ERACAN: Defending Against an Emerging CAN Threat Model

Zhaozhou Tang
Georgia Institute of Technology
Atlanta, USA
ztang306@gatech.edu

Khaled Serag
Qatar Computing Research Institute
Doha, Qatar
kseragalsharif@hbku.edu.qa

Saman Zonouz
Georgia Institute of Technology
Atlanta, USA
saman.zonouz@gatech.edu

Z. Berkay Celik
Purdue University
West Lafayette, USA
zcelik@purdue.edu

Dongyan Xu
Purdue University
West Lafayette, USA
dxu@purdue.edu

Raheem Beyah
Georgia Institute of Technology
Atlanta, USA
rbeyah@coe.gatech.edu

Abstract

The Controller Area Network (CAN) is a pivotal communication protocol extensively utilized in vehicles, aircraft, factories, and diverse cyber-physical systems (CPSs). The extensive CAN security literature resulting from decades of wide usage may create an impression of thorough scrutiny. However, a closer look reveals its reliance on a specific threat model with a limited range of abilities. Notably, recent works show that this model is outdated and that a more potent and versatile model could soon become the norm, prompting the need for a new defense paradigm. Unfortunately, the security impact of this emerging model on CAN systems has not received sufficient attention, and the defense systems addressing it are almost nonexistent. In this paper, we introduce ERACAN, the first comprehensive defense system against this new threat model. We first begin with a threat analysis to ensure that ERACAN comprehensively understands this model's capabilities, evasion tactics, and propensity to enable new attacks or enhance existing ones. ERACAN offers versatile protection against this spectrum of threats, providing attack detection, classification, and optional prevention abilities. We implement and evaluate ERACAN on a testbed and a real vehicle's CAN bus to demonstrate its low latency, real-time operation, and protective capabilities. ERACAN achieves detection rates of 100% and 99.7%+ for all attacks launched by the conventional and the enhanced threat models, respectively.

CCS Concepts

• **Security and privacy → Network security**.

Keywords

Automotive Security; Controller Area Network; Intrusion Detection

ACM Reference Format:
Zhaozhou Tang, Khaled Serag, Saman Zonouz, Z. Berkay Celik, Dongyan Xu, and Raheem Beyah. 2024. ERACAN: Defending Against an Emerging CAN Threat Model. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690267

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690267

1 Introduction

Since its introduction in the 1980s, the Controller Area Network (CAN) has solidified its position as the primary in-vehicle network and extended its influence to diverse cyber-physical systems (CPSs), allowing hundreds of electronic control units (ECUs), which govern various sensors, actuators, and CPS control functions, to communicate. Decades of widespread adoption have spurred substantial security research. Initially, accessing the bus was believed viable only through physical access, typically granted to authorized users. However, increased ECU connectivity challenged this assumption, allowing malicious attackers to exploit wireless channels such as WiFi, cellular, and Bluetooth to remotely compromise ECUs [8, 40, 45, 46, 68]. This shift ushered in a once-deemed unrealistic threat model: remote attackers.

From an OSI standpoint, the standard outlines the communication rules of the physical and data link layers for a broadcast-based bus and leaves the upper layers open to allow flexibility for various use cases. Typical ECUs connect to the bus through a CAN protocol controller and a transceiver, which enforce the specification rules for the data link and physical layers, respectively. The existing literature assumed that remote attackers could control the ECU but not the controller or transceiver, effectively leaving the data link and physical layers intact. This meant that they could only read or write entire messages assembled by an unbroken CAN controller. With these two capabilities, many works have shown that attackers could launch a plethora of attacks, including fake message injection, masquerading, flooding, error injection, and suspending other ECUs [2, 4, 8, 9, 33, 39, 40, 45, 46, 58, 59, 68, 69].

As a response, researchers proposed several defense approaches. Some explored protecting against certain attack types (e.g., masquerade) using techniques such as MACs or secret numbers to provide source authentication. However, due to limitations including message length, busload, key management, and the limited processing powers of most ECUs, intrusion detection systems (IDSs) [10, 13, 19, 30–32, 52, 56, 57, 61, 72] gained more traction. These approaches, which contain a super-node handling the bulk of the security work, are more suitable for CAN systems due to their performance-friendliness. The node monitors traffic and detects anomalies leveraging features, including message frequency, payload, timing, and physical signal characteristics.

Despite the ostensible maturity of CAN security research, recent developments suggest otherwise. While the literature assumes that

remote attackers cannot control the link layer, recent works indicate that this assumption may be obsolete. Techniques including manipulating peripheral clock gating or remapping transceivers' IO ports allow attackers substantial link layer control on many ECUs [6, 14, 34, 63]. Unlike the Conventional Remote Attacker Model (CRAM), where only two basic abilities–sending and receiving entire messages through an intact controller–enabled various attacks, the Enhanced Remote Attacker Model (ERAM) possesses expansive capabilities, including injecting pulses, incomplete frames, and edge control. These boosted abilities introduce more advanced attacks whose security implications are not fully investigated. Research demonstrates that the ERAM abilities may improve existing attacks [34, 43, 48], enable new attacks [64, 73], and circumvent current defense systems. Surprisingly, no defense systems addressing this threat model are proposed, and a comprehensive analysis of its security impact remains absent.

In response to this critical gap, we introduce ERACAN, the first comprehensive defense system against ERAM attackers. ERACAN provides detection for all ERAM and CRAM attacks, with optional prevention where feasible. As ERAM spans various attacks, ERACAN also provides attack classification, making clear the specific type of attack it is reacting against. Defending against ERAM poses major challenges. **C1:** First, a thorough analysis needs to identify the full range of ERAM abilities, attacks, and security implications, which is absent from prior works. **C2:** Second, confronting an attacker able to manipulate low-level events requires monitoring features spanning both the physical and link layers. A complete feature set covering all ERAM attacks is difficult to find due to the extensive ERAM capabilities. Existing defenses focus on single feature categories, hindering the detection of most ERAM attacks. For example, defenses that identify message senders and check if they are legitimate using voltage features cannot detect messages transmitted by a legitimate sender using ERAM techniques to achieve malicious goals [64, 73]. **C3:** Finally, detection must ensure reliability and abide by deadlines for effective incident response (e.g., destroying malicious messages before they are received). Crafting a monitoring strategy satisfying both goals presents another significant obstacle. Current sender identification approaches cannot detect attacks injecting short pulses. Similarly, using the GPIO to surveil the link layer is unreliable and computationally expensive.

We first address **C1** by thoroughly analyzing the ERAM model to understand its capabilities and security impacts through literature review and extrapolation. We then identify the needed feature set including bit timing, voltage, and link layer events to cover all ERAM attacks and address **C2**. Finally, to monitor these features and meet the requirements of **C3**, ERACAN uses a dedicated monitor node adopting a dual-faceted *delegation*, and *smart checking* strategy. It deploys a customized FPGA controller (ERACAN controller) for autonomous link layer surveillance to ensure reliability, customizability, and parallel execution. For the physical layer, ERACAN deviates from traditional sender identification. It mainly uses features to model valid message properties with simple equations and performs checks selectively based on attack scenarios. This reduces complexity and simplifies processing to meet deadlines. ERACAN is cost-efficient and requires minimal hardware changes, merely attaching a single monitor node. To help the research community build on ERACAN, we open source our FPGA design.[1]

Figure 1: Format of a standard CAN data frame.

For inclusivity, we evaluate the performance and security of ERACAN on a testbed and a real vehicle's CAN bus. ERACAN achieves 100% detection for CRAM attacks, 99.7%-100% detection for ERAM attacks, and an attack classification accuracy of 98.8%-100%. Overall, we make the following contributions:

- We introduce ERACAN, the first defense system against the ERAM model, offering real-time detection, classification, and optional prevention abilities for CRAM as well as ERAM attacks.
- We systematically threat-analyze ERAM to understand its features, defense evasion tactics, and the attacks it enables or improves, as a basis for ERACAN and future ERAM defenses to build on.
- We propose a new autonomous surveillance mechanism for intricate link layer events by delegating it to a configurable FPGA controller (ERACAN controller) to ensure parallel execution and real-time performance. Additionally, we introduce a *smart checking* approach for physical layer features. Finally, we provide open access to our FPGA controller design to support further research.
- We offer a performance analysis as well as a security analysis of ERACAN against various ERAM attacks and evasion tactics.
- We demonstrate ERACAN's feasibility, real-time abilities, performance, and protective capabilities by evaluating it on a testbed and a real vehicle's CAN bus and achieve excellent results.

2 Background

2.1 CAN Basics

Bit Encoding. CAN uses differential voltage between CANH and CANL to encode bits. A positive (dominant) voltage denotes 0 and zero (recessive) voltage denotes 1. When two nodes send a 1 and 0 concurrently, all nodes read a 0. If five identical bits are transmitted consecutively, a *stuff* bit of the opposite value is inserted.

Frame Format. Fig. 1 shows the various message fields. The ID determines the priority of a message, with lower IDs indicating a higher priority. When two nodes start transmission at the same time, they perform arbitration. Each node sends one bit at a time. The first node to transmit a 1 yields. A message terminates with the End of Frame (EOF) field (seven 1s). The next consecutive message is separated by at least three additional Inter Frame Space bits (IFS).

Error Handling. The CAN standard defines five kinds of errors: *bit, stuff, form, CRC*, and *acknowledgement* errors. Upon detecting an error, nodes signal with an error frame. A CRC error is signaled after the ACK field. Other errors are signaled at the next bit after where they are detected.

Error States. Every node keeps a Transmit Error Counter (TEC) and a Receive Error Counter (REC) to keep track of errors encountered during transmission or reception, respectively. If TEC or REC

[1] https://tinyurl.com/5n77avxu

exceeds 127, nodes enter the error-passive state, where stricter error signaling and transmission rules are enforced. If TEC or REC exceeds 255, they enter the bus-off state and stop communicating.
Sampling. CAN controllers divide a single bit into four segments of configurable durations: *synchronization, propagation delay segment*, and the *phase buffer 1 and 2 segments*. Controllers interpret the bit value at the *sample point* at the end of the *phase buffer 1 segment*.
Synchronization. A node expects 1→0 edges from other nodes within the the *synchronization* segment. If it observes the edge outside the *synchronization* segment, it *resynchronizes* either by lengthening *phase buffer 1* or shortening *phase buffer 2* segment.

2.2 Conventional Remote Attacker Model

2.2.1 Basic Abilities. CRAM attackers possess three basic abilities:
Valid Message Reception. They can read data or remote frames after they are fully received and validated by the CAN controller.
Valid Message Transmission. They can only transmit valid data or remote frames using the CAN controller and need to abide by all protocol rules, such as arbitration.
Simultaneous Transmission. This is a crucial technique to inject collisions. The attacker transmits a message with the same ID as the victim at the same time but with a different payload, so they both win arbitration and gain bus access. To synchronize the messages, the attacker needs to predict when the victim's message arrives, inject one or more preceded messages with a higher-priority ID slightly earlier, and then transmit a message with the target ID.

2.2.2 Possible Attacks. Two primary attack categories are possible:
Masquerading Attacks. Since CAN does not provide authentication, a CRAM attacker can masquerade as other ECUs by transmitting messages under their IDs to *forge* or *replay* messages and alter system functions the victim is in charge of.
Error-Handling Attacks. With *simultaneous transmission*, attackers can inject errors in a victim's message to exploit CAN error handling rules. They can *destroy messages*, push victims to the error-passive state to *map the network* [58], or push them to the bus-off state to achieve *targeted DoS* [9]. They can also inject collisions to corrupt messages' physical signals, *poison the retraining process of physical signal based defenses*, and render them ineffective [4].

3 Related Work

Timing Based Approaches. These approaches use message timing features such as clock skews and message intervals to detect anomalies [11, 53, 61, 72]. Despite being lightweight, some of them may not extend their protection to non-periodic messages. Evasion tactics for some approaches already exist under CRAM [54] and new ERAM tactics further undermine their security (Sec. 4.3).
Payload Inspection Approaches. These approaches inspect message payloads, extract statistical features, and use machine learning to check their plausibility [1, 38, 67]. Although they are good at detecting injection attacks, some may only detect anomalies consisting of message flows and could allow low-level attacks to pass unnoticed. Under CRAM, researchers have demonstrated evasion attacks that reduce the performance of some of these systems. [7]. Moreover, ERAM presents new evasion tactics against them (Sec. 4.3).
Cryptographic Approaches. Some researchers proposed providing sender and content authenticity using MACs [21, 26, 49, 50, 66], or sender authenticity only using secret tokens embedded in messages [25, 27, 70]. Researchers have made progress in making these approaches more lightweight, but these improvements are still not enough to enable wide adoption.
Secret Delay Approaches. These approaches embed authentication information in secret delays between messages instead of their payloads [22, 23, 57, 71]. Among them, ZBCAN additionally detects and prevents error-handling attacks. However, it does not protect against ERAM attackers (Sec. 4.3). Furthermore, approaches using secret delays require changing message schedules and could lead to bus load increases and priority inversions.
Physical Signal Approaches. These approaches identify message senders with ECUs' physical signal features, such as voltage [10, 12, 13, 19, 31, 32], bit timing [47, 56, 74], and time difference of arrival [44, 52, 55]. They offer good security against masquerading attacks but do not detect other attacks, except for VoltageIDS [13] which detects some error-handling attacks. Moreover, researchers have demonstrated poisoning attacks against some of these defenses [4]. Finally, they only consider CRAM attacks and are vulnerable under ERAM in several ways (Sec. 4.3).
Hardware Approaches. Some defenses use gateways or relays to isolate attackers [24, 29]. Although very effective against some attacks, they require significant hardware modifications and are too expensive to be adopted for all nodes on the bus.

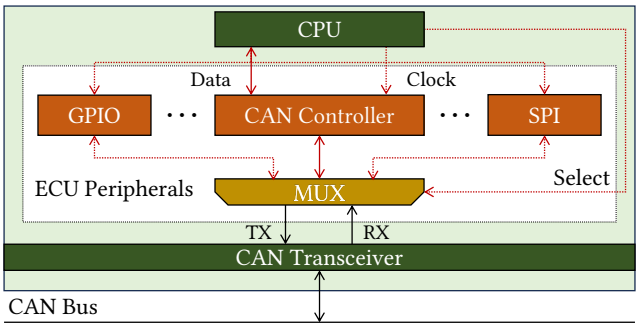


Figure 2: Typical Architecture of an ECU.

4 The ERAM Threat Model

Fig. 2 shows the architecture of a typical ECU with several peripherals to facilitate various functions. During normal operation, it connects to the bus through a CAN controller and a transceiver. Previously, it was assumed that attackers could not change the path to the bus or influence the controller operation in any way. However, recent work [34] showed that, by manipulating peripheral clock gating, the attacker could partially control the controller. Another recent work [14] showed that, by re-mapping the IO ports of the transceiver, they could completely disconnect the controller and instead connect it to other peripherals, such as the GPIO, SPI, UART, or others depending on the ECU's architecture.

In this paper, we assume a strong Enhanced Remote Attacker Model (ERAM). We assume the attacker can connect the transceiver to any other peripheral of the ECU and the ECU has all the commonly used peripherals. This means the attacker has full control over the data link layer and the layers above it, but still abides by

the physical layer's rules as it uses the transceiver. In this section, we first identify ERAM attackers' capabilities and the features they unlock. Then, by thoroughly reviewing and extrapolating existing literature on link layer attacks [14, 43, 62], we analyze their security impacts on aspects such as enabling new attacks, improving existing ones, and circumventing existing defenses.

4.1 Main Capabilities and Features

In addition to all the capabilities of the conventional CRAM model, ERAM possesses more sophisticated ones, each unlocking various features as discussed below.

Ubiquitous Link Layer Visibility. In CRAM, only valid frames received, assembled, and checked by the controller are visible to the attacker. However, an ERAM attacker can read the bus level through the transceiver at any point in time. This unlocks several previously unattainable features, such as:

Real-time bit inspection: ERAM attackers could read all message bits in real-time, without waiting for the controller to assemble them first. This includes the ID [14], payload, or the various protocol fields, such as the CRC. This ability has several uses, such as attacking messages with specific IDs or content.

Timing bus events: The attacker can accurately time various bus events, such as the start of a frame, the transmission of a particular ID [14] or field value, and many others. This is crucial for accurate attack timing, such as injecting signals at a specific message field.

Atypical event visibility: ERAM visibility spans all events, including valid but rare ones, such as error or overload frames, and invalid ones, such as pulses, incomplete, and invalid data frames. This ability has various uses, such as helping the attacker track the errors a specific ECU encounters.

Omnipotent Link Layer Write-Ability. CRAM attackers can only inject entire frames, assembled and checked by the CAN controller and abide by all bus access rules, such as arbitration. Contrastingly, ERAM attackers can directly inject signals onto the bus at any time. The nature of the signals could vary from pulses to complete frames. This unlocks several previously unattainable features, such as:

Arbitrary frame injection: ERAM attackers could inject bits one-by-one to form any complete frame at any time. Beyond valid data frames, this could vary from overload [64] and error frames [14, 34, 43, 48], to invalid frames, such as frames with a wrong CRC.

Arbitrary non-frame injection: ERAM attackers could inject non-frames, such as pulses, bits [43], and partial frames [17, 29, 62]. This feature is significant as it allows the attacker to flip bits of valid frames, overwrite frames, or cause synchronization problems.

Arbitrary edge and width control: ERAM attackers can control the edges or widths of their injected signals [43, 64]. This could allow attackers to force nodes to resynchronize or exploit sample point differences among different CAN controllers.

Link Layer Rule Non-Compliance. The attacker's control over the link layer extends to all its rules, such as arbitration, error states, and acknowledgment. The attacker may completely ignore the rules or apply their own rules. For example, they may continue sending a frame after being interrupted by an error frame, not transition to the error-passive or bus-off states, or acknowledge their own frames. This could help them in many scenarios, such as circumventing defense systems that rely on link layer rules (Sec. 4.3).

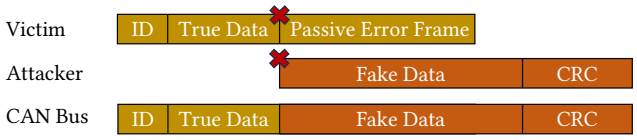


Figure 3: Frame hijacking attack.

4.2 ERAM Attacks

4.2.1 New Attacks. Besides all CRAM attacks (Sec. 2.2.2), ERAM attackers can launch many completely new attacks.

Frame Hijacking. As demonstrated by [17, 62], ERAM attackers could push a victim to the error-passive state and hijack frames mid-transmission. As shown in Fig. 3, the attacker flips a recessive bit into a dominant one and then continues transmission including a valid CRC. The victim detects a bit error and sends a passive error flag consisting of six 1s. However, it is overwritten by the attacker's payload and not visible to other nodes. Other nodes receive the attacker's forged data and a valid CRC and treat it as a valid message from the victim. This could be very useful if the beginning portion of the message is used for authentication or anomaly detection.

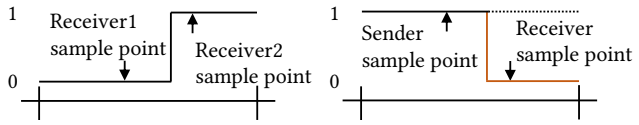
Double Receive. As shown in [64], an ERAM attacker could flip the last bit of EOF from 1 to 0. The receiver treats the message as valid as it contains no errors up to the second-to-last bit of EOF [20]. However, the sender detects a bit error and retransmits the message, causing the receiver to get the same message twice. This could cause problems in several scenarios. For example, the receiver may act on the same message twice when messages do not use sequence numbers, or nodes may lose synchronization when they need to agree on message counters.

Freeze Doom Loop. As proposed by [64], an ERAM attacker can send an overload frame at the first bit of IFS to falsely indicate that a node needs more time to process the received message. This cannot be done under CRAM because CAN controllers cannot be programmed by software to generate overload frames. On receiving an overload frame, other nodes also send overload frames and bus traffic is delayed. This could repeat indefinitely without increasing any ECUs' error counters, making it difficult for some IDSs to detect.

Unorthodox Frames. Based on the *arbitrary frame injection* capability, we find that the attacker could forge and inject frames that do not fully comply with the CAN standard, such as a message with a data field longer than 8 bytes, or a remote frame with data. This could achieve various purposes such as causing errors or discovering implementation mismatches between different CAN controllers of insufficiently defined parts of the standard.

Arbitration Denial. By flipping a 1 to a 0 in the ID field of a message in [14, 29, 43], the victim loses arbitration and the message is delayed, potentially missing its deadline. The attacker could perform this repeatedly to prevent the victim from ever gaining bus access. To cover his tracks, the attacker can continue transmitting a frame after winning arbitration so other nodes see a valid data frame and do not detect errors.

Synchronization Disruption. When a victim transmits a 1, the attacker can inject a 0 pulse after the synchronization segment [14, 43]. The sender and receivers see a 1→0 edge outside synchronization segments and resynchronize (Sec. 2.1). Depending on their bit time



(a) Janus frame. **(b) Counterfeit frame.**

Figure 4: Attacks exploiting sample point differences.

settings, they may adjust their bit durations differently, lose synchronization, and experience communication errors.

Janus Frames. In Fig. 4a, when two nodes have different sample points and an attacker sends a bit with a 1⟶0 or 0⟶1 transitions between their sample points, they read different values. The attacker can send a carefully chosen frame with such transitions so two nodes receive different contents and do not detect errors [64, 73].

Counterfeit Frames. Extrapolating the Janus frame idea, we deduce and verify that an attacker can flip bits from 1 to 0 in a sender's message without causing errors. For example in Fig. 4b, when the sender sends a 1, the attacker injects a 0 pulse after the sender's sample point that lasts after the receiver's sample point. The sender samples a 1 and does not detect bit errors, while the receiver samples a 0. The attacker can flip bits in both the data and CRC fields to modify a message received by the receiver without causing errors.

4.2.2 Improved Attacks. ERAM abilities can improve certain CRAM attacks and provide more stealth, reliability, and flexibility.

ERAM Error Injection. While CRAM attackers can inject errors by simultaneous transmission which usually involves injecting preceded messages to synchronize two messages (Sec. 2.2.1), an ERAM attacker could inject an error directly, making it stealthier and more deterministic [14, 34, 43, 48]. Further, while CRAM has limited control over the location or type of the error, ERAM attackers have substantial control over both by injecting bits or error frames at any location of their choosing in a victim message.

Physical Fingerprint Corruption. Building on voltage corruption attacks proposed by Bhatia et al. [4], we conceive an improved physical fingerprint corruption attack. Bhatia's technique involves causing several errors, transitioning a node into the error-passive state, cooperating between two attacking ECUs, and other requirements. In the improved attack, however, ERAM attackers directly inject pulses that overlap with parts of a victim's message to corrupt their physical characteristics without any such needs, making it stealthier and more convenient.

4.3 Impacts on Existing Defenses

On Physical Signal Based Approaches. ERAM abilities are problematic to systems using physical signals to identify attackers or detect intrusions. For example, many approaches use physical signal features from a specific part of the message to check authenticity [12, 32, 55, 56]. ERAM attackers could manipulate these systems in several ways. For example, attackers could leave such parts of the message intact, but hijack the frame after they elapse. For systems that take several samples all over the message with online updates [10], they only detect spoofing of entire messages and struggle with ERAM attacks that only require injecting short pulses, as [14] points out. Moreover, the attacker may gradually corrupt or hijack small parts of the frame to change the system's definition of a valid signal. Transmitting messages with GPIO, the attacker may also directly control bit timing of his messages to emulate the characteristics of other ECUs and evade bit timing based approaches [47, 56, 74].

On Error Handling Defenses. Some defenses attempt to prevent CRAM error-handling attacks by making it difficult for the attacker to transmit a message simultaneously with the victim. This is done by randomizing the message's transmission time or parts of its ID. ERAM's ability to time attacks accurately and inject errors arbitrarily bypasses any of these defenses as it could launch the attack once the message or the fixed part of its ID appears.

On Cryptographic Approaches. Some cryptographic approaches embed secret tokens in fields such as the ID to provide sender authentication [25, 70]. They are not secure under ERAM because attackers can wait after these tokens are transmitted and then hijack the frame. Further, approaches that keep message or freshness counters between senders and receivers [26, 50, 66] may be vulnerable to the double receive attack (Sec. 4.2.1) as it could cause the legitimate transmitter to send a message with the same counter twice. Finally, for content authentication defenses that use a central authenticator [35], Janus and counterfeit frame attacks (Sec. 4.2.1) could be used to falsify a legitimate frame to keep it looking valid for the authenticator but containing false data for some or all receivers, depending on their sample points.

On Timing Based Approaches. Defenses using message timings to assess authenticity, such as natural intervals [53, 61, 72] or secret delays [57, 71] between messages, are vulnerable under ERAM. Attackers can hijack or counterfeit messages to modify their contents without changing their transmission time, evading such defense.

On Payload Inspection Approaches. Since these approaches only process application layer information, they cannot detect low-level ERAM attacks, as noted by [14]. Moreover, they may be vulnerable to Janus and counterfeit frames posturing a benign message to them while containing malicious data for other receivers.

On Using Link Layer Rules for Defense. Some approaches use certain link layer rules for defense purposes. For example, researchers suggest identifying a message's sender by pushing it to the error passive state [58], which some defenses use to identify attackers [60]. Other papers suggest pushing attacker nodes to the bus-off state [57]. Due to the *non-compliance* capability (Sec. 4.1), none of these techniques could be used against ERAM attackers. Notably, CopyCAN [37] calculates ECUs' error counters by monitoring the link layer and reading error frames. Although it could limit some ERAM attacks that cause errors (e.g., frame hijacking), it could not detect attacks that do not or distinguish genuine errors.

5 ERACAN Design

5.1 Architecture and Operation Overview

ERACAN consists of a single node that connects to the bus and comprehensively monitors the data link and physical layers. It extracts specific features to detect and classify all ERAM attacks (Sec. 4.2). For certain attack types, ERACAN offers an attack *prevention* option to be enabled or disabled by the system administrator. To address performance challenges of such ubiquitous monitoring, ERACAN adopts a dual approach of *delegation* and *smart checking*. It delegates all link layer surveillance to a customized CAN controller (ERACAN

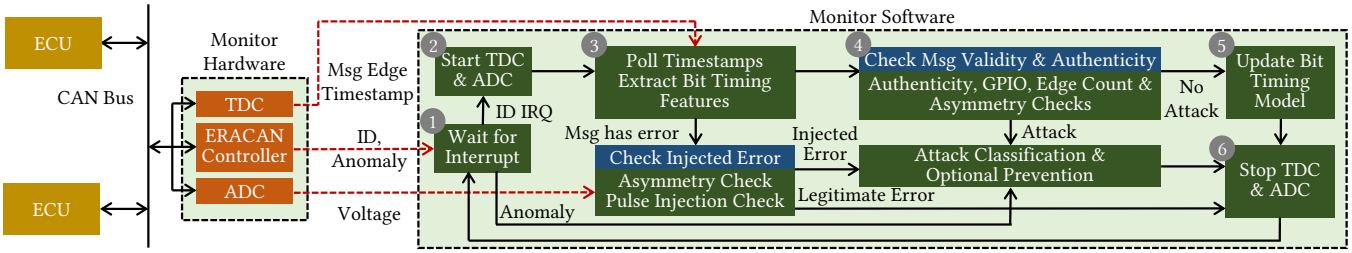


Figure 5: ERACAN monitor node architecture, workflow, and deployment on a CAN bus.

controller). For the physical layer, it uses a time-to-digital converter (TDC) to monitor time events, and an analog-to-digital converter (ADC) to monitor voltage levels. However, they are monitored using *smart checking* and only checked selectively based on attack scenarios. This greatly reduces the processing overhead. Below, we further describe these components and smart checking.

ERACAN Controller. This customized CAN controller can be configured to autonomously monitor certain link layer events. It stores information in registers and interrupts the software once events occur. The software then queries and clears them after each interrupt. Moreover, if attack prevention is enabled, ERACAN controller can inject errors to destroy malicious frames.

TDC. ERACAN uses this to measure the timestamps of 1⟶0 and 0⟶1 edges of the signal from the transceiver. They are used to extract and model two features: *bit-period* and *asymmetry*. We discuss their definition, extraction, and modeling in Sec. 5.2.

ADC. ERACAN uses this to measure the differential voltage levels of the bus, which it uses to check for attackers' error injections.

Smart Checking Workflow. Fig. 5 shows ERACAN's workflow. During operation, ERACAN controller continuously watches for suspicious link layer events and reads the ID of messages that appear on the bus. TDC and ADC measurements are started only when a message's ID field completes transmission. Initially, only TDC measurements are used to extract the frame's *bit period* and *asymmetry* in real-time. When its ACK field starts, ERACAN checks the message's *bit period* and *asymmetry* for validity and authenticity. If all checks pass, ERACAN updates a model for these features. However, if a message is interrupted with an error, ERACAN checks if the error is legitimate or injected by the attacker. ERACAN checks bit timing first, and if no anomalies are found, only then does it process and check ADC measurements. When an attack is detected, ERACAN performs attack classification, and prevention if it is optionally enabled. We explain these procedures in detail in this section.

5.2 Feature Extraction and Modeling Details

Bit Period: Equation 1 calculates an ECU's bit period T from the time $t_{\downarrow\downarrow}$ between consecutive 1⟶0 edges with $n_{\downarrow\downarrow}$ bits in between:

$$T = \frac{t_{\downarrow\downarrow}}{n_{\downarrow\downarrow}} \tag{1}$$

Asymmetry: Equation 2 computes an ECU's asymmetry A using the time $t_{\downarrow\uparrow}$ and the number of bits $n_{\downarrow\uparrow}$ between a 1⟶0 edge and the next 0⟶1 edge and its bit period T:

$$A = t_{\downarrow\uparrow} - n_{\downarrow\uparrow}T \tag{2}$$

Asymmetry is an ECU's unique physical fingerprint and depends on its transceiver switching characteristics [28, 56], signal reflection [36, 65], and load between the ECU and measuring unit [28, 56].

Modeling Legitimate Bit Timing. We compute bit period and asymmetry measurements using all edges between a message's ID and ACK fields. We model each ECU's expected bit period and asymmetry using normal distributions $N(\mu_T, \sigma_T^2)$ and $N(\mu_A, \sigma_A^2)$.

Bit Timing Model Recreation. Bit timing models are recreated when a CAN bus is turned on after a period of inactivity. ECUs send calibration messages and ERACAN uses them to compute their distribution parameters and bit period variance within a message.

Securing Model Recreation. ERACAN needs a cryptographic scheme that guarantees source authenticity and obfuscates the payload so attackers cannot predict it. Any scheme meeting these requirements can be used. In Appendix A, we explain an example lightweight scheme adopted from [57] for this step.

Bit Timing Model Online Updates. If a message contains no errors and fails no legitimacy checks, ERACAN uses it to perform online updates to account for feature drift due to environmental conditions. With each new measurement x, ERACAN updates distribution parameters using Equations 3 and 4:

$$\mu_n = w\mu_{n-1} + (1-w)x \tag{3}$$

$$\sigma_n^2 = w[\sigma_{n-1}^2 + (1-w)(x-\mu_{n-1})^2] \tag{4}$$

A smaller w gives new measurements a higher weight and helps the model adapt to changes faster. For large environmental variations or low-frequency messages whose bit timing can accumulate substantial changes between messages, w should be reduced.

Voltage Levels. Unlike sender identification approaches, ERACAN does not use voltage to achieve fine distinctions between ECUs, but only to distinguish between normal voltage levels and anomalies caused by attacks. It uses a single feature in two scenarios: voltage level variance to check for attack when a message contains errors (Sec. 5.3), and mean voltage level to confirm if an attack is launched by simultaneous transmission (Sec. 5.4). Since such distinctions are far more significant than differences among ECUs or under environmental variations, fixed detection thresholds enable reliable performance (Sec. 8). Thus, ERACAN measures voltage levels once in a secure setting (e.g., during manufacturing). It records the expected mean and variance of ECUs' stable dominant voltage levels and uses these to set fixed detection thresholds.

Link Layer Information. We configure ERACAN controller to report the following information: message IDs after their ID fields terminate, the edge count between ID and ACK fields, errors and

Algorithm 1 Message Authenticity Check

Input: Mean asymmetry in CRC field ($\bar{A}$), authorized sender (e)
Output: **true** if the message is authentic, **false** if not

$\mu_A \leftarrow \{\mu_{A1}, \mu_{A2}, ..., \mu_{An}\}$ ▷ Sorted list of ECUs' asymmetries
$\sigma_A \leftarrow \{\sigma_{A1}, \sigma_{A2}, ..., \sigma_{An}\}$ ▷ Created after model recreation
if $\bar{A} > \mu_{Ae} + 5\sigma_{Ae}$ **or** $\bar{A} < \mu_{Ae} - 5\sigma_{Ae}$ **then**
  return false
end if
$z_e \leftarrow |(\bar{A} - \mu_{Ae})/\sigma_{Ae}|$
$z_{e-1} \leftarrow |(\bar{A} - \mu_{Ae-1})/\sigma_{Ae-1}|$
$z_{e+1} \leftarrow |(\bar{A} - \mu_{Ae+1})/\sigma_{Ae+1}|$
return $z_e < z_{e-1}$ **and** $z_e < z_{e+1}$

their types, overload frames, discrepancies in sampled bits, and abnormal frame formats. These can be expanded or customized based on security requirements and identified attack vectors.

5.3 Legitimacy Checks

Authenticity Check. ERACAN computes the mean asymmetry of the CRC field to determine if a message is authentic using Algorithm 1. ERACAN maintains a sorted list of all ECUs' expected asymmetries. It first compares if a message's asymmetry deviates too much from the authorized sender. If not, it computes the distance to the authorized sender and two ECUs with the closest asymmetry. It then checks if the distance to the authorized sender is the closest. A message is authentic only if it passes both checks.

Optional Additional Authentication. For systems experiencing high environmental variations where ECUs' asymmetries may change abruptly, an optional check is added to eliminate any possible false positives. If a message fails Algorithm 1, ERACAN compares its asymmetry with the last message from the authorized sender and two ECUs with the closest asymmetry. It is deemed illegitimate if the difference with the authorized sender is not the smallest.

GPIO Check. This detects if a message is transmitted using GPIO, the most convenient and flexible link layer manipulation technique. These messages' bit timing is not derived from the ECU's oscillator but programmed by software. The variance of bit periods within the message multiplies due to greater uncertainty of software timing. ERACAN considers a message suspicious if its intra-message bit period variance is greater than the expected variance of its authorized sender by at least twice.

Asymmetry Check. This detects simultaneous transmission and pulse injection near edges in a message, which inevitably increases asymmetry (Sec. 6). ERACAN considers a message suspicious if any of its asymmetry measurements satisfies the following equation:

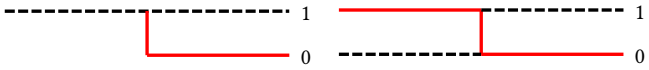
$$A > \mu_{Ae} + n\sigma_{Ae} \tag{5}$$

n is a configurable threshold. Increasing n to cover a larger part of the distribution increases both false negative and false positive rates. Its setting is optimal when false positive and false negative rates are equal and should be determined empirically for each ECU.

Edge Count Check. ERACAN checks if the TDC measures more timestamps than the expected edge count acquired by ERACAN controller, in case additional edges are injected by an attacker.

Pulse Injection Check Using Bit Timing. This checks if an error in a message is caused by an attacker's pulse injection. An attacker could inject a 0 when the victim is transmitting multiple 1s (Fig. 6a). Since the attacker's pulse is not transmitted based on the victim's bit period, its 1⟶0 edge is not aligned with the expected bit boundary. ERACAN checks all edges up to the start of an error frame for this anomaly (If the error is a CRC error, the start of the error frame is not checked because it is signaled by receivers). It calculates bit period T with all pairs of consecutive 1⟶0 edges using Equation 1. It considers the message suspicious if any T satisfies Equation 6, where n is the same threshold as asymmetry check:

$$|T - \mu_{TV}| > n\sigma_{TV} \tag{6}$$



(a) During consecutive 1s. (b) At a 0⟶1 transition.

Figure 6: Two possible locations of pulse injection.

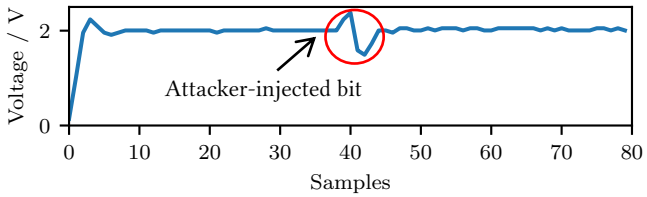


Figure 7: Voltage of pulse injection at a 0⟶1 transition.

Pulse Injection Check Using Voltage. If pulse injection check using bit timing does not find anomalies, ERACAN further checks voltage. This accounts for the attacker injecting a pulse at a victim's 0⟶1 transition (Fig. 6b). Due to imperfect bus termination and signal reflections [65], voltage level oscillates after the victim's falling edge. Such oscillations are superposed on the injected pulse, causing distortions in Fig. 7. They increase voltage level variance by orders of magnitude. Fig. 8 shows where ERACAN looks for abnormal voltage levels. After the attacker injects a 0, the sender transmits an error frame at the next bit. Depending on where the bit is injected, receivers transmit an error frame at the next bit or until they detect a stuff error after six consecutive 0s. Up to twelve consecutive 0s are observed [20]. The receivers transmit the last six bits, while voltage level anomalies are within the previous bits. Therefore, ERACAN finds bits before the last six in superposed error flags (highlighted in Fig. 8), extracts voltage samples in a 500ns window around every bit boundary, and computes the variance within each window. An anomaly is detected if the variance within any window is larger than the expected variance of the victim's stable dominant voltage levels by at least ten times.

Overload Frames Check. ERACAN controller signals to software if an overload frame is observed. On modern networks they must be transmitted by an attacker as modern CAN controllers do not initiate overload frames and only react to them [64]. For legacy networks where legitimate overload frames could arise, as the standard specifies at most two consecutive overload frames can be generated [20], ERACAN controller alerts if additional ones are observed.

Last EOF Bit Check. ERACAN controller signals to software if a message's last EOF bit is 0 but other fields are valid. The 0 could

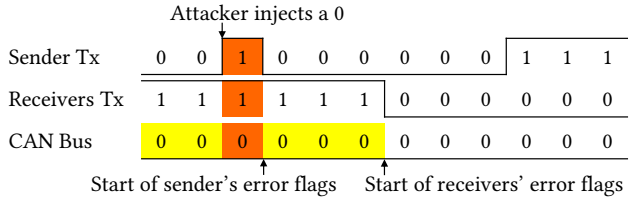


Figure 8: Error flags caused by error injection.

only be transmitted by an attacker because no CAN controllers should send a 0 in this field if all previous message fields are valid.

Sampled Bits Discrepancies Check. ERACAN controller has two sets of sampling logic with different sample points. Their sampled bits are compared in real-time. Any discrepancies indicate an attack and ERACAN controller signals to software. To ensure discrepancies in sampled bits between any nodes are visible, ERACAN controller's sample points could be set to the earliest (55.6%) and latest (90.9%) allowed sample points according to protocol specifications [20].

Frame Format Check. ERACAN controller signals to software if a message does not fully comply with the CAN standard. This is an unorthodox frame from an attacker. Depending on their CAN controller designs, legitimate ECUs could transmit certain kinds of unorthodox frames. Since ERACAN controller is implemented using an FPGA, check policies can be customized based on what frame formats are not expected to be transmitted normally.

CRC Errors Check. ERACAN controller signals to software if a message is properly acknowledged by receivers but a CRC error is signaled. This means the message is valid. The sender is operating correctly, and the CRC error could be injected by an attacker.

5.4 Attack Classification

If any check fails, ERACAN determines the attack type as follows.

Attacks Failing Controller Checks. *Freeze doom loop*, *double receive*, *Janus / counterfeit frame*, and *unorthodox frame* attacks are classified based on the respective controller check they fail.

Attacks Causing Errors. ERACAN distinguishes between *synchronization disruption*, *simultaneous transmission*, or *ERAM error injection*. Injecting pulses to disrupt synchronization causes additional edges. Simultaneous transmission increases voltage levels after the ID field. ERACAN uses edge count and voltage levels to distinguish them. Otherwise, ERAM error injection is the remaining possibility.

Attacks Failing GPIO Check. ERACAN distinguishes between *arbitration denial* and *bit timing poisoning*. Bit timing poisoning could also fail GPIO check if the attacker injects pulses near 1→ 0 edges, changes a message's bit period, and increases its variance. ERACAN confirms the attack is arbitration denial if none of the message's asymmetry measurements match the legitimate sender since the message is transmitted with another peripheral whose bit timing does not resemble its CAN controller. Otherwise, the attack is bit timing poisoning since asymmetry measurements not altered by the attacker still match the sender.

Attacks Failing Authenticity Check. If an attack fails authenticity but not GPIO check, ERACAN distinguishes between *masquerading attacks* and *frame hijacking*. ERACAN performs sender identification using asymmetry in the CRC field and the first asymmetry measurement of the message. It chooses the node with the closest expected asymmetry to the measurements as the sender. If the senders are different, the attack is frame hijacking since the first asymmetry measurement should match the legitimate sender but the CRC field matches the attacker. Otherwise, it is a masquerading attack as the entire message is sent by the attacker.

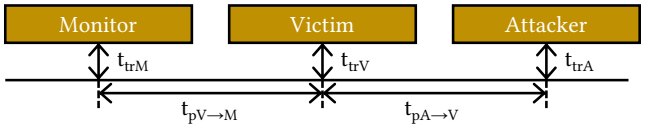


Figure 9: An example bus layout and propagation delays.

Attacks Failing Asymmetry or Edge Count Checks Only. These could only be *bit timing poisoning* using *pulse injection* or *simultaneous transmission*. ERACAN again leverages voltage levels to distinguish each technique.

5.4.1 Attack Prevention Options. By default ERACAN only detects and classifies attacks. Since ERACAN detects attacks with low false positive rates in real-time, it can translate detection into prevention for attacks compromising message integrity, including masquerading, frame hijacking, Janus, and counterfeit frame attacks. ERACAN offers prevention options for them that can be enabled per attack. If enabled, ERACAN destroys messages with error frames if relevant checks fail (authenticity check for masquerading / frame hijacking or sampled bits discrepancies check for Janus / counterfeit frames).

6 Security Analysis

Here we consider how ERACAN detects each ERAM attack, an attacker's potential evasion tactics, and ERACAN's mitigations.

Masquerading and Frame Hijacking. Both attacks require the attacker to transmit the entire CRC field and fail authenticity check using its asymmetry. If an attacker controls messages' bit timing to emulate a victim by transmitting with GPIO, he fails GPIO check even if he passes authenticity check.

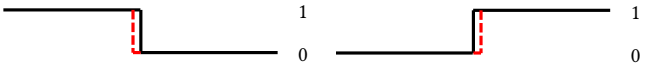
Arbitration Denial. First, authenticity check confines an attacker to launch the attack with his own ID. Then, since he must bypass the CAN controller, it is detected by GPIO check.

ERAM Error Injection. ERACAN controller CRC errors check detects injecting CRC errors after the ACK field. Pulse injection check detects injecting bit errors. A smart attacker could attempt to evade pulse injection check by accurately timing his injection to fall within the bound in Equation 6. The key difficulty is to accurately account for signal propagation delays through cables t_p and transceiver delays t_{tr} to translate the differential voltage into digital signals. We consider an example bus layout in Fig. 9. The victim starts transmission at t_0. The monitor and attacker each see the victim start transmission at t_1 (Equation 7) and t_2 (Equation 8). The attacker then delays Δt to inject a pulse at the m_{th} bit in the message and the pulse arrives at the monitor at t_3 (Equation 9):

$$t_1 = t_0 + t_{trV} + t_{pV\to M} + t_{trM} \tag{7}$$

$$t_2 = t_0 + t_{trV} + t_{pA\to V} + t_{trA} \tag{8}$$

$$t_3 = t_0 + t_{trV} + 2t_{pA\to V} + 2t_{trA} + t_{pV\to M} + t_{trM} + \Delta t \tag{9}$$



(a) Advance a 1→0 edge. (b) Delay a 0→1 edge.

Figure 10: Manipulating bit timing by injecting pulses.

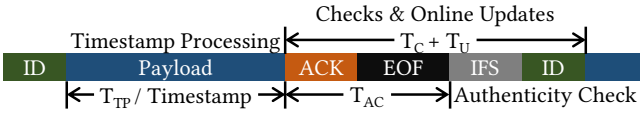


Figure 11: Processing by ERACAN in each message field.

Equation 10 derives the value of T the monitor uses for pulse injection check based on Equation 1. Substituting it into Equation 6, the attacker must satisfy Equation 11 to evade detection:

$$T = \frac{t_3 - t_1}{m} = \frac{2t_{pA\to V} + 2t_{trA} + \Delta t}{m} \tag{10}$$

$$\left|\frac{2t_{pA\to V} + 2t_{trA} + \Delta t}{m} - \mu_{TV}\right| \le n\sigma_{TV} \tag{11}$$

The attacker must accurately estimate $t_{pA\to V}$ and t_{trA} and adjust Δt. This is not possible without physical access and direct measurements. He could use the typical 5ns/m to estimate t_p [18] and obtain reference t_{tr} from the transceiver's datasheet, but these estimates are highly inaccurate due to environmental conditions and manufacturing variations. Thus, an attacker has minimal chances to evade detection and inject a single error. To inject more errors and change the victim's error state, the probability of consistently evading detection decreases exponentially.

Synchronization Disruption. An attacker must inject a pulse after the synchronization segment in a sender's recessive bit. Since the synchronization segment is at least 1/25 of a bit time [20], the attacker's pulse is at least 1/25 of a bit time after a bit boundary and can always be detected by pulse injection check using Equation 6.

Attacks Using Simultaneous Transmission. This technique is used to inject errors or poison bit timing. When two nodes transmit simultaneously, they do not see the end of the preceded message at the same time due to propagation delays [51]. They do not start transmission at exactly the same time. Their misaligned pulses overlap, increasing pulse width and asymmetry. This fails asymmetry check. The attacker could not evade detection since the increase in asymmetry depends on propagation delays outside his control.

Poisoning Attacks on Bit Timing. To manipulate bit timing measurements, an attacker could inject small pulses into the victim's message to introduce extra timestamps. This is detected by edge count check. Alternatively, he could inject pulses close to victim's edges to advance a 1→0 edge (Fig. 10a) or delay a 0→1 edge (Fig. 10b). According to Equation 2, both increase asymmetry and are detected by asymmetry check. Similar to error injection, evading asymmetry check requires accurate timing and has low success rates. Moreover, even if the attacker succeeds, he can only increase a single asymmetry measurement to at most $\mu_A + 5\sigma_A$. The change to μ_A modeled by ERACAN is bounded. To significantly change μ_A, the attacker must inject multiple pulses in the same message to poison a large portion of measurements. All of them need to evade detection, and the chance of success decreases exponentially.

Securing Model Recreation. A cryptographic scheme providing source authenticity and payload obfuscation is required to prevent spoofing and bit timing poisoning. Since attackers cannot predict the payload, they cannot poison bit timing by preparing the same message for simultaneous transmission or anticipating edge positions and injecting pulses. Moreover, we do not use the CRC fields and stuff bits for feature calculation since the attacker can predict them by reading the preceding content. A message is not used if it is retransmitted after an error. This is in case the attacker learns a message's content, injects an error, and then poisons bit timing in the retransmitted message. We choose the scheme in Appendix A as it is lightweight and meets both requirements. Any schemes meeting both requirements, such as [27, 49], can also be used.

Attacks Detected by Controller Checks. *Freeze doom loop*, *double receive*, *Janus / counterfeit frames*, and *unorthodox frames* are detected by ERACAN controller overload frames, last EOF bit, sampled bits discrepancies, and frame format checks respectively (Sec. 5.3).

7 Performance Analysis

7.1 Performance Deadlines

Fig. 11 shows ERACAN's mandatory processing steps for every message (steps 1 to 6 in Fig. 5). Other processing is only required after an error or attack. They have the following deadlines.

Timestamp Processing. After a message's ID field completes, ERACAN takes T_{TP} to retrieve and process each edge timestamps. This must finish before the next edge arrives, in the worst case within one bit time (2μs on 500kbps bus). As we show in Sec. 8.4, this can often finish well within the deadline and provide opportunities to leverage the idle time before the next edge for other processing.

Authenticity Check. After the ACK field starts, ERACAN performs authenticity check in T_{AC}. It must finish before the last EOF bit to enable attack prevention using error frames. The deadline is (2 bit ACK field + 6 bit EOF) = 8 bit time (16μs on 500kbps bus). ERACAN calculates a message's mean asymmetry in the CRC field iteratively using each new measurement during timestamp processing. This adds to T_{TP} but helps authenticity check meet its deadline because ERACAN only has to run Algorithm 1 after the ACK field.

Legitimacy Checks and Online Updates. Besides authenticity check, ERACAN also performs GPIO, edge count, and asymmetry checks, in a time totaling T_C. It then performs online updates in T_U. They must finish before the next message's ID field completes. In the worst case assuming 100% bus load and no bit stuffing, the deadline is (2 bit ACK field + 7 bit EOF + 3 bit IFS + 1 bit SOF + 11 bit ID) = 24 bit time (48μs on 500 kbps bus). ERACAN computes a running bit period variance for GPIO check and performs asymmetry check once a measurement is acquired. This amortizes the cost and leaves more time for the most time-consuming online updates.

7.2 Memory Overhead

ERACAN's arithmetic operations use 4-byte floating point numbers. For each ECU, ERACAN stores 5 model parameters: μ_A, σ_A, μ_T, σ_T, and intra-message bit period variance. For N ECUs, this requires ($20\times N$) bytes. ERACAN also needs to buffer a message's timestamps, bit period, and asymmetry measurements until they are used by

Table 1: Testbed setup for CRAM experiments.

Node	MCU	Transceiver	Distance To Monitor
ECU1	Arduino Due	TJA1051	40cm
ECU2	Arduino Due	TJA1051	60cm
ECU3	STM32F334	SN65HVD230	110cm
ECU4	STM32H755	TJA1051	170cm
ECU5	STM32F334	SN65HVD230	200cm

online updates. If bits alternate between 1 and 0 in the data and CRC fields of an 8-byte message, it contains 84 timestamps, and 84 measurements accordingly. No valid messages could contain more timestamps. To be conservative, ERACAN needs enough memory to buffer this amount of data ($84 \times 4 = 336$ bytes).

8 Evaluation

We implement ERACAN to evaluate its security and performance on a testbed and a real vehicle (2011 Chevy-Impala).

Implementation. We use a PYNQ-Z2 board as the monitor node. PYNQ-Z2 offers an SoC with FPGA fabric and a 650MHz dual-core ARM Cortex-A9 processor. It supports security features such as TrustZone, key storage, and secure boot. Following prior work [52], we implement the TDC in the FPGA fabric based on [5]. We design ERACAN controller by adding features to an open-source Verilog CAN controller design [42]. They interact with the ARM processor using interrupts and memory-mapped interfaces. To evaluate ERACAN's security, we collect TDC measurements, ERACAN controller information, and acquire voltage using a PicoScope 5244D oscilloscope at a sampling rate of 20MS/s and a resolution of 8 bit. We further implement bit timing features extraction, modeling, and checks on the ARM processor to evaluate its performance.

Choosing w. We collect messages, compute a moving average of asymmetry in the last 1 second, and exhaustively search between 0.9 and 1 for a w in Equation 3 that best approximates this value. w are 0.99975 and 0.99972 for the testbed and vehicle respectively.

Choosing Detection Threshold. To choose the threshold n in Equations 5 and 6, we run error injection attack and search for the value that yields equal false positive and negative rates. The chosen threshold is 4.8 and 4.5 respectively for the testbed and vehicle.

8.1 CRAM Security Evaluation on Testbed

We set up a 500kbps testbed with 5 ECUs configured according to Table 1 to test ERACAN against CRAM attacks in Table 2.

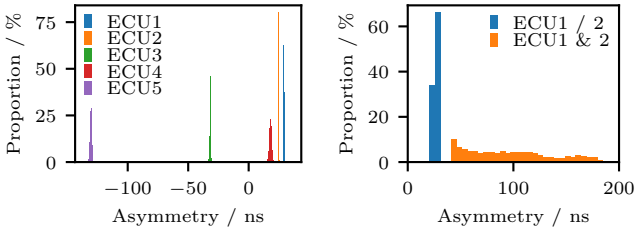
Sender Identification. We let each ECU transmit 8-byte messages with random data every 10ms. We collect 50000 messages and use the first 100 messages from each ECU for model creation. As shown in Fig. 12a, ECUs have distinct asymmetries. Therefore we identify senders of all messages with 100% accuracy.

Masquerading Attacks. We let each ECU transmit 10000 messages, 1/5 of which are under the ID of each ECU. Thus, bus traffic contains legitimate messages and masquerading attacks for every attacker-victim pair. We achieve 100% detection and no false positives.

Environmental Impacts. We reduce message periods to 100ms and repeat sender identification under temperature variations. We start at 25°C (77°F), heat the room to 28°C (82°F), cool it down to 18°C (64°F), and bring the temperature back to 25°C (77°F). We collect 550000 messages in three hours. As mentioned in Sec. 5.2, w should be reduced in this scenario. To evaluate its impact, we first test three values: 0.99975 (default w for other experiments), 0.994875, and 0.99 without enabling the optional additional authentication (Sec. 5.3). We get 7, 4, and 3 misclassifications respectively. Next, we enable the additional authentication. This eliminates all misclassification for all three w, reducing the false positive rate to 0%.

Table 2: Experiments results of CRAM attacks on the testbed.

Attack	Masquerading	Bus-Off	Bit Timing Poisoning
Detection Rate	100%	100%	100%
False Positive	0%	0.02%	0.02%



(a) In the absence of attacks. **(b) Under error handling attacks.**

Figure 12: Proportion of ECUs' asymmetry per range.

Error-Handling Attacks. We use one ECU to launch *bus-off* (Sec. 2.2.2) and *bit timing poisoning* (Sec. 6) attacks on another ECU using simultaneous transmission. We test all attacker-victim pairs. For each pair we collect 2500 messages from each ECU and 2500 messages under attack. We achieve 100% detection and a 0.02% false positive rate. Fig. 12b shows an example of ECU1 attacking ECU2. Asymmetries of all attacked messages are larger than normal ones from any single ECU by a significant margin.

8.2 ERAM Security Evaluation on Testbed

We build a 500kbps testbed with an attacker, a transmitter, and a listener. The attacker and transmitter both transmit normal messages every 10ms. The attacker launches ERAM attacks (Table 3) on some of the transmitter's messages using GPIO. For each attack, we record all bus traffic until we collect 2500 attacked messages.

Arbitration Denial. We compare the intra-message bit period variance of attacked messages from the attacker's GPIO to messages from its CAN controller, and observe at least a four-fold increase. With GPIO check, we achieve 100% detection and no false positives.

Detecting Error Injection Using Bit Timing. We test two cases. First, we estimate delays and accurately time the attacker's pulse injection according to Equation 11. Second, we do not estimate delays. We achieve 99.9% and 99.8% detection respectively and a 0.04% false positive rate. Estimating delays does not help the attacker evade detection since estimations are far from accurate.

Detecting Error Injection Using Voltage. We observe that attacks increase voltage level variance by at least 12 times. We achieve 100% detection and no false positives.

Table 3: Experiments results of ERAM attacks on the testbed.

Attack	Frame Hijacking	Error Injection	Arbitration Denial	Synch. Disruption	Janus Frame	Counterfeit Frame	Freeze D. Loop	Double Receive	Unorthodox Frame	Bit Timing Poisoning
Detection Rate	100%	99.8-100%	100%	100%	100%	100%	100%	100%	100%	99.9-100%
False Positive	0%	0-0.04%	0%	0.04%	0%	0%	0%	0%	0%	0.02%

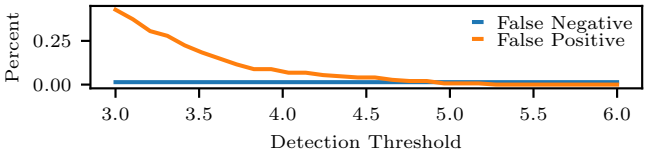


Figure 13: Influence of detection threshold.

Table 4: Attack prevention results.

Attack	Masquerading	Frame Hijacking	Janus Frame	Counterfeit Frame
Prevention	100%	100%	100%	100%

Synchronization Disruption. We set different bit segments for the transmitter and listener so they adjust bit timing differently when resynchronizing. We let the attacker inject pulses when the transmitter sends recessive bits and confirm the attack succeeds when the listener raises a CRC error. We achieve 100% detection and a 0.04% false positive rate.

Janus and Counterfeit Frames. We set the listen's sample point to 88.9%, and adjust the transmitter's sample point until attacks succeed reliably when set to 77.8%. The attacker transmits Janus frames read differently by the transmitter and listener, and launches counterfeit frame attacks on some of the transmitter's messages. Without setting its sample points to the extreme (Sec. 5.3), ERACAN controller already detects attacks with 100% accuracy and raises no false positives when its sample points are set to 65% and 90%.

Unorthodox Frames. We let the attacker transmit two types of unorthodox frames: frames with data fields longer than 8 bytes and remote frames with data. We implement their detection policies in ERACAN controller and achieve 100% detection with no false positives. For other kinds of unorthodox frames, ERACAN controller can be conveniently extended with respective detection policies.

Bit Timing Poisoning. We let the attacker corrupt the transmitter's bit timing with pulse injection. The detection rate is 99.9% if four measurements in a message are altered, and tends to 100% if more are altered. The false positive rate is 0.02%. In our test vehicle a message contains 18.6 measurements on average. Attackers can only poison a small part of a single message, and the chance to poison a large portion of ERACAN training set is negligible.

Frame Hijacking, Freeze Doom Loop, and Double Receive. All these attacks are detected 100% with no false positives.

Detection Threshold Impacts. To evaluate the impact of the detection threshold, we test thresholds from 3 to 6 for error injection. As shown in Fig. 13, its impact on false positives is far greater. Increasing the threshold from 3 to 5 increases the coverage of the normal bit timing distribution from 99.7% to 99.9999% and reduces false positives by orders of magnitude. Conversely, when the threshold increases, the allowed time window for attackers to inject a pulse without being detected (Equation 11) only increases by a few nanoseconds. Their chance of evading detection, and consequently the false negative rate, does not change significantly.

Attack Prevention. We let the attacker launch each attack in Table 4 2500 times. ERACAN detects and destroys all attacked messages with error frames, achieving 100% prevention.

Attack Classification. We launch seven attacks on a 5-ECU testbed: masquerading, error-handling attacks using simultaneous transmission, synchronization disruption, ERAM error injection, frame hijacking, arbitration denial, and bit timing poisoning with pulse injection. Other attacks are not tested because they can be easily classified using ERACAN controller checks. Each attack is launched 2500 times. We achieve a 99.8% overall classification accuracy. We misclassify attack types only in two scenarios. 0.3% of masquerading attacks are misclassified as frame hijacking. This happens when two nodes' asymmetries are close, and the first asymmetry measurement in the message matches a node other than the attacker due to its natural variations. 1.2% of arbitration denial attacks are misclassified as bit timing poisoning. This happens when some asymmetry measurements of the message happen to match the legitimate sender, although its chance is low, as our results show.

8.3 Security Evaluation on Real Vehicle

Our test vehicle has four ECUs on a 500kbps CAN bus. We connect ERACAN monitor node and an attacker node to its OBD-II port and perform experiments in Table 5.

Sender Identification. We collect ECUs' bit timing over four days with varying weathers and temperatures. We identify message senders with 100% accuracy. This translates to 100% detection for masquerading attacks by in-vehicle ECUs and no false positives.

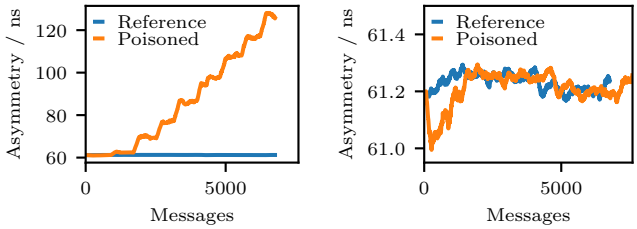
Frame Hijacking. We achieve 100% detection. CRC fields of attacked messages contain bit timing of the attacker. Although ERACAN has never modeled the attacker's bit timing before, it can still distinguish them from in-vehicle ECUs.

Detecting Error Injection Using Bit Timing. We achieve 99.7% detection. We observe that ECUs' bit periods have very small variations (maximum σ_T is 672ps). According to Equation 6, it is unrealistic for a remote attacker to evade detection since he must inject an error in a window shorter than 7ns.

Detecting Error Injection Using Voltage. We measure the variance of ECUs' stable dominant voltage levels on the first day and set a fixed detection threshold. We launch attacks on the following two days. We achieve 100% detection on both days using the fixed threshold despite changes in ECUs' voltage features across days.

Table 5: Real vehicle experiments results.

Experiment	Sender Identification	Frame Hijacking	Error Injection	Synch. Disruption	Freeze D. Loop	Double Receive
Accuracy	100%	100%	99.7-100%	100%	100%	100%



(a) Without asymmetry check. **(b) With asymmetry check.**

Figure 14: Asymmetry learned from online updates.

Bit Timing Poisoning. We let the attacker poison bit timing using pulse injection on one ID of an ECU. The attacker starts by corrupting a small portion of a message, then gradually increases the amount of corruption. We perform online updates on the IDs not poisoned by the attacker and use this as a reference for the victim's true bit timing. We then perform online updates using all messages including poisoned ones, with or without asymmetry check enabled, and compare the learned asymmetry with the reference. Without asymmetry check, the attacker successfully tricks ERACAN into gradually learning a larger asymmetry as in Fig. 14a. With asymmetry check the attack fails. The ECU's asymmetry learned by ERACAN closely resembles the reference as in Fig. 14b.

Double Receive, Freeze Doom Loop, and Synch. Disruption. We achieve 100% detection for all of these attacks using ERACAN controller checks or pulse injection check.

8.4 Performance Evaluation

To test ERACAN's feasibility and abilities to operate in real-time, we connect it to a testbed with 5 ECUs operating at 500kbps and profile its latency as it processes 20000 messages.

Timestamp Processing. ERACAN takes 0.4μs on average to retrieve one TDC timestamp and another 0.21μs to calculate bit period or asymmetry. This is well within the 2μs deadline. Therefore, ERACAN uses the remaining time for other processing, such as asymmetry checks. Adding them brings the processing time to 0.72μs on average and 1.34μs in the worst case, still within the deadline.

Authenticity Check. For 5 ECUs, authenticity check takes 0.26μs on average and 0.81μs in the worst case. We experiment with up to 10 ECUs and show their average latency in Table. 6. The latency is almost constant and it only takes 0.27μs with 10 ECUs. Deadlines are always met despite the different number of ECUs on the testbed.

GPIO and Edge Count Checks. *GPIO check* takes 33.8ns on average and 70.8ns in the worst case. *Edge count check* takes 0.20μs on average and 0.22μs in the worst case.

Online Updates. It takes 0.23μs on average and 0.78μs at most to process one asymmetry and bit period measurement. If every update takes the worst-case 0.78μs and the number of measurements in messages is at maximum (Sec. 7.2), online updates take 32.8μs. This most pessimistic estimate is still within the 48μs deadline.

Table 6: ERACAN average authenticity check latency.

N_{ECU}	5	6	7	8	9	10
Latency / μs	0.263	0.250	0.250	0.256	0.265	0.265

Table 7: ERACAN operation latencies.

Processing Stage	Average	Worst-Case	Deadline
Timestamp Processing	0.72μs	1.34μs	2μs
Authenticity Check	0.26μs	0.81μs	16μs
Checks & Online Updates	4.77μs	21.4μs	48μs

Real-Time Capability. We calculate the average and worst-case latencies of each processing stage and compare them with the deadlines in Table 7. To calculate online updates latency, we use our profiling results and the average and maximum number of measurements per message from our test vehicle (18.6 and 26). All operations meet deadlines. Therefore, ERACAN can operate on a 500kbps CAN bus loaded up to 100% and guarantee all detected masquerading and frame hijacking attacks can be prevented.

Memory Footprint. We measure the code and data size of our implementation and find them to be 16.7kB and 1.44kB, respectively. As many recent commercial automotive-grade FPGAs offer at least 256kB on-chip memory [3], this overhead is reasonable.

9 Benchmark Comparison

ERACAN is designed for ERAM attacks which other defenses do not protect from, so it is hard to compare its performance with other defenses on the same attack set. Instead, we compare ERACAN with separate defense categories, followed by its performance on CRAM attacks that other systems also defend against.

Compared to Cryptography and Secret Delay IDS. As shown in Table 8, these approaches guarantee message authenticity under CRAM. They do not protect against error-handling attacks except for ZBCAN [57]. Under ERAM, all secret delay approaches and some cryptographic approaches lose their security guarantees (Sec. 4.3). Furthermore, they offer no security against the wide range of new ERAM attacks (Sec. 4.2). ERACAN detects both CRAM attacks as well as all ERAM attacks. It is the first to offer attack classification, enabling intrusion responses to build on its output. Furthermore, ERACAN does not increase busload or reschedule bus traffic, which is required for some of these approaches.

Compared to Physical Signal IDS. As shown in Table 8, they only detect CRAM masquerading attacks but not error-handling attacks, except for VoltageIDS [13]. However, some are evadable

Table 8: How ERACAN compares with other defense systems.

Defense Approach	Attacks					Features		Cost	
	Spoofing		Error Handling		ERAM	Single-Msg	Classify	Increase	Modify
	CRAM	ERAM	CRAM	ERAM	Attacks	Detection	Attacks	Busload	Traffic
Cryptography [25, 35, 49, 50]	●	◐	○	○	○	✓	X	✓	X
Secret Delay IDS [57, 71]	●	○	◐	○	○	✓	X	X	✓
Physical Signal IDS [13, 32, 52, 56]	●	◐	◐	○	○	✓	X	X	X
Interval / Clk-Skew IDS [11, 53, 72]	●	○	○	○	○	X	X	X	X
Payload Inspection IDS [1, 38, 67]	●	○	○	○	○	X	X	X	X
ERACAN	●	●	●	●	●	✓	✓	X	X

Table 9: Benchmarking ERACAN authenticity check latency.

EASI [32]	**ASSASSIN** [56]	**SPARTA** [55]	**ERACAN**
$1.02\mu s^* N_{ECU}$	$0.94\mu s^* N_{ECU}$	$1.15\mu s$	$0.26\mu s$

Table 10: Performance comparison on masquerading attacks.

Defense System	Detection	Prevention	Deployment Cost
EASI [32]	99.66%	-	1 Monitor Node
ASSASSIN [56]	99.02%	-	1 Monitor Node
EdgeTDC [52]	100%	-	1 Monitor Node + Double Cable Length
ZBCAN [57]	98.5%	98.5%	1 Monitor Node + Reschedule Bus Traffic
ERACAN	99.99%+	99.99%+	1 Monitor Node

under ERAM and they do not detect any new ERAM attacks. ERACAN offers stronger security by detecting all CRAM and ERAM attacks.

Compared to Interval and Payload Inspection IDS. As Table 8 shows, these approaches only detect CRAM message injection but can be evaded by ERAM tactics. ERACAN protects against the full range of CRAM and ERAM attacks. Furthermore, ERACAN offers single message detection to ensure no low-level attacks can pass unnoticed, which some of these defenses do not offer.

Operation Latency. Table 9 compares ERACAN's average message authenticity check latency with three recent lightweight defenses, assuming all systems use a 650MHz processor. For a fair comparison, we exclude the time to take measurements and extract features. The remaining processing steps are performed by the CPU and their latency does not depend on the measuring equipment used. ERACAN's latency is the lowest. Compared to EASI and ASASSIN, ERACAN's latency is independent of the number of ECUs and allows it to scale to a CAN bus with more ECUs.

Masquerading Attacks. Table 10 compares ERACAN's performance on masquerading attacks with four latest defenses offering single message detection. ERACAN's detection rate beats all systems except for EdgeTDC, whose high performance comes at the cost of significant hardware changes by doubling the cable length [52]. ERACAN offers guaranteed prevention of all detected masquerading attacks, which only ZBCAN offers [57]. Unlike ZBCAN, ERACAN achieves this without rescheduling bus traffic.

10 Discussion and Limitations

Corrupted Payloads. ERACAN mainly aims to fill the research gap in emerging ERAM attacks. Detecting an ECU corrupting the payloads of its own messages, an old CRAM tactic, requires payload inspection. This is orthogonal to ERACAN's goal. Good solutions already exist [1, 38, 67] and can be integrated. ERACAN offers additional security as it cannot be deceived by Janus frames that look benign to the monitor but malicious to other nodes (Sec. 4.3).

False Positive Consequences. The only two cases where ERACAN has non-zero false positive rates are with error-handling (e.g., error injection) and bit timing poisoning attacks. Specifically, for error-handling attacks, there is a 0-0.04% chance ERACAN treats a genuine error as an attack. These misclassifications only relate to whether errors (uncommon on a healthy bus) are malicious and do not affect normal messages. We can build on CopyCAN's [37] idea of reading error frames and tracking ECUs' error counters to further reduce false alarms. Namely, ERACAN can alert only when an ECU's error counter reaches a threshold and a large portion of its past errors are suspicious, instead of for single errors, because it is much less likely that multiple genuine errors are all classified as malicious. Finally, ERACAN does not take any intrusive actions against error-handling attacks. As such, the consequences of misclassifications are limited to raising an alarm. For bit timing poisoning attacks, the false positive rate is 0.02%. The only aim of this attack is to poison the online updates process, not to falsify data or impersonate other ECUs. Consequently, ERACAN does not destroy these messages and only excludes them from online updates.

Possible Extensions. In Sec. 5.3, authenticity check only uses the mean asymmetry of the CRC field. We can extend it with other statistical measures to slightly improve its accuracy. We can calculate both a message's mean asymmetry and variance and use a t-test to assess if its distribution is the same as its authorized sender. This could be more reliable as it accounts for the variability of samples and remains robust if ECUs' bit timing is not normally distributed. However, it incurs more processing time. Similarly, ERACAN could be extended to monitor several buses operating at different security levels as was proposed for other buses [15, 16].

Non-GPIO Peripherals. In our evaluations, we focus mainly on the GPIO as it is the most versatile and convenient peripheral. Although ERAM attacks can be launched with other peripherals, ERACAN will maintain high detection performance since most of

its checks do not depend on the technique to launch attacks. Its performance could weaken only in exceptional cases of *arbitration denial* attacks. Specifically, if the attacker could manage to find a peripheral whose bit timing closely resembles the CAN controller, abide by both the peripheral's and CAN's valid formats, and be originally authorized to transmit a high-priority ID. Even in this case, since arbitration denial with a single message only delays the victim's message by its duration, the attacker must launch the attack continuously to maximize the chance that the message misses its deadline or prevent it from gaining bus access. ERACAN can be extended to monitor message frequency to detect such scenarios.

Safeguarding the Monitor. Like most IDSs, we assume trust in a central monitor. Nonetheless, we took measures to minimize the risk of its failure or compromise. First, we connected it in parallel, not as a pass-through gateway. Thus, in the event of failure, it fails safe and bus communication continues. Second, to minimize the risk of compromise, we used hardware with security features: secure boot and secure key storage allow crucial assets (e.g., program image, bitstream files for FPGA configuration) to be signed and encrypted, preventing tampering. Finally, except for its connection to the CAN bus, the monitor is air-gapped with no other entry points.

Bypassing Physical Layer Rules. In theory, remote attackers could bypass physical layer rules by controlling more than 8 ECUs [41]. This assumption is unrealistic and not considered by ERACAN. To account for this, ERACAN could easily integrate existing solutions to detect physical layer manipulations by flipping 0 to 1 [53].

11 Conclusions

In this paper, we aimed to bridge a critical gap in CAN security research: the escalating threat of remote attackers gaining extensive link layer control (ERAM model). We introduced ERACAN, the first comprehensive defense system tailored explicitly to counter this attacker model in addition to the conventional model (CRAM), offering detection, classification, and prevention abilities against both models. We started with a security analysis of the ERAM model, focusing on its capabilities, attacks enabled, and impacts on conventional defenses. We then designed ERACAN to monitor essential link and physical layer features for securing against all ERAM attacks. ERACAN addresses complex performance and reliability challenges posed by such meticulous monitoring by delegating link layer surveillance to an autonomous ERACAN controller and employing innovative smart-checking to leverage physical signals efficiently. We analyzed ERACAN's security against various ERAM attacks and evasion tactics. Finally, we validated ERACAN's feasibility, security, performance, and real-time capabilities by evaluating it on a testbed and a real vehicle's CAN bus.

Acknowledgments

We thank the anonymous reviewers for their invaluable feedback and our shepherd for the guidance in the revision process. This work was supported in part by the National Science Foundation (NSF) under the Secure and Trustworthy Cyberspace (SaTC) program and Grant CNS-2144645, as well as the Office of Naval Research (ONR) under Grants N00014-22-1-2671 and N00014-18-1-2674. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of our sponsors.

References

[1] Natasha Alkhatib, Lina Achaji, Maria Mushtaq, Hadi Ghauch, and Jean-Luc Danger. 2023. WIP: AMICA: Attention-based Multi-Identifier model for asynchronous intrusion detection on Controller Area networks. In *Symposium on Vehicles Security and Privacy (VehicleSec)*.
[2] Khaled Serag Alsharif. 2023. PROACTIVE VULNERABILITY IDENTIFICATION AND DEFENSE CONSTRUCTION – THE CASE FOR CAN. (2023).
[3] AMD. 2024. XA Automotive Product Selection Guide. https://docs.amd.com/v/u/en-US/xa-portfolio-product-selection-guide.
[4] Rohit Bhatia, Vireshwar Kumar, Khaled Serag, Z Berkay Celik, Mathias Payer, and Dongyan Xu. 2021. Evading Voltage-Based Intrusion Detection on Automotive CAN.. In *Network and Distributed System Security Symposium (NDSS)*.
[5] Benjamin Blase. 2015. tdc-fpga: Time to digital converter for use on a Xilinx 7-series FPGA. https://github.com/benbr8/tdc-fpga.
[6] Tim Brom. 2018. CANT. https://github.com/bitbane/CANT.
[7] Paolo Cerracchio, Stefano Longari, Michele Carminati, and Stefano Zanero. 2024. Investigating the Impact of Evasion Attacks Against Automotive Intrusion Detection Systems. In *Symposium on Vehicles Security and Privacy (VehicleSec)*.
[8] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security Symposium*.
[9] Kyong-Tak Cho and Kang G. Shin. 2016. Error Handling of In-vehicle Networks Makes Them Vulnerable. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[10] Kyong-Tak Cho and Kang G. Shin. 2017. Viden: Attacker Identification on In-Vehicle Networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[11] Kyong-Tak Cho and Kang G. Shin. 2016. Fingerprinting electronic control units for vehicle intrusion detection. In *USENIX Security Symposium*.
[12] Wonsuk Choi, Hyo Jin Jo, Samuel Woo, Ji Young Chun, Jooyoung Park, and Dong Hoon Lee. 2018. Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks. *IEEE Transactions on Vehicular Technology* (2018).
[13] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. 2018. VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Transactions on Information Forensics and Security* (2018).
[14] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. 2022. CANflict: Exploiting Peripheral Conflicts for Data-Link Layer Attacks on Automotive Networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[15] Josh D Eckhardt, Thomas E Donofrio, and Khaled Serag. 2019. System and method of monitoring data traffic on a MIL-STD-1553 data bus. US Patent 10,467,174.
[16] Josh D Eckhardt, Thomas E Donofrio, and Khaled Serag. 2020. Multiple security level monitor for monitoring a plurality of MIL-STD-1553 buses with multiple independent levels of security. US Patent 10,685,125.
[17] Bernd Elend and Tony Adamson. 2017. Cyber security enhancing CAN transceivers. In *International CAN Conference*.
[18] International Organization for Standardization (ISO). 2016. *Road Vehicles — Controller area network (CAN)*. Part 2: Highspeed medium access unit.
[19] Mahsa Foruhandeh, Yanmao Man, Ryan M. Gerdes, Ming Li, and Thidapat Chantem. 2019. SIMPLE: single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks. In *Annual Computer Security Applications Conference (ACSAC)*.
[20] Robert Bosch GmbH. 1991. CAN Specification. (1991).
[21] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. 2012. Libra-can: a lightweight broadcast authentication protocol for controller area networks. In *International Conference on Cryptology and Network Security*.
[22] Bogdan Groza, Lucian Popa, and Pal-Stefan Murvay. 2018. INCANTA - INtrusion Detection in Controller Area Networks with Time-Covert Authentication. In *Security and Safety Interplay of Intelligent Software Systems*.
[23] Bogdan Groza, Lucian Popa, and Pal-Stefan Murvay. 2021. CANTO - Covert AutheNtication With Timing Channels Over Optimized Traffic Flows for CAN. *IEEE Transactions on Information Forensics and Security* (2021).
[24] Bogdan Groza, Lucian Popa, Pal-Stefan Murvay, Yuval Elovici, and Asaf Shabtai. 2021. CANARY - a reactive defense mechanism for Controller Area Networks based on Active RelaYs. In *USENIX Security Symposium*.
[25] Kyusuk Han, André Weimerskirch, and Kang G. Shin. 2015. A practical solution to achieve real-time performance in the automotive network by randomizing frame identifier. In *ESCAR Europe*.
[26] Oliver Hartkopp and R Schilling. 2012. Message authenticated CAN (MaCAN). In *ESCAR*.
[27] Ahmed Hazem and HA Fahmy. 2012. Lcap-a lightweight can authentication protocol for securing in-vehicle networks. In *EASCAR*.
[28] Magnus-Maria Hell. 2015. The physical layer in the CAN FD world-The update. In *International CAN Conference*.

[29] Abdulmalik Humayed, Fengjun Li, Jingqiang Lin, and Bo Luo. 2020. CANSentry: Securing CAN-Based Cyber-Physical Systems against Denial and Spoofing Attacks. In *European Symposium on Research in Computer Security (ESORICS)*.
[30] Sungwoo Kim, Gisu Yeo, Taegyu Kim, Junghwan "John" Rhee, Yuseok Jeon, Antonio Bianchi, Dongyan Xu, and Dave (Jing) Tian. 2022. ShadowAuth: Backward-Compatible Automatic CAN Authentication for Legacy ECUs. In *ACM ASIA Conference on Computer and Communications Security*.
[31] Marcel Kneib and Christopher Huth. 2018. Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[32] Marcel Kneib, Oleg Schell, and Christopher Huth. 2020. EASI: Edge-Based Sender Identification on Resource-Constrained Platforms for Automotive Networks. In *Network and Distributed System Security Symposium (NDSS)*.
[33] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. 2010. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy (S&P)*.
[34] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. 2021. CANNON: Reliable and Stealthy Remote Shutdown Attacks via Unaltered Automotive Microcontrollers. In *IEEE Symposium on Security and Privacy (S&P)*.
[35] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. 2014. CaCAN-centralized authentication system in CAN (controller area network). In *ESCAR*.
[36] Hansang Lim, Gyunha Kim, Seungsu Kim, and Dongok Kim. 2019. Quantitative analysis of ringing in a controller area network with flexible data rate for reliable physical layer designs. *IEEE Transactions on Vehicular Technology* (2019).
[37] Stefano Longari, Matteo Penco, Michele Carminati, and Stefano Zanero. 2019. CopyCAN: An Error-Handling Protocol based Intrusion Detection System for Controller Area Network. In *ACM Workshop on Cyber-Physical Systems Security & Privacy*.
[38] Stefano Longari, Carlo Alberto Pozzoli, Alessandro Nichelini, Michele Carminati, and Stefano Zanero. 2023. Candito: improving payload-based detection of attacks on controller area networks. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*.
[39] Charlie. Miller and Chris. Valasek. 2013. Adventures in automotive networks and control units. *Def Con* (2013).
[40] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* (2015).
[41] Abdullah Zubair Mohammed, Yanmao Man, Ryan Gerdes, Ming Li, and Z Berkay Celik. 2022. Physical layer data manipulation attacks on the can bus. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.
[42] Igor Mohor. 2017. CAN Protocol Controller. https://opencores.org/projects/can.
[43] Pal-Stefan Murvay and Bogdan Groza. 2017. DoS Attacks on Controller Area Networks by Fault Injections from the Software Layer. In *International Conference on Availability, Reliability and Security (ARES)*.
[44] Pal-Stefan Murvay and Bogdan Groza. 2020. TIDAL-CAN: Differential Timing Based Intrusion Detection and Localization for Controller Area Network. *IEEE Access* (2020).
[45] Sen Nie, Ling Liu, and Yuefeng Du. 2017. Free-fall: Hacking tesla from wireless to can bus. *Black Hat USA* (2017).
[46] Sen Nie, Ling Liu, Yuefeng Du, and Wenkai Zhang. 2018. Over-the-air: How we remotely compromised the gateway, BCM, and autopilot ECUs of Tesla cars. *Black Hat USA* (2018).
[47] Shuji Ohira, Araya Kibrom Desta, Ismail Arai, and Kazutoshi Fujikawa. 2021. PLI-TDC: Super fine delay-time based physical-layer identification with time-to-digital converter for in-vehicle networks. In *ACM Asia Conference on Computer and Communications Security*.
[48] Andrea Palanca, Eric Evenchick, Federico Maggi, and Stefano Zanero. 2017. A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks. In *Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*.
[49] Mert D Pesé, Jay W Schauer, Junhui Li, and Kang G. Shin. 2021. S2-CAN: Sufficiently Secure Controller Area Network. In *Annual Computer Security Applications Conference (ACSAC)*.
[50] Andreea-Ina Radu and Flavio D Garcia. 2016. LeiA: A lightweight authentication protocol for CAN. In *European Symposium on Research in Computer Security (ESORICS)*.
[51] Stuart Robb and East Kilbride. 1999. CAN bit timing requirements. *Motorola Semiconductor Application Note, AN1798* (1999).
[52] Marc Roeschlin, Giovanni Camurati, Pascal Brunner, Mridula Singh, and Srdjan Capkun. 2023. EdgeTDC: On the Security of Time Difference of Arrival Measurements in CAN Bus Systems. In *Network and Distributed System Security Symposium (NDSS)*.
[53] Matthew Rogers, Phillip Weigand, Jassim Happa, and Kasper Rasmussen. 2023. Detecting CAN Attacks on J1939 and NMEA 2000 Networks. *IEEE Transactions on Dependable and Secure Computing* (2023).
[54] Sang Uk Sagong, Xuhang Ying, Andrew Clark, Linda Bushnell, and Radha Poovendran. 2018. Cloaking the clock: Emulating clock skew in controller area networks. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*.
[55] Oleg Schell and Marcel Kneib. 2023. SPARTA: Signal Propagation-based Attack Recognition and Threat Avoidance for Automotive Networks. In *ACM Asia Conference on Computer and Communications Security*.
[56] Oleg Schell, Claudio Oechsler, and Marcel Kneib. 2022. Asymmetric Symbol and Skew Sender Identification for Automotive Networks. *IEEE Transactions on Information Forensics and Security* (2022).
[57] Khaled Serag, Rohit Bhatia, Akram Faqih, Muslum Ozgur Ozmen, Vireshwar Kumar, Z. Berkay Celik, and Dongyan Xu. 2023. ZBCAN: A Zero-Byte CAN Defense System. In *USENIX Security Symposium*.
[58] Khaled Serag, Rohit Bhatia, Vireshwar Kumar, Z. Berkay Celik, and Dongyan Xu. 2021. Exposing New Vulnerabilities of Error Handling Mechanism in CAN. In *USENIX Security Symposium*.
[59] Khaled Serag, Vireshwar Kumar, Z Berkay Celik, Rohit Bhatia, Mathias Payer, and Dongyan Xu. 2022. Attacks on can error handling mechanism. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.
[60] Jiwoo Shin, Hyunghoon Kim, Seyoung Lee, Wonsuk Choi, Dong Hoon Lee, and Hyo Jin Jo. 2023. RIDAS: Real-time identification of attack sources on controller area networks. In *USENIX Security Symposium*.
[61] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. 2016. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In *International Conference on Information Networking (ICOIN)*.
[62] Ken Tindell. 2020. *CAN Bus Security: Attacks on CAN bus and their mitigations*. Technical Report. Canis Automotive Labs.
[63] Ken Tindell. 2020. CANHack. https://github.com/kentindell/canhack.
[64] Ken Tindell. 2020. Three new CAN protocol hacks. https://kentindell.github.io/2020/01/20/new-can-hacks/.
[65] Ken Tindell. 2022. Running high speed signals through CAN bus wiring. https://kentindell.github.io/2022/11/15/canbus-wiring/.
[66] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. 2011. CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography*.
[67] Armin Wasicek, Mert D Pesé, André Weimerskirch, Yelizaveta Burakova, and Karan Singh. 2017. Context-aware intrusion detection in automotive control systems. In *ESCAR USA*.
[68] Haohuang Wen, Qi Alfred Chen, and Zhiqiang Lin. 2020. Plug-N-Pwned: Comprehensive Vulnerability Analysis of OBD-II Dongles as A New Over-the-Air Attack Surface in Automotive IoT. In *USENIX Security Symposium*.
[69] Marko Wolf, André Weimerskirch, and Christof Paar. 2004. Security in automotive bus systems. In *Workshop on Embedded Security in Cars*.
[70] Samuel Woo, Daesung Moon, Taek-Young Youn, Yousik Lee, and Yongeun Kim. 2019. CAN ID shuffling technique (CIST): Moving target defense strategy for protecting in-vehicle CAN. *IEEE Access* (2019).
[71] Xuhang Ying, Giuseppe Bernieri, Mauro Conti, and Radha Poovendran. 2019. TACAN: Transmitter authentication through covert channels in controller area networks. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*.
[72] Clinton Young, Habeeb Olufowobi, Gedare Bloom, and Joseph Zambreno. 2019. Automotive intrusion detection based on constant can message frequencies across vehicle driving modes. In *ACM Workshop on Automotive Cybersecurity*.
[73] Li Yue, Zheming Li, Tingting Yin, and Chao Zhang. 2021. Cancloak: Deceiving two ecus with one frame. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.
[74] Jia Zhou, Prachi Joshi, Haibo Zeng, and Renfa Li. 2019. Btmonitor: Bit-time-based intrusion detection and attacker identification in controller area network. *ACM Transactions on Embedded Computing Systems (TECS)* (2019).

A Secure Model Recreation Details

Each ECU has a secret key pre-shared only with the monitor. Details on establishing the key are outside the scope of this paper. Using these keys, each ECU securely generates and exchanges a random seed with the monitor when model recreation starts. Using the seed, pre-shared key, and an agreed-upon pseudo-random function (PRF), the ECU and monitor generate a session key. Next, using the seed, session key, and PRF, they generate the same random sequence. For each calibration message, the ECU uses the next 64 bits of the sequence as its data field. The monitor compares the data field to the next 64 bits in the ECU's sequence to check a message's authenticity, before using it to create bit timing model.