A Holistic Approach to Power Efficiency in a Clock Offset Based Intrusion Detection Systems for Controller Area Networks

Subir Halder[a,*], Mauro Conti[a], Sajal K. Das[b]

[a]*Department of Mathematics, University of Padua, Padua 35121, Italy*
[b]*Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409, USA*

Abstract

Controller Area Network (CAN) is an in-vehicle communication protocol, which provides an efficient and reliable communication link between Electronic Control Units (ECUs) in real time. Recent studies have shown that attackers can take remote control of the targeted vehicle by exploiting the vulnerabilities of the CAN protocol. Motivated by this fact, we propose an Intrusion Detection System (IDS), called Clock Offset-based IDS (COIDS), to monitor in-vehicle network activities to detect any intrusion. Precisely, COIDS measures and then exploits the clock offset of transmitter ECU's clock for fingerprinting ECU. COIDS next leverages the derived fingerprints to construct a baseline of ECU's normal clock behavior using an active learning technique. Based on the baseline of normal behavior, COIDS uses the Cumulative Sum method to detect any abnormal deviation in clock offset. Further, COIDS uses sequential change-point detection technique to determine the exact time of intrusion. Generally, COIDS has to run on every ECU to monitor the network behavior. This can be a significant power overhead for a hardware-constrained ECU. Thus, we next develop a cooperative game model to optimize the active time duration of COIDS in an ECU. We performed exhaustive experiments on real world publicly available datasets primarily to assess the effectiveness of COIDS against various in-vehicle network attacks. Our results show that COIDS detects intrusions faster than the best performed IDS in the state-of-the-art. Further, the results show that our designed cooperative game model significantly reduces the power overhead of the ECU without compromising the performance.

Keywords: Clock Offset, Clock Skew, Cooperative Game, Controller Area Network, Cumulative Sum Method, Intrusion Detection Systems

1. Introduction

In recent years, we have been witnessing a significant transformation of the automotive industry. Almost every next day, new advanced functions and features are added into the modern vehicles, which make them not only safe, but also connected, smart and intelligent [1]. However, as modern vehicles have become more connected, security has become an important factor for real concern [2]. Recently, researchers [3, 4, 5] have analyzed the remote exploitation technique using different attack vectors (e.g., Cellular, Bluetooth) and showed that in-vehicle Electronics Control Units (ECUs) can be compromised for remote cyber attacks. The cyber attacker can control the vehicle by injecting packets in the in-vehicle network through the compromised ECU. Recently, Miller and Valasek [6] have been able to hack and remotely stop a Jeep Cherokee on a highway, which triggered a recall of 1.4 million vehicles by the Chrysler automobile company. More recently, Nie et al. [7] have been able to compromise and remotely gain control of a Tesla Model S vehicle, which triggered

*Corresponding author.
Email addresses: subir.halder@math.unipd.it (Subir Halder), conti@math.unipd.it (Mauro Conti), sdas@mst.edu (Sajal K. Das)

the introduction of a code signing protection into vehicles by the Tesla Motors company. These incidents of remote cyber attacks on vehicles have made automobile security as one of the most vital issues [8]. The security of modern vehicle is a challenging job mainly due to complexity, numerous attack surfaces, and unsafe and old technologies.

Today's road vehicles are more intelligent than ever before. Automobile manufacturers are embedding several ECUs to enhance many safety and comfort relevant functionalities such as braking, steering and traction controls. All these ECUs continuously exchange messages through in-vehicle Controller Area Network (CAN) protocol. In almost every country, all recently sold vehicles implement the CAN protocol as one of the on-board diagnostics signal protocols. In spite of the widespread popularity and high reliability of the CAN protocol, from security perspective, the major problem of the CAN protocol is the lack of message authentication. Recently, Mazloom et al. [9] have shown that attackers can take control of the targeted vehicle remotely by exploiting the vulnerabilities of the CAN protocol. As a defensive mechanism against such attacks on CAN, mainly, two types of defensive mechanisms are followed: (i) message authentication [10, 11], and (ii) intrusion detection [12, 13]. Although, message authentication provides a level of security, however, due to limited space available for adding a Message Authentication Code (MAC), e.g., HMAC with SHA256 in a CAN message, hinders its applicability in CAN protocol. In contrast, Intrusion Detection Systems (IDSs) are drawing attention as a promising technique to detect suspicious behaviors on the in-vehicle CAN, as IDS provides security without generating computational overhead in the CAN protocol [14].

In the recent past, several types of state-of-the-art IDSs were proposed [12, 13, 15, 16, 17]. The main philosophy of these IDSs is to monitor the physical invariants, e.g., message contents, message periodicity, voltage distribution of the systems, and validate whether there is any significant deviation in them. It is worth mentioning that ECUs generally transmit messages of fixed length and at fixed periodicity, and the message contents do not vary drastically over time. However, there are still some critical attacks, e.g., fuzzy attack, impersonation attack, where existing IDSs fail in detection or prevention. The possible reasons for this inefficiency is: (a) CAN messages do not carry transmitter information, and hence, it is difficult to tell whether a message has originated from a genuine transmitter or not; and (b) lack of transmitter's information makes the job of an IDS nearly impossible to detect, which ECU has launched an attack.

Contribution. To protect against various vehicle attacks, we propose a novel anomaly-based IDS, called Clock Offset-based IDS (COIDS). COIDS monitors the interval of periodic messages, and then exploits them to estimate the clock offsets of transmitter ECUs' clock, which are then used to fingerprint the transmitter ECU. Unlike the existing state-of-the-art IDSs, where clock skew is used as a fingerprint, COIDS exploits clock offset for fingerprinting transmitter ECUs. This makes COIDS invulnerable to adversaries who can manipulate inter-departure times of messages. Based on the extracted fingerprints from the message periodicity, COIDS first constructs a baseline of ECUs' normal clock behavior model using an active learning technique [18]. COIDS then uses the Cumulative Sum (CUSUM) method [19] to derive the cumulative sums of deviations from the baseline of normal behavior for detecting adversaries. Finally, COIDS uses the Sequential Change-Point Detection (SCPD) technique to determine the exact time of attack. This allows COIDS to detect not only standard attacks that are discussed in existing literatures, but also those that are more intelligent and cannot be detected by existing IDSs, e.g., impersonation attack. Further, the use of SCPD algorithm enables COIDS to detect an anomaly in real time. As an ECU may have limited hardware resource, thus, running COIDS incessantly on every ECU may be a significant power overhead. To overcome this challenge, we propose a probabilistic model to minimize the active duration of the COIDS in the ECUs. We performed extensive simulation on publicly available real CAN traffic traces. Our results demonstrate that COIDS can detect three most potential attacks on CAN, i.e., Denial of Service (DoS), impersonation

and fuzzy attacks almost in real time. The results also show that our designed cooperative game model reduces power consumption considerably by optimizing the active time of the COIDS without compromising its performance.

Organization. The rest of this paper is organized as follows. Section 2 discusses the related work. We describe the system and adversary models considered for this work in Section 3. Section 4 presents the detailed design of COIDS. In Section 5, we first define a problem to optimize the active time duration of COIDS. We next present a multiplayer cooperative game-theoretic analysis to the problem in Section 5. In Section 6, we provide a qualitative analysis of COIDS. The performance of COIDS is evaluated by providing simulation results in Section 7. Finally, we conclude the paper in Section 8.

2. Related Work

In many recent works, due to the lack of cryptography primitives in CAN, researchers have preferred anomaly-based IDSs over message authentication mechanism to secure in-vehicle CAN bus. The existing anomaly-based IDSs have (i) analyzed the data traffic on the CAN bus, frequency/time, and entropy, (ii) exploited physical characteristics of ECUs, and (iii) exploited the characteristics of the CAN protocol. Unlike the CAN bus data traffic, imitating physical characteristics of ECUs is highly challenging for an attacker. In this work, we classified existing anomaly-based IDS designed by levering physical characteristics into three categories, namely, voltage-based, clock-based and message periodicity-based. Section 2.1 briefly summarizes the existing voltage-based IDS. We present clock-based IDS in Section 2.2. Finally, in Section 2.3, we discuss message periodicity-based IDS, most relevant to our context.

2.1. Voltage-based IDS

In an early work, Hoppe et al. [20] first introduced the idea of IDS for in-vehicle network. They proposed to exploit the traffic pattern of CAN bus, e.g., message frequency for designing an efficient IDS. Motivated by the work [20], Choi et al. [12] proposed a novel automotive IDS, called VoltageIDS, by exploiting the selected features of time and frequency domains of the electrical signals in CAN bus. In a similar work, Cho and Shin [21] proposed voltage-based attacker identification scheme, called Viden, for in-vehicle network. Viden initially measures the voltage of the electrical signals in CAN bus to create and update the transmitter ECUs' voltage profiles. Finally, Viden uses the voltage profiles to identify any anomaly in the in-vehicle network. Kneib and Huth [17] proposed an IDS, called Scission, by leveraging physical characteristics of electrical signals in CAN bus. Unlike VoltageIDS and Viden, Scission exploits variations in the resistor, signal reflections in addition to voltage. Through empirical studies, the authors show significant improvement of performance in terms of false positive rate compared to VoltageIDS and Viden. Although, Scission has shown impressive performance to detect intruders in CAN, however, real time measurement and processing of variations in resistor, signal reflections and voltage are highly challenging for ECUs due to limited resources. Further, in [22], researchers have shown that due to the requirement of additional cable, voltage-based IDSs introduce new attack surface for various voltage-based attackers. In an interesting work, Foruhandeh et al. [23] proposed a SIngle-fraMe based Physical-LayEr (SIMPLE) solution to detect intrusion and identify the specific ECU generating a CAN frame. SIMPLE exploit voltage based features in the time-domain from every CAN frame for fingerprinting transmitter ECU. Different from existing voltage-based IDS, SIMPLE securely updates the training data at regular interval to compensate environmental changes, e.g., supply voltage, temperature.

2.2. Clock-based IDS

Different from earlier works, Cho and Shin [15] proposed an anomaly-based IDS, called Clock-based IDS (CIDS), for in-vehicle network by leveraging clock skew of quartz crystal clock associated with ECUs. In a similar work, Sagong et al. [24] proposed a clock skew-based IDS for in-vehicle network. The authors first proposed an intelligent attack for CAN bus, called cloaking attack, where an adversary regulates message timing and cloaks its clock to equate the clock of the targeted ECU. Next, the authors defined a new metric called Maximum Slackness Index (MSI) to measure the efficacy of the proposed IDS for detecting cloaking attack. Further, the authors extended the work in [13] by providing formal models to accurately predict and characterize the attack success probability curves. Recently, Zhou et al. [25] propose an IDS, called Bit-time-based CAN Bus Monitor (BTMonitor) by exploiting the physical discrepancies between clocks of different ECUs. In particular, BTMonitor measures the anomalies of bit time in CAN frames from different ECUs for fingerprinting the sender ECU. To reduce the necessity for a high sampling rate, the authors next determine the bit time of recessive and dominant bits, respectively, and extract their statistical features as fingerprint.

2.3. Message periodicity-based IDS

In a recent work, Han et al. [26] proposed a host-based IDS for in-vehicle network. Particularly, the authors used the survival analysis model for estimating the survival function, which in turn helps to detect anomalies. Lee et al. [27] proposed an IDS for in-vehicle network by exploiting the remote frame. Specifically, the proposed IDS broadcasts remote frame periodically in the CAN bus and receives a response from the sender ECU. Based on the received response, the IDS calculates offset and time interval. If calculated offset and time interval exceeds a predefined threshold, IDS declares it as intrusion. In another work, Marchetti and Stabili [28] exploited the message ID, an unique identifier of the message used by each ECU, to design an IDS for in-vehicle network. In a similar work, Groza and Murvay [16] proposed an IDS for in-vehicle network. Interestingly, the proposed IDS took the advantage of Bloom filtering to check frame periodicity according to message ID. Kalutarage et al. [29] proposed a context-aware IDS for monitoring cyber-attacks in CAN. Particularly, the authors exploit CAN message sequence for fingerprinting ECU. To extract message sequences from the CAN bus, they used a sequence modelling technique, called n-gram distribution. Yu and Wang [30] proposed an IDS based on network topology construction and subsequent verification. They initially used message periodicity to estimate the communication links among the ECUs. The authors next exploited communication link information to construct a network topology. The proposed scheme periodically verifies the topology construction process and if it finds any deviation in the number of participating ECUs, IDS declares it as an intrusion.

In the context of physical properties based IDS, we have the following observations:

- None of the works exploited clock offset of quartz crystal clock inbuilt in ECUs while designing IDS for in-vehicle network. Motivated by this fact, in this work, we design COIDS leveraging clock offset of transmitter ECUs'.

- Except the works in [15, 22], none of the works consider real time intrusion detection scenario. However, it is utmost important to detect intruders in real time to minimize possible escalation of fatalities due to cyber attacks. Therefore, motivated by this fact, we employed CUSUM, the most powerful method for detecting irregular patterns in a real time process, in our proposed COIDS.

- None of the works considered optimizing the active time duration of IDS without compromising its effectiveness. Motivated by this fact, we propose a cooperative game model to analyze the performance of COIDS with reduced activity within the ECU.

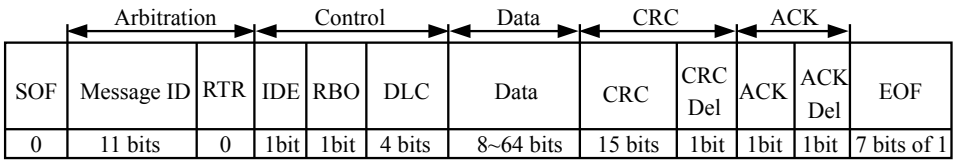


Figure 1: Format of a CAN data frame. Every CAN data frame consists of Start Of Frame (SOF), Arbitration, Control, Data, CRC, ACK and End Of Frame (EOF) fields.

- In contrast to existing works, we employed an active learning technique in COIDS to learn traffic patterns and SCPD to determine the exact time of attack.

3. Overview of CAN and Threat Model

In this section, we recall the required concepts related to our COIDS. Particularly, in Section 3.1, we present the background information about the CAN protocol. Section 3.2 discusses the clock related concept. We then present a brief discussion on the SCPD technique in Section 3.3. Section 3.4 briefly illustrates the CUSUM method. Finally, in Section 3.5, we present the adversary model and attack scenarios.

3.1. CAN Background

In in-vehicle networking, CAN communication protocol is one of the most popular standards, which interconnects various ECUs (or, nodes) using a multi-master, message broadcast bus system. As CAN is a broadcast bus system, hence, ECUs connected on the bus can transmit any messages to any ECU as well as monitor ongoing message transmissions. To preserve data consistency and take control decision, ECU exchange messages among themselves through CAN frames. Figure 1 depicts the typical format of a CAN data frame. Since CAN is a simple message oriented communication protocol, as an alternative of containing transmitter and/or receiver address, each CAN data frame contains a unique message ID. For example, a data frame with message ID 0x20 may comprise wheel speed of a vehicle. It is worth noting that a CAN data frame does not contain encryption, authentication fields.

The CAN bus is designed to behave as a wired-AND gate, particularly, contending ECUs give higher priority to a message with a smaller message ID. This procedure is known as arbitration.For example, if two ECUs $\mathbb{X}$ and $\mathbb{Y}$ are contending for transmitting messages 0x01 and 0x11 over the CAN bus, respectively. Since ECU $\mathbb{X}$ sends message with lower ID, ECU $\mathbb{X}$ wins arbitration, and acquires exclusive access of the CAN bus for message transmission. The ECU $\mathbb{Y}$, which has lost the arbitration, again attempts for transmission once the CAN bus becomes idle. It is worth mentioning that CAN bus lacks clock synchronization. However, SOF field and bit stuffing in CAN data frame provide the alignment of bit edges during the message transmission. In absence of clock synchronization, time instants for ECUs are given by their own quartz crystal clocks. In practice, a quartz crystal clock of an ECU runs at diverse frequencies, resulting in as much as a drift of 2400ms over a period of 24 hours [15]. For easy reference, we have summarized the notations in Table 1.

3.2. Clock Related Concepts

In this paper, we follow the standard nomenclature of clocks as defined in Network Time Protocol (NTP) [31]. Let C_t be a true clock that runs at a constant rate, $C_t(t) = t$, and C_A be a non-true clock kept by a clock A. We define the clock offset, frequency and clock skew as follows:

- **Clock Offset:** Clock offset $O_A(t)$ is the time difference between the non-true clock C_A and true clock C_t, i.e., $O_A(t) = C_A(t) - C_t(t)$.

Table 1: Summary of important notations

Symbol	Description	Symbol	Description
$C_t(t)$	True clock at time t	$C_A(t)$	Non-true clock at time t
$O_A(t)$	Clock offset at time t	T	Message interval
t_i	Transmission time of i-th message	a_i	Arrival time of i-th message
O_i	Accumulated clock offset of i-th message	d_i	Network delay of i-th message
n_i	Quantization noise of i-th message	J_i	Jitter of i-th message
O_k^{avg}	Average clock offset of k-th batch message	N	Number of batches of messages
O_k^{ac}	Accumulated clock offset of k-th batch message	f	Probability density function
$\mathbb{E}[T_{ar,i}]$	Expected inter-arrival time between message $(i-1)$ and i	M_{jk}	j-th message at k-th batch
L	Instantaneous log-likelihood ratio	S	Cumulative sum
t_a	Abrupt change time	D	Intrusion decision function
σ^2	Standard deviationof Gaussian distribution	μ	Mean of Gaussian distribution
ϑ	Number of neighbors of an ECU	γ	Security level
p	Probability of neighborhood monitoring	R	A set of players, $R = \{1, 2, \ldots, \psi\}$
V	Coalition	v	Characteristic function
δ	Number of players in a game, $\delta = 1, \ldots, \psi$	Sh_r	Shapley value of a player r

- **Frequency:** Frequency is the rate at which non-true clock runs. Hence, the frequency at a time t is given as: $C'_A = dC_A(t)/dt$.

- **Clock Skew:** Clock skew is the frequency difference between the non-true clock C_A and true clock C_t, i.e., $S_A(t) = dC_A(t)/dt - dC_t(t)/dt$.

At any instant of time, if two clocks have clock skew as 0, we consider those clocks are synchronized, otherwise, they are asynchronized. A positive clock skew means that C_A runs faster than C_t, whereas a negative clock skew means that C_A runs slower than C_t. Generally, the unit for measuring clock skew is microseconds per second ($\mu s/s$) or parts per million.

3.3. Sequential Change Point Detection

Generally, in CAN, intrusions occur at unknown points in time and resulting significant changes in the statistical properties of a data sequence [32]. To determine the precise time of intrusion, we need to analyze the observed data sequence using a statistical approach, where the number of observations is time varying. There are broadly two different techniques for detecting abrupt variations in stochastic data sequence model, namely, fixed size batch detection and sequential change point detection. Among these two change detection techniques, sequential change point detection is quicker than fixed size batch detection, and suitable for real time scenario [32, 33, 34]. The sequential change point detection technique characterizes the change point as the pre-change with unknown post-change in time, resulting in quickest change detection. Basically, a change point in the change point detection technique is a particular time instant where the statistical properties of data before and after this time instant are significantly different. In this work, motivated by the works [32, 33, 34], we model the precise estimation of intrusion time problem as a change-point detection problem.

3.4. CUSUM Method

The CUSUM method is one of the most powerful methods for detecting irregular patterns quickly in a real time process [19]. To detect irregular patterns or anomalies, CUSUM uses hypothesis testing developed over independent identically distributed (i.i.d.) random variables. Specifically, to detect an anomaly, CUSUM periodically calculates two sums, i.e., the upper threshold and the lower threshold, which signify the cumulative deviations between the observed and expected values. When the upper or lower control threshold exceeds a particular threshold, CUSUM classifies it as irregular pattern or anomaly. It is worth mentioning that, in CUSUM, the anomaly detection rule is a comparison between the cumulative sum and adaptive threshold. It is also worth mentioning that we not only can update the value of adaptive threshold in real time, but also can keep track of the memory usage of past observations. In CUSUM, a batch processing approach is used to detect a

small variation in statistical parameters, e.g., mean with respect to the regular patterns. Finally, the outcome of the CUSUM method is the list of anomalies associated with the plot in the time series. The CUSUM method has been used in various applications, including intrusion detection in networks, speech and image processing, signal processing. Due to the simplicity and cost-effectiveness, we used the CUSUM method in in-vehicle networks for intrusion detection. Nevertheless, we propose to connect a memory stick, running COIDS, with the vehicle via OBD-II port and monitoring CAN bus for possible anomalies.

3.5. Adversary Model and Attack Scenarios

In this work, we assume that an adversary is capable of performing the read and write operations on the CAN bus. Particularly, an adversary can perform eavesdropping and intercepting messages by reading CAN bus. In contrast, by write operation, an adversary can perform replaying, forging, and transmitting unauthenticated messages on the CAN bus. Further, we assume that an adversary can remotely compromise ECUs or gain access to the CAN bus via various attack surfaces, e.g., Bluetooth, mechanics tools, cellular connectivity. However, we do not assume that adversaries have physical access to the vehicle through an OBD-II port such as CANtact.

In this work, based on the above adversary model, we assume the following three most potential attack scenarios that can significantly hamper in-vehicle networks: DoS, Impersonation and Fuzzy.

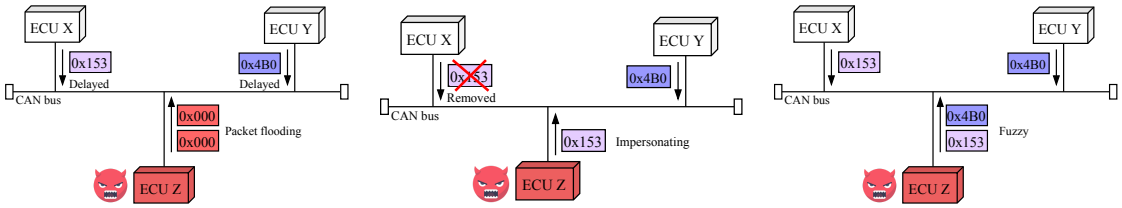
a) **DoS Attack.** To mount DoS attack, an adversary injects high priority messages in a short time interval on the CAN bus. Due to flooding of many high priority messages, the CAN bus becomes busy all the time and unavailable to other ECUs. Generally, an adversary mounts a DoS attack by injecting messages with theoretically highest priority message ID [27]. For example, as shown in Figure 2(a), an attacker ECU $\mathbb{Z}$ injects several high priority messages with ID=0x000. Since both ECU $\mathbb{X}$ and ECU $\mathbb{Y}$ share the same CAN bus, increasing occupancy of CAN bus generates delay for both message ID=0x153 and ID=0x4B0.

b) **Impersonation Attack.** To launch impersonation attack, an adversary ceases message transmission by controlling the victim (or, target) ECU and successfully gains the identity of the victim ECU to pose as an impersonating ECU. Thus, an impersonating ECU periodically broadcast a data frame and responds to a data frame as victim ECU. For example, as shown in Figure 2(b), an attacker ECU $\mathbb{Z}$ successfully gains the identity of ECU $\mathbb{X}$ and ceases all message transmissions from ECU $\mathbb{X}$. ECU $\mathbb{Z}$ next injects message ID=0x153 into the CAN bus impersonating ECU $\mathbb{X}$.

c) **Fuzzy Attack.** To mount fuzzy attack, an adversary injects randomly spoofed messages with various identifiers. As a result, ECUs in the in-vehicle network receive a significant number of messages. This may, in turn, leads to unintended CAN bus behavior, e.g., message priority inversion, deadline violation, etc. For example, as shown in Figure 2(c), an attacker ECU $\mathbb{Z}$ injects spoofed messages with ID=0x153 and ID=0x4B0 into the CAN bus randomly. Due to the random insertion of spoofed messages, fuzzy attack paralyzes the various functions of a vehicle, including tremendous shaking of the steering wheel, instrument panel blinking in countless ways and automatic changing of gear shift.

4. Proposed IDS: COIDS

In this section, we present a detailed description of COIDS. Particularly, in Section 4.1, we discuss the clock offset estimation procedure adapted in COIDS. Section 4.2 presents clock offset anomaly detection mechanism. Finally, we put forward intrusion detection approach in Section 4.3.



(a) DoS attack (b) Impersonation attack (c) Fuzzy attack

Figure 2: Three type of attack scenarios on in-vehicle CAN.

4.1. Clock Offset Estimation

Let us assume a scenario where an ECU $\mathbb{X}$ transmits messages periodically at every T ms and an ECU $\mathbb{Y}$ receives those messages. As only ECU $\mathbb{Y}$'s timestamp is available, we assume its clock as the true/reference clock. Let $t_0 = 0$ be the time when ECU $\mathbb{X}$ transmits its first message. In ideal scenario when transmitter and receiver clocks are synchronized, message i will be transmitted at $t_i = iT$ in ECU $\mathbb{Y}$'s clock. For the sake of convenience, we depict the timing diagram in Figure 3. Then, due to the clock skew, the actual transmission time is $t_i = iT + O_i$ in ECU $\mathbb{Y}$'s clock, where O_i is the accumulated clock offset of ECU $\mathbb{X}$ as first message sent at $t_0 = 0$. Due to an incurred network delay of d_i between transmission and reception, timestamp of the i-th message arriving at the incoming buffer of ECU $\mathbb{Y}$ is:

$$a_i = iT + O_i + d_i + n_i, \tag{1}$$

where n_i denote the noise introduced by ECU $\mathbb{Y}$'s timestamp quantization. Similarly, timestamp of the $(i-1)$-th message arriving at the incoming buffer of ECU $\mathbb{Y}$ is:

$$a_{i-1} = (i-1)T + O_{i-1} + d_{i-1} + n_{i-1}. \tag{2}$$

From eqs. (1) and (2), the expected value of the inter-arrival time between message $(i-1)$ and i, $\mathbb{E}[T_{ar,i}]$, is given by:

$$\begin{aligned}\mathbb{E}[T_{ar,i}] &= \mathbb{E}[a_i - a_{i-1}] \\ &= \mathbb{E}[T + \Delta O_i + \Delta d_i + \Delta n_i] \\ &= T + \mathbb{E}[\Delta O_i + \Delta d_i + \Delta n_i],\end{aligned}$$

where $\Delta O_i = O_i - O_{i-1}$ is the clock offset, Δd_i $(= d_i - d_{i-1})$ is the difference in network delay, $\Delta n_i (= n_i - n_{i-1})$ is the difference in noise. In CAN, as the lengths of messages of same ID usually are constant over time, without loss of generality, we assume $\mathbb{E}[\Delta d_i] = 0$. Let n_i be a zero-mean Gaussian distribution, therefore, it is reasonable to assume $\mathbb{E}[n_i] = 0$, and hence $\mathbb{E}[\Delta n_i] = 0$. So, in the ideal case, we have $\mathbb{E}[T_{ar,i}] = T + \mathbb{E}[\Delta O_i] \approx T$. It is evident from the discussion that the inter-arrival time between any two successive messages is approximately T, i.e., same as message periodicity. Therefore, to obtain a significant value of clock offset estimation, we must measure for a batch of incoming messages instead of two successive messages. Next, we discuss how we can estimate clock offset for a batch of messages.

Considering the arrival timestamp of the first message at ECU $\mathbb{Y}$'s end as $d_0 + n_0$ and the expected inter-arrival time between messages, $\mathbb{E}[T_{ar,i}]$, we can determine the the expected arrival time of the i-th message as $i\mathbb{E}[T_{ar,i}] + d_0 + n_0$. In contrast, the actual i-th message arrival time is $a_i = iT + O_i + d_i + n_i$. According to the NTP specification, we model the accumulated clock offset from 1st to i-th message as a random variable, $O_i = iO + J_i$, where O is the clock offset per message period T, and J_i is the ECU jitter, caused by variations in the task scheduling, execution time. It is worth mentioning that due to the randomness of thermal noise, jitter follows a Gaussian distribution. As jitter follows a

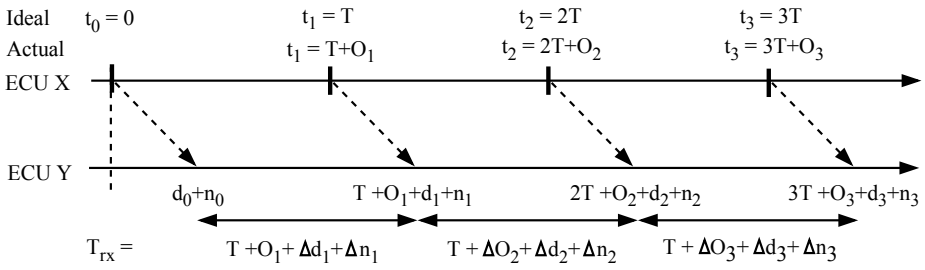


Figure 3: Timing analysis of message arrivals in CAN bus.

Gaussian distribution, we assume that J_i and J_{i-1} as outcomes of a Gaussian random variable $J \sim \mathbb{N}(0, \sigma^2)$, where σ^2 is the deviation of jitter. Thus, the actual i-th message arrival time is $a_i = iT + iO + J_i + d_i + n_i$. From $\mathbb{Y}$'s perspective, the message period is T with respect to ECU $\mathbb{X}$'s clock, which corresponds to $T_{ar,i} = a_i - a_{i-1}$ in ECU $\mathbb{Y}$'s clock. By the definition of the clock offset, the observed offset is:

$$\begin{aligned} \bar{O}_i &= (a_i - a_{i-1}) - T \\ &= (O + \Delta J_i + \Delta d_i + \Delta n_i), \end{aligned}$$

where $\Delta J_i = J_i - J_{i-1}$. To estimate the clock offset, we processed a batch of N received messages and determine the average clock offset in the k-th batch, O_k^{avg}, where $k = 1, \ldots, K$. We determine O_k^{avg} as follows:

$$\begin{aligned} O_k^{avg} &= \frac{1}{N} \sum_{i=1}^{N} \bar{O}_i \\ &= \frac{1}{N} \sum_{i=1}^{N} (O + \Delta J_i + \Delta d_i + \Delta n_i). \end{aligned} \tag{3}$$

Since we measured the offset for every N received messages, using eq. (3), the accumulated clock offset till the last message of the k-th batch is given as:

$$O_k^{ac} = O_{k-1}^{ac} + N O_k^{avg}. \tag{4}$$

It is worth mentioning that Cho and Shin [15] used the original value of O_k^{avg} instead of the absolute value. From eq. (4), it is clear that by calculating the clock offset from observation of message periodicity, transmitter ECUs can be fingerprinted. In this work, we exploit this characteristic in designing COIDS, a clock offset-based IDS for in-vehicle networks.

4.2. Clock Offset Anomaly Detector

To detect anomalies in accumulated clock offset, COIDS uses the CUSUM method, which is the core of the SCPD algorithm. The CUSUM method is a sequential detection method suitable for detecting any anomaly that causes changes in measurement. With a limited computation resource, the CUSUM method uses the feature of sequential and non-parametric examinations to detect any attack in time series data. Specifically, in COIDS, we processed a batch of N messages and then, we applied the CUSUM method to detect small changes in statistical parameters, e.g., mean with respect to the regular pattern.

Let $M_{jk} = \{m_{1k}, m_{2k}, \ldots, m_{Nk}\}$ be the j-th message at k-th batch sent periodically over time t_j, where $j = 1, \ldots, N$ and $k = 1, \ldots, K$. Also, let $O_j = \{o_1, o_2, \ldots, o_{N-1}\}$ be a set of clock offsets of j messages and i.i.d. following a Gaussian distribution with mean μ and variance σ^2. Under normal scenario, as each message in CAN bus is transmitted periodically, each clock offset O_j follows a Probability Density Function (PDF), $f(O_j, \alpha)$ based on a deterministic parameter α, e.g., μ or σ^2 of O_j. Under attack scenario, O_j may

contain an abrupt change at some time t_a, where $t_a \in t_j$. It is worth mentioning that by the abrupt change we mean the changes with both large and small magnitudes of O_j. Using the CUSUM method, we can model this abrupt change by an instantaneous modification of the value of deterministic parameter α at time t_a. Hence, $\alpha = \alpha_0$ before t_a, and $\alpha = \alpha_1$ from t_a to the present time. Under this scenario, the whole PDF of the clock offset measured between t_1 and the present time t_j can be divided into two categories of hypotheses (h):

- Under no change hypothesis ($h = h_0$), the PDF of clock offset is given as:

$$f_{h_0} = \prod_{t_j=t_1}^{t_N} f(O_j, \alpha_0). \tag{5}$$

- Under a change hypothesis ($h = h_1$), the PDF of clock offset is given as:

$$f_{h_1} = \prod_{t_j=t_1}^{t_a} f(O_j, \alpha_0) \prod_{t_j=t_a+1}^{t_N} f(O_j, \alpha_1). \tag{6}$$

It is clear from eqs. (5) and (6) that to determine the clock offset, the PDF of each sample $f(O_j, \alpha)$ and the values of the deterministic parameter t_a need to be known. Further, we have to determine the time of the abrupt change between t_1 and t_N. Since here the problem is to decide between the two hypotheses h_0 and h_1 from the PDF of the measured clock offset, we can call this problem as a binary hypothesis testing problem [19]. According to [32], instantaneous log-likelihood ratio test is the best possible solution technique for the binary hypothesis testing problem. Therefore, following the solution technique, the instantaneous log-likelihood ratio at time t_j is given as:

$$s[t_j] = L[t_j, t_j] = \ln\left(\frac{f(O_j, \alpha_1)}{f(O_j, \alpha_0)}\right), \tag{7}$$

and the cumulative sum from t_1 to t_N, i.e.,:

$$S[t_N] = \sum_{t_j=t_1}^{t_N} s[t_j]. \tag{8}$$

From eqs. (7) and (8), we can rewrite the instantaneous log-likelihood ratio as:

$$\begin{aligned} L[t_N, t_a] &= \sum_{t_j=t_a}^{t_N} \ln\left(\frac{f(O_j, \alpha_1)}{f(O_j, \alpha_0)}\right) \\ &= S[t_N] - S[t_a - 1]. \end{aligned} \tag{9}$$

It is worth noting that t_a is unknown in eq. (9). To estimate t_a, we can apply a standard statistical approach based on the maximum likelihood principle. Following the maximum likelihood principle, we obtain the intrusion decision function $D[t_N]$ and the estimated abrupt change time $\hat{t_a}$ as:

$$D[t_N] = S[t_N] - \min_{1 \leqslant t_a \leqslant N} S[t_a - 1], \tag{10}$$

$$\hat{t_a} = \min_{1 \leqslant t_a \leqslant N} S[t_a - 1]. \tag{11}$$

It is worth noting from eq. (10) that the decision function is the present cumulative sum minus its present minimum value. On the contrary, in eq. (11), we notice that the abrupt change time estimate is the time of the present minimum of the cumulative sum. It is clear from the above discussion that using eq. (10), one can decide whether any intrusion has

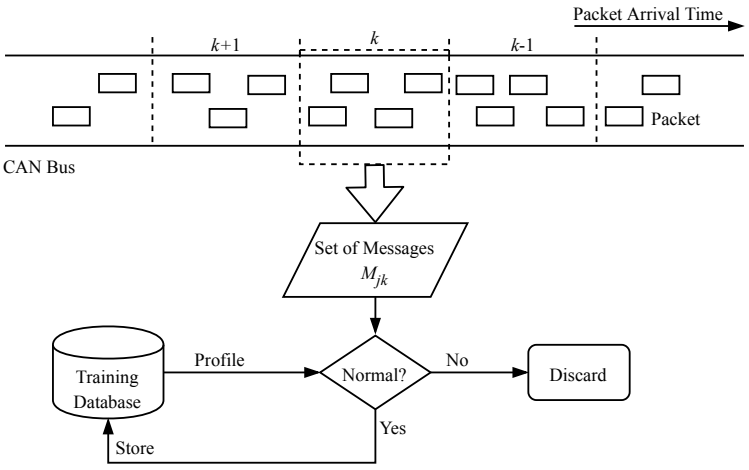


Figure 4: Flow chart of learning mechanism used in COIDS.

actually occurred or not. Similarly, using eq. (11), one can estimate the time of intrusion. As far as the decision whether any intrusion has actually occurred or not, in this work, we consider that if $D[t_N]$ exceeds a positive threshold h_{th}, we conclude that the intrusion has occurred, otherwise, no intrusion. Since our objective is to detect intrusions in real time, hence, we rewrite the eq. (8) in a recursive form and the same is given as:

$$S[t_j] = S[t_j - 1] + s[t_j]. \tag{12}$$

Further, as we assume that in case intrusion decision function exceeds a positive threshold, we can rewrite eq. (10) as:

$$D[t_N] = \{D[t_N - 1] + s[t_j]\}^+, \tag{13}$$

where $\{x\}^+ = sup(x, 0)$. Summarily, from our above analysis, we can take the decision about the possible intrusion using eq. (13). In contrast, we can estimate the time of intrusion using eq. (11) from the set of received messages, e.g., $\{m_{1k}, m_{2k}, \ldots, m_{Nk}\}$ efficiently. Therefore, the size of the set of received messages, i.e., N, determines the amount of past memory held by the CUSUM algorithm and the correct choice of h_{th} are the vital parameters for efficient intrusion detection in real time.

4.3. Intrusion Detection Approach

In this work, we exploit the clock offset of in-vehicle messages for fingerprinting transmitter ECU. Generally, an ECU transmits messages through CAN bus at regular intervals or frequency, where frequency is determined by the quartz crystal of that particular ECU. If an adversary injects a spoofed message from an ECU that is distinct from the spoofed ECU, the difference in clock frequency significantly changes the clock offset of messages [15]. In this work, to characterize clock offset, we follow a batch processing approach, where the size of a batch is N. Specifically, we determine the average clock offset of messages for every N received messages from the attack-free dataset. Figure 4 shows the flow chart of learning mechanism used in COIDS. In particular, we used an active learning strategy to accelerate the training speed. In a normal state, the clock offsets of messages O_j in a particular batch k are measured and form the training set. This procedure of clock offset estimation continues iteratively. We then measure the mean of the clock offsets of messages O_j. If the mean of the clock offsets of messages deviates $\pm 20\%$ from a normal state [18], we consequently discard the training dataset.

Algorithm 1 illustrates how we estimate clock offset and intrusion time. We determine the PDF of O_j by calculating the deterministic parameters μ and σ^2 of the k-th batch messages.

Algorithm 1: Clock offset and intrusion time estimation

Initialize: set the threshold of decision function $h_{th} > 0$, $S[t_1] = D[t_1] = 0$, $j = 1$
for $k = 1$ to K **do**
  if $h_{th} \geqslant D[t_j]$ **then**
    collect set of messages $M_{jk} = \{m_{1k}, m_{2k}, \ldots, m_{Nk}\}$
    calculate $s[t_j] = \ln\left(\frac{f(O_j,\alpha_1)}{f(O_j,\alpha_0)}\right)$
    calculate $S[t_j] = S[t_j - 1] + s[t_j]$
    if $D[t_j] > h_{th} > 0$ **then**
      $\hat{t_a} \leftarrow t_j$
      reset or stop the algorithm
    end if
    $j = j + 1$
  end if
end for

We then compute the cumulative sum $S[t_N]$ as presented in eq. (8) using the instantaneous log-likelihood ratio $s[t_j]$ given as follows:

$$S[t_N] = \frac{\mu_{\alpha_1} - \mu_{\alpha_0}}{\sigma_\alpha^2}\left(O[t_N] - \frac{\mu_{\alpha_1} + \mu_{\alpha_0}}{2}\right). \tag{14}$$

In this work, for determining intrusion detection function and subsequently generating an alarm signal, COIDS computes atleast ten Average Run Length (ARL) of mean and variance of clock offset. At each step of ARL computation, COIDS updates the deviation of mean and variance of clock offset by comparing ARL under attack-free scenario, i.e., h_0 and ARL under attack scenario, i.e., h_1. If either of the control point, h_0 or h_1, exceeds the threshold h_{th}, i.e., an unexpected negative or positive change in value has been detected, respectively, and hence COIDS proclaims this change as an intrusion. It is worth mentioning that as COIDS is based on the cumulative sum, even a small drifting in the mean and variance of clock offset from the normal value leads to steadily decreasing or increasing cumulative sum values. In CUSUM, as a general rule of thumb, the value of h_{th} ranges between 0 and 3 [19].

5. Proposed Probabilistic Model

Typically, our proposed COIDS has to run incessantly in every ECU to oversee network behavior and subsequent detection of any anomalous activity. Such behavior of COIDS have the potential to increase the power overhead on an ECU. Hence, in this section, we attempt to address the challenge: how to limit the active time duration of COIDS without compromising its performance. We initially present a probabilistic model to optimize the active time duration of COIDS in Section 5.1. Thereafter, in Section 5.2, we proposed a multiplayer cooperative game based solution to the optimization problem.

5.1. Optimization Problem Formulation

To address the typical challenge of COIDS, we tackle the problem from the point of view of an ECU being monitored by its immediate neighbor. We develop an optimization problem for addressing the challenge and examine the optimization problem using multiplayer game theory [35]. Let us assume a network scenario, where several ECUs are connected with each other through a CAN bus and COIDS is running in each ECU for detecting malicious activity within its immediate neighborhood.

Let us assume that an ECU $\mathbb{X}$ has ϑ active neighbors at a specific time instant. Hence, each ϑ neighbor monitors the traffic activity of ECU $\mathbb{X}$. Now, during a certain instant of time, all or some of the ϑ neighbors might detect an anomalous activity of ECU $\mathbb{X}$ and trigger an

alarm upon crossing certain predefined detection rate. Interestingly, while monitoring ECU $\mathbb{X}$, the neighbors spend their significant amount of computational resources and energy. Nevertheless, it might not be essential to keep the COIDS running on every ECU all the time. Therefore, we attempt to minimize this redundancy, and subsequently saving the scarcest resource of ECU. Here we assume that each ECU is equipped with COIDS and COIDS monitors the traffic of its immediate neighbors incessantly. It is worth mentioning that the number of COIDSs running and monitoring a neighborhood relies on the level of security that is expected at some specific time instant. In this paper, we define the security level γ as: an ECU is monitored by at least γ of its immediate neighbors at any time instant. From definition, the security level offers a tradeoff between overheads and security. The higher value of γ, the greater number of immediate neighbors that monitor an ECU at a particular time, which ultimately results in higher energy and computational overheads. We introduce the concept of the security level so that COIDS can be used in various application scenarios with varying security requirements. For example, some application scenarios may tolerate some trivial ECU(s) like seat, door to be compromised. It means $\gamma = 1$ is limited for most security sensitive applications, where in-vehicle network cannot tolerate intruders. Hence, based on the tolerance of the highest number of compromised ECUs, the security level can be adjusted. Further, in an application scenario, an IDS component in an ECU might observe a part of its neighbor's behavior. This might lead to irregularities with respect to the observed data in various IDSs. By setting the security level, one can limit the number of ECUs observing an ECU's behavior at any instant of time. Furthermore, if the validation needs consensus of more neighboring ECUs, the security level can be raised.

Let us consider that ECU $\mathbb{X}$ has ϑ immediate neighboring COIDSs running on respective ECUs at a specific time instant. Further, let us consider that each ECU monitors independently its immediate neighbors with a probability of p. Hence, the probability that ECU $\mathbb{X}$ is monitored for anomalous activity with security level γ is given as:

$$P(\gamma/\vartheta) = \sum_{l=\gamma}^{\vartheta} \binom{\vartheta}{l} p^l (1-p)^{\vartheta-l}. \tag{15}$$

Based on eq. (15), our objective can be formulated by the following optimization problem:

$$\min p \tag{16}$$

subject to

$$\sum_{l=\gamma}^{\vartheta} \binom{\vartheta}{l} p^l (1-p)^{\vartheta-l} \geqslant \omega, \tag{17}$$

where $\omega + \delta = 1$ and δ is a very small positive number. Here, ω represents a threshold value, which is the minimum probability required to maintain the desired level of γ. We can set the value of ω considering the application scenario. Therefore, an optimal solution of our formulated problem eqs. (16)-(17) will provide the minimum p with which each immediate neighbor has to monitor.

5.2. Game-Theoretic Solution

In this section, we determine the solution to the optimization problem presented in eqs. (16)-(17). It is here worth mentioning that our design solution must be profitable from the cooperative IDS's point of view. Alternatively, the design solution must ensure the significant resource saving in balance way among the ECUs. To achieve this goal, we design a multiplayer cooperative game model to designate the interactions among the COIDSs running on immediate neighboring ECUs.

In our game model, we set the objective of each player or ECU as that of monitoring the immediate neighboring ECUs at the prerequisite security level for detecting any malicious activity. Additionally, we set objective of conserving resources, particularly, energy. Therefore, among these two objectives, we consider monitoring objective as primary goal and energy saving as secondary goal. It is worth mentioning that if we were set the precedence of energy saving objective over monitoring objective, each ECU would independently choose to sleep entire time duration, resulting in a completely ineffective COIDS. As ECUs are operated independently, hence, each ECU must cooperate with each other to achieve our aforementioned goals. Therefore, we can model our scenario as a ψ-player cooperative game.

To get an ECU monitored with γ security level, each of its immediate neighbors must participate in monitoring with the minimum probability determined through eqs. (16)-(17). To determine minimum probability, we can model our problem as a ψ-player cooperative Transferable Utility (TU) game [36]. A cooperative TU game is defined as a pair (R, v), where R is a finite set of players with $|R| \geqslant 2$, and $v : 2^R \rightarrow \mathbb{R}$ is a characteristic function that associates a real number $v(V)$ with each subset V of R such that $v(\phi) = 0$ [36]. For a coalition $V \subseteq R$, $v(V)$ is called the worth of coalition V and it can be divided into any possible way among its players. Considering that the energy consumption of a COIDS is linear, we can represent $v(V)$ as:

$$v(V) = \begin{cases} \delta(1 - p_\delta)E, & \text{if } \delta \geqslant \gamma \\ 0, & \text{if } \delta < \gamma \end{cases}, \tag{18}$$

where E is the energy consumed by the COIDS when it incessantly monitors, $\delta = |V|$, p_δ is the probability (refer to eqs. (16)-(17)) with which every player monitors in a coalition comprising of δ players, and γ is the security level. It is worth noting that the utility of our game is to save energy of each player. If $\delta \geqslant \gamma$, COIDS achieve the desired security level, i.e., γ, and hence, the payoff $v(V) = \delta(1 - p_\delta)E$. Otherwise, COIDS cannot achieve the desired security level, and thus, $v(V) = 0$. According to [37], a solution to a cooperative game of each player r is given by the Shapley value as follow:

$$\mathrm{Sh}_r[v] = \sum_{V, r \in V} \frac{(\delta - 1)!(\psi - \delta)!}{\psi!}[v(V) - v(V - r)]. \tag{19}$$

In eq. (19), the summation is taken over all subsets of V of which player r is a member. According to eq. (18), the value of $v(V)$ depends on δ. Hence, we group the subsets based on their cardinality, and summation is determined over these groups of subsets such that $\delta = 1, \ldots, \psi$. We can determine the number of subsets of size δ of which player r as follow:

$$\begin{aligned} \mathrm{Sh}_r[v] =& \sum_{\delta=1}^{\psi} \binom{\psi - 1}{\delta - 1} \frac{(\delta - 1)!(\psi - \delta)!}{\psi!}[\delta(1 - p_\delta) - (\delta - 1)(1 - p_{\delta-1})]E \\ =& \frac{E}{\psi} \sum_{\delta=1}^{\psi} [1 - \delta p_\delta + (\delta - 1)p_{\delta-1}] \\ =& (1 - p_\psi)E, \end{aligned} \tag{20}$$

where p_ψ is the probability with which every player monitors the whole coalition that comprises of ψ players. From eq. (20), we can deduce a few observations, as follows:

- *Observation 1:* The Shapley value allocation for the game $\mathrm{Sh}[v] = [(1 - p_\psi)E, \ldots, (1 - p_\psi)E]$ is individually rational for all the players as $\mathrm{Sh}_r[v] \geqslant v(\{r\})$.

Based on the nature of equation $v(r) = (1 - p_1)E$, we can write $\mathrm{Sh}_r[v] = (1 - p_\psi)E$ as subset V comprises of only one player. According to the aforementioned definition of

the characteristic function, p_δ is the probability with which every member monitors in a coalition that comprises of δ players, and p_δ is determined using our optimization problem eqs. (16)-(17). Hence, we can write: $p_1 > p_\psi$, and thereby, $(1-p_\psi)E > (1-p_1)E$. It means none of the players have any problem of accepting this payoff as it is preferable for a player to monitor in a group instead of alone.

- *Observation 2:* The Shapley value allocation for the game $\text{Sh}[v] = [(1-p_\psi)E, \ldots, (1-p_\psi)E]$ is an imputation as $\text{Sh}_r[v] \geqslant v(\{r\})$ and $\sum_{r=1}^{\vartheta} \text{Sh}_r[v] = v(R)$.

Here, it is worth noting that $\sum_{r=1}^{\vartheta} \text{Sh}_r[v] = \psi(1-p_\psi)E$, and $v(R) = \psi(1-p_\psi)E$. It means that in our game, imputation is an individually rational payoff that allocates the maximum amount (Pareto-optimality condition). Therefore, every player receives the maximum payoff possible [38].

- *Observation 3:* The Shapley value allocation for the game $\text{Sh}[v] = [(1-p_\psi)E, \ldots, (1-p_\psi)E]$ is collectively rational as $\sum_{r\in V} \text{Sh}_r[v] \geqslant v(V)\ \forall V \subset R$.

Here, it is worth noting that $\sum_{r\in V} \text{Sh}_r[v] = \delta(1-p_\psi)E$, $v(V) = \delta(1-p_\delta)E$, and $\delta = |V|$. Furthermore, we can derive p_ψ and p_δ by solving our optimization problem, i.e., eqs. (16)-(17), and there are more players in R than in V (i.e., $\psi > \delta$). Therefore, $p_\psi < p_\delta$. So, we notice that $\delta(1-p_\psi)E > \delta(1-p_\delta)E$. It means that in our game no player has the incentive to deviate from the grand coalitions and form a relatively smaller coalition with neighboring players (or, ECUs).

All these three observations reveal that energy saving achieved by COIDS with the help of the optimization problem eqs. (16)-(17) is in balance (or, equilibrium). Alternatively, a solution of our game model in equilibrium is to prove that it is in the core of the game. According to Lemaire [38], *the core of the game is the set of all collectively rational payoffs.* In Observation 3, it is revealed that the Shapley value achieved is collectively rational. Hence, the Shapley value of the game $\text{Sh}[v] = [(1-p_\psi)E, \ldots, (1-p_\psi)E]$ is in the core.

6. Qualitative Analysis

We present the theoretical analysis of our designed framework in this section. In particular, we first derive the computational complexity of our intrusion detection technique presented in Section 4.3. We then examine the message complexity of COIDS. Finally, we analyze the optimization problem presented in Section 5.2.

Computational Complexity. We estimate the clock offset and the intrusion time using Algorithm 1. We here derive the computational complexity of Algorithm 1 in Theorem 1.

Theorem 1. The proposed clock offset and intrusion time estimation algorithm has a polynomial time complexity.

Proof. The proposed clock offset and intrusion time estimation algorithm effectively calculate the instantaneous log-likelihood ratio of $f(O_j, \alpha_1)$ and $f(O_j, \alpha_0)$ before and after t_a, i.e., $s[t_j]$ for the k-th batch of received messages (line 5). Let us assume that the complexity of computing $s[t_j]$ is β_1, where β_1 depends on the size of a batch of message, i.e., N. Particularly, for the larger value of N, the value of β_1 will be higher or vice-versa. Our algorithm also calculates the cumulative sum $S[t_j]$, where $j = 1, \ldots, N$ in the k-th batch of received messages (line 6). Let us assume that the complexity of computing $S[t_j]$ is β_2, where β_2 depends on the size of a batch of message, i.e., N. Similar to β_1, for the larger value of N, the value of β_2 will be higher or vice-versa. Our proposed algorithm has a loop (lines 2-13). Therefore, for executing this loop, the complexity is $O(K(\beta_1 + \beta_2))$. In totality, the computational complexity of Algorithm 1 is $O(K(\beta_1 + \beta_2))$ and it is polynomial time complex. □

Message Complexity. Generally, in CAN bus, a message is broadcast by the ECUs (see Section 3.1). Each active ECU receives messages from each of its neighbors. In our proposed COIDS, an attacker is detected using only local information. Hence, in COIDS, the worst-case complexity is $O(\Gamma)$, where Γ is the maximum number of active neighbors at any instant of time.

Security Level. In Section 5, we present a probabilistic model to determine the ideal probability based on which a COIDS will decide to remain active to achieve the desired security level. Here, we now determine the minimum monitoring probability based on the solution to the optimization problem given in eqs. (16)-(17).

Theorem 2. The minimum monitoring probability to ensure that each ECU r is monitored at the desired security level γ is given as p_r^{min}, where p_r^{min} is derived using the minimum degree of neighbors x_r of ECU r in eq. (17).

Proof. We prove this theorem by contradiction. Suppose that each ECU r is monitored with security level γ. Further, assume that for each ECU r, p_r^{min} is derived using a positive integer y such that $y > x_r$.

Let $p^{(x_r)}$ be the solution to the problem given in eqs. (16)-(17). Therefore, $p^{(y)}$ signifies the corresponding solution, where x_r is substituted by y. Suppose ECU q be the neighbor of ECU r having the minimum degree of neighbors among all its neighbors. If we notice eq. (17), the left hand side is representing the probability that at least γ immediate neighbors are monitoring out of the y neighbors. Hence, $p^{(y)}$ decreases with the increase of y. Alternatively, $p^{(y)} < p^{(x_r)}$, as $y > x_r$. It is worth noting that x_r is the degree of neighbors of ECU q. Thus, $p^{(x_r)}$ is the minimum monitoring probability to ensure that ECU q is monitored by its neighbors to achieve security level γ. As $p^{(y)} < p^{(x_r)}$, ECU q is monitored with desired security level γ. This contradicts our initial assumption. Therefore, it proves our theorem. □

7. Experimental Evaluation

In this section, we evaluate the performance of our proposed COIDS under practical CAN bus application scenarios. More specifically, we compare this dynamic security enabled COIDS with our main competitor, OTIDS [27], and Conventional COIDS (C-COIDS) [39]. Here, we first describe the attack scenarios considered during simulation in Section 7.1. We next present the experimental setup used to evaluate the performance of the competing schemes in Section 7.2. Finally, we discuss the experimental results performed under three attack scenarios in Section 7.3.

7.1. Attack Scenarios

To measure the performance of all competing schemes, we use real world publicly available datasets [40]. The datasets were constructed by logging CAN traffic through the OBD-II port of a real vehicle. This real world datasets were collected during attack free state and three different attack scenarios, namely, DoS, impersonation and fuzzy. Particularly, during attack free state 2,369,868 messages were generated and collected from the CAN bus. In contrast, during the DoS attack scenario, an attack message is injected in every 0.003 sec and in total 656,579 messages are injected within CAN bus, where the dominant message ID is 0x000. Similarly, during the impersonation attack scenario total 659,990 messages are injected at the rate of 0.001 sec by impersonating the drive gear message ID. Finally, during the fuzzy attack scenario total 591,990 messages are injected with random message IDs at the rate of 0.005 sec to interfere the normal vehicle operations.

7.2. Experimental Setup

We performed all experiments in a specially designed OMNeT++ simulator for CAN [41]. In CAN bus, we can learn the behavior of a message by inspecting the average number of message instances and intervals between successive messages generated from the same ECU. Therefore, to identify whether messages originate from the same ECU, we consider a naive method [15] while simulating all the schemes. We initially run our intrusion detection algorithm on attack free dataset to train COIDS using an active learning technique. To speedup the training process, we used an active learning technique [18]. Further, to detect an abnormal behavior, COIDS computes ten ARL functions of the mean and variance of clock offset. Generally, the ARL function is the estimated number of samples to detect any shift in the process mean and subsequently generating an alarm signal. The ARL function takes h_0 and h_1 as inputs and is represented by: $ARL = E_h[t_d]$, where t_d is the change in detection time of the CUSUM method. Similar to [33], we implemented a SCPD technique to determine the exact time of attack while simulating the COIDS.

During simulation, we assume that the values of the various parameters used for the CUSUM method as suggested by Olufowobi et al [34]. Specifically, in [34], they have determined the optimal parameter setting to detect small changes in the process mean in real time for the CUSUM method. For example, they found that $h_{th} = 3$ and the value of reference parameter of CUSUM method is 0.5σ as the optimal value for detecting any small change in real time. Additionally, we set $\omega = 0.995$ considering the fact that $\omega + \delta = 1$ and δ is a very small positive number. While running COIDS, we calculate clock offsets in every 20 received messages, i.e., $N = 20$. Finally, during simulation, we consider two variants of COIDS. In first variant, we set $\gamma = 1$, i.e., at least one ECU is monitoring the CAN bus for detecting adversaries. In the second variant, we set $\gamma = 3$, i.e., at least three ECUs are monitoring the CAN bus for detecting adversaries. While simulating COIDS, we consider γ as the threshold value so that in the detection scenario when at least one ECU is monitoring, at least one alarm signal is essential to declare an adversary as detected. Finally, while simulating C-COIDS and OTIDS, we did not consider the security level.

7.3. Experimental Results

In this section, we discuss the performance of both COIDS and OTIDS under three different attack scenarios mentioned in Section 7.1. Specifically, we first evaluate the validity of fingerprinting of the transmitted ECUs based on accumulated clock offset in Section 7.3.1. We then present our experimental results under DoS attack scenario in Section 7.3.2. Section 7.3.3 discusses the experimental results under impersonation attack scenario. In Section 7.3.4, we put forward the experimental results under fuzzy attack scenario. Finally, we discuss the power efficiency of the COIDS in Section 7.3.5.

7.3.1. Clock Offset as a Fingerprint

We first determine the validity of COIDS's fingerprinting of the transmitted ECUs based on accumulated clock offset. Figure 5 shows our simulation results of accumulated clock offsets for different messages generated from three ECUs. Specifically, messages {0x153, 0x164} sent from ECU $\mathbb{X}$, messages {0x4B0, 0x4B1} sent from ECU $\mathbb{Y}$ and finally message 0x5A0 sent from ECU $\mathbb{Z}$. We notice from the plot that all accumulated clock offsets are linear in time. Here, it is worth mentioning that the slope of the plot is representing the estimated clock offset. Interestingly, the plot shows that for the same message source the value of the clock offset is almost same. For example, messages {0x153, 0x164} sent from ECU $\mathbb{X}$, exhibited almost the same clock offset range between 0 and 1.08 sec in Figure 5. Likewise, messages {0x4B0, 0x4B1} sent from ECU $\mathbb{Y}$, exhibited almost the same clock offset range between 0 and 0.70 sec. These experimental results indicate that COIDS can use clock offset to distinguish ECUs, and hence can be used as the fingerprints of the respective ECUs.

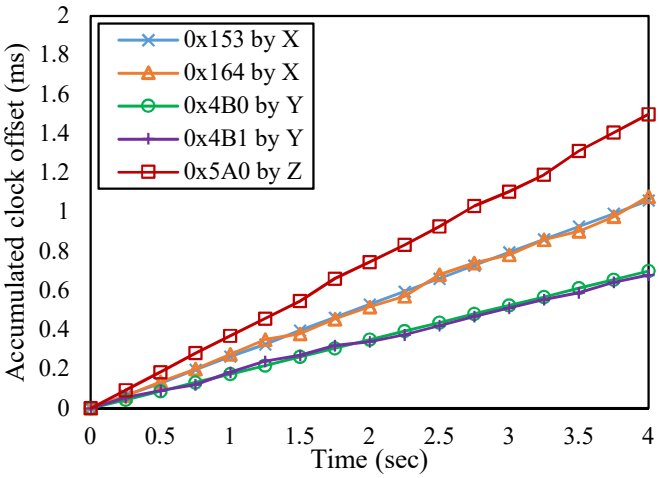


Figure 5: Accumulated clock offsets determined by COIDS in our experimental evaluation setting.

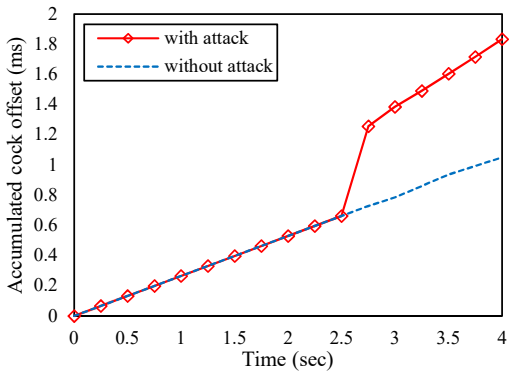
7.3.2. Defense Against DoS Attack

To evaluate the capability of COIDS for defending DoS attack on the CAN bus at a desired security level, we program ECU $\mathbb{Z}$ as the attacker, which injects periodic messages from $t = 2.5$ sec at 0.003 sec intervals with message ID 0x000, i.e., the highest priority message. Further, we program ECU $\mathbb{X}$ as the victim ECU and ECU $\mathbb{Y}$ as the monitoring ECU running COIDS.

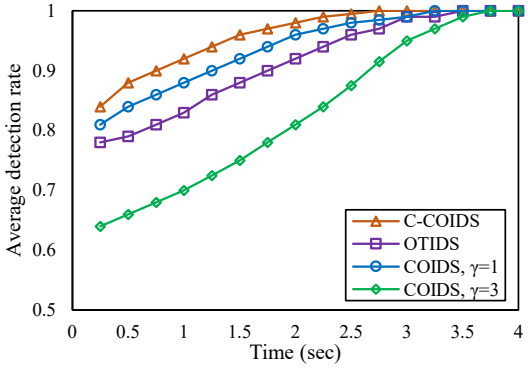
Figure 6(a) shows how the value of accumulated clock offset changes in the absence and presence of DoS attack. The plot shows that as soon as ECU $\mathbb{Z}$ mounts DoS attack, the value of accumulated clock offset suddenly changes in the positive direction. Due to this sudden shift, the change hypothesis, h_1, of CUSUM method suddenly increases and exceeds its threshold $h_{th} = 3$, i.e., detects an intruder.

We measure the effectiveness of detecting DoS attack for all competing schemes considering the average detection rate as the performance metric. We plot the average detection rate in Figure 6(b). While simulating, we considered the number of ECUs as 20 and number of malicious ECUs as 5. We also considered that at any instant of time any one of the 5 malicious ECUs is chosen for injecting messages at the rate of 0.05 sec. Figure 6(b) shows that all schemes are successful in detecting DoS attack after a certain time. In all schemes, we observe that the average detection rate increases with the increase of time. We also observe that C-COIDS has the highest detection rate, whereas COIDS, $\gamma = 3$ has the lowest detection rate. It is due to the different number of votes that are required to finally convict an ECU as malicious.

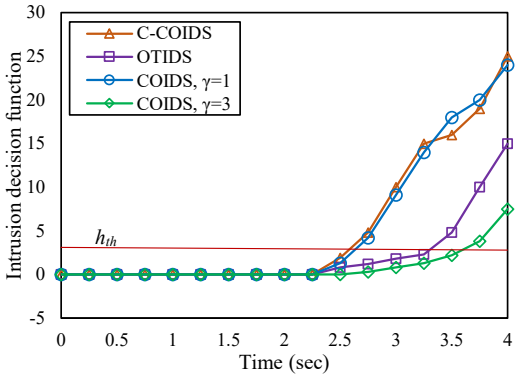
We plot the intrusion detection time by varying the time in Figure 6(c). The plot reveals that C-COIDS detects the injected message ID 0x000 in a very short delay, in fact, almost in real time compared to other schemes. Similar to C-COIDS, COIDS, $\gamma = 1$ also shows a similar kind of plot as in both cases at least one ECU is sufficient for declaring adversaries. The possible reason for this very short delay is the inclusion of SCPD technique. Precisely, we observe that the first injection of message ID 0x000 was at 2.5 sec, while C-COIDS and COIDS, $\gamma = 1$ signaled the intrusion detection alarm at $t_d = 2.704$ sec and $t_d = 2.711$ sec, respectively. This indicates that the delay of detecting DoS attack is 0.204 sec and 0.211 sec for C-COIDS and COIDS, $\gamma = 1$, respectively. It means that the delay in detecting DoS attack is slightly more in COIDS, $\gamma = 1$ compared to C-COIDS. This is mainly due to the stricter requirement for $\gamma = 1$ in COIDS compared to C-COIDS. All these results indicate that both C-COIDS and COIDS, $\gamma = 1$ are most suitable for security sensitive applications, where in-vehicle network cannot tolerate intruders. Now, if we notice the plot of OTIDS, the intrusion detection alarm is generated at $t_d = 3.346$ sec, i.e., delay of 0.846 sec. In summary, the performance of C-COIDS and COIDS, $\gamma = 1$ have significantly improved



(a) Accumulated clock offsets derived by COIDS



(b) Intrusion detection rate



(c) Intrusion detection instant

Figure 6: DoS attack - Variations in accumulated clock offset, intrusion detection rate and intrusion detection instant.

compared to OTIDS. It is primarily due to the inclusion of SCPD technique and an active learning technique. Finally, the plot of COIDS, $\gamma = 3$ shows the maximum delay of detecting intrusion. A possible reason is that stricter security level requires more alarm signals from ECUs to declare the detection of an adversary. Particularly, in COIDS, $\gamma = 3$, there is a requirement of at least 3 ECUs to generate alarms for deciding an adversary.

7.3.3. Defense Against Impersonation Attack

To evaluate the capability of COIDS for defending impersonation attack on the CAN bus, we program ECU $\mathbb{Z}$ as attacker and impersonating ECU $\mathbb{W}$. We assume that ECU $\mathbb{Z}$ impersonating message ID 0x2C0, i.e., drive gear message and injecting periodic message from $t = 2.5$ sec at 0.001 sec intervals. Further, we program ECU $\mathbb{X}$ as the victim ECU and ECU $\mathbb{Y}$ as the monitoring ECU running COIDS.

Since the attacker ECU $\mathbb{Z}$ is impersonating ECU $\mathbb{W}$, so it sends the injected message at the same frequency as sent by the ECU $\mathbb{W}$. To identify the originating ECU for the message ID 0x2C0, we initially evaluate the probability mass function of the interval of message ID 0x2C0, before and after the attack was mounted. We plot only the significantly large values of the PMF before and after the attack in Figure 7(a). Figure 7(a) shows the abnormal deviation of PMF plot from the mean. This is due to the mistimed impersonation attack. Particularly, we observe from the plot that the mean of the PMF before and after the attack is 0.75 and 0.58, respectively.

We plot the changes of accumulated clock offset in the absence and presence of impersonating attack in Figure 7(b). It is interesting to show that in presence of impersonation attack the value of accumulated clock offset changes in the positive direction. It is because for an impersonating ECU it is difficult to control the arrival of response for a message within a particular time. As a matter of this significant change in accumulated clock offset,

the changed hypothesis, h_1, exceeds its threshold $h_{th} = 3$, i.e., COIDS declares detection of an intruder.

Similar to DoS attack, we measure the effectiveness of detecting the impersonation attack for all competing schemes, considering the average detection rate as performance metric. Figure 7(c) shows the average impersonation attack detection rate for all schemes. While simulating, we considered the same experimental setup as a DoS attack. Figure 7(c) shows that all schemes are successful in detecting impersonation attack after a certain time. The plot also shows that C-COIDS outperforms the other schemes in terms of average detection rate. Specifically, we notice that C-COIDS has the highest detection rate, whereas COIDS, $\gamma = 3$ has lowest detection rate. It is mainly due to the different number of votes that are required to finally convict an ECU as malicious.

Now, if we compare the performance of all schemes, C-COIDS and COIDS, $\gamma = 1$ significantly outperforms COIDS, $\gamma = 3$ and OTIDS. It is because both C-COIDS and COIDS, $\gamma = 1$ exploits the accumulated clock offset for detecting the intruder as well as includes the SCPD technique to reduce intrusion detection time and at least one alarm signal is required to declare an adversary as detected, whereas OTIDS exploits the time interval of message. It is quite clear that due to the intelligent attacker, time interval of message may not change. However, it is difficult for an intelligent attacker to keep unchanged accumulated clock offset. As far as the detection time is concerned in Figure 7(d), C-COIDS and COIDS, $\gamma = 1$ exhibit similar plot patterns and detect impersonation attack at 2.736 sec and 2.741 sec, respectively. Hence, the delay of detecting impersonation attack is 0.236 sec and 0.241 sec in C-COIDS and COIDS, $\gamma = 1$, respectively. It means, similar to DoS attack, the delay in detecting impersonation attack is slightly more in COIDS, $\gamma = 1$ compared to C-COIDS. This is primarily due to the stricter requirement for $\gamma = 1$ in COIDS compared to C-COIDS. Conversely, OTIDS and COIDS, $\gamma = 3$ detects impersonation attack at 3.472 sec and 3.687 sec, respectively. Thus, the delay of detecting impersonation attack is 0.972 sec and 1.187 sec in OTIDS and COIDS, $\gamma = 3$, respectively. Interestingly, we notice from Figure 7(d) that the detection time increases with the increase in security levels, i.e., γ. It is due to the fact that a higher security level signifies a stricter IDS, where more alarm signals are required from ECUs to finally convict an activity or ECU as malicious. Summarily, C-COIDS and COIDS, $\gamma = 1$ outperform both OTIDS and COIDS, $\gamma = 3$ in terms of quick impersonation attacker detection. Alternately, in C-COIDS and COIDS, $\gamma = 1$ have the least intruder tolerance capability than the other schemes.

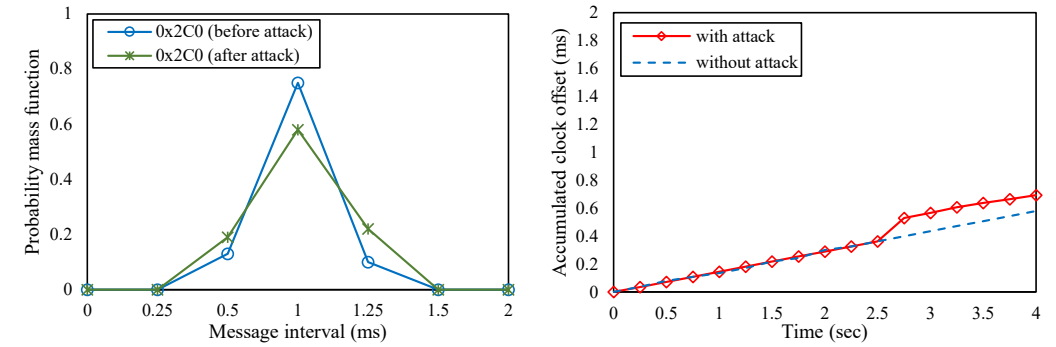
7.3.4. Defense Against Fuzzy Attack

In this section, we evaluate the capability of COIDS for defending fuzzy attack. We program ECU $\mathbb{Z}$ as attacker randomly injecting message IDs 0x2C0 and 0x5A2 at every 0.005 sec. Further, similar to the two earlier attack scenarios, we program ECU $\mathbb{X}$ as the victim ECU and ECU $\mathbb{Y}$ as monitoring ECU running COIDS.

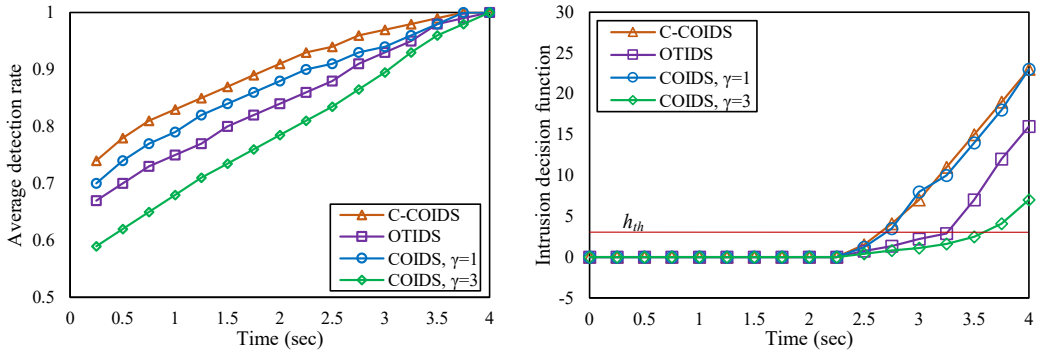
We plot the value of accumulated clock offset changes in the absence and presence of fuzzy attack in Figure 8(a). The plot shows that as soon as ECU $\mathbb{Z}$ mounts fuzzy attack, the value of accumulated clock offset suddenly changes in the positive direction. Particularly, the plot shows that fuzzy attack mounted at 2.5 sec. Now, due to injection of 0x2C0 and 0x5A2 messages, the accumulated clock offset abruptly changes from 0.937 sec to 1.367 sec.

Similar to the earlier two attacks, we measure the effectiveness of detecting impersonation attack for all competing schemes considering the average detection rate as the performance metric. We considered the same experimental setup as DoS attack while measuring the average detection rate. Figure 8(b) shows the average impersonation attack detection rate for all schemes. We notice from Figure 8(b) that all schemes are successful in detecting fuzzy attack after a certain time. We also notice from the plot that C-COIDS has the highest detection rate, whereas COIDS, $\gamma = 3$ has the lowest detection rate. It is mainly due to the different number of votes that are required to finally convict an ECU as malicious.

In Figure 8(c), we observe that the first set of message injections for the spoofing gear is



(a) Probability mass function of message intervals (b) Accumulated clock offsets derived by COIDS



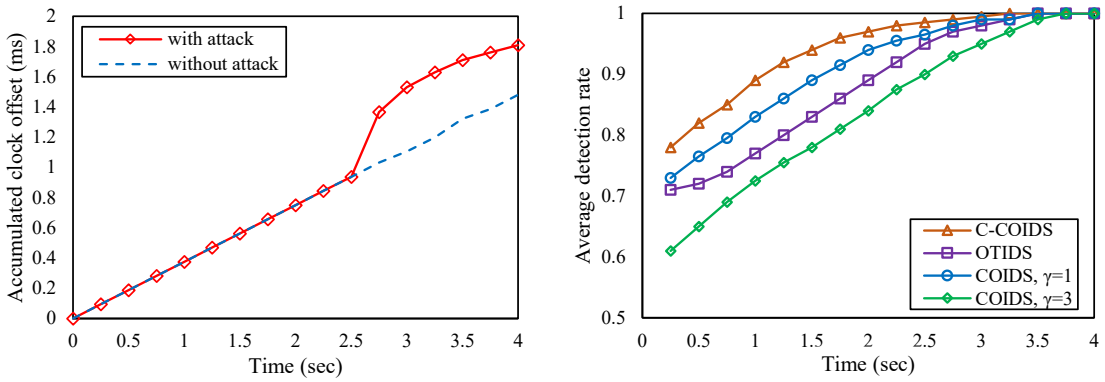
(c) Intrusion detection rate (d) Intrusion detection instant

Figure 7: Impersonation attack - Probability mass function of message intervals, variations in accumulated clock offset, intrusion detection rate and intrusion detection instant.

at 2.5 sec, whereas C-COIDS signals the alarm at $t_d = 2.719$ sec. It means that the detection delay of C-COIDS for the gear data injection is 0.219 sec, due to the inclusion of the SCPD technique to reduce the intrusion detection time. Like C-COIDS, COIDS, $\gamma = 1$ also show similar kind of plot as in both schemes alarm signal from at least one ECU is required to declare an adversary as detected. Now, if we compare the performance between C-COIDS, COIDS, $\gamma = 1$ and OTIDS, the detection delay of C-COIDS and COIDS, $\gamma = 1$ is much less than OTIDS. In particular, OTIDS signals the alarm at 3.392 sec, i.e., delay of 0.892 sec. Finally, as expected, COIDS, $\gamma = 3$ shows the highest detection delay, particularly, $(3.603 - 2.5) = 1.103$ sec. As explained previously, at higher security level, COIDS is more stricter, and thus, requires more alarm signals to convict an ECU as an adversary. It means that in COIDS where $\gamma = 3$, has more intruder tolerance capability than in COIDS where $\gamma = 1$.

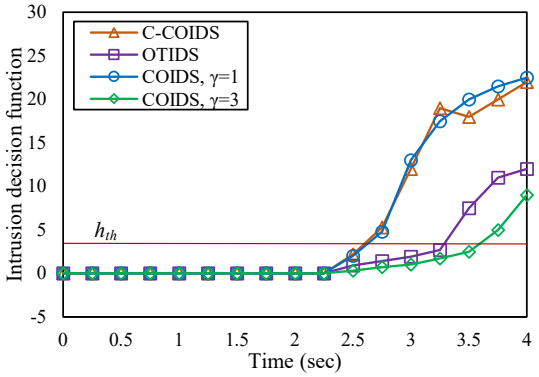
7.3.5. Power Efficiency

In this section, through simulation, we show how power consumption is minimized using our probability model (see Section 5.1). Here, we first introduce the power model used during simulation. In our power model, we determine the power consumption in an ECU as a function of its transceiver and controller. Particularly, we assume an ECU embeds a Microchip MCP2551 transceiver and a Microchip MCP2515 controller [42], where the controller is considered to require constant power irrespective of operating state. According to the data sheet of MCP2551 [43], the transmitter consumes more power than the receiver. Whereas idle state consumes equal power to that of receiving. Specifically, based on the values obtained from the data sheet, the power consumption for transmitting and receiving by an ECU in the CAN bus are 177 mW and 105 mW, respectively [42]. Here it is worth mentioning that although CAN bus supports a data transfer rate of 1 Mbps, it is unusual to use more than 50% of the maximum, due to its collision handling scheme. While measuring



(a) Accumulated clock offsets derived by COIDS

(b) Intrusion detection rate



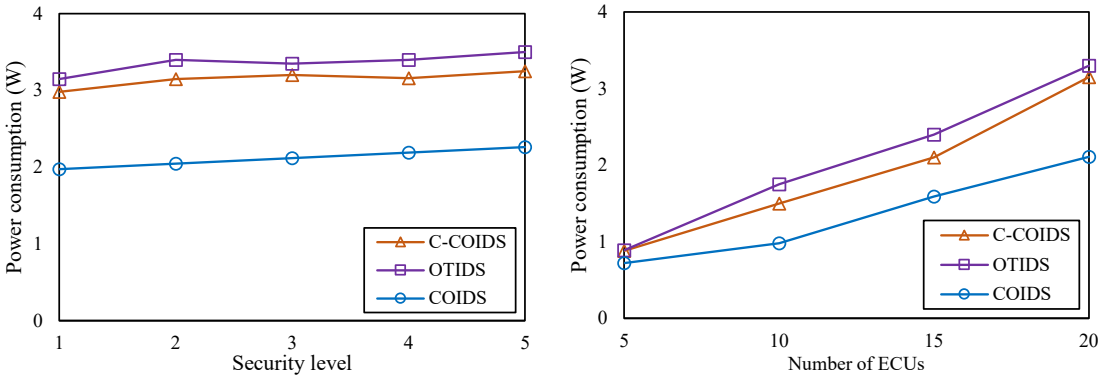
(c) Intrusion detection instant

Figure 8: Fuzzy attack - Variations in accumulated clock offset, intrusion detection rate and intrusion detection instant.

the power efficiency, we vary the number of ECUs from 5 to 20. We define the power efficiency as the minimum power consumed to provide the desired security level. We performed two sets of experiments for evaluating the power efficiency of the various schemes. One set of experiments measures power efficiency with varying desired security levels and the second one measures power efficiency with varying number of ECUs. Without loss of generality, we consider fuzzy attack scenario while performing these two sets of experiments.

Figure 9(a) depicts the average power consumption of various schemes with varying desired security levels. While simulating, we consider the number of ECUs connected through CAN bus as 20. For all three schemes, the power consumed by IDS is increasing linearly with the increase in security level, i.e., γ. It is obvious that to maintain a higher value of γ, comparatively more power needs to be consumed. However, the power saving in COIDS is clearly visible and quite significant. It is because, in both C-COIDS and OTIDS, ECUs are active all the time, irrespective of γ, resulting in significant power consumption. On the contrary, in COIDS, power consumption is significantly reduced by minimizing the active time duration of ECU based on the devised multiplayer cooperative game model. We notice from Figure 9(a) that the average power consumption in COIDS is 33.05% and 37.21% reduced compared to C-COIDS and OTIDS, respectively. Therefore, the results reveal that a significant amount of power is saved during the entire lifetime of the vehicle.

We next evaluate the power efficiency by varying the number of ECUs, and the results are plotted in Figure 9(b). While simulating, we set $\gamma = 3$ for all three schemes. We notice from the plot that the power consumption increases with the increase in number of ECUs for all three schemes. However, the power consumption in both C-COIDS and OTIDS is significantly more than COIDS. It is obvious as previously explained that, in C-COIDS and OTIDS, ECUs are active all the time, resulting in considerable power consumption. On the contrary, in COIDS, most of the ECUs remain in the idle state, resulting in significant



(a) Effect of security level

(b) Effect of number of ECUs

Figure 9: Average power consumption of different competing schemes.

power conservation. In Figure 9(b), we observe that COIDS reduces the average power consumption by 31.57% and 35.09% compared to C-COIDS and OTIDS, respectively. Here, again, we observe that the use of our designed multiplayer cooperative game model results in significant saving of power.

8. Conclusion

We present COIDS, a new anomaly based IDS for the in-vehicle network. COIDS monitors the intervals of periodic messages and exploits them to estimate the clock offset of the transmitter ECUs' clock for fingerprinting ECUs. COIDS then leverages the derived fingerprints to construct a baseline of ECU's normal clock behavior using an active learning technique. Based on the baseline of normal behavior, COIDS detects any abnormal deviation in clock offset via the CUSUM method. Further, to determine the exact time of intrusion, COIDS uses SCPD. As COIDS incessantly runs on every hardware-constrained ECU, we present the minimization of the active time duration of COIDS in an ECU as an optimization problem. We next devise a multiplayer cooperative game model where the main objective of the COIDS is to monitor the neighboring ECUs at a desired security level for detecting anomalous activities, whereas the secondary objective of the COIDS is to conserve power as much as possible. To attain these objectives, each ECUs has to cooperatively participate in monitoring its neighbors with a minimum probability. The evaluated results show that COIDS outperforms the state-of-the-art IDS, OTIDS under three most potential attacks on CAN, i.e. DoS, impersonation and fuzzy. The experimental results show that COIDS is not only effective in defending all these three attacks, but also ensures intrusion detection almost in real time. Further, the results show that our devised cooperative game model significantly improves the power efficiency of the ECU without compromising the performance. In the future, we intend to design a secure authentication scheme for ECUs on the legacy CAN bus by exploiting the covert channels. Further, we seek to investigate the performance of the COIDS under more sophisticated attacks, like cloaking attack [13].

References

[1] V. H. Le, J. D. Hartog, Z. Zannone, Security and privacy for innovative automotive applications: A Survey, Computer Communications 132 (2018) 17–41.

[2] S. Halder, A. Ghosal, M. Conti, Secure over-the-air software updates in connected vehicles: A survey, Computer Networks (2020) 107343.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, Comprehensive experimental analyses of automotive attack surfaces, in: Proc. of USENIX Security Symposium, 2011, pp. 77–92.

[4] L. Constantin, Researchers hack Tesla Model S with remote attack, Accessed on April 20, 2020. [Online]: http://www.pcworld.com/article/3121999/security/researchers-demonstrate-remote-attack-against-tesla-model-s.html.

[5] E. Weise, Chinese group hacks a Tesla for the second year in a row, [Online] Accessed on April 20, 2020. [Online]: https://eu.usatoday.com/story/tech/2017/07/28/chinese-group-hacks-tesla-second-year-row/518430001/.

[6] C. Miller, C. Valasek, Remote exploitation of an unaltered passenger vehicle, in: Black Hat USA, 2015, pp. 1–91.

[7] S. Nie, L. Liu, Y. Du, Free-fall: Hacking tesla from wireless to can bus, in: Black Hat USA, 2017, pp. 1–16.

[8] J. Liu, S. Zhang, W. Sun, Y. Shi, In-vehicle network attacks and countermeasures: Challenges and future directions, IEEE Network 31 (2017) 50–58.

[9] S. Mazloom, M. Rezaeirad, A. Hunter, D. McCoy, A security analysis of an in-vehicle infotainment and app platform, in: Proc. of 10th USENIX Workshop on Offensive Technologies, 2016, pp. 1–12.

[10] A. I. Radu, F. D. Garcia, LeiA: a lightweight authentication protocol for can, in: Proc. of European Symposium on Research in Computer Security (ESORICS), 2016, volume 9879 of LNCS, pp. 283–300.

[11] B. Groza, S. Murvay, A. V. Herrewege, I. Verbauwhede, Libra-can: Lightweight broadcast authentication for controller area networks, ACM Trans. on Embedded Computing Systems 16 (2017) 1–25.

[12] W. Choi, K. Joo, H. J. Jo, M. C. Park, D. H. Lee, VoltageIDS: low-level communication characteristics for automotive intrusion detection system, IEEE Trans. on Information Forensics and Security 13 (2018) 2114–2129.

[13] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, R. Poovendran, Shape of the cloak: Formal analysis of clock skew-based intrusion detection system in controller area networks, IEEE Trans. on Information Forensics and Security 14 (2019) 2300–2314.

[14] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, K. Li, A survey of intrusion detection for in-vehicle networks, IEEE Trans. on Intelligent Transportation Systems 21 (2020) 919–933.

[15] K. Cho, K. Shin, Fingerprinting electronic control units for vehicle intrusion detection, in: Proc. of USENIX Security Symposium, 2016, pp. 911–927.

[16] B. Groza, P. S. Murvay, Efficient intrusion detection with bloom filtering in controller area networks, IEEE Trans. on Information Forensics and Security 14 (2019) 1037–1051.

[17] M. Kneib, C. Huth, Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks, in: Proc. of ACM Conference on Computer and Communications Security (ACM CCS), 2018, pp. 787–800.

[18] K. Huang, Q. Zhang, C. Zhou, N. Xiong, Y. Qin, An efficient intrusion detection approach for visual sensor networks based on traffic pattern learning, IEEE Trans. on Systems, Man, and Cybernetics: Systems 47 (2017) 2704–2713.

[19] M. Basseville, I. V. Nikiforov, et al., Detection of abrupt changes: theory and application, volume 104, Prentice Hall Englewood Cliffs, 1993.

[20] T. Hoppe, S. Kiltz, J. Dittmann, Security threats to automotive can networks–practical examples and selected short-term countermeasures, in: Proc. of International Conference on Computer Safety, Reliability, and Security, 2008, pp. 235–248.

[21] K.-T. Cho, K. G. Shin, Viden: Attacker identification on in-vehicle networks, in: Proc. of ACM Conference on Computer and Communications Security (ACM CCS), 2017, pp. 1109–1123.

[22] S. U. Sagong, X. Ying, R. Poovendran, L. Bushnell, Exploring attack surfaces of voltage-based intrusion detection systems in controller area networks, ESCAR Europe (2018) 1–13.

[23] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, T. Chantem, Simple: single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks, in: Proc. of 35th Annual Computer Security Applications Conference, 2019, pp. 229–244.

[24] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, R. Poovendran, Cloaking the clock: emulating clock skew in controller area networks, in: Proc. of ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), 2018, pp. 32–42.

[25] J. Zhou, P. Joshi, H. Zeng, R. Li, Btmonitor: Bit-time-based intrusion detection and attacker identification in controller area network, ACM Trans. on Embedded Computing Systems (TECS) 18 (2019) 1–23.

[26] M. L. Han, B. I. Kwak, H. K. Kim, Anomaly intrusion detection method for vehicular networks based on survival analysis, Vehicular communications 14 (2018) 52–63.

[27] H. Lee, S. H. Jeong, H. K. Kim, Otids: A novel intrusion detection system for in-vehicle network by using remote frame, in: Proc. of 15th Annual Conference on Privacy, Security and Trust (PST), 2017, pp. 57–5709.

[28] M. Marchetti, D. Stabili, Anomaly detection of can bus messages through analysis of id sequences, in: Proc. of IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 1577–1583.

[29] H. K. Kalutarage, M. O. Al-Kadri, M. Cheah, G. Madzudzo, Context-aware anomaly detector for monitoring cyber attacks on automotive can bus, in: Proc. of ACM Computer Science in Cars Symposium, 2019, pp. 1–8.

[30] T. Yu, X. Wang, Topology verification enabled intrusion detection for in-vehicle can-fd networks, IEEE Communications Letters 24 (2020) 227–230.

[31] D. Mills, Network Time Protocol (Version 3) specification, implementation and analysis, Technical Report, University of Delaware, 1992, pp. 1-92.

[32] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, H. Kim, A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods, IEEE Trans. on Signal Processing 54 (2006) 3372–3382.

[33] A. G. Tartakovsky, A. S. Polunchenko, G. Sokolov, Efficient computer network anomaly detection by changepoint detection methods, IEEE Journal of Selected Topics in Signal Processing 7 (2012) 4–11.

[34] H. Olufowobi, U. Ezeobi, E. Muhati, G. Robinson, C. Young, J. Zambreno, G. Bloom, Anomaly detection approach using adaptive cumulative sum algorithm for controller area network, in: Proc. of ACM Workshop on Automotive Cybersecurity (AutoSec), 2019, pp. 25–30.

[35] C. Kiennert, Z. Ismail, H. Debar, J. Leneutre, A survey on game-theoretic approaches for intrusion detection and response optimization, ACM Computing Surveys (CSUR) 51 (2018) 1–31.

[36] B. Peleg, P. Sudhölter, Introduction to the theory of cooperative games, volume 34, Springer Science & Business Media, 2007.

[37] L. S. Shapley, A value for n-person games, Contributions to the Theory of Games 2 (1953) 307–317.

[38] J. Lemaire, Cooperative game theory and its insurance applications, ASTIN Bulletin: The Journal of the International Actuarial Association 21 (1991) 17–40.

[39] S. Halder, M. Conti, S. K. Das, Coids: A clock offset based intrusion detection system for controller area networks, in: Proc. of 21st ACM International Conference on Distributed Computing and Networking (ACM ICDCN), 2020, pp. 1–10.

[40] H. Lee, S. H. Jeong and H. K. Kim, CAN Dataset for intrusion detection (OTIDS), [Online]: http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset, 2018. Accessed on January 15, 2020.

[41] J. Matsumura, Y. Matsubara, H. Takada, M. Oi, M. Toyoshima, A. Iwai, A simulation environment based on omnet++ for automotive can–ethernet networks, Analysis Tools and Methodologies for Embedded and Real-time Systems (2013) 1–44.

[42] K. French, Energy Consumption of In-Vehicle Communication in Electric Vehicles: A comparison between CAN, Ethernet and EEE, Ph.D. thesis, Linköping University, 2019.

[43] N. Balbierer, T. Waas, J. Noebauer, J. Seitz, Energy consumption of ethernet compared to automotive bus networks, in: Proc. of 9th International Workshop on Intelligent Solutions in Embedded Systems, 2011, pp. 61–66.