predictable for the attacker. The attacker has to overshadow the victim packet without causing bit errors and without introducing additional edges. Unfortunately, CAN messages in modern cars are often entirely predictable, especially periodic safety-critical messages. Many packets contain static IDs, status bits and sensor readings that rarely change and result in nearly constant data bits [4]. Similar to spoofing, the poisoning attack requires synchronization. To transmit the attack signal in the correct time window, the adversary again needs to synchronize with the target node's transmission.

Analogous to the spoofing attack, the effective TDoA measured depends on the positions of the victim and attacker as well as the delay chosen by the attacker. If the attacker anticipates/delays edges of the victim's transmission, the effect on TDoA is determined by the amount of the shift and the polarity of the edge (falling, rising). An example for a poisoning attack based on delaying edges is shown in Figure 8, where an attacker introduces a delayed falling edge at measurement location 2, which is used in TDoA calculation. The introduced edge arrives at around the same time at measurement location 1 as the legitimate edge and thus only has a small effect on measurement location 1, subject to some slight variation due

to clock drift/jitter. The resulting TDoA value corresponds to a location between attacker and target node despite the message having been sent by the target. We confirm the effectiveness of poisoning in an ablation study in Section VII.

E. Clock Drift and Jitter

When an adversary overshadows another message, two transmissions collide on the bus. Since the transmitters have independently running oscillators, i.e., either faster or slower than an ideal oscillator with constant frequency, the edges contained in the combined message experience a clock drift, even if the contents of both messages match exactly and transmission started at the exact same time. Clock drift introduces a small misalignment of the bit timings that increases over time and shifts the (combined) edges apart. Moreover, the oscillators are subject to clock jitter that can further increase the delay between the edges. This phenomenon modifies the spoofed location represented by the overshadowed message (see Figure 6). For a physical layer attacker, drift and jitter are controllable to an extent, i.e., the attacker could use expensive clocks. A remote attacker, however, has to rely on the internal oscillators of the compromised CAN nodes which inevitably skew the edges of an overshadowed message. Clock drift and jitter constitute the root cause for an increase in intra-packet variance of TDoA readings and thus form the basis of the detection mechanism in EdgeTDC. At the same time, higher variance entails that acquiring just one TDoA value per packet (e.g., through measuring a single predefined edge) is not sufficient. In Section VII, we show that even a remote attacker can alter the TDoA of a node using our novel spoofing/poisoning techniques when intra-packet variance is not computed.

V. EdgeTDC

EdgeTDC is a TDoA-based IDS that can defend against any remote adversary, even if it is capable of poisoning or spoofing TDoA. It can efficiently measure the TDoA of each edge using a time-to-digital (TDC) converter on the digital output of two transceivers at the ends of the bus. Measuring each edge allows calculation of statistics that can detect tampering with (parts of) a CAN message despite not acquiring analog signal samples.

A. System Design

EdgeTDC needs—as any other IDS—an additional device that augments the CAN bus. It uses the concept of TDoA as described in Section II-B. The measurements are acquired at two points (ideally at both ends) of a linear CAN bus, see Figure 4. If the ends of the bus do not happen to lie close to each other, the measurement points can be connected to the IDS device using additional cables. In Figure 9, we show an overview of the IDS device which implements the concept of EdgeTDC. There are six main components: (1) the electrical translation from CAN to digital, (2) sampling and decoding of incoming CAN packets, (3) control logic that acquires the ID and arms the TDoA measurement block before the incoming edges, (4) the Time-to-Digital Converter (TDC) that measures the TDoA, (5) a memory buffer and a UART peripheral to transmit the results to the host, and (6) software that verifies the authenticity of the CAN packets by comparing the expected

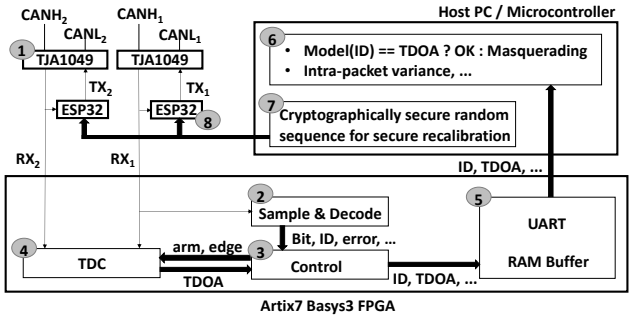


Fig. 9. Overview of EdgeTDC. The TDoA of each edge is measured while decoding incoming packets (1)–(5). TDoA consistency with the ID is then verified to detect masquerading attacks (6). To calibrate securely, EdgeTDC transmits a sequence of packets whose content is created by a cryptographically secure pseudo-random generator (7)–(8).

TDoA for the claimed ID and the real measured TDoA. In order to calibrate the model for the expected TDoA, EdgeTDC transmits a series of packets from a known location using the same transceivers as for reception (1) and two additional controllers (8). The content of these packets is generated by a cryptographically secure pseudo-random generator (7). Since the content is unpredictable to an attacker, the TDoA of the calibration packets cannot be poisoned or spoofed. We implement blocks (2) to (5) on an inexpensive ($150) Basys3 development board by Digilent, which is equipped with a Xilinx Artix7 FPGA running at 100 MHz base frequency. Component (1) can use any CAN transceiver with 3.3 V output (e.g., NXP TJA1049). Similarly, component (8) can use any micro-controller with a CAN controller peripheral (e.g., ESP32-based Atom CAN). Verification and calibration software (6)-(7) can run on any micro-controller or host computer (e.g., Raspberry Pi). If the two ends of the CAN bus are close they can be attached directly to EdgeTDC. Otherwise, EdgeTDC could be connected to the far side using an active CAN relay made of a long cable and two transceivers (see Figure 23 in Appendix).

B. Implementation

Electrical translation. EdgeTDC uses the NXP TJA1049 high-speed CAN transceiver (available on the Arduino MKR CAN Shield [40] for <$30), which turns the incoming CAN signal into a 3.3 V digital line for the FPGA. Two NXP TJA1049 are used, one for each end of the CAN bus—so that EdgeTDC can take TDoA measurements. These transceivers, driven by two ESP32-based controllers (M5Stack ATOM CAN [27]), are also used to transmit calibration packets on the CAN bus.

Sampling, decoding, control. The sample logic continuously detects and synchronises with each edge on the incoming RX signal, in order to schedule the best moment to sample the value of each bit. The decoding logic parses each bit, and then identifies each field of the frame. We borrow these components from an open-source Verilog implementation of a CAN controller available on OpenCores [31], with some modifications. Our custom control logic collects the ID of each CAN packet and triggers the measurement of the TDoA of its

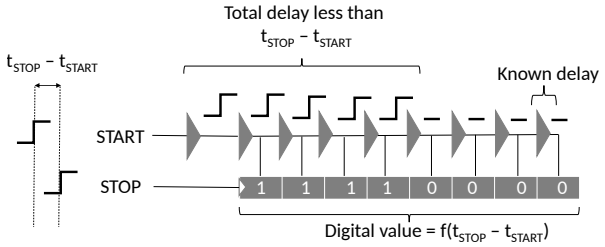


Fig. 10. Working principle of a TDC based on a delay line. Each flip-flop is fed with an increasingly delayed version of the START signal. When STOP arrives at the clock of the flip-flops, those flip-flops whose input delay is smaller than the TDoA between START and STOP will sample a '1', whereas the following ones with sample a '0'.

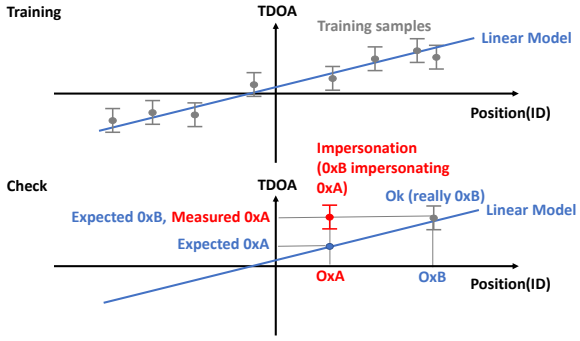


Fig. 11. Working principle of the linear model. EdgeTDC compares the expected TDoA with the measured one for each ID. The ID with the smallest error is the most likely. If it matches with the ID decoded in the message, then the message is authentic, otherwise it has been masqueraded by another node.

edges. The sampling and decoding logic informs the control logic about the expected polarity and time of the edges, which is used to arm the TDoA block.

TDoA Measurement. To acquire high-resolution TDoA measurements, EdgeTDC uses a TDC circuit on the FPGA. A TDC is a partially asynchronous digital circuit that converts the TDoA between a START and STOP signal into a digital number using a delay line, illustrated in Figure 10. The STOP signal samples several delayed copies of the START signal in a flip-flop chain. The number of flip-flops at '1' is then proportional to the delay between START and STOP. The resolution of the measurement is given by the resolution of the elements in the delay line, which can be controlled accurately by using digital logic as a delay element. The total delay of the line is chosen to match the duration of a clock cycle (220 MHz). This method allows measuring TDoA with sub-clock accuracy, without requiring to increase the clock frequency. If the TDoA value exceeds the duration of the delay line, then an additional standard digital counter takes care of measuring the extra delay. We borrow the implementation of the TDC from an open-source project [3], modifying the code in order to measure both positive and negative values of the TDoA, to introduce a signal that arms the circuit before each measurement, and to acquire both rising and falling edges. Following the recommendations in [3], we optimize the physical placement of EdgeTDC components on the FPGA fabric (see Figure 22 in Appendix). The entire design is placed in the same clock region. The elements of the delay lines are placed close together and form a column that shares the same clock lines. Finally, we use dedicated blocks designed for clock distribution to route START and STOP signals in our design. The TDoA between START and STOP is measured at the same location within the same clock region. We interface to the EdgeTDC peripheral via a 2 MHz serial link that can be connected to any micro-controller or host computer.

C. Model, Secure (Re-)Calibration, and Detection

Problem statement. Given a CAN message with a certain TDoA ($tdoa_{measured}$) that claims to originate from a certain ID ($id_{claimed}$), we want to find the ID (id_{best}) which is most likely attributed to the measured TDoA. If the ID claimed by the packet and the ID we expect from the TDoA are the same, the message is authentic, otherwise, a masquerade attack is detected. The overall $toda_{measured}$ of a message is the average over all the TDoAs of each edge of the message.

Topology. While knowing the topology of the CAN bus (i.e., the nominal position of all the node that transmit using a certain ID) is not strictly necessary for classification, it is very useful to constrain the possible positions of the nodes, making the system more robust to poisoning attacks. The topology of the bus is known by design or it can be measured directly. If this is not known by design, we can utilize EdgeTDC to collect messages and estimate the position for each ID from the average TDoA for a given ID and the propagation speed:

$$position(id) = speed \cdot tdoa_{id} \quad \forall id \in IDs \tag{1}$$

This step must be performed once during manufacturing/maintenance and in absence of any attack.

Initial calibration. We need a model that, given an ID, can predict the expected TDoA. We know that the TDoA is linear with the position of the node (travel time is proportional to distance divided by speed). Therefore, we use a linear model as follows:

$$\widehat{model}(id) = \alpha + \beta \cdot position(id) \quad \forall id \in IDs \tag{2}$$

To estimate α and β, we use linear regression over a set of calibration measurements collected in absence of any attack. This has to be done once, for example during manufacturing.

Secure re-calibration. A re-calibration is necessary every time the CAN bus is turned on again after a period of inactivity, as changes in the environment might have slightly altered the model. Once the CAN bus is deployed, we cannot rerun the initial calibration procedure (an attacker might have compromised some of the nodes). To solve this, we leverage the two trusted transceivers ①/⑧ part of EdgeTDC. Every time the bus is turned on, EdgeTDC emits a sequence of calibration packets from its transceivers at the extremities of the bus. EdgeTDC then uses linear regression to estimate $\hat{\alpha}$ and $\hat{\beta}$, similar to the initial estimation of α and β, but without relying on other nodes. By re-calibrating the model each time the bus is turned on, EdgeTDC compensates for possible variations that occurred during the off time. EdgeTDC can also compensate for a systematic deviation between initial and secure re-calibration.[3]

[3] EdgeTDC runs the secure re-calibration for the first time right after initial calibration and computes the correction terms $\Delta = \alpha - \hat{\alpha}$ and $\Omega = \beta/\hat{\beta}$. In all subsequent re-calibrations the systematic deviations are re-applied to the estimates of $\hat{\alpha}$ and $\hat{\beta}$ by computing $\hat{\alpha}' = \hat{\alpha} + \Delta$ and $\hat{\beta}' = \Omega\hat{\beta}$.

To prevent poisoning/spoofing attacks, EdgeTDC must generate the calibration sequence using a cryptographically secure pseudo-random generator. We propose the same generator used by the IEEE804.15.4z standard [1] to protect Ultra Wide Band (UWB) distance measurements from manipulation and overshadowing attacks. It is based on AES-128 in counter mode. The length of the sequence can be chosen to provide both, a good calibration estimate and a high security-level.

Online update/detection. During normal operation of the CAN bus, whenever the IDS receives a message with $ID = id_{claimed}$ and $TDOA = tdoa_{measured}$, it has to determine if the ID is genuine or not. We compute the following discriminant for each possible ID of the CAN bus:

$$\widehat{error}(id) = \left| \widehat{model}(id) - tdoa_{measured} \right| \quad \forall id \in IDs \quad (3)$$

We can find the id that has most likely generated $tdoa_{measured}$ as the one with minimum error $\widehat{error}$:

$$id_{best} = \underset{id}{\operatorname{argmin}}(\widehat{error}(id)) \quad \forall id \in IDs \quad (4)$$

We check if the message is authentic by comparing $id_{claimed}$ decoded from the message and id_{best} predicted by the model:

$$authentic = (\ node(id_{claimed}) == node(id_{best})\ ) \quad (5)$$

To account for nodes that transmit more than one ID, we check that $id_{claimed}$ and id_{best} belong to the same node, i.e., EdgeTDC uses a predefined mapping. Finally, if the message is authentic, we can use the $tdoa_{measured}$ to update the $\widehat{model}$ online (in order to deal with small legitimate fluctuations of the model due to temperature, for example):

$$\widehat{model} = \text{update}\left(\widehat{model}, tdoa_{measured}, id_{claimed}\right) \text{ if } authentic \quad (6)$$

We extend linear regression to perform an efficient online update.[4] The reactivity of the model update is controlled by the forgetting factor $\lambda \in [0, 1]$.

Rising and falling edges. EdgeTDC knows whether to expect a rising or falling edge, based on the current value of the bit. For falling edges, EdgeTDC simply inverts the start and stop lines. After having decoded the next bit, EdgeTDC detects if the (rising or falling) edge really occurred, or if there was no transition. Two separate linear models are kept for rising and falling edges, to account for possible differences on the intercept due to the inversion and possible differences on the slope due to attacks that target only one of the edge polarities. EdgeTDC can detect inconsistencies between the number of TDoA measurements and the number of bit transitions.

Edge selection. EdgeTDC needs to carefully select the bits it considers for TDoA measurement: it needs to exclude the arbitration part and the ACK bit which could have been sent by a different node. Those bits distort the measurement otherwise.

Intra-message variance check. EdgeTDC measures the TDoA of one message as the average of the TDoAs of all edges (selected as explained above). During calibration EdgeTDC also estimates the (pooled) variance of TDoA within a message. During normal operation, EdgeTDC checks that the variance of each received message does not exceed the expected values by orders of magnitude, which occurs due to clock drift when two attacker nodes transmit at the same time as explained in Section VI.

Physical interpretation. The linear model has a physical interpretation. It is an estimate of the relation between time and position. We expect α to be close to zero and β to be approximately the inverse of the propagation speed of CAN messages over the bus, i.e., $\frac{1}{\beta} \approx c \cdot vf$, where c is the speed of light and vf is the velocity factor of the cable, often around 0.7 for automotive standards J2284 and J1939. Despite additional variables such as delays in receiver circuitry and driver, we can use this interpretation to check if the parameter we estimate falls in the expected range for this physical quantity, and rule out values that are physically impossible and might be caused by an attack. Figure 11 depicts the principle behind the previous equations in an intuitive way.

VI. Security Analysis of EdgeTDC

Message integrity and overshadowing detection. EdgeTDC measures the time of arrival of all rising and falling edges excluding ID and ACK (in contrast to previous work that measures only one edge). Launching a single attack by delaying or anticipating an edge is not sufficient and is detected. EdgeTDC measures intra-message average and variance of the TDoA values occurring in each message. Affecting only a small subset of the edges in a frame does not modify the average TDoA significantly.

Intra-message variance and clock drift. If an adversary interferes with several edges, the intra-message variance increases and the variance check described in Section V triggers. This is due to the fact that, during overshadowing, two nodes A and B transmit at the same time and the combined message is subject to two (different) clocks with drift and imprecise synchronization, leading to an unavoidable increase in variance. This is especially true for remote attackers—a physical layer attacker with two colluding nodes and perfect clock stability, on the other hand, might pass the variance check if the delay for every single edge in a frame is modified successfully.

We model the variance based on the inaccuracies of the two clocks of nodes A and B and determine the required clock stability to pass the test. We use the following clock model and define the offset between A and B at time t as

$$\delta(t) \ = \ \delta_0 + f_{\text{offs}} \cdot t \quad (7)$$

where δ_0 is the inital offset and f_{offs} is the worst-case frequency offset between the nodes (we do not model frequency drift as the transmissions are on the order of a few hundred nano-seconds). We then define the TDoA (for rising and falling edges) over an overshadowed packet as follows, assuming $d = \text{dist}(A, B)$. Additionally, $\delta(t) < 0$ if A sends before B.

$$\text{TDoA}_R(t) = \begin{cases} \text{TDoA}_B & \delta(t) \leq -d/(c \cdot vf) \\ \text{TDoA}_A & \delta(t) \geq d/(c \cdot vf) \\ \text{TDoA}_A + \delta(t) + \frac{d}{c \cdot vf} & \text{else} \end{cases} \quad (8)$$

and

$$\text{TDoA}_F(t) = \begin{cases} \text{TDoA}_A & \delta(t) \leq -d/(c \cdot vf) \\ \text{TDoA}_B & \delta(t) \geq d/(c \cdot vf) \\ \text{TDoA}_B - \delta(t) + \frac{d}{c \cdot vf} & \text{else} \end{cases} \quad (9)$$

[4] https://scaron.info/blog/simple-linear-regression-with-online-updates.html

In order to create an outcome different from the existing TDoA values of A and B, i.e., TDoA_A and TDoA_B, the adversary has to make sure that

$$-\frac{d}{c \cdot vf} < \delta(t) < \frac{d}{c \cdot vf} \tag{10}$$

In addition, edge consistency requires $\text{TDoA}_F(t) = \text{TDoA}_R(t)$ which is only the case if $\delta(t) = \frac{1}{2}(\text{TDoA}_B - \text{TDoA}_A) - \frac{d}{c \cdot vf}$. However, this difficult to achieve since the delay needs to be changed throughout the packet and clock drift needs to be compensated which is only possible for a physical layer attacker, but requires tight synchronization on the order of nano-seconds. A remote attacker does certainly not have the ability to selectively delay edges between compromised nodes with fine-grained precision because a remote attacker can not modify or tune the oscillators.

Since explicit delay modifications are infeasible, the adversary can hope that, due to the variability in $\delta(t)$, the average TDoA of the packet falls into the interval in Equation 10. If we assume a normally distributed error term $\mathcal{N}(0, \sigma_\epsilon^2)$ for the (remaining) time/phase jitter and the measurement uncertainty that are not captured in the clock model, we can write $\widehat{\text{TDoA}}(t) = \text{TDoA}(t) + \mathcal{N}(0,\ \sigma_\epsilon^2)$ and estimate $\text{Mean}(\widehat{\text{TDoA}})$ and $\text{Var}(\widehat{\text{TDoA}})$. If the TDoA values are sampled at every edge in the packet, we can compute sample mean and variance over a packet. Assuming equally spaced bits and excluding the edges that do not lie in the window in Equation 10, sample mean and variance are[5]:

$$\text{Var}(\widehat{\text{TDoA}}) \approx \text{Var}(\ \{\delta(i \cdot r)\}_{i=0}^{N-1}\ ) \tag{11}$$

$$\text{Mean}(\widehat{\text{TDoA}}) \approx \text{TDoA}_F + \frac{d}{c \cdot vf} + \text{Mean}(\ \{\delta(i \cdot r)\}_{i=0}^{N-1}\ ) \tag{12}$$

with N being the number of edges in the window and r the bit time, e.g, 8 µs for a bus speed of 125 kbit/s. Plugging the definition of $\delta(t)$ into Equation 11 and Equation 12, respectively, we arrive at

$$\text{Var}(\widehat{\text{TDoA}}) \approx f_{\text{offs}}^2\ r^2 \frac{N^2 - 1}{12} + N^2 \sigma_\epsilon^2 \tag{13}$$

$$\text{Mean}(\widehat{\text{TDoA}}) \approx \text{TDoA}_F + \frac{d}{c \cdot vf} + \delta_0 + f_{\text{offs}}\ r \frac{(N-1)N}{2} \tag{14}$$

The N edges within the window form a linear sequence and thus the larger N, the higher the advantage for the attacker to steer the mean towards a TDoA/position that is consistent for both edges. However, the larger N, the higher the variance. Moreover, the larger the window and distance $d = \text{dist}(A, B)$ between the nodes, the larger N and thus higher variance in the measured TDoA values. We show in Section VII typical variance figures for both benign and overshadowed messages.

Edge count. Furthermore, `EdgeTDC` counts the number of edges and compares the result with the expected number of edges by using bit-level information. Attacker can not inject more edges than there are supposed to be in the received frame, let alone edges with the wrong polarity. Therefore, any attack that injects edges/pulses is detected, independent of the polarity. In `EdgeTDC`, the delay lines have to match a clock cycle, i.e., $4.55\text{ns} = \frac{1}{220\text{MHz}}$. Therefore, only one edge can be captured during that period and adversarial pulses shorter than 4.55ns can not be detected. However, since commercial CAN transceivers (such as TJA1049) have bus-to-logic delay times on the order of $30 - 70\text{ns}$, short pulses are filtered by the transceiver chip and neither reach the FPGA, nor affect the received data.

Propagation Channel Integrity. `EdgeTDC` leverages the topology of the bus. The variance of the measurements and position of nodes are used to compute confidence that TDoA of a message indeed matches the claimed ID. `EdgeTDC` models the system with specific propagation speed and a static offset between both extremities of the bus (parameters α and β). Since those parameters are continuously verified, a state where α or β are shifted outside their physical bounds or the propagation speed is not constant across the entire bus length is immediately detected. The propagation speed does require a certain margin as the cables and transceivers are subject to environmental changes in a vehicle. Section VII shows how propagation speed changes when the measurement device (FPGA) is heated up and how continuous training adjusts the model. Despite those environmental changes, the model of `EdgeTDC` can detect (selective) attacker-induced changes in propagation speed since the physical range for the propagation speed can be determined through measurements prior to system deployment.

Excluding perturbations. In addition to forcing the TDoA measurements into a linear model, `EdgeTDC` does not update the model in case a TDoA measurement falls outside the expected distribution of a node. This provides further resilience against (local) perturbations introduced by an attacker.

Re-calibration. An attacker can attempt to interfere with the re-calibration phase using compromised bus nodes by transmitting fake calibration packets or poisoning legitimate calibration packets. For a remote attacker, this is not possible if the packets contain unpredictable bits since, for successful poisoning, the attacker needs to know the packet's content. Anticipating and guessing the bits without knowledge of the key and the IV (which should be embedded securely in `EdgeTDC`) results in detectable bit errors. Injecting pulses (dominant or recessive) can be detected by counting the edges. In theory, a physical layer attacker can delay all the edges part of the calibration packets, but this requires over-driving the bus to a recessive state whenever a recessive-to-dominant edge is encountered and leads to states outside the CAN specifications (see Section IV).

Compared to regular `EdgeTDC` operation, re-calibration does require the CRC part to be removed from the TDoA calculation (in addition to the ID and ACK). The CRC can be computed based on the preceding bits, invalidating the assumption that the edges are unpredictable. In Section VII, we show the robustness of secure re-calibration under different temperature conditions.

VII. Experimental Evaluation

A. Automated Experimental Setup

Figure 12 shows an overview of our experimental setup. We connect `EdgeTDC` to a CAN bus with 10 nodes using MKR CAN Shields [40] based on the NXP TJA1049 CAN transceiver. Nodes D, E, H, I, L control the shield using an Arduino MKR, while nodes A, B, C, F, G use an ESP32-based controller from the M5Stack Atom [27]. Each node runs an

[5]Analysis done for falling edges. Rising edges are analoguous.

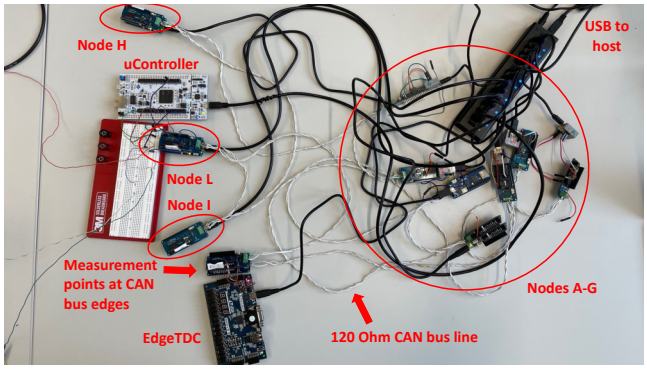


Fig. 12. Overview of the experimental setup.

Node	dist(Node, Prev.)	position(Node)	TDoA(Node)
RX1	0 cm	454 cm	-
A	50 cm	354 cm	(19 896 ± 78) ps
B	14 cm	326 cm	(18 374 ± 63) ps
C	30 cm	266 cm	(15 207 ± 81) ps
D	100 cm	66 cm	(4295 ± 63) ps
E	20 cm	26 cm	(1993 ± 58) ps
F	50 cm	-74 cm	-(2917 ± 76) ps
G	20 cm	-114 cm	-(5023 ± 88) ps
H	50 cm	-214 cm	-(10 743 ± 66) ps
I	50 cm	-314 cm	-(16 114 ± 63) ps
L	20 cm	-354 cm	-(18 428 ± 57) ps
RX2	50 cm	-454 cm	-

position(Node) = dist(Node,RX2) - dist(Node,RX1)

TABLE II. POSITION AND TDOA OF THE NODES.

implementation of *slcan* (i.e., CAN over serial) and is seen as a network interface from our Linux host, making systematic control of transmission and reception from Python possible. We have extended the basic *slcan* interface to also include control for our attacks. We can then transmit (normal or extended) frames (ID and some data) from each node, while `EdgeTDC` analyzes the messages. The bus lines are made of CAN cables taken from a real bus salvaged from a vehicle. The two ends of the bus have the nominal 120 Ω terminations. We performed transmission line engineering and verified that our bus line is indeed balanced. We selected a bus speed of 125 kHz.

Table II reports the positions of each node. RX1 and RX2 are the two ends of the bus. The first column lists the distance of each node from the previous node. The second column shows the difference of distance that each node has from RX2 and RX1. This is the *position* that we use in the linear model described in Section V, and it exhibits a linear dependence with the TDoA.

B. `EdgeTDC` *In Absence Of Attacks*

Initial Calibration and Measurement Accuracy: We collect the TDoA from a calibration set of 10000 messages from all nodes, to build an initial model of the relationship between position and TDoA. This initial calibration is performed once in a secure setting, with knowledge of the bus topology. Figure 13 shows the corresponding linear model, which is clearly a good fit of the TDoA measurements. The last column of Table II reports the values of the TDoA and their uncertainty. Under the reasonable assumption of Gaussian measurement noise, the uncertainty is computed as plus/minus three times the standard deviation (99.8 % confidence interval). In a practical

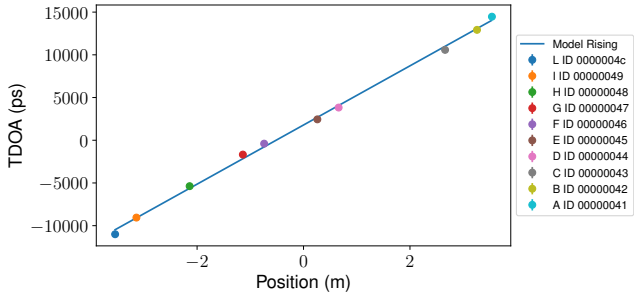


Fig. 13. Linear model between position and TDoA.

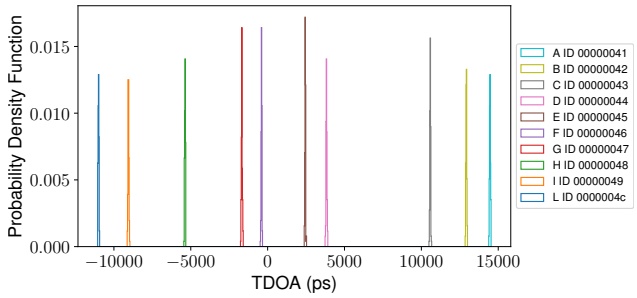


Fig. 14. Probability density functions for the TDoA of each ID. Every node can be clearly distinguished from the others.

deployment, it is important to check that the nodes intervals do not overlap given the bus topology and external conditions. For example, in our case the confidence interval is always below 90 ps, which corresponds to a distance of less than 2.7 cm (assuming that signals propagate at the speed-of-light or less). Figure 14 shows the probability density functions of the TDoA for each ID, measured over 10000 messages. The distributions are clearly separate, and each node can be distinguished from the others.

Secure Re-calibration and Online Update: We use 200 64-bit CAN messages for secure re-calibration, for a total of 12′800 bits generated from AES-128 in counter mode. Transmission at a 125 kHz bus speed takes around 0.3 s.

We run `EdgeTDC` at three different times under three different temperature conditions and show the results in Figure 15. We first run `EdgeTDC` at a constant low temperature (Step 1). Then we turn it off, heat it to a higher temperature, turn it on again and keep increasing the temperature (Step 2). Finally, we turn it off, let it partially cool, and turn it on again while it is still cooling down (Step 3). The upper plot shows the slope of the model for rising edges as time passes. The lower plot shows the false rejection rate (over a sliding window of 100 messages). With secure re-calibration (every time the bus is turned on again) and online model update (red) the false rejection rate is 0 %. If the model is not re-calibrated when the bus is turned on in different conditions (orange) a few packets might be wrongly rejected before the online update converges on the new value. Without online update (blue, green) the model cannot follow the temperature changes and the false rejection rate climbs to 50 % (with re-calibration) or even 100 % (without re-calibration).

On the effect of noise: The CAN bus is specifically designed to be resistant to noise injected in CAN High and Low.

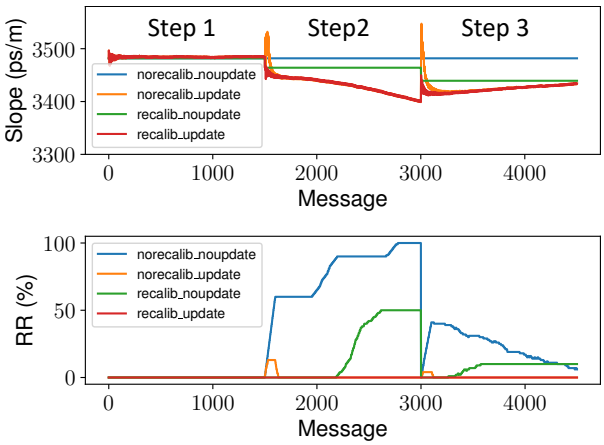


Fig. 15. Usage of EdgeTDC at three different times for varying temperature conditions within/between each run (Steps 1-3). Model slope for the rising edges (top) and false rejection rate (bottom). The false rejection rate is 0 % when secure re-calibration and online model update are performed (red), and climbs up to 100 % otherwise (orange, green, blue).

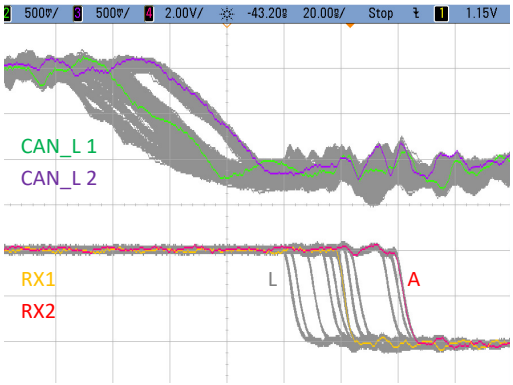


Fig. 16. Signal analysis at the two inputs of EdgeTDC. The oscilloscope is set to persistent display to visualize noise (grey shadows are the past values of the signal). Even if the CAN line is noisy, the digital RX signals (the ones used by EdgeTDC) at the output of the transceivers are clean and a unique value of TDoA can be clearly distinguished for each node.

In a differential bus, noise consists of two components: (i) differential (i.e., noise on the voltage difference between lines) and (ii) common mode (i.e., noise on the offset of both lines). Differential noise is minimized by using a twisted pair. Common mode noise is minimized by using split terminations (terminations providing a path to ground from common-mode noise) or other filtering techniques. CAN transceivers can operate with a common-mode voltage offset in the range of several volts, which can be nevertheless minimized with an appropriate power supply network. Finally, line matching with 120 Ω terminations minimizes reflections in the lines. As a result, the transceiver turns the noisy differential signal on the CAN lines into a clean digital RX signal that is then fed to the digital CAN controller and, in our case, to EdgeTDC. As shown in Figure 16, each node has a distinctive TDoA value between RX1 and RX2. The large margin between nodes is also visible in Figure 14 and Table II.

To further evaluate the effect of noise we build a noise injector, that we connect to the ground pin of the CAN connector of Node I using a 4.7 nF capacitor. We use a USRP

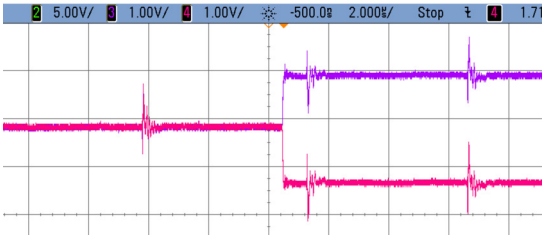


Fig. 17. Example of cross talk spikes on CANH and CANL caused by a 100 kHz square wave.

B210 Software Defined Radio (SDR) to inject random radio-frequency noise at 800 MHz. We also use a micro-controller to create a 100 kHz 5 V square-wave which, after the capacitive-coupling stage, forms short positive and negative common-mode spikes on the CANH and CANL lines. These two noise sources represent radio-frequency interference and cross-talk, respectively. Figure 17 shows an example. Running EdgeTDC in these conditions (one or both noise sources) for 10000 messages does not yield any false rejection. Indeed, though the variance of each node increases up to 137 ps, there is still a large margin between nodes. In Section VII we will further analyse the effect of malicious pulses injected by an attacker synchronously to the data bits.

CAN Relay: We re-test EdgeTDC with an additional 3 m relay from one of the side of the CAN bus to RX2 (see Figure 23 in Appendix). The relay does not alter the original bus and it only adds the same fixed offset to every TDoA. Therefore, EdgeTDC works as well as without relay, achieving zero rejection rate over 10000 messages.

C. Simple Masquerading Attacks

During a simple masquerading attack, the remote adversary tries to send a message from a compromised node claiming a victim node's ID without any further measures, i.e., the message is transmitted analogous to a legitimate CAN frame. The chances for such a simple attack to succeed are very low as our results show; irregardless of whether the system is subjected to temperature variations or not, the false acceptance rate (FAR) is 0.0 in both cases. We tested a set-up analogous to the previous where 10 nodes transmit a total of 100000 messages. $\frac{9}{10}$ of those messages constituted masquerading attacks, i.e., we had every node send regular frames as well as masquerade attempts. We tested all possible combinations of (attacker, victim) pairs. All of the masquerade attacks were detected while at the same time no legitimate transmission was rejected.

D. Remote TDoA Spoofing

As explained in Section IV, a remote adversary can launch a TDoA spoofing attack by compromising two adjacent nodes, one of which with higher priority than the victim. An example of attack is shown in Figure 18. The victim is node F with ID 0xF4. The attacker uses node D with ID 0x44 and node H with IDs 0x0F and 0xF4 (which have higher priority than the victim). First, node H sends three messages back to back: (i) a message with its own ID 0x04, (ii) a message with its own ID 0x0F, and (iii) an message with ID of the victim (0xF4) and malicious payload. As soon as it receives message (i), the

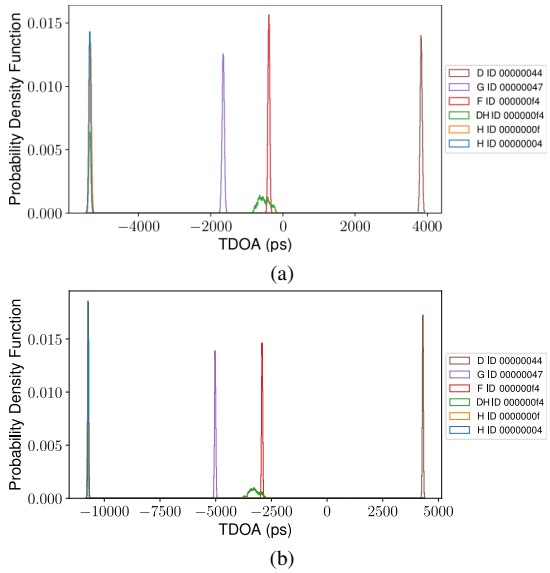


Fig. 18. Example of remote TDoA spoofing, where the remote adversary has compromised D and H. If D and H transmit at the same time, they can spoof the TDoA for node F for both rising (a) and falling (edges). D can easily synchronize with H by receiving a first message from H and transmitting during a subsequent one. However, the increased variance of the edges within the spoofed message can be easily detected by `EdgeTDC`.

	Intra-packet variance	Both edges	Update	A1	A2	Victim	FAR	FRR
1	yes	yes	yes	D	H	F	0 %	0 %
2	yes	no	yes	D	H	F	0 %	0 %
3	no	yes	yes	D	H	F	100 %	0 %
4	no	no	yes	D	H	F	100 %	0 %
5	yes	yes	yes	L	E	F	0 %	0 %
6	yes	no	yes	L	E	F	17 %	60 %
7	yes	no	no	L	E	F	41 %	0 %
8	no	no	no	L	E	F	91 %	0 %
9	no	yes	no	L	E	F	0 %	0 %

TABLE III. ABLATION STUDY FOR TDOA SPOOFING.

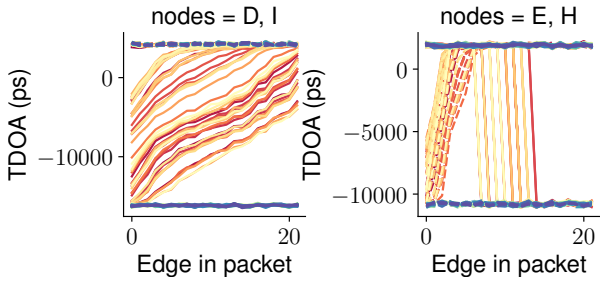


Fig. 19. Clock drift when remote attacker performs spoofing with two compromised nodes. Each line represents a packet (1000 in total). Yellow and red lines represent packets where TDoA value is between the nodes. Blue lines show messages that had no spoofing effect. The combined drift of E, H is higher than D, I.

second attacker node (D) initiates the transmission of message (iii) as well. Because the bus is still busy with a higher priority message (ii), transmission really begins only afterwards, on top of (iii). Because both H and D transmit the same message (iii) with minimal time difference due to different location, no bit error occurs. Due to clock jitter, the resulting TDoA values show a greater dispersion than a legitimate transmission, but are very likely to fall into the region halfway between the attacker node, as explained in Section IV. Message (iii) will then have the ID of the victim (0xF4) and also the TDoA of the victim (because of the combined effect of H and D as explained in Section IV. In Figure 18 the combined TDoA of H and D (HD, ID 0xF4, green) clearly overlaps with that of node F (0xF4, red) for both rising and falling edges.

In general, not all pairs will result in an an exploitable attack as D, H, F. We have analyzed all possible pairs of attackers with nodes D, E, H, I, L and we report additional results in Figure 24 in the Appendix. We focus on D, H, F because it is one of the most favorable ones from the attacker's point of view (both rising and falling edges occurring close to the victim in a balanced way). Despite this fact, `EdgeTDC` can still detect the attack by observing the higher intra-packet variance caused by concurrent transmission, as explained in Section V. In other cases, for example, using E and L the spoofing is possible on rising edges but not on falling edges, and will be easily detected by `EdgeTDC` because the TDoA of falling edges will reveal that the real sender is not the victim F. Finally, in other cases none of the edges will match the TDoA of a suitable victim.

We conduct an ablation study by running the remote TDoA spoofing attack while enabling/disabling some of `EdgeTDC` checks. Results are shown in Table III. In each case, we transmit 1000 legitimate messages from the victim and 1000 messages from the two attackers attempting to spoof TDoA. When performing the check on the variance of TDoA within each message, `EdgeTDC` detects all attacks achieving 0 % FAR and 0 % FRR (cases 1-2). An attack is detected if the variance of the edges within a message is 400 times bigger than the normal one computed during calibration. Without this check, the attack succeeds with 100 % (case 3-4).

The case of L, E, F reveals many interesting observations. When both edges and the variance check are enabled, the attack is always detected, and there is no false reject (case 5). However, when only the rising edges are used, some attacks succeed (FAR 17 %). Since the TDoA for rising edges does not exactly matches that of the victim, this likely causes *poisoning* and increases the FRR to 60 % with the message from legitimate node L mis-classified as a masquerading attack from G (this means that *poisoning* has created an opportunity to spoof G from F). We further confirm the role of *poisoning* in case 6 by disabling the model update and observing that the FRR goes back to 0 % because model poisoning has been stopped (case 7). In case 7 the FAR goes up to 41 %. This means that, while poisoning created room for spoofing G from F, it has also decreased the success rate of spoofing F from L and E. If we now disable the check on variance, the FAR climbs to 91 % (case 8), as expected since the TDoA for rising edges is close to that of the victim (but not as much as in case of H D). However, if we now enable both edges again, the FAR goes down to 0 % (case 9), as expected because for the case L E the TDoA of falling edges falls well far from the victim F.

Intra-message variance and clock drift: In Figure 19, we show how (combined) clock drift can effect the TDoA values throughout an overshadowed frame (red and yellow lines) and we observe how infeasible it is for a remote attacker to consistently spoof a location in-between the nodes with low variance. The drift "carries" the TDoA away towards one of

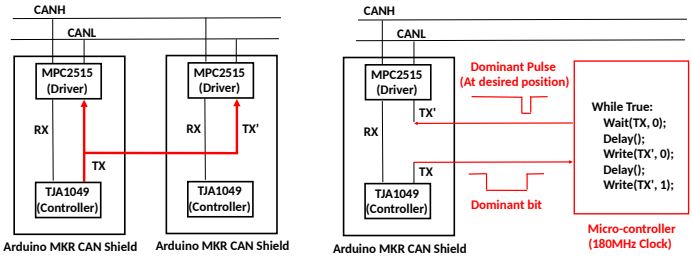


Fig. 20. Physical-layer attackers. Two transmitters driven by the same controller (left) and pulse injection (right).

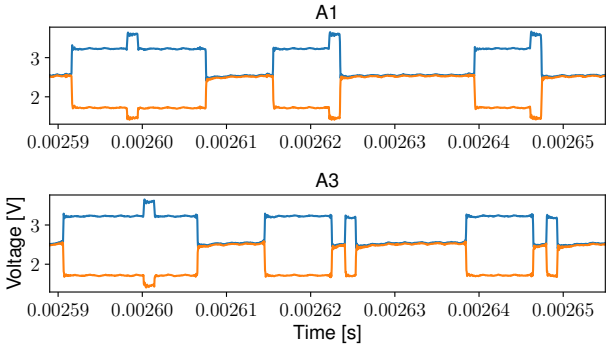


Fig. 21. Bit-level closeup of physical layer attacks. Delay 0→1 (A1) and inject dominant pulses (A3). Both attacks do not introduce CAN errors or affect decoding of the data.

the two values corresponding to the actual locations of the transmitting nodes (blue lines).

E. Physical Attacks

We present two additional physical layer attacks requiring physical access and additional hardware. The first physical layer extension, shown in Figure 20 (left) simply consists in synchronizing the two attacker nodes by connecting two transceivers at different locations to the same controller using a few tens of centimeter of jumper wire. We then vary the length of the cable to adjust the delay between the two transmitters attempting to match the TDoA of another node for both rising and falling edges. We succeed when attacking node H from nodes F and L. In this case, using the same controller minimizes the variance on the negative edges. However, positive edges still fail the variance check and `EdgeTDC` detects the attack. We could not find other better combinations.

The second physical layer extension, shown in Figure 20 (right) consists in inserting a fast micro-controller between CAN controller and CAN transceiver on the transmission side. The micro-controller can essentially delay or (if bits are known) anticipate edges by injecting short pulses right after rising (dominant-to-recessive) edges or right before falling (recessive-to-dominant) edges sent by a node at a different location. The advantage over the remote spoofing attack is the additional control on the pulse duration and position, that is however limited by the clock frequency (180 MHz in our implementation) and latency of the micro-controller. This makes it easier to balance the effect on rising and falling edges through accurate tuning. As long as the pulse is small and it falls at the beginning or end of the bit, it will not cause bit errors. `EdgeTDC` can however detect if the attacker has injected additional edges where they should not occur (e.g., one short dominant pulse between two recessive bits) because it knows the data bits and the expected number of edges. We have implemented this attack on node L. Figure 21 shows various examples of pulses at different positions in the bits.

VIII. Discussion and Limitations

The main limitation of any TDoA-based IDS is that TDoA is a physical feature that can, under specific threat models, be altered by an attacker. However, unlike other features such as voltage, TDoA correlates with physical properties that strongly constrain what an attacker can achieve. TDoA is a feature tightly coupled with the location of a node on the bus and compared to voltage-based features, and a single node can not alter its own TDoA. Applying checks such as those in `EdgeTDC`, TDoA is secure against remote attackers and mitigates many physical layer attacks.

The practical implementation of `EdgeTDC` has a limitation related to the measurement capability. Only one edge can be captured during one clock period, i.e., every 4.55 ns. Higher acquisition rates could be achieved with more capable hardware, but at an increased price.

Possible Extensions. Our proposal for secure TDoA acquisition is focused on the CAN legacy standard (ISO 11898). One future direction of work is to implement `EdgeTDC` with support for CAN-FD (Controller Area Network Flexible Data-Rate). CAN-FD is an extension to CAN used in modern high performance vehicles, which provides flexible data rate and thus features more data capacity than legacy CAN. Since the electrical layer is the same for CAN-FD, `EdgeTDC` could be extended by adding a CAN-FD capable decoder in FPGA-fabric and adjusting the arming signals for the delay lines.

Other possible directions for future work are: (1) including other physical layer features for each bit of the CAN message (e.g., bit duration, voltage) and environmental factors (e.g., temperature), (2) exploring different models for analysis and detection, (3) leveraging an echo node as proposed in [5] to measure TDoA only from one side of the bus, and (4) integrating `EdgeTDC` into a custom ASIC for deployment.

IX. Related Work

Network segmentation separates CAN nodes into multiple independent CAN buses based on their vulnerability to external attack, their necessity to operate a vehicle, and their interaction with other nodes. The works in [19] and [22] are two proposals of this category, providing protection mechanisms at the cost of increased design and maintenance costs and only partially mitigating the problem.

Encryption prevents the spoofing of messages and increases confidentiality. By encrypting data with a key dependent on the CAN message ID, such an approach prevents unauthorized nodes from generating valid messages, see, e.g., [14], [42]. However, often additional cryptographic hardware is needed and the bus load is increased due to the need of multiple messages for one encrypted payload. Another challenge is the complexity of handling and updating keys, which we briefly

discuss below when describing approaches aiming to achieve sender authentication.

Message or sender authentication attempts to verify whether a node on the bus is authorized to send messages with certain IDs. Similar to encrypting the payload, the content of a message is signed using a key specific to the message ID. This leads to an increased bus load and the need to manage keys among nodes over time (up to the lifetime of a car). Depending on the security-level, such systems could be broken if an adversary has decades to crack the keys. Sender authentication is proposed in, e.g., [18] and [48]. The latter includes potential real world applications, but the automotive industry has yet to adopt a solution for secure CAN bus based on cryptography. New designs for "secure" CAN appear to focus on protocol and physical-layer features [37].

IDSs. Intrusion Detection Systems (IDSs) are the most researched area of CAN security. Through various network and physical layer parameters or collections thereof, IDSs try to differentiate between legitimate and malicious messages. More recent ideas such as [41] use machine learning and artificial intelligence techniques to assemble a large variety of parameters extracted from the transmitted messages for detection. Others, such as [45] and [12], focus on physical characteristics. Voltage-based and message-based IDSs have been shown to be vulnerable to spoofing [4], [6].

X. Conclusion

We have studied the security implications of CAN bus IDSs that identify masquerading attacks based on TDoA measurements. We have identified novel remote spoofing and poisoning attacks. We have proposed `EdgeTDC`, a novel resilient approach that can successfully detect masquerading attacks launched by compromised CAN nodes, even when the attacker attempts to alter the TDoA measurements by spoofing and poisoning. We have implemented `EdgeTDC` with inexpensive off-the-shelf hardware and demonstrated its effectiveness with a thorough theoretical and experimental analysis.

Acknowledgment

This research has received funding from the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40_180545.

References

[1] I. S. Association, "IEEE Standard for Low-Rate Wireless Networks–Amendment 1: Enhanced Ultra Wideband (UWB) Physical Layers (PHYs) and Associated Ranging Techniques," *IEEE Std 802.15.4z-2020 (Amendment to IEEE Std 802.15.4-2020)*, pp. 1–174, 2020.

[2] R. Baker and I. Martinovic, "Losing the car keys: Wireless {PHY-Layer} insecurity in {EV} charging," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 407–424.

[3] Benjamin Blase, "tdc-fpga: Time to digital converter for use on a Xilinx 7-series FPGA." https://github.com/benbr8/tdc-fpga, accessed 2022-05-07.

[4] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu, "Evading voltage-based intrusion detection on automotive CAN," in *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/evading-voltage-based-intrusion-detection-on-automotive-can/

[5] E. Biham, S. Bitan, and E. Gavril, "Tcan: Authentication without cryptography on a CAN bus based on nodes location on the bus," in *2018 Embedded Security in Cars, November 2018*, 2018. [Online]. Available: https://www.cs.technion.ac.il/~biham/Workshops/Cyberday/2019/Slides/cyberday-2019-6-biham-tcan_presentation.pdf

[6] G. Bloom, "WeepingCAN: A Stealthy CAN Bus-off Attack," in *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, vol. 2021, 2021, p. 25.

[7] G. Bloom, G. Cena, I. C. Bertolotti, T. Hu, and A. Valenzano, "Supporting security protocols on CAN-based networks," in *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2017, pp. 1334–1339.

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

[9] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1044–1055.

[10] ——, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 911–927.

[11] ——, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1109–1123.

[12] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018.

[13] Comma.ai, "openpilot on the comma three," https://github.com/commaai/openpilot, accessed 2022-05-28.

[14] T. P. Doan and S. Ganesan, "CAN crypto FPGA chip to secure data transmitted through CAN FD bus using AES-128 and SHA-1 algorithms with a symmetric key," SAE Technical Paper, Tech. Rep., 2017.

[15] U. Ezeobi, H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Reverse engineering controller area network messages using unsupervised machine learning," *IEEE Consumer Electronics Magazine*, vol. 11, no. 1, pp. 50–56, 2022.

[16] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "SIMPLE: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 229–244.

[17] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Washington, D.C.: USENIX Association, Aug. 2015. [Online]. Available: https://www.usenix.org/conference/woot15/workshop-program/presentation/foster

[18] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "LiBrA-CAN: a lightweight broadcast authentication protocol for controller area networks," in *International Conference on Cryptology and Network Security*. Springer, 2012, pp. 185–200.

[19] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 11–25, 2011.

[20] T. Instruments, "SN65HVD233-HT 3.3V CAN Transceiver," https://www.ti.com/lit/ds/symlink/sn65hvd233-ht.pdf, accessed 2022-06-08.

[21] H. J. Jo, J. H. Kim, H.-Y. Choi, W. Choi, D. H. Lee, and I. Lee, "Mauth-can: Masquerade-attack-proof authentication for in-vehicle networks," *IEEE transactions on vehicular technology*, vol. 69, no. 2, pp. 2204–2218, 2019.

[22] R. Kammerer, B. Frömel, and A. Wasicek, "Enhancing security in CAN systems using a star coupling router," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*. IEEE, 2012, pp. 237–246.

[23] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 787–800.

[24] M. Kneib, O. Schell, and C. Huth, "EASI: Edge-Based Sender Identification on Resource-Constrained Platforms for Automotive Networks." in *NDSS*, 2020.

[25] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 447–462.

[26] D. Li, M. Tian, R. Jiang, and K. Yang, *Exploiting Temperature-Varied Voltage Fingerprints for In-Vehicle CAN Intrusion Detection*. New York, NY, USA: Association for Computing Machinery, 2021, p. 116–120. [Online]. Available: https://doi.org/10.1145/3472634.3472662

[27] M5STACK, "M5Stack Atom CAN," https://docs.m5stack.com/en/atom/atom_can, accessed 2022-06-06.

[28] M. Marchetti and D. Stabili, "READ: Reverse Engineering of Automotive Data Frames," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2019.

[29] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.

[30] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[31] Mohor, Igor, "CAN Protocol Controller," https://opencores.org/projects/can, accessed 2022-05-07.

[32] C. Moreno and S. Fischmeister, "Sender Authentication for Automotive In-Vehicle Networks through Dual Analog Measurements to Determine the Location of the Transmitter," in *Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019*, P. Mori, S. Furnell, and O. Camp, Eds. SciTePress, 2019, pp. 596–605. [Online]. Available: https://doi.org/10.5220/0007580105960605

[33] P. Murvay and B. Groza, "TIDAL-CAN: Differential Timing Based Intrusion Detection and Localization for Controller Area Network," *IEEE Access*, vol. 8, pp. 68 895–68 912, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2985326

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

[35] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking tesla from wireless to can bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.

[36] S. Nie, L. Liu, Y. Du, and W. Zhang, "Over-the-air: How we remotely compromised the gateway, BCM, and autopilot ECUs of Tesla cars," *Briefing, Black Hat USA*, 2018.

[37] NXP, "Secure HS-CAN Transceiver with Standby Mode," https://www.nxp.com/products/interfaces/can-transceivers/secure-can-transceivers/secure-hs-can-transceiver-with-standby-mode:TJA1152, accessed 2022-06-05.

[38] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "LibreCAN: Automated CAN Message Translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2283–2300. [Online]. Available: https://doi.org/10.1145/3319535.3363190

[39] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent transportation systems*, vol. 16, no. 2, pp. 546–556, 2014.

[40] A. SARL, "Arduino CAN Shield," https://store.arduino.cc/products/arduino-mkr-can-shield, accessed 2022-05-07.

[41] E. Seo, H. M. Song, and H. K. Kim, "GIDS: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–6.

[42] A. S. Siddiqui, Y. Gui, J. Plusquellic, and F. Saqib, "Secure communication over CANBus," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 1264–1267.

[43] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *2016 international conference on information networking (ICOIN)*. IEEE, 2016, pp. 63–68.

[44] STMicroelectronics, "L9616 - Automotive high speed CAN bus transceiver," https://www.st.com/en/automotive-analog-and-power/l9616.html, accessed 2022-06-08.

[45] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2015, pp. 45–49.

[46] S. Thakur, C. Moreno, and S. Fischmeister, "CANOA: CAN Origin Authentication Through Power Side-Channel Monitoring," *arXiv preprint arXiv:2006.06993*, 2020.

[47] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT workshop on Lightweight Cryptography*, vol. 2011. ECRYPT, 2011, p. 20.

[48] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *2014 International Conference on the Internet of Things (IOT)*. IEEE, 2014, pp. 13–18.

[49] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Transactions on intelligent transportation systems*, vol. 16, no. 2, pp. 993–1006, 2014.

APPENDIX

In this appendix we provide additional data on our systematic evaluation of attacks and defenses.

Figure 22 shows the physical layer desing of `EdgeTDC`. All components fall in the same clock region, and all the delay line elements (orange) are close together and share the same clock lines. Figure 23 shows how we can actively relay the CAN connection from one side of the CAN bus to `EdgeTDC`.

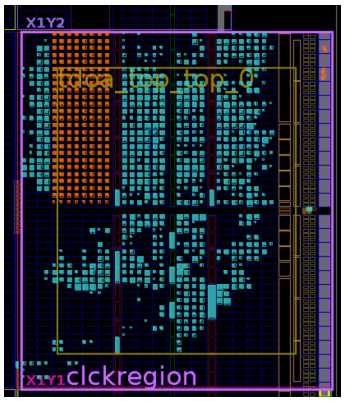


Fig. 22. Physical-layer design of `EdgeTDC`. Logic elements highlighted in orange form the TDCs / delay lines.

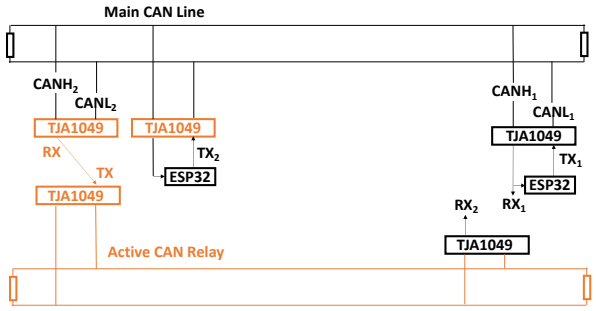
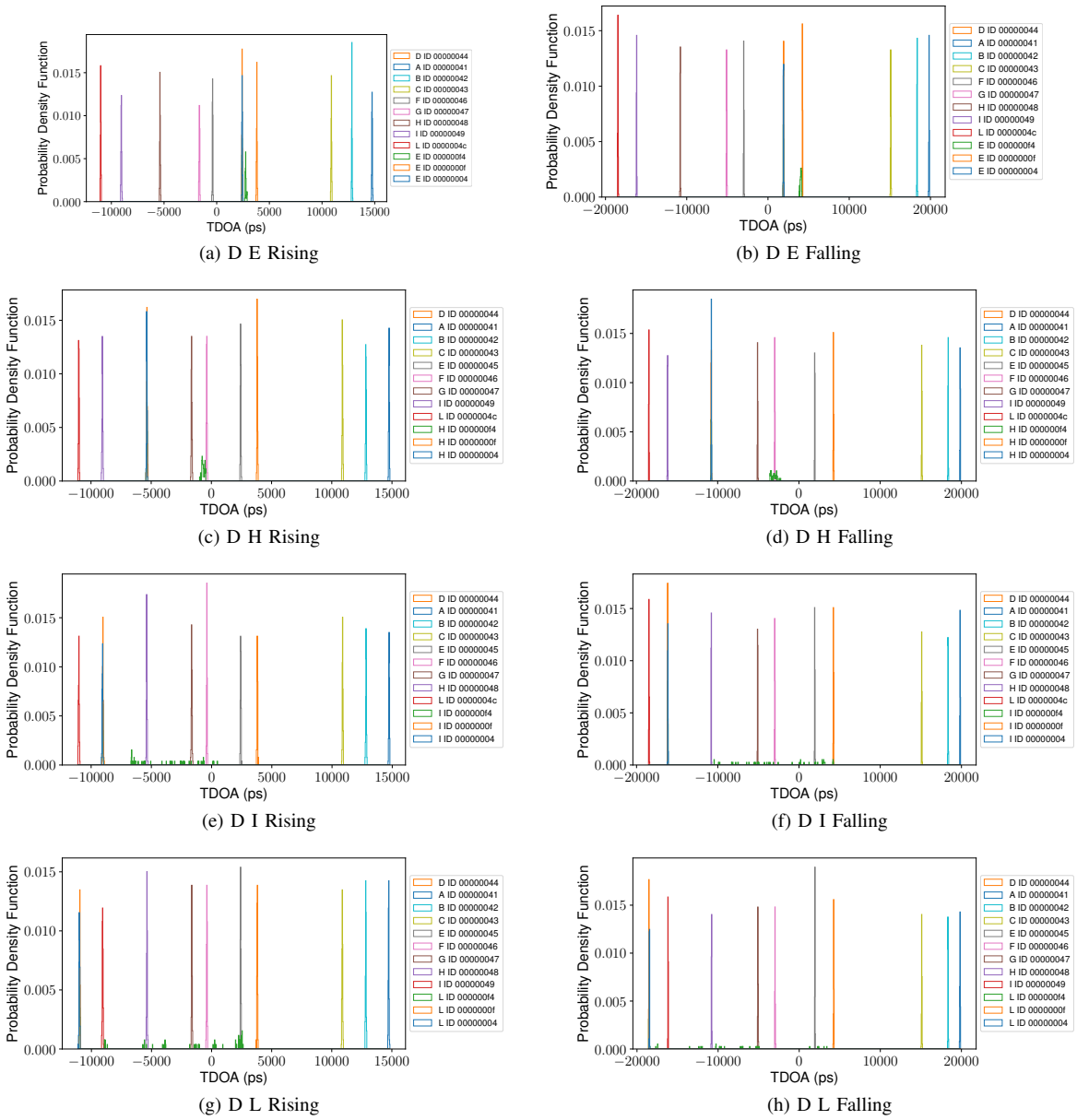


Fig. 23. Active CAN relay.



(a) D E Rising

(b) D E Falling

(c) D H Rising

(d) D H Falling

(e) D I Rising

(f) D I Falling

(g) D L Rising

(h) D L Falling

Fig. 24. Systematic analysis of node D transmitting on top of nodes E, H, I, L.

Figure 24 shows a systematic analysis of spoofing at the logical layer. We show what happens to the TDOA when node D transmits on top of nodes E, H, I, L. Other combinations have similar results and are omitted for brevity. In general, the resulting TDOA (green) falls in between the TDOA of the two attackers (e.g., D, H). In the case of the D H pair, the resulting TDOA matches the one of node F, opening an opportunity for successful masquerading attacks that also spoof the correct value of the TDOA. However, the variance of TDOA measured over each edge of the packets is much higher in this case, allowing for effective detection.