POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Physical-Layer Intrusion Detection for the CAN Bus: A Voltage-Based Approach Against Transport-Layer Attackers

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria Informatica

Author: **Pierantonio Mauro**

Student ID: 258737
Advisor: Prof. Stefano Longari
Academic Year: 2025-26

Abstract

The Controller Area Network (CAN) bus is the backbone of in-vehicle communication in modern automobiles, yet it was designed with no security mechanisms: no authentication, no encryption, and no access control. Any node on the bus can read every frame and inject frames with any identifier. This design weakness, acceptable when vehicles were closed systems, has become a significant vulnerability as cars gain wireless connectivity and expose standardized diagnostic interfaces to the outside world.

Intrusion Detection Systems (IDSs) have been proposed to address CAN bus threats. Payload-based, timing-based, and physical-layer fingerprinting approaches have demonstrated effectiveness against conventional attackers that operate through the intact CAN controller (the Conventional Remote Attacker Model, CRAM). However, recent work has demonstrated that a class of software-only techniques – peripheral clock gating (CAN-non) and pin conflict exploitation (CANflict) – allows a compromised ECU to bypass the CAN controller entirely and manipulate individual bus bits in real time. This Enhanced Remote Attacker Model (ERAM) undermines existing defences, including physical-layer IDSs, through fingerprint corruption, partial-frame attacks, and short-pulse injections.

This thesis investigates whether a physical-layer IDS can maintain effectiveness against an ERAM attacker. The key insight is that an ERAM attacker, while gaining full data-link layer control, still uses the victim ECU's physical CAN transceiver, whose analog output characteristics cannot be altered through software alone. A lightweight IDS is proposed that monitors the CAN bus differential voltage ($V_{\text{CAN-H}} - V_{\text{CAN-L}}$) in real time, deriving a detection threshold ($\mu + 3\sigma$) from a baseline capture of clean traffic and correlating two voltage-anomaly rules with CAN error flags detected in real time by a protocol-aware frame counter operating at 1 MSa/s.

The system was evaluated on a laboratory test bed against an ERAM-class bus-off attack implemented via the CANflict technique. A baseline session of 126.1 seconds and over 65 million samples produced zero false positives and zero rule fires. Two dedicated bus-off attack sessions achieved 100% error-flag detection rates (16/16 and 19/19 error flags confirmed by preceding voltage anomalies). A sustained attack session of 127.4 seconds with

575 error flags achieved an overall detection rate of 89.6% (515/575 confirmed). These results demonstrate the feasibility of error-flag-correlated voltage monitoring as a first step toward ERAM-aware physical-layer intrusion detection.

Keywords: CAN bus, Intrusion Detection System, physical-layer security, voltage monitoring, error-flag correlation, automotive security

Abstract in lingua italiana

Il Controller Area Network (CAN) bus costituisce il cuore della comunicazione interna dei veicoli moderni, ma è stato progettato senza alcun meccanismo di sicurezza: nessuna autenticazione, nessuna cifratura e nessun controllo degli accessi. Qualsiasi nodo sul bus può leggere ogni frame e iniettarne di nuovi con qualsiasi id. Questa debolezza progettuale, accettabile quando i veicoli erano sistemi chiusi, è diventata una vulnerabilità significativa con la diffusione della connettività wireless e l'esposizione di interfacce diagnostiche standardizzate verso l'esterno.

Per contrastare le minacce al bus CAN sono stati proposti sistemi di rilevamento delle intrusioni (Intrusion detection Systems, IDS). Gli approcci basati sul payload, sulla temporizzazione e sul fingerprinting a livello fisico hanno dimostrato efficacia contro gli attaccanti convenzionali, che operano attraverso il controller CAN intatto (Conventional Remote Attacker Model, CRAM). Tuttavia, ricerche recenti hanno dimostrato che una classe di tecniche puramente software — il blocco del clock periferico (CANnon) e lo sfruttamento di conflitti tra pin (CANflict) — consente a un ECU compromessa di aggirare completamente il controller CAN e manipolare i singoli bit sul bus in tempo reale. Questo Enhanced Remote Attacker Model (ERAM) mina le difese esistenti, inclusi gli IDS a livello fisico, attraverso la corruzione delle fingerprint, attacchi su frame parziali e l'iniezione di impulsi.

Questa tesi indaga se un IDS a livello fisico possa mantenere la propria efficacia contro un attaccante ERAM. L'intuizione chiave è che un attaccante ERAM, pur acquisendo pieno controllo del livello data-link, utilizza comunque il transceiver CAN fisico dell'ECU vittima, le cui caratteristiche di uscita analogica non possono essere modificate via software. Si propone un IDS leggero che monitora in tempo reale la tensione differenziale sul bus CAN ($V_{\text{CAN-H}} - V_{\text{CAN-L}}$), derivando una soglia di rilevamento ($\mu + 3\sigma$) da una sessione di acquisizione di traffico pulito, e correlando due regole di anomalia di tensione con gli error flag CAN rilevati in tempo reale da un contatore di frame protocol-aware operante a 1 MSa/s.

Il sistema è stato valutato su un testbench di laboratorio contro un attacco bus-off di

classe ERAM implementato tramite la tecnica CANflict. Una sessione di riferimento di 126,1 secondi e oltre 65 milioni di campioni ha prodotto zero falsi positivi e zero attivazioni delle regole. Due sessioni dedicate di attacco bus-off hanno raggiunto un tasso di rilevamento degli error flag del 100% (16/16 e 19/19 error flag confermati da anomalie di tensione precedenti). Una sessione di attacco sostenuto di 127,4 secondi con 575 error flag ha raggiunto un tasso di rilevamento complessivo dell'89,6% (515/575 confermati). Questi risultati dimostrano la fattibilità del monitoraggio di tensione correlato agli error flag come primo passo verso un rilevamento delle intrusioni a livello fisico consapevole dell'ERAM.

Parole chiave: CAN bus, sistema di rilevamento delle intrusioni, sicurezza a livello fisico, monitoraggio di tensione, correlazione error flag, sicurezza automotive

Contents

1 | Introduction

Means of transportation are an essential part of modern life, and cars in particular have a leading role in society, being the most common vehicle. Automotive manufacturers today compete not only on performance and comfort, but on the richness of their electronic systems: modern vehicles can feature over one hundred Electronic Control Units (ECUs) governing everything from engine management and antilock braking to infotainment and park-assist. Features such as lane keeping, adaptive cruise control and the first steps toward autonomous driving have made cars, in a very real sense, rolling computers on wheels.

This increasing electronic complexity has been accompanied by a parallel growth in connectivity. Vehicles are no longer isolated systems: they communicate over Bluetooth, Wi-Fi, and cellular links, expose standardised diagnostic interfaces through the OBD-II port, and are progressively integrated into Vehicle-to-Everything (V2X) architectures that exchange data with other vehicles, road infrastructure and cloud services. Every wireless interface is a potential entry point for a remote attacker.

The backbone of in-vehicle communication is the Controller Area Network (CAN) bus, a broadcast serial protocol standardised by ISO 11898 and introduced by Robert Bosch GmbH in 1986. CAN was designed for robustness and real-time performance in noisy automotive environments, not for security. The protocol provides no encryption, no message authentication and no access control: any node on the bus can read every frame and inject frames with any identifier. These properties were acceptable when vehicles were closed systems with no external connectivity; they are plainly inadequate in the face of modern attack surfaces.

Researchers have demonstrated the real consequences of this gap. In 2015, Miller and Valasek remotely exploited the infotainment system of a Jeep Cherokee and gained full control over steering and braking [13]. In 2017, Tencent's Keen Security Lab performed a similar remote compromise on a Tesla Model S [18]. As vehicles become more connected, the scale and sophistication of such threats will only grow.

1.1. The conventional attacker model and its limitations.

Most CAN security research has been conducted under what the recent literature calls the *Conventional Remote Attacker Model* (CRAM) [17]. Under this model, a remote attacker who has compromised an ECU can send and receive complete, well-formed CAN frames through the intact CAN controller — but cannot manipulate the controller itself, inject individual bits, or interact with the bus outside the rules enforced by the data-link layer hardware. Within these bounds, CRAM attackers can already execute dangerous attacks: message fabrication, masquerade, replay, flooding and targeted denial-of-service.

In response, the security community has developed a variety of Intrusion Detection Systems (IDSs) tailored to these threats. Payload-based IDSs flag anomalous message contents; timing-based IDSs exploit the near-constant transmission period of ECU-generated frames to build clock-skew fingerprints; and physical-layer IDSs monitor the analogue voltage waveform on the bus, exploiting the fact that each transceiver's hardware characteristics produce a unique electrical fingerprint.

1.2. The emerging threat: the Enhanced Remote Attacker Model.

Recent work has revealed that the CRAM assumption — that a remote attacker cannot control the data-link layer — may be obsolete. De Faveri Tron et al. showed with CANflict [8] that pin conflicts between the CAN controller and other on-chip peripherals (SPI, UART, I2C, GPIO) allow a compromised MCU to read and inject individual bits on the CAN bus entirely from software, bypassing the CAN controller's rule-enforcement logic and keeping up with a 1 Mbit/s bus even on low-end microcontrollers. Kulandaivel et al. demonstrated a similar capability through peripheral clock gating [10]. Taken together, these works establish the *Enhanced Remote Attacker Model* (ERAM), formalised and analysed comprehensively by Tang et al. in ERACAN [17].

An ERAM attacker retains all CRAM capabilities and adds several new ones: arbitrary frame injection (including error frames, overload frames, and malformed frames), pulse injection at any bit boundary, precise timing of bus events in real time, and the ability to ignore protocol rules such as error-state transitions and arbitration. This expanded capability set enables a range of attacks — frame hijacking, Janus and counterfeit frames, synchronisation disruption, double-receive, freeze doom loop — that are undetectable

by every class of IDS designed against the CRAM. In particular, ERAM attackers can directly inject short pulses that corrupt physical-layer signals or gradually poison the voltage fingerprint models on which physical-layer IDSs depend, without triggering the error patterns those IDSs monitor.

This thesis addresses the question: *can a physical-layer IDS be designed that is effective against an ERAM attacker?*

1.3. Thesis contribution.

The key insight is that an ERAM attacker, while gaining full data-link layer control, still uses the victim ECU's *physical* CAN transceiver. The transceiver's analogue characteristics — slew rate, output impedance, parasitic capacitance — depend on the silicon manufacturing process and printed-circuit-board layout, which the attacker cannot alter through software alone. A physical-layer IDS that monitors the voltage waveform emitted by the transceiver, rather than the digital content assembled by the controller, therefore retains its discriminating power even against ERAM attackers, provided it is also hardened against the specific fingerprint-corruption techniques the model enables.

This work makes the following contributions:

1. An analysis of the ERAM threat envelope as it applies to physical-layer IDSs, identifying three classes of vulnerability — fingerprint corruption, partial-frame attacks, and short-pulse injections — that existing voltage-based defences cannot address.
2. A physical-layer IDS prototype for a controlled laboratory environment, based on real-time statistical monitoring of the CAN bus differential voltage signal: a baseline (mean and standard deviation of burst-peak differentials) is computed offline from a clean traffic capture and used at runtime to flag samples that deviate beyond three standard deviations, with detections correlated against CAN error flags identified in real time by a protocol-aware frame counter, providing a transparent and interpretable anomaly-detection mechanism as a starting point for more advanced development.
3. An experimental evaluation on a laboratory CAN test bed of an ERAM-class bus-off attack implemented via the CANflict technique, with quantified error-flag detection rates (100% on dedicated bus-off sessions, 89.6% on a sustained attack session), a zero false-positive baseline, and a discussion of the system's limitations and directions for future work.

1.4. Structure of the thesis.

Chapter 2 provides the technical background on the CAN bus protocol. Chapter 3 analyses the security landscape: the vulnerabilities of CAN, the attack taxonomy from CRAM to ERAM, the existing IDS landscape, and the specific gaps that motivate this work. Chapter 4 presents the proposed IDS architecture and design choices. Chapter 5 describes the hardware and software implementation. Chapter 6 reports the experimental results. Chapter 7 concludes with a summary of contributions, limitations, and directions for future work.

2 | Background on CAN Bus

The Controller Area Network (CAN) is a serial, multi-master broadcast communication protocol invented by Robert Bosch GmbH in the early 1980s and introduced at the Society of Automotive Engineers (SAE) Congress in 1986 [14]. It was standardised as ISO 11898 in 1993 and has since become the dominant in-vehicle network protocol, used not only in cars but also in trucks, buses, aircraft, ships, industrial machinery and medical equipment. Its success derives from its cost-effectiveness, its real-time performance, and its robustness to electromagnetic noise, a critical property in the high-interference environment of an automotive engine bay.

2.1. Protocol Overview

ISO 11898 defines only the two lowest layers of the OSI model for CAN: the physical layer and the data link layer. Higher-layer protocols such as CANopen and SAE J1939 are built on top of the standard and are described briefly in Section 2.4; the Unified Diagnostic Services (UDS) protocol is discussed in Section 2.5.

A CAN network is a two-wire differential bus that interconnects all nodes in a broadcast topology. Every node receives every frame, and the protocol provides no addressing of individual receivers: each node independently decides whether an incoming message is relevant to it.

Two main variants of the protocol are in widespread use today. Classic CAN (CAN 2.0) supports a maximum bit rate of 1 Mbit/s and a maximum payload of 8 bytes. CAN Flexible Data-rate (CAN FD), standardised in ISO 11898-1:2015, extends both limits: the data phase of a frame can run at a higher bit rate (up to several Mbit/s with suitable transceivers), and the payload can be up to 64 bytes. Both variants share the same arbitration mechanism and error-handling philosophy; the key difference for physical-layer security is discussed in Section 2.4.

2.1.1. Physical Layer

At the physical level, bits are encoded as the differential voltage between two wires called CAN-H (CAN High) and CAN-L (CAN Low):

- **Dominant bit (logical 0):** CAN-H is driven to approximately 3.5 V and CAN-L to approximately 1.5 V, producing a differential of ≈2 V.
- **Recessive bit (logical 1):** Both lines float to approximately 2.5 V, producing a differential of ≈0 V, which is also the idle state of the bus.

The bus is a wired-AND: whenever any node drives a dominant bit, the differential bus level becomes dominant regardless of what other nodes are simultaneously outputting. This property underlies the arbitration mechanism. To prevent signal reflections that would distort waveforms at high bit rates, the standard mandates 120 Ω termination resistors at both ends of the bus.

Each node that participates in CAN communication contains a *CAN transceiver*, which is the analogue front-end responsible for converting the digital bit stream produced by the CAN controller into the differential voltage on CAN-H/CAN-L, and vice versa. The transceiver is physically the closest component to the bus wires and operates entirely at the physical layer. Common automotive transceivers include the NXP TJA1050, Texas Instruments SN65HVD230, and Microchip MCP2551.

A critical observation for this thesis is that the transceiver's output characteristics — its slew rate (the rate at which the bus voltage transitions between dominant and recessive), output impedance, and parasitic capacitances — are determined by the silicon manufacturing process, the specific component tolerances, and the physical layout of the printed circuit board. Advanced physical-layer IDSs exploit the fact that even nominally identical transceivers may exhibit measurable differences in these parameters, treating those differences as a per-node *fingerprint* (see Section 3.4). The IDS proposed in this work takes a simpler, complementary approach: rather than per-node fingerprinting, it monitors the differential voltage ($V_{\text{CAN-H}} - V_{\text{CAN-L}}$) against a statistical baseline (mean + 3σ) computed during a benign traffic reference window, and correlates voltage anomalies with CAN error flags to flag injection attacks, as described in Chapter 4.

2.1.2. Data Link Layer

The data link layer is implemented by the *CAN controller*, which is typically an on-chip peripheral of the ECU's microcontroller. The controller is responsible for frame assembly

and disassembly, bit stuffing and de-stuffing, CRC calculation and verification, arbitration, acknowledgment, and error handling. The microcontroller application communicates with the controller by providing the ID and payload of a frame to transmit or by reading the ID and payload of a received frame; all lower-level protocol mechanics are handled autonomously by the controller hardware.

The data link layer defines four frame types:

- **Data Frame:** the standard bearer of application data.
- **Remote Frame:** a request to another node to transmit a data frame with a specific ID; structurally identical to a data frame with the RTR bit set and no data field.
- **Error Frame:** generated autonomously by the controller upon detection of a protocol error; it consists of an error flag (six dominant bits in the error-active state, or six recessive bits in the error-passive state) followed by an error delimiter (eight recessive bits).
- **Overload Frame:** inserted by a receiver to request additional delay before the next data or remote frame.

Data frame structure. A standard (11-bit ID) CAN data frame consists of the following fields:

- **Start of Frame (SOF):** one dominant bit that signals the beginning of a frame after an idle bus.
- **Arbitration Field:** the 11-bit message identifier (or 29 bits in the extended format), followed by the RTR bit. The identifier serves simultaneously as a content label and as a priority token in the arbitration process.
- **Control Field:** includes the IDE bit (standard vs. extended format), a reserved bit, and the 4-bit Data Length Code (DLC) specifying the payload length in bytes.
- **Data Field:** 0 to 8 bytes of application payload.
- **CRC Field:** a 15-bit Cyclic Redundancy Check over all preceding fields, followed by a recessive CRC delimiter.
- **ACK Field:** the transmitter sends a recessive bit; any correctly receiving node overwrites it with a dominant bit. The transmitter verifies that the ACK slot was acknowledged.
- **End of Frame (EOF):** seven recessive bits.

- **Intermission (IFS):** three recessive bits separating consecutive frames.

Bit stuffing. To guarantee sufficient signal transitions for receiver synchronisation, the CAN standard requires that whenever five consecutive bits of the same polarity are transmitted in the region from SOF to the end of the CRC field, a *stuff bit* of the opposite polarity is inserted. Receivers strip stuff bits before passing data to the application. A violation of the stuffing rule is one of the five recognised error types (see Section 2.3).

2.2. Arbitration and Bus Access Control

CAN implements Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Before transmitting, a node senses the bus; if idle (all recessive), it begins transmitting. Because multiple nodes may independently decide to start at the same time, a deterministic collision resolution mechanism is needed.

Arbitration is performed bit by bit over the ID field exploiting the wired-AND property of the bus. Each transmitting node reads back the bus level after placing each bit. A node transmitting a recessive bit (1) that reads a dominant bit (0) back has lost arbitration — another node with a lower ID is transmitting a dominant bit, which overwrites the recessive — and immediately ceases transmission without generating an error. The node with the lowest numerical ID value always wins. Because a lower ID corresponds to more dominant bits in the most-significant positions, safety-critical messages are assigned low IDs to guarantee they are always prioritised.

This arbitration mechanism has an important security implication: an attacker that continuously injects frames with ID 0x000 can prevent every other node from winning arbitration, achieving a complete denial of service. More subtly, an ERAM attacker can force a specific node to lose arbitration on its own frames by injecting a dominant bit over the ID field, delaying or silencing that node without winning arbitration itself (arbitration denial attack [17]).

2.3. Error Handling

The CAN protocol defines five error types detected during frame transmission or reception:

- **Bit error:** a transmitting node reads a bus level different from the bit it placed (outside the arbitration field, where overwriting is expected).
- **Stuff error:** more than five consecutive bits of the same polarity are detected, violating the stuffing rule.

- **CRC error:** the CRC calculated by the receiver does not match the CRC field in the received frame.
- **Form error:** a bit field with a fixed value (CRC delimiter, ACK delimiter, EOF, IFS) contains an illegal bit.
- **Acknowledgment error:** the transmitter detects that no node acknowledged the frame.

Upon detecting an error, the node that detects it transmits an error frame to abort the current transmission and notify all other nodes. Error handling is governed by two counters maintained by each CAN controller: the *Transmit Error Counter* (TEC) and the *Receive Error Counter* (REC). The TEC is incremented by 8 on each transmission error and decremented by 1 on each successful transmission; the REC is incremented by 1 (or 8 if the receiver caused the error) on each reception error.

The values of TEC and REC determine the node's error state:

- **Error Active** (TEC ≤ 127 and REC ≤ 127): normal operation; errors are signalled with an *active error flag* (six dominant bits), which destroys the offending frame and forces retransmission.
- **Error Passive** (TEC > 127 or REC > 127): errors are signalled with a *passive error flag* (six recessive bits), which does not disrupt other nodes' reception. The node must also observe an extra 8-bit *suspend transmission* period before retransmitting.
- **Bus Off** (TEC > 255): the CAN controller ceases all transmission and stops processing received frames. The node can resume normal operation only after it has observed $128 \times 11 = 1408$ consecutive recessive bits on the bus, ensuring that no other node is in a persistent error state before communication is restored.

The error-handling mechanism was designed for fault tolerance in genuinely noisy environments, but it constitutes a significant security vulnerability: an attacker that can inject errors into a victim's transmissions can systematically drive the victim's TEC toward 256 and silence it (bus-off attack). The ERAM model makes this attack more powerful and harder to detect, as discussed in Chapter 3.

2.4. CAN FD and Other Extensions

CAN FD (ISO 11898-1:2015) retains the classic CAN arbitration phase but switches to a higher bit rate for the data phase after the arbitration field has been resolved. The payload capacity is extended to 64 bytes. From a physical-layer security perspective, the

bit-rate switch introduces a change in the voltage waveform characteristics: the slew rate and transition times observed during the data phase differ from those in the arbitration phase. Physical-layer fingerprinting approaches must account for this, either by restricting feature extraction to the arbitration field or by training separate models for each phase.

Other protocols are CANopen and SAE J1939, which are higher-layer protocols built on top of classic CAN. They define structured message formats, network management, and time-synchronisation mechanisms, but do not alter the physical or data-link layer and therefore do not affect the physical fingerprinting approach.

2.5. Diagnostics: OBD-II and UDS

The On-Board Diagnostics II (OBD-II) port is a standardised connector that has been mandatory in all vehicles sold in the European Union since 2001 and in the United States since 1996. It provides physical access to the vehicle's CAN buses without opening the vehicle. While intended for authorised diagnostic use, it is a commonly exploited attacker entry point: any device with a standard OBD-II adapter can attach a node to the CAN bus. The OBD-II port is a commonly exploited attacker entry point, since any device with a standard adapter can attach a node to the CAN bus. Other attack vectors have also been demonstrated: the 2015 Jeep Cherokee remote compromise was carried out via the vehicle's cellular infotainment link, while the Toyota RAV4 theft technique described by Tindell involved injecting CAN signals directly into the vehicle's headlight wiring rather than through the diagnostic port.

The Unified Diagnostic Services (UDS) protocol (ISO 14229) runs over CAN and defines a request–response service interface for reading and writing ECU data, reprogramming firmware, controlling ECU functions, and triggering diagnostic sessions. UDS requires no additional authentication beyond what CAN itself provides (i.e., none). An attacker with access to the bus can therefore exploit UDS to read sensitive data, overwrite control parameters, reset ECUs, or enter extended diagnostic sessions that unlock otherwise restricted functions.

2.6. Structure of a CAN Node

Understanding the internal architecture of a CAN node is essential for appreciating both the attack surface exposed by the ERAM model and the limitations of existing defences.

A typical ECU consists of three main components:

- **Microcontroller Unit (MCU):** executes the application firmware, decides which messages to transmit and how to react to received messages.
- **CAN controller:** implements the data-link layer; it may be an on-chip peripheral of the MCU (the most common modern configuration) or a stand-alone chip connected to the MCU via SPI. It enforces protocol rules, manages error counters, and presents a simple send/receive interface to the application.
- **CAN transceiver:** implements the physical layer; it converts the digital CAN-TX signal from the controller into the differential voltage on CAN-H/CAN-L, and converts the differential bus voltage into a digital CAN-RX signal for the controller.

As noted in Section 2.1.1, the transceiver's analogue output is the source of the hardware fingerprint exploited by physical-layer IDSs. Crucially, because the MCU software interfaces only with the CAN controller — which presents a digital, abstracted view of the bus — a conventional attacker (CRAM) cannot alter the physical-layer fingerprint. The ERAM model, however, allows the attacker to *bypass* the CAN controller entirely by redirecting the MCU's GPIO or other peripherals to the CAN-TX and CAN-RX pins, as explained in Chapter 3.

3 | Threat Model and Security in the CAN Bus

3.1. Security Concerns in the Automotive Domain

Despite its central role in vehicle safety, the CAN bus was designed with no security features whatsoever. The protocol lacks encryption (all messages travel in plaintext), message authentication (any node can transmit any ID with any payload), integrity protection beyond error detection (the CRC guards only against accidental bit errors, not intentional manipulation), and access control (there is no mechanism to restrict which nodes may send or receive which messages). These deficiencies were not considered problematic at the time of the protocol's inception: vehicles were closed systems with no external connectivity, and physical access was assumed to be restricted to authorised users.

This assumption has been progressively eroded. Cars today incorporate Bluetooth, Wi-Fi, cellular (4G/5G), and V2X communication interfaces, each of which constitutes a potential remote entry point. The OBD-II diagnostic port, mandatory in all modern vehicles, provides immediate physical access to the CAN bus. Third-party devices — telematics dongles, aftermarket infotainment systems, and connected accessories — are regularly connected to the OBD-II port, each one potentially carrying its own vulnerabilities.

The consequences of a successful attack on the CAN bus can be severe. The real-world incidents cited in the Introduction — the Jeep Cherokee and Tesla Model S remote exploits [13, 18] and the Toyota RAV4 relay attack — are emblematic of a broader trend: CAN bus vulnerabilities have moved from theoretical concern to demonstrated risk, and the attack surface continues to expand as vehicles incorporate more wireless interfaces and remote-service features.

3.2. Attacker Models

Security research on CAN bus has historically assumed a single attacker model. Tang et al. in ERACAN [17] formalise a clean taxonomy distinguishing two models of increasing capability.

3.2.1. Conventional Remote Attacker Model (CRAM)

The Conventional Remote Attacker Model (CRAM) represents the state of the art prior to recent low-level attack research. A CRAM attacker has compromised an ECU remotely — through a wireless interface, the OBD-II port, or firmware exploitation — and can execute arbitrary code on the ECU's operating system. The attacker's access to the CAN bus is mediated entirely by the intact CAN controller, which enforces all protocol rules. Specifically, a CRAM attacker can:

- **Receive valid frames:** read the ID and payload of correctly received frames after the controller has assembled and validated them.
- **Transmit valid frames:** inject well-formed data or remote frames with any ID and payload; the controller enforces arbitration, bit stuffing, CRC generation, and acknowledgment rules.
- **Exploit simultaneous transmission:** by timing a frame injection to begin just as the victim starts transmitting under the same ID (both win arbitration), the attacker can introduce a dominant bit where the victim sends recessive during the data or CRC field. The victim detects a bit error and increments its TEC. Repeated across many transmission cycles, this gradually drives the victim toward the error-passive or bus-off state without requiring any physical access.

These three capabilities enable a wide range of attacks, including fabrication, masquerade, DoS, bus-off, WeepingCAN, and limited forms of error injection.

3.2.2. Enhanced Remote Attacker Model (ERAM)

Recent work has demonstrated that the CRAM assumption — the controller's data-link layer logic being inviolable from software — is obsolete for a large proportion of modern ECUs. Two mechanisms have been identified:

1. **Peripheral clock gating (CANnon [10]):** on MCUs where the CAN controller's clock can be suspended by software, gating the clock during transmission freezes the controller's finite state machine, causing it to hold the last transmitted bit

indefinitely. This enables targeted single-frame bus-off attacks without the timing uncertainty of simultaneous transmission.

2. **Pin conflict exploitation (CANflict [8]):** in most modern MCU designs, the CAN-TX and CAN-RX signals are connected to the same physical pins as other peripherals (SPI COPI/CIPO, UART TX/RX, I2C SDA, GPIO) through an internal multiplexer configurable from software. By redirecting these pins to a less-constrained peripheral, an attacker can read and inject individual bits on the CAN bus entirely from software, bypassing the CAN controller's rule enforcement logic entirely. De Faveri Tron et al. demonstrated this on high-, mid-, and low-end automotive-grade MCUs, achieving reliable bit-level access at 1 Mbit/s.

The *Enhanced Remote Attacker Model* (ERAM), as comprehensively analysed in ERA-CAN [17], assumes an attacker who has compromised an ECU and is able to exploit one of these (or equivalent) mechanisms. ERAM subsumes all CRAM capabilities and adds three additional categories:

- **Ubiquitous link-layer visibility:** the attacker can read individual bits in real time, including the ID, payload, and protocol fields of frames currently in progress; error frames, overload frames, and other atypical bus events; and precise timestamps of any bus event. This enables, for example, attacks targeting a frame only after a specific ID or payload value is observed.
- **Omnipotent link-layer write-ability:** the attacker can inject arbitrary signals at any time, including valid frames of any type, error frames, overload frames, partial frames, pulses, and sequences that violate the CAN standard. This includes the ability to flip a single bit in an ongoing transmission without necessarily triggering detectable errors.
- **Link-layer rule non-compliance:** the attacker is not bound by protocol rules. It can transmit during arbitration without backing off, continue transmitting after an error frame, ignore error counter transitions, and acknowledge its own frames.

This work adopts the ERAM model as its threat model. This is the appropriate choice for an IDS based on physical-layer signals, because:

1. ERAM attackers can launch all CRAM attacks; in particular, bus flooding, sustained dominant injection, and differential collapse all produce differential voltage anomalies that fall outside the statistical baseline monitored by the proposed IDS;
2. ERAM attackers can specifically target and degrade physical-layer IDSs through fingerprint corruption and frame hijacking, which CRAM attackers cannot;

3. the number of ECU platforms vulnerable to ERAM exploitation is large and growing, as pin conflicts between CAN and other peripherals are a structural feature of general-purpose automotive MCU design [8].

3.3. Attacks on the CAN Bus

The following sections describe the attack taxonomy relevant to this thesis, progressing from CRAM to ERAM attacks.

3.3.1. CRAM Attacks

Fabrication and Replay

A fabrication attack consists of injecting frames with a valid ID and an arbitrary or crafted payload. Because the bus has no authentication, all receiving ECUs process the injected frame as legitimate. A replay attack is a special case in which a previously captured legitimate frame is retransmitted at a later time to reproduce its effect, requiring no knowledge of the payload semantics.

Fuzzing

Fuzzing floods the bus with frames carrying randomised IDs and/or payloads. Used as a vulnerability discovery technique, it can also trigger unexpected ECU behaviour or degrade bus performance.

Denial of Service (DoS)

By continuously injecting frames with the lowest possible ID (e.g., 0x000), an attacker always wins arbitration and prevents all other nodes from transmitting, achieving a complete network denial of service. A targeted DoS silences a single victim ECU by using an ID slightly lower than the victim's.

Bus-Off Attack

A bus-off attack systematically forces a victim ECU's TEC above 255 by repeatedly injecting errors into the victim's transmissions using the simultaneous transmission technique. Once bus-off, the victim is silenced for at least 128×11 recessive bit times. Multiple variants have been described, exploiting bit errors, stuff errors, or sustained dominant injection [4].

Masquerade Attack

A masquerade attack combines a bus-off attack to silence a victim ECU with a subsequent fabrication attack that impersonates it under the same CAN ID. The attacker uses the correct ID and a plausible payload, defeating both payload-based and timing-based IDSs. Advanced voltage-fingerprinting approaches (see Section 3.4) can further identify the attack by comparing the transceiver characteristics of the masquerading node against the enrolled victim profile. The system proposed in this thesis detects a masquerade attempt indirectly, through the voltage anomalies produced during the bus-off phase used to silence the victim (e.g., sustained dominant injection).

WeepingCAN

WeepingCAN, introduced by Bloom [3], is a stealthy variant of the bus-off attack that deliberately controls the victim's error counters to avoid the distinctive patterns (systematic error flags, passive error state transitions) that timing- and event-based IDSs use for detection. By keeping the error increment rate low and irregular, the attack extends over minutes to hours while remaining below detection thresholds.

3.3.2. ERAM-Specific Attacks

Frame Hijacking

An ERAM attacker can silently push a victim to the error-passive state and then hijack a frame mid-transmission: after the legitimate header has been transmitted (which may contain authentication tokens or fingerprinting-sensitive data), the attacker flips a recessive bit to dominant and continues transmitting a forged payload with a valid CRC. The victim detects a bit error and sends a passive error flag (six recessive bits), but this is overwritten by the attacker's dominant bits and is invisible to other nodes. Other nodes receive the attacker's data as a valid frame attributed to the victim.

Janus and Counterfeit Frames

When the sample points of two receiving nodes differ, an ERAM attacker can craft a frame with bit transitions timed to fall between the two sample points [20]. One receiver samples a 1 while the other samples a 0, causing them to receive different payloads from the same frame without either generating an error. A counterfeit frame is a variant in which the attacker flips bits in an ongoing legitimate transmission by injecting a pulse that starts after the sender's sample point and ends after the receiver's, modifying the

received value without the sender detecting a bit error.

Synchronisation Disruption

By injecting a dominant pulse after the synchronisation segment of a recessive bit, an ERAM attacker causes all nodes to observe a $1 \rightarrow 0$ edge outside the synchronisation window and triggers resynchronisation. Depending on each node's bit-time configuration, they may adjust their bit boundaries differently, accumulating timing mismatches that eventually cause CRC errors between the transmitter and receivers.

Double Receive and Freeze Doom Loop

An ERAM attacker can flip the last bit of the EOF field from recessive to dominant, causing the sender to detect a bit error and retransmit, while receivers — which have already accepted the frame — receive the same message twice. Alternatively, sending an overload frame at the first bit of the intermission period triggers a freeze doom loop in which all nodes continuously exchange overload frames, halting bus traffic indefinitely without incrementing any error counter [19].

Physical Fingerprint Corruption

Building on the DUET attack [2] — a technique in which an attacker uses a second compromised ECU to inject subtle voltage pulses during the victim's transmissions, gradually shifting the IDS model toward the attacker's fingerprint — an ERAM attacker can carry out the same poisoning with a single node, since bit-level write access allows pulse injection without a hardware implant. Because physical-layer IDSs that use online model updates re-train on recent measurements, such a gradual and stealthy fingerprint corruption attack can slowly shift the IDS's model of the victim toward the attacker's physical characteristics, ultimately enabling an undetected masquerade.

3.4. Existing Intrusion Detection Systems for CAN Bus

Intrusion detection systems represent the most practical security layer for CAN bus, as they require no modification of the underlying protocol, introduce no timing constraints on message scheduling, and can be retrofitted to existing vehicles as a dedicated monitor node. The existing literature can be organised along two axes: the protocol layer at which monitoring is performed (data-link vs. physical), and the detection strategy (signature-

based vs. anomaly-based).

3.4.1. Payload-Based IDSs

Payload-based IDSs operate at the data-link layer. They monitor the ID and/or the payload content of CAN frames and flag anomalies against a learnt model of normal traffic.

Statistical approaches such as those of Ohira et al. use sliding-window frequency analysis to detect DoS attacks. Machine learning approaches, exemplified by CANnolo [11] (LSTM autoencoder), CANet (per-ID LSTM), CANdito [12] (RNN/LSTM), and GIDS (generative adversarial network), learn the expected payload distributions per ID and flag deviations.

The fundamental limitation of payload-based IDSs is that they are entirely blind to the identity of the transmitting node: they can detect that an anomalous payload is present but cannot determine whether it was injected by an attacker or generated by a malfunctioning legitimate ECU. Against masquerade attacks — in which the attacker sends a valid payload under the victim's ID — they are ineffective. Under ERAM, Janus and counterfeit frame attacks can deliver payloads that appear normal to the IDS monitor while being malicious to other receivers.

3.4.2. Timing-Based IDSs

Timing-based IDSs exploit the observation that ECU-generated messages are transmitted with near-constant inter-message intervals, and that the CAN controller's oscillator introduces a unique, measurable clock skew.

CIDS by Cho and Shin [5] estimates each ECU's clock skew using Recursive Least Squares and builds per-ID sender fingerprints, enabling it to detect masquerade attacks by flagging frames whose inter-arrival timing does not match the enrolled clock skew. ClockIDS extends this with Dynamic Time Warping for improved source identification. TIDAL-CAN and CAN Radar measure signal propagation times and reflectometry patterns to localise nodes physically on the bus.

Timing-based IDSs offer sender authentication without any hardware modification beyond a high-resolution timestamp source. However, they require long observation windows to build reliable clock-skew estimates, are susceptible to timing attacks that mimic the victim's period (demonstrated by Sagong et al. [16]), and cannot detect single-frame attacks. Under ERAM, counterfeit and Janus frame attacks modify a frame's content

without altering its transmission time, making them completely invisible to interval-based approaches. Frame hijacking likewise preserves the original transmission timing.

3.4.3. Voltage-Based / Physical Fingerprinting IDSs

Physical-layer IDSs monitor the analogue voltage waveform on the CAN bus and extract the transceiver's hardware fingerprint. This approach provides a fundamentally stronger identity guarantee than timing-based IDSs because the fingerprint is rooted in unalterable physical hardware characteristics.

Viden [6] extracts the steady-state dominant-bit voltage level of each ECU as a per-node fingerprint, achieving sender identification at a sampling rate of only 1 MSa/s. Because the dominant-state voltage varies measurably between nodes even with the same transceiver model, a single scalar feature — the mean CANH voltage during dominant bits — suffices for reliable identification. The approach in the present work is conceptually related: it also monitors the voltage level of the CAN bus signal, but instead of building a per-node fingerprint it uses a statistical baseline (mean + 3σ) of the differential voltage computed from a clean reference window, and correlates anomalous samples with CAN error flags detected in real time by a protocol-aware frame counter. VoltageIDS [7] uses the differential CAN-H/CAN-L signal, extracting features from rising edges, falling edges, and stable dominant states in both time and frequency domains, achieving high accuracy at the cost of a 2.5 GSa/s oscilloscope. SIMPLE [9] restricts features to rising and falling edge shapes and applies Fisher's Discriminant Analysis for dimensionality reduction, enabling deployment on low-cost hardware. EASI and ASSASSIN extract edge asymmetry features using a Time-to-Digital Converter (TDC), achieving latency independent of the number of ECUs. EdgeTDC [15] achieves 100% masquerade detection by doubling cable length to increase propagation-delay differences between nodes.

These approaches have demonstrated high effectiveness against CRAM masquerade attacks. However, as discussed in the following section, they are vulnerable in important ways to the ERAM model.

3.5. Limitations of Existing Approaches and the Need for an ERAM-Aware IDS

The ERACAN paper [17] provides the most comprehensive analysis of how the ERAM model undermines existing defences. For physical-layer IDSs in particular, three classes of vulnerability are identified.

Vulnerability to fingerprint corruption. Physical-layer IDSs that use online model updates to adapt to temperature drift and battery charge variation are vulnerable to a gradual fingerprint poisoning attack. An ERAM attacker can inject pulses that partially overlap with the victim's transmissions, incrementally shifting the measured waveform toward the attacker's own characteristics. Because the IDS's update rule cannot distinguish a legitimate environmental drift from an adversarial perturbation, the model converges to the attacker's fingerprint over time, enabling an eventual undetected masquerade. Bhatia et al. [2] demonstrated a CRAM version of this attack (requiring two attacker ECUs and simultaneous transmission); ERAM makes it stealthier and requires only one attacker.

Vulnerability to partial-frame attacks. Most voltage-based IDSs extract features from a specific field of the frame — typically the arbitration field or the CRC field. An ERAM attacker performing frame hijacking leaves that field intact (transmitted by the legitimate node with the correct fingerprint) and only modifies the payload fields that follow. The IDS authenticates the sender based on the unmodified portion and passes the frame as legitimate, while the receivers accept the forged payload as valid.

Inability to detect ERAM-specific attacks. Short pulse injections (synchronisation disruption, arbitration denial, partial bit flips) do not produce complete frames and are therefore invisible to IDSs that process only fully assembled frames. GPIO-transmitted frames evade asymmetry-based fingerprinting if the attacker can match the CAN controller's bit period, but betray themselves through increased intra-frame bit-period variance (the GPIO lacks the hardware-disciplined timing of a CAN controller oscillator). Janus and counterfeit frames evade all IDSs that only monitor the frame as seen by the monitor node.

These gaps illustrate the significant open challenges that a fully ERAM-resistant physical-layer IDS would need to address. The IDS proposed in this thesis takes a first, deliberately scoped step in this direction: rather than targeting the full ERAM attack spectrum, it focuses on detecting the voltage-level anomalies that are characteristic of common bus-level attacks (flooding, differential collapse, sustained dominant or recessive injection) in a controlled laboratory environment. The system is conceived as a proof of concept and a baseline for future, more comprehensive development; its scope and limitations are discussed in detail in Chapters 4 and 7.

4 | System Design

4.1. Design Goals and Requirements

The proposed IDS is designed around four primary requirements derived from the constraints of the laboratory prototype and the threat model of Chapter 3.

Passive monitoring. The IDS must not alter the CAN bus topology, introduce any additional network nodes, or modify the firmware of any existing ECU. It operates exclusively as a passive observer, leaving all protocol mechanics intact.

Real-time detection. Alerts must be generated on a millisecond timescale. The streaming acquisition pipeline processes samples continuously; each detection rule evaluates every individual sample so that the worst-case detection latency is bounded by the alert cooldown interval (500 ms by default), not by any processing backlog.

Statistically derived thresholds. Rather than fixed hard-coded thresholds, the IDS derives detection limits from the actual bus during normal operation. A dedicated calibration phase (Section 4.5) captures the statistical distribution of the voltage signal under clean traffic and sets each threshold at three standard deviations from the mean, accommodating transceiver-to-transceiver variation and normal bus-load effects.

Low false-positive rate. A threshold at $\mu + 3\sigma$ corresponds to a false-alarm probability below 0.13% under a Gaussian assumption. An additional confirmation filter requires three consecutive anomalous samples before raising an alert for the sustained rule (R_2), further suppressing transient noise spikes.

The system is explicitly scoped to a controlled laboratory environment with a fixed, known set of nodes. It is designed as a proof of concept and a baseline for future development, not as a production-ready automotive IDS.

4.2. System Architecture

The IDS is organised into two sequential pipelines that share the same hardware front-end but serve distinct purposes: a *calibration pipeline* run once before deployment to establish the statistical baseline, and a *detection pipeline* run continuously during operation. Figure 4.1 illustrates the logical components and the data flow between them.

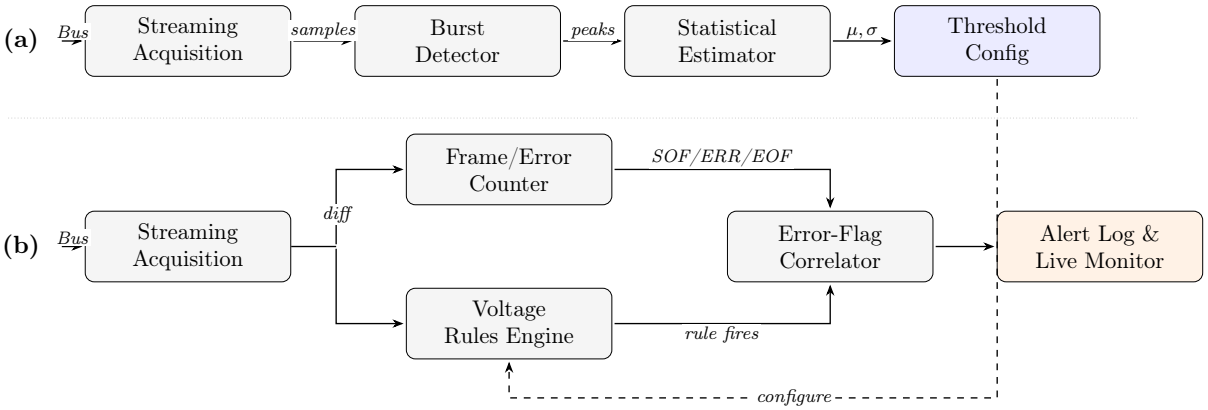

Figure 4.1: Logical architecture of the IDS. (a) Calibration pipeline: raw voltage samples are segmented by the Burst Detector into per-frame events; the Statistical Estimator accumulates burst-peak differential values and derives the $\mu + 3\sigma$ threshold stored in the Threshold Config. (b) Detection pipeline: the Streaming Acquisition module feeds the differential signal into two parallel paths — the Frame/Error Counter (which identifies SOF, error flags, and EOF events from dominant/recessive run lengths) and the Voltage Rules Engine (which evaluates rules R_1–R_2). The Error-Flag Correlator checks whether a voltage rule fired within the lookback window preceding each detected error flag, classifying each flag as confirmed or missed. Thresholds produced by calibration are supplied as a static configuration to the Voltage Rules Engine.

The six logical components are the following.

Streaming Acquisition. Interfaces with the oscilloscope hardware and delivers a continuous stream of raw voltage samples at 1 MSa/s on both CAN-H and CAN-L channels to the downstream processing stage. Both pipelines share the same physical hardware and the same acquisition parameters. The differential signal $V_{\text{diff}} = V_{\text{CAN-H}} - V_{\text{CAN-L}}$ is computed per sample and fed to both the Frame/Error Counter and the Voltage Rules Engine.

Burst Detector. Used only during calibration. A state machine that segments the continuous sample stream into per-frame burst events by monitoring the differential signal $|V_{\text{CAN-H}} - V_{\text{CAN-L}}|$. It outputs a pair of scalar features per burst: the peak CANH voltage

and the peak differential.

Statistical Estimator. Used only during calibration. Accumulates the burst-peak sequences produced by the Burst Detector and computes the sample mean μ and standard deviation σ of the peak differential series. The output is the Threshold Config: $\mu_{\text{diff}} + 3\sigma_{\text{diff}}$ that parameterises the detection rules.

Frame/Error Counter. Used during detection. A vectorised state machine that processes the differential signal to identify CAN frame boundaries and error flags from dominant/recessive run lengths. A dominant-to-recessive transition after an inter-frame space of at least 3 bit-times signals a Start of Frame (SOF); a dominant run exceeding 5.5 bit-times within a frame signals an error flag; a recessive run of at least 7 bit-times signals an End of Frame (EOF). The counter tracks total, normal, and error frames and emits SOF, ERR, and EOF events to the Error-Flag Correlator.

Voltage Rules Engine. Applied to every dominant-phase sample during detection. Evaluates the two rules R_1–R_2 (Section 4.6) in parallel, each with its own violation counter. The threshold is supplied by the Threshold Config produced during calibration. Rules are reset at each recessive-to-dominant transition.

Error-Flag Correlator. Receives ERR events from the Frame/Error Counter and rule-fire events from the Voltage Rules Engine. For each detected error flag, it checks whether any rule fired within a configurable lookback window (default: 500 samples, i.e. 500 µs). If at least one rule fired within the window, the error flag is classified as *confirmed* (the voltage anomaly preceded and likely caused the error); otherwise it is classified as *missed*. The detection rate is the ratio of confirmed to total error flags. At each EOF event, the correlator also classifies the completed frame into one of four categories: *true positive* (error frame with a preceding voltage anomaly), *false negative* (error frame without one), *false positive* (normal frame where a rule fired — in practice this occurs during active attacks when the attacker's injection raises the bus voltage above threshold but fails to trigger an error flag), or *true negative* (normal frame with no rule fire). Alerts are written to a timestamped log and reflected in the Live Monitor display.

4.3. Signal Acquisition Architecture

The acquisition subsystem must sample both CAN-H and CAN-L continuously at a rate sufficient to resolve the steady-state voltage levels that characterise normal operation and the anomalies produced by attacks. A CAN bus running at 500 kbit/s has a nominal bit period of $T_{\text{bit}} = 2\,\mu\text{s}$; one sample per microsecond ($f_s = 1\,\text{MSa/s}$) provides two samples

per bit, which is adequate for amplitude monitoring. Approaches that measure edge rise and fall times (such as VoltageIDS [7] and SIMPLE [9]) require 10–100 MSa/s; because the present system monitors only amplitude and dominant-phase duration, 1 MSa/s is sufficient and allows the use of the lower-rate streaming mode of the PicoScope, which is also more CPU-friendly on the host PC.

The PicoScope 2205A MSO oscilloscope is used as the acquisition front-end. It provides two analogue input channels, a bandwidth of 25 MHz, and an 8-bit ADC with a configurable full-scale range of ±50 mV to ±20 V. The device is interfaced to the host PC via USB and controlled through the oscilloscope SDK. The oscilloscope is operated in *streaming mode*: samples are transferred continuously to the host without gaps, using a sample interval of 1 µs and a ±5 V full-scale range on both channels. At this range the ADC quantisation step is approximately 39 mV, which is more than adequate to distinguish a CAN-H dominant level (≈3.5 V) from a recessive level (≈2.5 V) with ample margin.

Channel A is connected to CAN-H and Channel B to CAN-L through standard passive BNC probes attached directly to the bus wiring. No external triggering is required: the acquisition is free-running, and frame boundaries are identified in software during the calibration phase by the burst detector described in Section 4.4. The differential signal $V_{\text{diff}} = V_{\text{CAN-H}} - V_{\text{CAN-L}}$ is computed per-sample in the processing thread.

4.4. Signal Preprocessing

Signal preprocessing differs between the two operating phases of the system.

Calibration phase — burst segmentation. The raw CANH and CANL voltage streams are processed by a *burst detector*, a state machine that segments the continuous acquisition into individual message events. A burst begins when $|V_{\text{diff}}|$ exceeds a configurable threshold (default 300 mV, roughly one-seventh of the nominal dominant differential) and ends when $|V_{\text{diff}}|$ has remained below that threshold for a configurable idle gap (default 10 µs, i.e. 10 samples). Bursts shorter than a minimum duration (default 5 µs) are discarded: ACK-bit pulses, which last only 1–2 µs, are thereby eliminated. For each surviving burst the *peak differential* — the maximum absolute value of $V_{\text{CAN-H}} - V_{\text{CAN-L}}$ observed during the burst — is extracted as scalar feature and forwarded to the statistical estimation stage (Section 4.5).

Detection phase — per-sample comparison. In the online detection phase, preprocessing is minimal: raw ADC counts are converted to millivolts by a linear scaling function, and V_{diff} is computed per sample. No low-pass filtering is applied; the detection

rules incorporate their own hysteresis via consecutive-sample confirmation counters that reject isolated noise spikes (see Section 4.6).

4.5. Feature Extraction and Threshold Derivation

The calibration phase converts raw burst-peak values into the two detection thresholds that the runtime IDS enforces.

Statistical baseline from burst-peak distributions. During the calibration phase only the single ECU whose frames will be targeted by the attack is active on the bus. The Calibration Pipeline processes the continuous voltage stream through the Burst Detector described in Section 4.4: each detected burst corresponds to one CAN frame, and its *peak differential* ($\max|V_{\text{CAN-H}} - V_{\text{CAN-L}}|$) is recorded as a scalar feature. As bursts accumulate, the Statistical Estimator computes the sample mean μ and standard deviation σ of the peak differential series and reports $\mu + 3\sigma$ as the upper bound. Under a Gaussian assumption this bound encloses 99.87% of the normal-traffic distribution. The calibration session on the test bed yielded $\mu + 3\sigma \approx 1876$ mV for the burst-peak CANH−CANL differential.

Threshold derivation. This burst-peak bound is directly used to set the per-sample threshold of the Voltage Rules Engine, giving the two detection rules:

- R_1 — **DIFF spike**: $V_{\text{diff}} > 1.876$ V for ≥ 1 sample.
- R_2 — **DIFF sustained**: $V_{\text{diff}} > 1.876$ V for ≥ 3 consecutive samples.

The operator may override the threshold at the command line; the value above is that used in the evaluation described in Chapter 6.

4.6. Anomaly Detection Rules

At runtime, the IDS applies two independent detection rules to every dominant-phase sample, monitoring a single physical quantity: the CANH−CANL differential voltage. The rules form a complementary pair: a *spike* rule (fires on a single anomalous sample) and a *sustained* rule (requires three consecutive anomalous samples).

Rule R_1 — DIFF spike. An alert is raised when $V_{\text{diff}} = V_{\text{CAN-H}} - V_{\text{CAN-L}} > 1.876$ V for at least one sample. A transient differential excursion above the calibrated upper bound indicates that the bus is being driven beyond the level produced by the legitimate CAN transceiver alone, which can result from an ERAM-style SPI injection momentarily

adding drive to an ongoing dominant bit. During the attack experiments, typical detected values ranged from +1.899 V (marginally above threshold) to +2.373 V (a clear anomaly), confirming that the attacker's transceiver produces a measurably different differential drive.

Rule R_2 — DIFF sustained. An alert is raised when $V_{\text{diff}} > 1.876$ V for three or more consecutive samples. The three-sample requirement (3 µs at 1 MSa/s, or 1.5 bit periods at 500 kbit/s) suppresses isolated noise transients while catching injections that sustain an elevated differential state across multiple sampling instants.

Rule state machine. Each rule within the Voltage Rules Engine maintains an independent violation counter that increments on each anomalous sample and resets to zero on any non-anomalous sample. The engine sets a per-rule *fired* flag on the first sample that satisfies the minimum-count criterion and clears it only when the signal returns below the threshold, ensuring that each contiguous excursion above the threshold produces exactly one fire event regardless of its duration. Rules are evaluated only during dominant phases (when V_{diff} exceeds the dominant threshold of 500 mV); on recessive samples the violation counters are reset, preventing spurious accumulation across bit boundaries.

4.7. Handling Edge Cases

ACK-bit pulses. Each correctly receiving node pulls the ACK slot dominant for approximately one bit time (2 µs at 500 kbit/s). During calibration, ACK-bit bursts are shorter than the minimum-burst-duration parameter (5 µs) and are therefore discarded by the burst detector, so they do not contribute to the statistical baseline. During detection, whether such pulses trigger R_1 depends on the drive strength of the receiving transceivers. Because ACK-bit pulses are excluded from the calibration data, the $\mu + 3\sigma$ bound reflects only full-frame dominant-bit peaks; the runtime threshold (1.876 V for the differential) may therefore not fully account for ACK-induced per-sample excursions, which is why a zero false-positive rate must be verified empirically in the baseline evaluation before deploying the IDS.

Single ECU during calibration. Calibration is performed with only the ECU whose frames are targeted by the attack active on the bus. Because only one node is transmitting, frame merging between multiple nodes cannot occur. The only merging risk arises if that ECU transmits at a rate high enough that the inter-frame gap falls below the idle-gap parameter (10 µs); at the typical test rate of approximately 10 frames/s the inter-frame gap exceeds 90 ms, so this is not a concern in practice.

Calibration in the presence of the attacker node. The calibration pipeline is run with the attacker node inactive. If the attacker node were active during calibration, its frames would contribute to the burst-peak statistics, potentially inflating the estimated mean and standard deviation and shifting the $\mu + 3\sigma$ bound upward, making the derived thresholds less sensitive. The correct procedure is to disconnect or silence the attacker node before running calibration.

Recalibration after hardware changes. If a node is replaced or its transceiver ages significantly, the calibration phase must be re-run to update the thresholds. The calibration pipeline is designed to complete in under two minutes on a live bus, so periodic recalibration is operationally feasible.

5 | Implementation

5.1. Hardware Platform

The laboratory test bed follows the hardware configuration described in [1]. It consists of two CAN nodes on a shared differential bus, monitored passively by an oscilloscope.

Legitimate traffic nodes. Two Arduino development boards, each equipped with a CAN transceiver module, act as the legitimate ECU simulators. Each board runs a firmware sketch that periodically transmits CAN data frames with a fixed identifier and synthetic payload, simulating two independent ECUs exchanging sensor data at different rates. The two nodes are assigned distinct CAN identifiers so that their traffic is distinguishable at the data-link layer, although the IDS does not exploit the identifier in its detection logic.

Attacker node. An STM32 Nucleo development board, equipped with a CAN transceiver, serves as the attack injector. The STM32 microcontroller on the Nucleo board integrates a bxCAN controller peripheral, which provides configurable bit timing, a programmable message transmit mailbox, and flexible identifier filtering. This setup allows precise control over the injection rate, identifier, and payload of attack frames. The Nucleo is connected to the same bus as the two Arduino nodes and injects frames independently, without any synchronisation with the legitimate traffic.

Signal acquisition. The bus voltage is captured by a **PicoScope 2205A MSO** oscilloscope, connected to the bus through two passive BNC probes (one on CAN-H, one on CAN-L). The PicoScope is interfaced to the host PC via USB. Key specifications relevant to this work are:

- Analogue bandwidth: 25 MHz.
- Maximum real-time sample rate: 500 MSa/s (two-channel mode).
- ADC resolution: 8 bits.
- Used sample rate in this work: $f_s = 1\,\text{MSa/s}$ (1 µs interval), via the oscilloscope

SDK's streaming interface.

- Input range: ±5 V (both channels), DC coupling.
- Voltage resolution at ±5 V range: ≈39 mV per ADC step.

Bus topology. All three nodes and the oscilloscope probes are connected to a short CAN bus terminated with 120 Ω resistors at both ends, consistent with the ISO 11898 requirement for a properly matched bus. The short wire length ensures that propagation delays between nodes are negligible (<1 ns) and that the oscilloscope probes observe the same voltage as all nodes simultaneously.

5.2. Software Stack

The software pipeline is split into two phases executed sequentially: an offline calibration phase that establishes the voltage baseline, and an online detection phase that monitors the bus in real time.

Arduino firmware. Each Arduino node runs a simple firmware sketch using a CAN-bus library. The sketch configures the on-board CAN module at 500 kbit/s and enters a loop that transmits a data frame with a node-specific identifier every few milliseconds, generating a steady stream of normal bus traffic.

Nucleo firmware. The STM32 Nucleo runs firmware built with the STM32 HAL library. The bxCAN peripheral is configured at 500 kbit/s with normal operational mode. The firmware accepts attack commands over the Nucleo's USB-serial interface, specifying the attack type, target identifier, and injection rate, and executes them on demand without any modification to the legitimate nodes.

Calibration Pipeline. Before the IDS is deployed, the Calibration Pipeline is run with only the single ECU whose frames are targeted by the attack active on the bus. The Streaming Acquisition module opens the oscilloscope in streaming mode at 1 MSa/s and delivers raw ADC sample blocks to the Burst Detector, which segments them into per-frame burst events. For each burst the peak CANH−CANL differential is extracted and forwarded to the Statistical Estimator, which maintains running mean and standard deviation. At the end of the session the Estimator outputs the Threshold Config: the $\mu + 3\sigma$ bound for each signal, which is then supplied to the Detection Pipeline as the rule-threshold configuration. A live display shows the raw waveforms, real-time histograms of the burst-peak distributions with μ and $\mu + 3\sigma$ markers, and a running statistics summary to help the operator verify convergence before finalising calibration.

Detection Pipeline. The Detection Pipeline opens the oscilloscope with the same hardware configuration as the calibration run and processes every acquired sample through two parallel paths. The Frame/Error Counter uses a vectorised run-based approach: NumPy identifies dominant/recessive transitions across each sample buffer, and Python iterates only over the resulting runs, feeding each run's polarity and length into a state machine that tracks SOF, error-flag, and EOF events. The Voltage Rules Engine evaluates rules R_1–R_2 (Section 4.6) sample-by-sample, but only on dominant-phase samples (approximately 0.2% of total samples on a 20 msg/s bus), so the full 1 MSa/s stream is consumed without sample loss. A multithreaded architecture separates the Streaming Acquisition callback — running in a dedicated background thread and draining blocks of up to 50 000 samples every 1 ms — from the Live Monitor display in the main thread, which refreshes every 500 ms. The Error-Flag Correlator emits a timestamped log entry for every SOF, rule fire, and error-flag event, classifying each error flag as confirmed or missed. The pipeline supports two evaluation modes: *baseline mode* (clean traffic, verifying zero false-positive rate) and *attack mode* (injection active, measuring error-flag detection rate). All detection thresholds are supplied as configuration parameters so that the values derived from calibration can be applied without modifying the pipeline internals.

5.3. Attack Injection Module

Attacks are injected by the Nucleo node, which participates in the CAN bus as a third node alongside the two Arduino nodes.

ERAM-model bus-off attack. For the bus-off experiment evaluated in Chapter 6, the Nucleo runs a dedicated attack firmware that uses the CANflict technique (Section 3.2). The Nucleo's SPI peripherals are redirected to the CAN-TX and CAN-RX pins, bypassing the bxCAN controller entirely and granting direct bit-level read and write access to the bus: SPI1 monitors bus bits in real time, and SPI2 injects dominant bits at arbitrary bit positions within ongoing frames. This is an ERAM-class attack; the controller's error-state machine, stuffing rules, and arbitration logic are completely circumvented.

Each attack session is recorded in the Alert Log produced by the Detection Pipeline, which includes timestamps, rule identifiers, measured voltage values, and threshold values. The known attack onset time (when the Nucleo firmware begins injecting) serves as the ground truth for computing detection latency in Chapter 6.

5.4. Dataset Collection

The dataset consists of two categories of acquisition sessions recorded on the laboratory test bed.

Baseline sessions (calibration). Several calibration runs were performed with only the target ECU (one Arduino node) active and the Nucleo silenced. Each run lasted 30 to 120 seconds and produced a sequence of burst-peak values from which the Statistical Estimator extracted the per-signal mean and standard deviation. The consistency of the threshold values across multiple runs was verified to confirm that the statistical baseline is stable and reproducible.

Attack sessions (evaluation). Multiple injection sessions were recorded with the Detection Pipeline running in attack mode. The Alert Log records the precise timestamps of all SOF events, rule fires, and error-flag detections with their correlation status (confirmed or missed), providing a complete audit trail of the IDS's decision process.

The raw data consists of the mV time series acquired by the PicoScope (indirectly, through the alert log and threshold comparisons performed at runtime by the IDS) and the corresponding alert logs. The dataset is intentionally compact, consistent with the proof-of-concept scope of this work. The experimental results and an analysis of the system's limitations on this small-scale test bed are reported in Chapters 6 and 7.

6 | Experiments and Results

6.1. Experimental Setup

The experiments were conducted on the laboratory test bed described in Chapter 5. The hardware configuration was unchanged: two Arduino nodes transmitting normal CAN traffic at 500 kbit/s, a PicoScope 2205A MSO acquiring both CAN-H and CAN-L at 1 MSa/s, and the Detection Pipeline running on the host PC with the detection thresholds derived from calibration:

- R_1 (DIFF spike): $V_{\text{diff}} > 1.876\,\text{V}$ for $\geq$1 sample;
- R_2 (DIFF sustained): $V_{\text{diff}} > 1.876\,\text{V}$ for $\geq$3 consecutive samples.

The Frame/Error Counter was configured with a dominant threshold of 500 mV, a samples-per-bit ratio of 2 (for 500 kbit/s at 1 MSa/s), and an error-flag minimum of 11 dominant samples (>5.5 bit-times), consistent with the CAN specification that an active error flag consists of 6 dominant bits. The lookback window for error-flag correlation was set to 500 samples (500 µs).

The only attack class evaluated experimentally in this work is the bus-off attack. The attacker firmware on the STM32 Nucleo board implements an ERAM-class bus-off attack via the CANflict technique, as described in Section 5.3. Concretely, the firmware monitors the bus through SPI1 (configured as a slave receiver whose clock and data lines are physically shared with the CAN-RX pin), identifies each frame carrying the target identifier (0x48C), and immediately injects 6 consecutive dominant bits via SPI2 (whose data output is shared with the CAN-TX pin). The 6-bit dominant injection is delivered directly onto the bus, bypassing the bxCAN controller: because it occurs during the victim's ongoing frame transmission, it introduces a stuff violation or bit error, and the victim's CAN controller increments its TEC by 8.

An FSM in the Nucleo firmware monitors the disappearance of the 0x48C identifier from the bus: if the identifier is absent for more than 20 consecutive frame periods, the firmware transitions to a waiting state; if it remains absent for more than twice the expected period,

bus-off is confirmed. The Nucleo reports the attack progression and the total injection count over its UART interface at 115200 baud.

The IDS calibration phase was run with only the target ECU (one Arduino node) active, establishing clean voltage baselines. Four evaluation sessions were then recorded with both Arduino nodes active:

1. A *baseline session* (126.1 s, 65 399 810 samples) with the Nucleo silenced, to verify zero false positives.
2. A *sustained attack session* (127.4 s, 64 549 827 samples) with the Nucleo continuously injecting, producing 575 error flags over the full recording.
3. Two *dedicated bus-off sessions* (69.5 s and 45.1 s) in which the Nucleo drove the victim to bus-off, producing 16 and 19 error flags respectively.

Detection performance was quantified using error-flag correlation: for each error flag detected by the Frame/Error Counter, the Error-Flag Correlator checks whether at least one voltage rule (R_1 or R_2) fired within the preceding 500-sample lookback window. The detection rate is the ratio of confirmed error flags to total error flags.

6.2. Performance Metrics

Given the limited scope of this prototype evaluation (a single attack class, four sessions), the following metrics are reported.

Error-flag detection rate. The fraction of error flags detected by the Frame/Error Counter for which at least one voltage rule (R_1 or R_2) fired within the preceding lookback window. This is the primary sensitivity metric: it measures how reliably the IDS can correlate a protocol-level error event with a preceding voltage-level anomaly. The metric is reported both globally (across all error flags in a session) and per-rule (the fraction of error flags for which each individual rule contributed a confirmation).

False positive rate. The number of rule fires and error flags during the clean baseline session. A zero count on both confirms that the statistical thresholds pass all legitimate traffic without spurious alerts and that normal CAN traffic produces no error flags.

Per-rule fire statistics. For each rule, three counts are reported: *total fires* (the total number of contiguous threshold excursions), *distinct frames* (the number of distinct CAN frames in which at least one fire occurred), and *confirmed* (the number of error flags for which the rule contributed a confirmation). These statistics characterise the rule's sensitivity and specificity independently of the other rule.

Frame-level classification. At each End of Frame, the completed frame is classified as true positive (TP), false negative (FN), false positive (FP), or true negative (TN), as described in Section 4.2. Example voltage traces for each category are captured automatically and presented in Section 6.3.

More advanced metrics (precision, recall, AUC-ROC, per-class F1 score) require ground-truth labelled datasets covering multiple attack classes and attack-free periods, which are not available for this prototype. Their evaluation is identified as future work (Chapter 7).

6.3. Attack-Scenario Results

Attack mechanics. The attacker's firmware targets frames with CAN identifier 0x48C. Each time a frame with this identifier is detected, the firmware injects 6 consecutive dominant bits via SPI2, bypassing the bxCAN controller. Each injection causes a stuff violation or bit error in the victim's frame, incrementing the victim's TEC by 8. Because the victim also decrements its TEC by 1 on each successfully completed transmission, the net rate of TEC increase depends on the relative frequency of attacker injections versus successful victim transmissions. In practice, approximately 40 injections are required to drive TEC above 255 and achieve bus-off.

Why individual injections sometimes evade the IDS. Each 6-bit dominant injection is delivered via the Nucleo's CAN transceiver (SPI2 output routed to the CAN-TX pin). The transceiver drives a differential voltage close to the 1.876 V threshold calibrated from the victim's transmissions alone. Whether a specific injection triggers R_1 (DIFF spike, $\geq$1 sample above 1.876 V) depends on the instantaneous bus-drive conditions at sample time: when the injected dominant bits overlap with dominant bits simultaneously driven by the victim, the combined bus drive elevates the differential above the calibrated single-transceiver baseline; in other cases the voltage remains within the calibrated range and no alert fires. The ADC quantisation step of $\approx$39 mV near the threshold means that borderline voltage excursions are resolved stochastically across injection events. Figure A.4 shows an example waveform of a missed injection, where the differential remains within the calibrated range despite the error flag.

Mechanism of detected alerts. Rule R_1 (DIFF spike) fires when $V_{\text{diff}} > 1.876$ V for at least one sample, and R_2 (DIFF sustained) fires when the elevated level persists for $\geq$3 consecutive samples. During the attack experiments, detected differential values ranged from +1.899 V (marginally above threshold) to +2.373 V. When an error flag is detected by the Frame/Error Counter (a dominant run exceeding 5.5 bit-times), the Error-Flag Correlator checks whether either rule fired within the preceding 500-sample lookback

window. If so, the error flag is classified as *confirmed*; otherwise as *missed*. Figure A.1 shows an example of a confirmed error flag, where the differential clearly exceeds the threshold before the error flag.

Baseline results. The baseline session (126.1 s, 65 399 810 samples, 1315 frames) with the Nucleo silenced produced zero rule fires and zero error frames, confirming that: (i) the calibration-derived thresholds pass all legitimate two-ECU traffic without spurious alerts; (ii) the Frame/Error Counter does not misidentify normal dominant runs as error flags. The frame rate of 10.4 frames/s is consistent with two ECUs transmitting at approximately 10 Hz each, with roughly half the frames detected per acquisition buffer cycle.

Attack results. The evaluation results for all attack sessions are summarised in Table 6.1.

Table 6.1: Error-flag correlation results (ERAM bus-off via CANflict, 500 kbit/s).

Metric	**Sustained**	**Bus-off 1**	**Bus-off 2**
Duration (s)	127.4	69.5	45.1
Samples	64 549 827	39 700 000	24 000 000
Total frames	1943	548	385
Normal frames	1368	532	366
Error frames	575	16	19
Error rate	29.6%	2.9%	4.9%
Frames/s	15.3	7.9	8.5
Error flags	575	16	19
Confirmed	515	16	19
Missed	60	0	0
Detection rate	**89.6%**	**100%**	**100%**
R_1 fires / frames / confirmed	1228 / 680 / 515	33 / 18 / 16	35 / 19 / 19
R_2 fires / frames / confirmed	539 / 422 / 375	19 / 16 / 14	22 / 15 / 15
R_1 per-flag rate	89.6%	100%	100%
R_2 per-flag rate	65.2%	87.5%	78.9%

Example voltage traces. The Detection Pipeline can automatically capture one representative voltage trace per classification category (TP, TN, FP, FN) during an evaluation session. These traces — snapshots of the CAN-H, CAN-L, and differential waveforms surrounding the classified frame — provide visual confirmation of the IDS's detection mechanism and are presented in Appendix A. Figure A.1 shows a confirmed error flag (true positive), where the differential clearly exceeds the threshold before the error flag. Figure A.4 shows a missed injection (false negative), where the differential remains within the calibrated range despite the error flag — this explains the mechanism behind the 10.4% missed rate in the sustained session. Figures A.2 and A.3 show the corresponding

normal-frame categories.

Discussion. The dedicated bus-off sessions achieved 100% error-flag detection rates (16/16 and 19/19), demonstrating that when the attacker's injections are concentrated and the bus conditions consistently produce overlapping dominant states, every error flag is preceded by a detectable voltage anomaly.

The sustained attack session, with 575 error flags over 127.4 s, achieved an overall detection rate of 89.6% (515 confirmed, 60 missed). The 60 missed error flags (10.4%) correspond to injections where the attacker's differential contribution did not exceed the $\mu + 3\sigma$ threshold within the 500-sample lookback window. This can occur when the injection happens during a recessive phase (where the differential is near zero) and the attacker's transient dominant pulse is too brief to be sampled above threshold at 1 MSa/s.

Rule R_1 (DIFF spike) is the primary detection mechanism: it confirmed 89.6% of error flags in the sustained session and 100% in both bus-off sessions. Rule R_2 (DIFF sustained, $\geq$3 consecutive samples) is more restrictive and confirmed 65.2–87.5% of error flags, but does not contribute additional detections beyond R_1 in any session. This is expected: a single-sample spike is sufficient to confirm an error flag, and the sustained criterion only adds value when the anomaly persists for multiple microseconds.

The baseline session (126.1 s, 65 399 810 samples, 1315 frames) produced zero rule fires and zero error frames across both rules, confirming that the calibration-derived threshold of 1.876 V passes all legitimate two-ECU traffic without spurious alerts.

False positives during attack sessions. During the sustained attack session, the R_1 rule fired in 680 distinct frames, substantially more than the 575 error frames detected. Since the baseline session produced zero rule fires, these additional fires in normal frames are not caused by calibration error or noise. Instead, they arise from attack attempts that the IDS correctly detected at the voltage level: the attacker's dominant-bit injection via SPI2 raises the bus differential above the $\mu + 3\sigma$ threshold, but the injection does not always succeed in producing a bit or stuff error in the victim's CAN controller. In these cases the frame completes normally despite the attack attempt, and the correlator classifies it as a false positive. From a security perspective, these events are informative rather than spurious — they indicate ongoing attack activity even when the individual injection did not succeed at the protocol level.

6.4. Computational Cost and Deployment Feasibility

The runtime detection pipeline processes one sample per microsecond. The Frame/Error Counter uses a vectorised run-based approach: NumPy identifies dominant/recessive transitions across each buffer, and Python iterates only over runs (not individual samples). The Voltage Rules Engine evaluates two scalar comparisons and maintains two hysteresis counters, but runs only on dominant-phase samples (approximately 0.2% of total samples on a 20 msg/s bus), making the per-buffer CPU cost negligible. On the host PC, the practical bottleneck is the USB transfer cycle from the PicoScope: the Streaming Acquisition module polls every 1 ms and drains a 50 000-sample ring buffer that holds up to 50 ms of bus data, delivering ≈1 000 new samples per poll at 1 MSa/s. Per-batch processing time is well under 0.5 ms on a modern laptop, so the pipeline has zero risk of falling behind the acquisition rate.

The end-to-end detection latency from the first anomalous sample to a logged alert is bounded by the USB polling interval (approximately 1 ms) plus the sustained-rule confirmation window (≥3 samples, i.e. 3 µs at 1 MSa/s); in practice, it is dominated by the 1 ms polling period. For a 500 kbit/s bus this corresponds to approximately 50–100 CAN frames, which is acceptable for a proof-of-concept monitor but would need to be reduced for a production deployment where faster response is required.

For deployment in a production vehicle the USB/PC acquisition architecture would need to be replaced by an embedded front-end. The detection logic — two scalar comparisons, two integer counters, and a run-length state machine per sample — is trivially implementable in firmware on a low-cost microcontroller. The main engineering challenge is achieving 1 MSa/s throughput at adequate voltage resolution: the PicoScope's 8-bit ADC at ±5 V provides a ≈39 mV quantisation step, which is sufficient to resolve the threshold level with comfortable margin. Many embedded ADCs achieve comparable specifications; alternatively, a dedicated CAN physical-layer monitor IC (combining an analogue front-end with a microcontroller) could integrate acquisition and detection in a single chip, reducing cost and form factor to levels compatible with series automotive production.

7 | Conclusions and Future Developments

7.1. Summary of Contributions

This thesis has made three main contributions toward a physical-layer IDS capable of detecting attacks launched by an ERAM-class attacker.

1. Threat analysis of the ERAM model for physical-layer IDSs. Chapter 3 provided a systematic analysis of how the Enhanced Remote Attacker Model undermines existing physical-layer defences. Three classes of vulnerability were identified: (i) fingerprint corruption, in which an attacker injects pulses that gradually poison the IDS's voltage model without triggering detectable errors; (ii) partial-frame attacks, in which frame hijacking leaves the authenticated portion of a frame intact while modifying the payload fields that follow; and (iii) short-pulse attacks (synchronisation disruption, arbitration denial), which do not produce complete frames and are therefore invisible to IDSs that process only assembled frames. This analysis clarifies which aspects of the ERAM threat envelope are and are not addressable by a voltage-threshold approach.

2. IDS prototype based on error-flag-correlated voltage monitoring. Chapters 4 and 5 presented a lightweight, transparent anomaly-detection system that operates on raw 1 MSa/s voltage samples without requiring per-node fingerprinting models. The prototype was developed and evaluated on the laboratory test bed introduced by Azzarà [1], which provided the hardware infrastructure (Arduino ECU simulators, STM32 Nucleo attacker node, PicoScope 2205A MSO oscilloscope) reused in this work. Calibration uses a single-ECU burst-peak analysis to derive a $\mu + 3\sigma$ threshold for the CANH−CANL differential; two detection rules (R_1–R_2) enforce this threshold with single-sample or three-consecutive-sample confirmation, and a protocol-aware Frame/Error Counter identifies CAN error flags in real time for correlation with voltage anomalies. The calibration procedure is non-parametric and requires no knowledge of the attack type, making it applicable to any new deployment without retraining.

3. Experimental evaluation of an ERAM bus-off attack. Chapter 6 evaluated the IDS against an ERAM-class bus-off attack implemented via the CANflict technique, targeting CAN identifier 0x48C. A baseline session (126.1 s, 65 399 810 samples, 1315 frames) confirmed zero false positives and zero rule fires, validating the calibration-derived threshold against clean traffic. Two dedicated bus-off sessions achieved 100% error-flag detection rates (16/16 and 19/19 confirmed). A sustained attack session (127.4 s, 64 549 827 samples, 575 error flags) achieved an overall detection rate of 89.6% (515/575 confirmed), with the spike rule R_1 as the primary detection mechanism.

7.2. Limitations

Oscilloscope dependency. The acquisition pipeline is built around a PicoScope 2205A MSO oscilloscope connected via USB to a host PC. The oscilloscope is a benchtop laboratory instrument whose retail cost is substantially higher than what automotive OEMs accept for any per-vehicle monitoring component. Beyond cost, the form factor and PC dependency make this architecture fundamentally incompatible with mass production: a running vehicle cannot accommodate an oscilloscope and a laptop as permanent components of its electronic system. This is the primary barrier between the proof-of-concept prototype and a deployable product.

Limited test bed scope. All experiments were conducted on a minimal bench (two Arduino ECU nodes and one Nucleo attacker) under controlled laboratory conditions. A real in-vehicle CAN network may comprise 50–100 ECUs across multiple bus segments, several metres of twisted-pair wiring, and realistic traffic loads. The bus voltage distribution on such a network — and therefore the calibration-derived thresholds — may differ significantly from those observed in the laboratory. These results should be treated as proof-of-concept evidence rather than production-validated performance figures.

Single attack class evaluated. The experimental evaluation covered only the ERAM bus-off attack. Detection behaviour for fabrication, masquerade, replay, fuzzing, and ERAM-specific attacks (frame hijacking, Janus frames, synchronisation disruption) was analysed qualitatively only.

Missed error flags in sustained attack. The 89.6% detection rate in the sustained attack session means that 10.4% of error flags (60 out of 575) were not preceded by a detectable voltage anomaly within the lookback window. The system achieves 100% detection on dedicated bus-off sessions, but cannot guarantee detection of every individual injection event in a sustained, high-rate attack scenario.

Static thresholds. The calibrated thresholds are fixed after the calibration session. Temperature variation (the automotive operating range is −40 °C to +85 °C), battery voltage fluctuation (9–16 V), and transceiver ageing can shift the bus voltage distribution over time, potentially increasing the false-positive rate or degrading the detection rate without any explicit recalibration trigger. All experiments in this work were conducted at room temperature with a fixed, low bus load (approximately 10–20% utilisation); the system's behaviour under varying temperature, higher bus loads (50%–90% utilisation), or different sampling rates remains an open question.

No systematic comparison with other IDS approaches. The evaluation measures the proposed system's detection rate in isolation. A quantitative comparison against timing-based IDSs (e.g., CIDS [5]) and payload-based IDSs (e.g., CANnolo [11]) on the same attack scenarios was not performed, so the relative advantage of the physical-layer approach over existing defences for the ERAM bus-off attack is not empirically established. Such a comparison would require implementing or adapting reference IDSs to the same test bed, which is beyond the scope of this proof-of-concept work.

7.3. Future Developments

Multi-condition calibration with all ECUs active. The current calibration procedure runs with only the single target ECU transmitting. A more robust approach would run calibration with all ECUs simultaneously active — as they would be in normal vehicle operation — and across the full range of operating conditions: ambient temperatures spanning the automotive range, battery voltage levels from 9 V to 16 V, and bus utilisation from low (<10%) to high (>80%). Repeated calibration sessions under these conditions would reveal how much the burst-peak distributions drift with the vehicle's state and enable condition-aware threshold adaptation. If the drift is systematic and monotone (e.g., dominant-bit voltage rises linearly with battery voltage), a simple compensation function could maintain the $\mu + 3\sigma$ bound without requiring a full recalibration. This would make the IDS substantially more robust to the conditions encountered during the full vehicle lifecycle.

Embedded implementation on low-cost hardware. Replacing the PicoScope and host PC with a dedicated embedded monitor is the critical step toward production deployability. The STM32 Nucleo board — the same platform used as the ERAM attacker in this work — is a natural prototype vehicle for such an embedded IDS: STM32 devices integrate 12-bit ADCs capable of sampling rates of 5–18 MSa/s (device-dependent), which comfortably exceeds the 1 MSa/s used here, and the ARM Cortex-M4/M7 core

can execute the two-rule detection logic (two scalar comparisons, two integer counters, and a run-length state machine per sample) at negligible CPU cost. The main open engineering challenge is the analogue front-end: a suitable differential amplifier circuit to interface directly with the CAN-H/CAN-L pair and scale the ≈0–5 V differential range into the ADC's input range at adequate resolution. A fully embedded monitor would reduce the hardware cost to a few tens of euros per unit, eliminate the PC and oscilloscope dependency, and enable integration as a dedicated passive node on the vehicle's CAN bus.

Extension to multiple attack classes and ablation study. The next evaluation step is to implement and test the remaining attack classes — fabrication, masquerade, fuzzing, and dominant flooding — on the test bed, quantify the per-class false-positive and true-positive rates of each rule, and conduct a formal ablation study to determine the individual contribution of R_1 and R_2. This would also determine whether additional rules on the CANH signal (rather than the differential alone) provide complementary detection capability for attack classes beyond bus-off.

Evaluation on a more realistic test bed. The two-node bench should be extended toward a network more representative of a real vehicle, following the requirements framework defined by Azzarà [1] for objective CAN IDS benchmarking. Adding more ECU nodes, longer bus wiring, and realistic multi-identifier traffic loads would stress-test both the calibration procedure and the detection rules, and would reveal whether the voltage anomaly signatures observed in the two-node setting generalise as the network grows.

Defence-in-depth combination with higher-layer IDSs. The physical-layer approach is complementary to payload-based IDSs (e.g., CANnolo [11]) and timing-based IDSs (e.g., CIDS [5]): voltage monitoring detects attacks invisible to higher layers (ERAM injection, bit-level manipulation), while higher-layer approaches detect semantic attacks invisible to voltage monitoring (Janus frames, counterfeit frames with identical voltage profiles). A defence-in-depth architecture that combines all three layers would address a substantially broader fraction of the ERAM threat envelope than any single layer alone.

Bibliography

[1] D. Azzarà. CANdemonium: An evaluation test bed for objective CAN intrusion detection systems benchmarking. Master's thesis, Politecnico di Milano, 2024.

[2] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu. Evading voltage-based intrusion detection on automotive CAN. In *28th Annual Network and Distributed System Security Symposium*, NDSS '21, 2021. doi: 10.14722/ndss.2021.23013. URL `https://www.ndss-symposium.org/ndss-paper/evading-voltage-based-intrusion-detection-on-automotive-can/`. Introduces the DUET dual-ECU fingerprint-poisoning technique.

[3] G. Bloom. WeepingCAN: A stealthy CAN bus-off attack. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec), co-located with NDSS 2021*, AutoSec '21, 2021. doi: 10.14722/autosec.2021.23002. URL `https://www.ndss-symposium.org/ndss-paper/auto-draft-102/`.

[4] K.-T. Cho and K. G. Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1044–1055. ACM, 2016. doi: 10.1145/2976749.2978302. URL `https://dl.acm.org/doi/10.1145/2976749.2978302`.

[5] K.-T. Cho and K. G. Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th USENIX Security Symposium*, USENIX Security '16, pages 911–927. USENIX Association, 2016. URL `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho`.

[6] K.-T. Cho and K. G. Shin. Viden: Attacker identification on in-vehicle networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17. ACM, 2017. doi: 10.1145/3133956.3134001. URL `https://dl.acm.org/doi/10.1145/3133956.3134001`.

[7] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee. VoltageIDS: Low-level communication characteristics for automotive intrusion detection system. *IEEE Trans-*

actions on Information Forensics and Security, 13(8):2114–2129, 2018. doi: 10.1109/TIFS.2018.2812149. URL https://ieeexplore.ieee.org/document/8306904.

[8] A. De Faveri Tron, S. Longari, M. Carminati, M. Polino, and S. Zanero. CANflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pages 711–723. ACM, 2022. doi: 10.1145/3548606.3560618. URL https://dl.acm.org/doi/10.1145/3548606.3560618.

[9] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem. SIMPLE: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pages 229–244. ACM, 2019. doi: 10.1145/3359789.3359834. URL https://dl.acm.org/doi/10.1145/3359789.3359834.

[10] S. Kulandaivel, S. Jain, J. Guajardo, and V. Sekar. CANNON: Reliable and stealthy remote shutdown attacks via unaltered automotive microcontrollers. In *2021 IEEE Symposium on Security and Privacy*, SP '21, pages 195–210. IEEE, 2021. doi: 10.1109/SP40001.2021.00122. URL https://ieeexplore.ieee.org/document/9519391.

[11] S. Longari, D. H. Nova Valcarcel, M. Zago, M. Carminati, and S. Zanero. CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network. *IEEE Transactions on Network and Service Management*, 18(2):1913–1924, 2021. doi: 10.1109/TNSM.2020.3038991. URL https://ieeexplore.ieee.org/document/9262960.

[12] S. Longari, C. A. Pozzoli, A. Nichelini, M. Carminati, and S. Zanero. CANdito: Improving payload-based detection of attacks on controller area networks. In *Cyber Security, Cryptology, and Machine Learning (CSCML 2023)*, volume 13914 of *Lecture Notes in Computer Science*. Springer, 2023. doi: 10.1007/978-3-031-34671-2_10. URL https://link.springer.com/chapter/10.1007/978-3-031-34671-2_10.

[13] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. Black Hat USA 2015 technical white paper, 2015. URL https://illmatics.com/Remote%20Car%20Hacking.pdf. Presented at Black Hat USA and DEF CON 23, Las Vegas, NV, August 2015.

[14] Robert Bosch GmbH. *CAN Specification Version 2.0*. Robert Bosch GmbH, Stuttgart, Germany, 1991. URL https://www.tekeye.uk/downloads/can2spec.pdf.

[15] M. Roeschlin, G. Camurati, P. Brunner, M. Singh, and S. Capkun. EdgeTDC: On the security of time difference of arrival measurements in CAN bus systems. In *30th Annual Network and Distributed System Security Symposium*, NDSS '23. Internet Society, 2023. doi: 10.14722/ndss.2023.24271.

[16] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran. Cloaking the clock: Emulating clock skew in controller area networks. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '18, pages 32–42. ACM/IEEE, 2018. URL `https://labs.ece.uw.edu/nsl/papers/sang_iccps2018.pdf`.

[17] Z. Tang, K. Serag, Z. B. Celik, S. Zonouz, D. Xu, and R. Beyah. ERACAN: Defending against an emerging CAN threat model. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, CCS '24. ACM, 2024. doi: 10.1145/3658644.3690267. URL `https://dl.acm.org/doi/10.1145/3658644.3690267`. Distinguished Paper Award.

[18] Tencent Keen Security Lab. Car hacking research: Remote attack Tesla cars. Technical blog post, Tencent Keen Security Lab, 2016.

[19] K. Tindell. Three new CAN protocol hacks. Canis Automotive Labs technical blog, 2020. URL `https://kentindell.github.io/2020/01/20/new-can-hacks/`.

[20] K. Tindell. The Janus attack. Canis Automotive Labs technical blog, 2021. URL `https://kentindell.github.io/2021/07/15/janus-attack/`.

A | Raw Voltage Waveform Examples

This appendix presents representative voltage traces captured automatically by the Detection Pipeline for each frame-level classification category. Each trace shows the CAN-H, CAN-L, and differential ($V_{\text{CAN-H}} - V_{\text{CAN-L}}$) waveforms surrounding the classified frame. The horizontal dashed line marks the 1.876 V detection threshold.

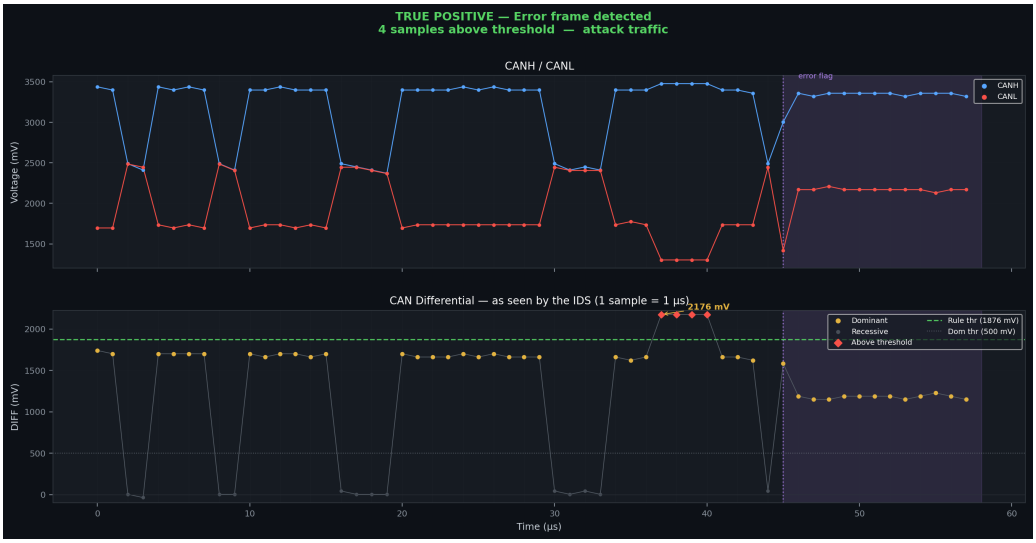

Figure A.1: Example trace: **True Positive** — an error frame preceded by a differential voltage excursion above the 1.876 V threshold. The voltage rule fired within the lookback window, and the error flag was classified as confirmed.

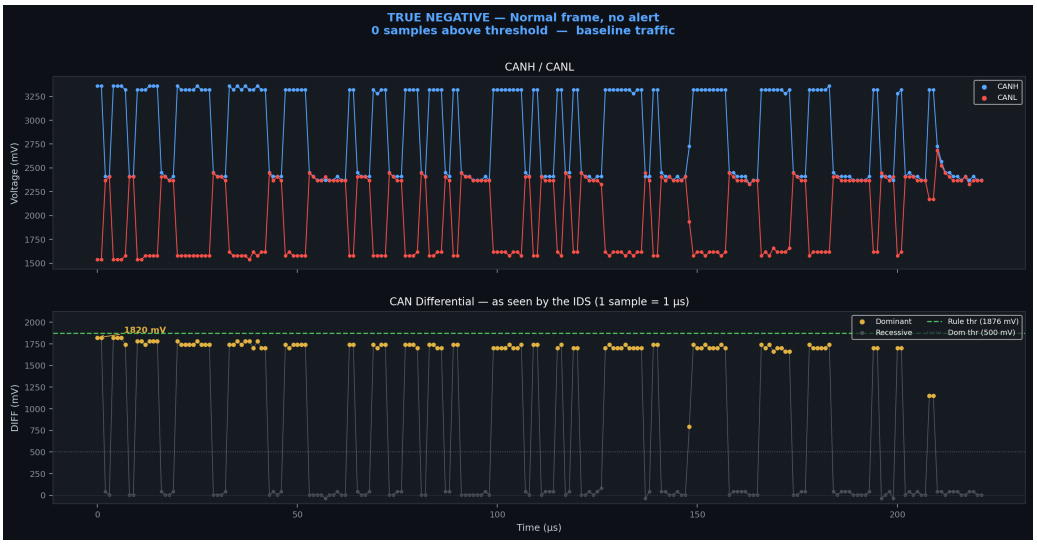

Figure A.2: Example trace: **True Negative** — a normal frame with no voltage anomaly. The differential remains within the calibrated baseline throughout the frame, and no rule fires.

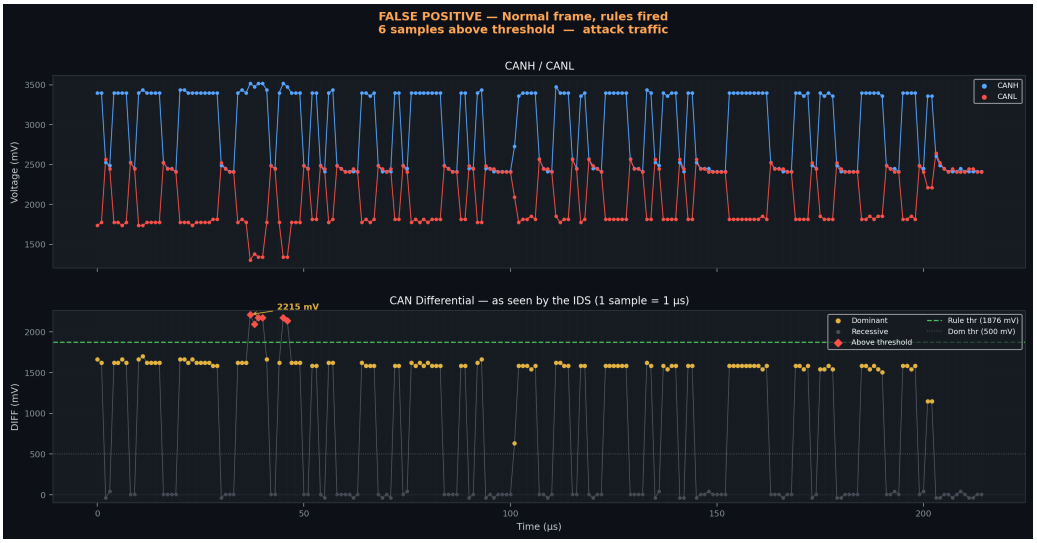

Figure A.3: Example trace: **False Positive** — a normal frame in which a voltage rule fired (the differential momentarily exceeded 1.876 V) but no error flag was detected. These events occur during an active attack: the attacker injects dominant bits attempting to cause a bit or stuff error, and the injection raises the bus voltage above the detection threshold, but fails to corrupt the frame — the victim's CAN controller completes it normally. The baseline session produced zero false positives, confirming that legitimate traffic alone does not trigger the voltage rules.

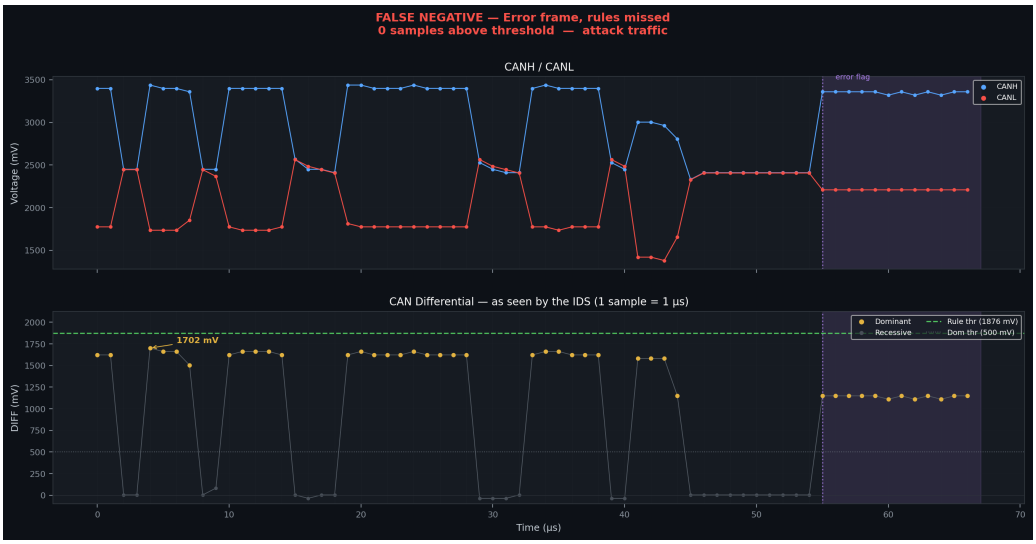

Figure A.4: Example trace: **False Negative** — an error frame not preceded by a voltage anomaly within the lookback window. The differential voltage during the attacker's injection remains at or below the 1.876 V threshold: this occurs when the attacker's dominant pulse does not overlap with the victim's own dominant bits, so the bus is driven by a single transceiver rather than two in superposition, and the resulting differential stays within the calibrated normal range. The error flag (a sustained dominant run >5.5 bit-times) is still detected by the Frame/Error Counter, but no corresponding voltage rule fire precedes it.

List of Symbols

Symbol	Description	Unit
V_{diff}	Differential bus voltage ($V_{\text{CAN-H}} - V_{\text{CAN-L}}$)	V
μ	Mean of the burst-peak differential distribution	mV
σ	Standard deviation of the burst-peak differential distribution	mV
R_1	Detection rule: DIFF spike ($\geq$1 sample above threshold)	–
R_2	Detection rule: DIFF sustained ($\geq$3 consecutive samples)	–
f_s	ADC sampling frequency	Sa/s
T_{bit}	CAN nominal bit time	μs
TEC	Transmit Error Counter	–
REC	Receive Error Counter	–

Acknowledgements