Chapter

Advancements in Vehicle Intrusion Detection Systems: Enhancing Cybersecurity in Intelligent Transportation

Mahdi Sahlabadi, Md Rezanur Islam,
Munkhdelgerekh Batzorig and Kangbin Yim

Abstract

Advancements in in-vehicle Intrusion Detection Systems (IDS) are critical to addressing cybersecurity challenges in modern intelligent transportation. This chapter explores vulnerabilities inherent in Controller Area Network (CAN) protocols, such as DoS, spoofing, and replay attacks, and their impacts on vehicular systems. By analyzing the LISA Vehicle Dataset collected through innovative techniques like gateway-based methods and custom testbeds as injection tools, the study provides insights into abnormal behaviors and attack patterns. It evaluates IDS approaches, including rule-based and machine learning-based systems, highlighting the strengths of deep learning models like RNNs and CNNs for robust anomaly detection. Feature selection and optimized chunking methodologies, such as mutual information analysis, are discussed to enhance IDS efficiency while reducing computational demands. The chapter also emphasizes real-world simulation using synchronized CAN datasets and testbeds, contributing to the development of scalable, real-time IDS frameworks for secure and reliable vehicular communication.

Keywords: vehicle intrusion detection systems (IDS), controller area network (CAN) security, in-vehicle network (IVN) vulnerabilities, deep learning for anomaly detection, feature selection and chunking in IDS

1. Introduction

This chapter delves into the advancements in Vehicle Intrusion Detection Systems (IDS), addressing the critical need for enhanced cybersecurity in intelligent transportation systems. It builds on practical research conducted in the LISA laboratory, incorporating findings from real-world testbeds, including Connected and Autonomous Vehicles (CAV) and Vehicle-to-Everything (V2X) systems. The chapter begins with the foundational Vehicles Dataset, detailing data collection processes, environments, and the characteristics of the collected datasets, which form the basis for understanding IDS development. It then transitions into an exploration of In-Vehicle Network (IVN) Vulnerabilities, highlighting risks within automotive networks and presenting practical solutions such as simulation environments and V2X data frameworks. Finally, the chapter focuses on Data Analysis and Feature Selection, discussing methodologies for analyzing data, identifying critical features, and building robust intrusion detection models using advanced machine learning techniques. Together, these sections provide a cohesive narrative, blending theoretical insights with practical experimentation to offer actionable solutions for mitigating cyber threats in modern vehicular networks, ensuring their safety and reliability in an increasingly connected and autonomous era.

2. Vehicles dataset

Modern vehicles use various communication protocols, such as FlexRay, Media Oriented Systems Transport (MOST), Local Interconnect Network (LIN), and automotive Ethernet (100BASE-T1). However, the Controller Area Network (CAN) message protocol is the most widely used because of its cost-effectiveness and efficient data transfer. Despite its advantages, the CAN protocol lacks robust security features, which can lead to severe risks, including damage to the vehicle and potential harm to human life. To address these vulnerabilities, vehicles require security systems, such as IDS [1].

The LISA Vehicle Dataset for Intrusion Detection is a collection of CAN message datasets gathered from actual vehicles while they are in motion. During the data collection, attack messages were intentionally injected into the vehicle's internal network. A specialized tool was used to collect the resulting dataset, creating an attack dataset for analysis on Volkswagen GOLF (2014), KIA SOUL (2015), BMW 118d (2015), KIA SOUL Booster (2019), Tesla Model 3 (2020), Genesis G80 (2022), as shown in **Figure 1**.

2.1 Dataset collection

2.1.1 Collection environment

The OBD-II port, or onboard diagnostics, is typically used to collect CAN messages. It monitors and controls a vehicle's operation. Initially, it was used to improve

Figure 1.
Test vehicles used in the study: Volkswagen GOLF (2014), KIA SOUL (2015), BMW 118d (2015), KIA SOUL Booster (2019), Tesla Model 3 (2020), and Genesis G80 (2022).

the efficiency of vehicle maintenance. However, due to security concerns related to using the OBD-II port, newer vehicles have limited data transfer and installed filters on the port. The other method we found for collecting IVN data is the gateway-based EDA (Gateway ECU Direct Approach) method [2]. The IVN gateway is connected to all vehicle networks to maintain the vehicle system. In this method, we first locate the gateway position based on the manufacturer's guidelines, as shown in **Figure 2**. We then compared data collection from the OBD-II port and the gateway-based collection method. As shown in **Figure 3**, we can see that the gateway-based method collects an average of 4340 messages per millisecond. The OBD-II port collects an average of only 50 messages per millisecond [3].

We developed our own collection and injection tools for CAN messages using APIs provided by PEAK and VECTOR collection tools. This allowed us to collect and analyze various CAN message data, providing a wealth of information.

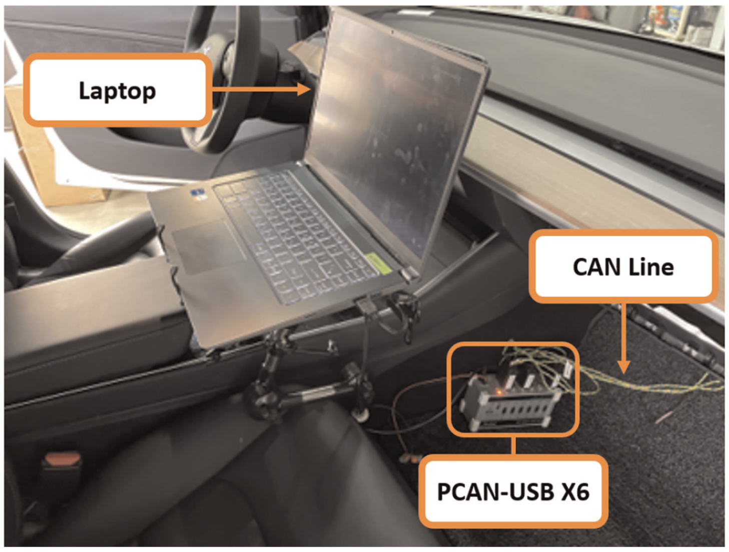


Figure 2.
Illustration of the data collection method via the ICU.

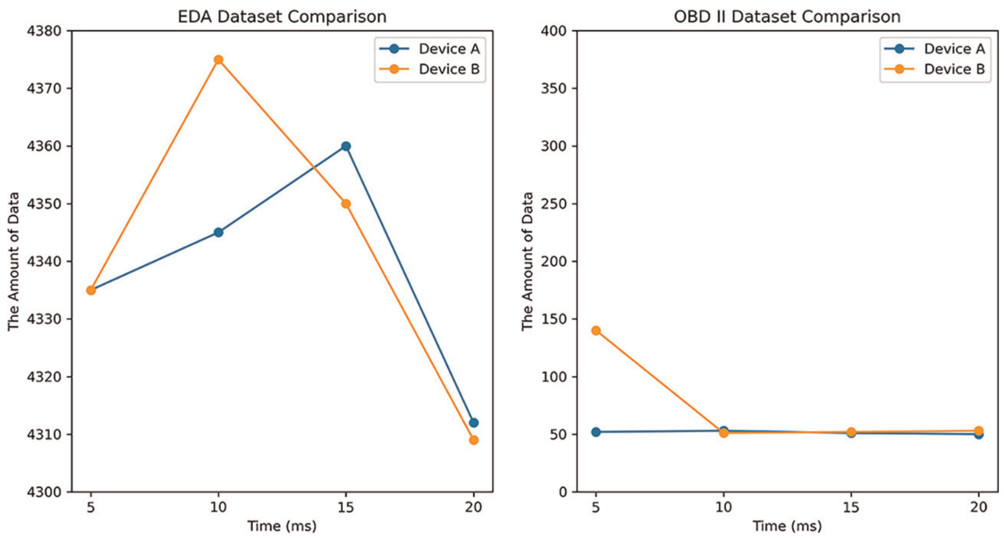


Figure 3.
Data collection difference between ICU and BOD-II.

2.1.2 Data collection scenario

Establishing an accurate scenario for dataset collection is crucial, as it is just as important as building the collection environment. Data analysis could lead to accurate results if the dataset needs to be corrected. If such accurate results are used in machine learning or deep learning models, it could result in misdetection, making it easier to evaluate the model. Therefore, it is essential to collect the dataset from the beginning. We begin by configuring the dataset collection settings within the vehicle. Once the environment is set up, we revert the vehicle to its original state before the driver unlocks the door, and after that, we begin the data collection. During the collection, we have an attack scenario for each attack based on its specification. The data collection process continues until the driver shuts down the engine, exits the vehicle, locks it, and takes the smart key with them [4]. **Figure 4** demonstrates the Stages of the vehicle state: Stage 0 (Original state before it is unlocked), Stage 1 (Before the engine is started), Stage 2 (After the engine is started), Stage 3 (After the engine is shut down), Stage 4 (After the vehicle is locked).

2.1.3 Dataset description

In a vehicle, as shown in **Figure 5**, there are typically multiple CAN networks. Each of these networks is responsible for different functions within the vehicle's electronic control unit (ECU), allowing various systems within the vehicle to communicate. For example, the Chassis CAN (C-CAN) network communicates at a high speed of 500 Mbit/s and controls systems such as the Transmission Control Unit (TCU) and Anti-lock Braking System (ABS). The Body CAN (B-CAN) network controls systems unrelated to the vehicle's power components, such as the Body Control Module (BCM), Smart Key Module, and Power Window. The Multimedia CAN (M-CAN) network operates slower than 100 Mbit/s and controls in-vehicle multimedia systems such as Audio, Video, Navigation (AVN) systems and instrument panels.

The Diagnosis CAN (D-CAN) network is used for vehicle diagnostics, while the Powertrain CAN (P-CAN) network controls the vehicle's power transmission system. These different CAN networks work together to ensure that various systems within

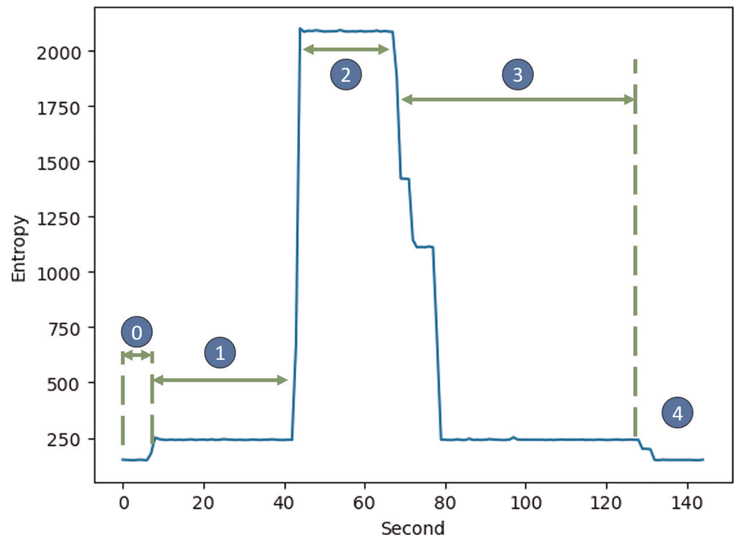


Figure 4.
Event triggered data scenario.

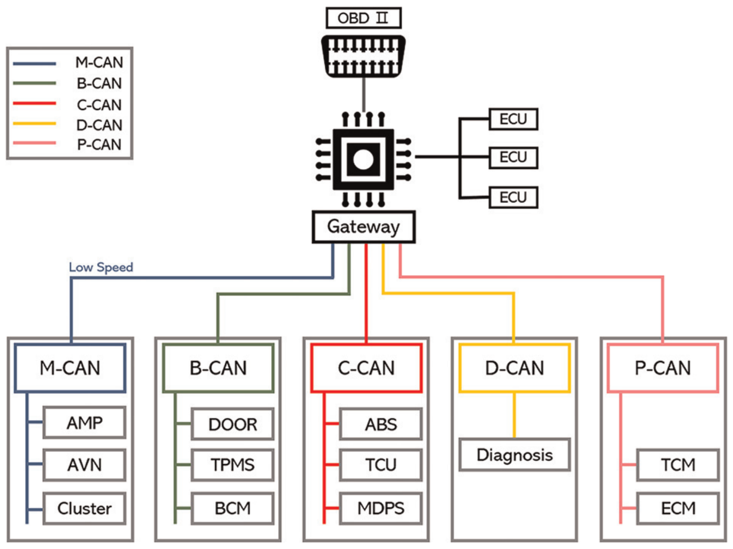


Figure 5.
IVN structure.

the vehicle can operate smoothly and efficiently. A vehicle's gateway is a system that plays a critical role in safely transmitting data between different CAN networks. When designing distributed CAN networks, it is essential to centralize information collection on vehicle networks. This makes expanding high-speed networks such as CAN-FD (CAN-Flexible D), Ethernet, and FlexRay easier. Additionally, these gateways can route different CAN channels flexibly based on traffic and environmental conditions. This allows for the transmission of signals between CAN domains and the ability to check the entire vehicle or devices for abnormal operation.

Our data collection contains columns such as “Time_offset_ms,” “Type,” “CAN_ID,” “DLC,” “Data Field,” and “Label.” The CAN FD dataset may include additional columns specific to the CAN FD protocol that are not found in other datasets. A detailed explanation of these columns is provided in **Table 1**, while an example of the collected dataset is shown in **Figure 6**.

2.2 CAN message details

See **Table 1**.

3. In-vehicle network vulnerabilities

In-vehicle networks (IVNs) rely on the Controller Area Network (CAN) protocol as the backbone for communication between Electronic Control Units (ECUs) [5]. CAN operates without built-in security mechanisms, making it vulnerable to a range of cyber-attacks [6]. The protocol transmits messages in a broadcast manner, allowing all ECUs to receive every message on the network [6]. However, ECUs process messages based on the priority of their identifiers (IDs), which determines the sequence of handling. This lack of authentication and encryption in CAN communication opens the door to various attack vectors, such as Denial of Service (DoS), Fuzzing, Replay, Spoofing, Malfunction, and Impersonation, which exploit the broadcast nature of the network [2, 7, 8]. Additionally, ECU-specific attacks like Suspension, Fabrication, and Masquerading can manipulate or disrupt the functionality of targeted ECUs [9].

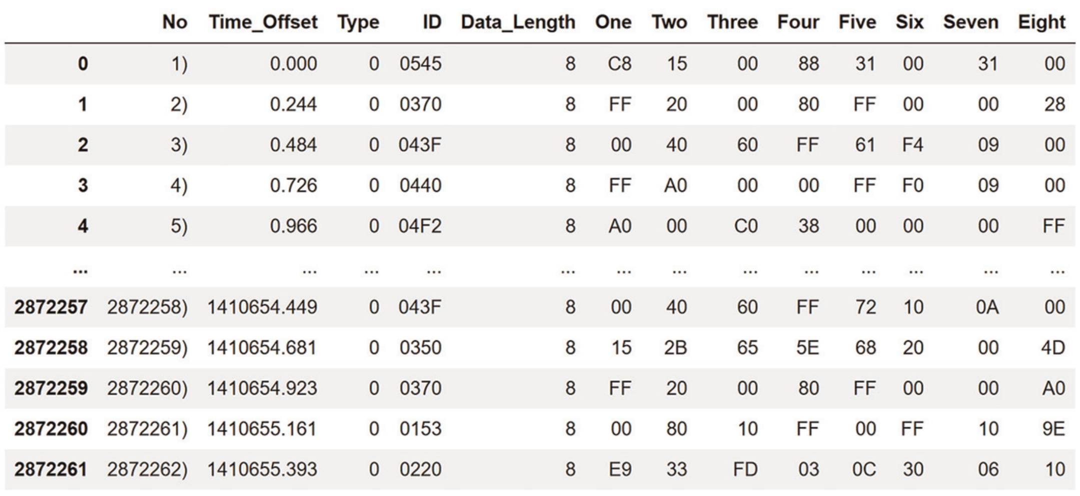
Field name	Description	Application and value
Message Number	The message number indicates the order in which messages were collected by the collection tool.	Provides the exact order of each CAN message. Values start from 1 and continue until the collection tool stops recording.
Time Offset	Displays the timestamp of each message occurrence, measured in milliseconds.	Provides precise timing of CAN message collection. Values start from 0 and continue in float format until the collection tool stops.
Type	Identifies different message categories and explains their functions and usage.	Categorizes and filters messages based on type. Common values include: • 0x00: Standard Frame (11-bit identifier, Rx). • 0x01: Remote-Transfer-Request Frame. • 0x02: Extended Frame (29-bit identifier). • 0x04: FD frame (CiA Specs). • 0x08: FD bit rate switch. • 0x10: FD error state indicator. • 0x40: Error frame. • 0x80: PCAN status message.
CAN ID	Serves as an identifier for each ECU and determines priority when multiple messages are received simultaneously.	Messages can be filtered and categorized using hexadecimal CAN IDs. Confidential information links CAN ID to ECU.
Data Length Code (DLC)	Indicates the size of the data field in bytes.	Represents the number of bytes in the payload (0–8). Helps in understanding message content.
Data Bytes	Hold the information transmitted by the ECU. Can contain up to 8 bytes depending on DLC.	Analyze signals packaged in the payload: • Constants (unchanging values). • Multi-Values (changing bits, e.g., door status). • Counters (cyclic counters for transmitted messages). • Check Codes (CRC for error prevention). • Physical Values (real-world values). Bytes are represented as hexadecimal (00 to FF). NaN values are replaced with −1.

Table 1.
Detailed description of CAN message fields.

Addressing these vulnerabilities is critical to ensuring the security and reliability of IVNs, especially in connected and autonomous vehicle systems.

3.1 Denial of service (DoS) attack

- *Definition*: A DoS attack involves flooding the CAN bus with high-priority messages, monopolizing bus access, and preventing legitimate ECUs from communicating.



	No	Time_Offset	Type	ID	Data_Length	One	Two	Three	Four	Five	Six	Seven	Eight
0	1)	0.000	0	0545	8	C8	15	00	88	31	00	31	00
1	2)	0.244	0	0370	8	FF	20	00	80	FF	00	00	28
2	3)	0.484	0	043F	8	00	40	60	FF	61	F4	09	00
3	4)	0.726	0	0440	8	FF	A0	00	00	FF	F0	09	00
4	5)	0.966	0	04F2	8	A0	00	C0	38	00	00	00	FF
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2872257	2872258)	1410654.449	0	043F	8	00	40	60	FF	72	10	0A	00
2872258	2872259)	1410654.681	0	0350	8	15	2B	65	5E	68	20	00	4D
2872259	2872260)	1410654.923	0	0370	8	FF	20	00	80	FF	00	00	A0
2872260	2872261)	1410655.161	0	0153	8	00	80	10	FF	00	FF	10	9E
2872261	2872262)	1410655.393	0	0220	8	E9	33	FD	03	0C	30	06	10

Figure 6.
CAN dataset examples.

- *Impact on CAN*: This disrupts normal operations by overwhelming the bus, leading to delays or complete loss of communication.
- *Example*: An attacker floods the bus with high-priority messages such as 0x100, causing critical ECU messages like brake or airbag signals to be delayed or dropped.

3.2 Fuzzing attack

- *Definition*: Random or malformed CAN frames are injected into the network, using a range of low- to high-priority IDs with varying payloads, to exploit undefined behaviors or crash ECUs.
- *Impact on CAN*: Can cause unpredictable ECU behavior, crashes, or trigger unintended actions in the vehicle.
- *Example*: An attacker sends random frames with varying priorities, e.g., ID: 0x050, Payload: [0xFF, 0xAA, 0x00, 0xEE] and ID: 0x200, Payload: [0x01, 0x02, 0x03, 0x04], to identify vulnerabilities in specific ECUs such as the engine or infotainment system.

3.3 Replay attack

- *Definition*: Captured CAN frames are retransmitted to mimic legitimate ECU communication, leading to repeated or unauthorized actions.
- *Impact on CAN*: Allows an attacker to replicate previous actions, such as unlocking doors or disabling alarms.
- *Example*: An attacker captures and replays a frame ID: 0x250, Payload: [0x01, 0x00, 0x01, 0xFF] used for door unlocking, causing the doors to unlock repeatedly.

3.4 Malfunction attack

- *Definition*: A malfunction attack targets specific CAN IDs extracted from the network and manipulates the associated data fields. By injecting crafted messages with altered payloads or values, the attacker induces abnormal behavior in the vehicle's ECUs.

- *Impact on CAN*: This type of attack disrupts normal vehicle operation by causing unintended actions, such as incorrect sensor readings or unexpected system responses.

- *Example*: An attacker manipulates the data field of a CAN message, such as ID: 0x153, by setting certain bytes to [0x00, 0x00, 0x00, 0x00], resulting in repeated beeping sounds in the vehicle. Similarly, modifying the third byte of ID: 0x43F causes the headlights to blink and the engine/emissions warning light to activate.

3.5 Impersonation attack

- *Definition*: An impersonation attack occurs when an attacker disables a legitimate ECU and replaces it with a malicious node that mimics the behavior of the original ECU. The impersonating node sends spoofed frames or responds to remote frames to appear as the legitimate ECU.

- *Impact on CAN*: This attack compromises the authenticity and integrity of messages, allowing the attacker to override legitimate commands without disrupting overall network availability.

- *Example*: An attacker disables the legitimate steering ECU and replaces it with a malicious node. The impersonating node sends spoofed frames, such as ID: 0x110, Payload: [0x01, 0x02, 0x03, 0x04], causing unintended changes in the vehicle's steering direction.

3.6 Spoofing attack

- *Definition*: Similar to impersonation, spoofing injects fake frames with valid-looking IDs to mislead ECUs about their origin.

- *Impact on CAN*: Deceives ECUs into executing unauthorized actions or sending false responses.

- *Example*: A spoofed frame ID: 0x400, Payload: [0xAA, 0xBB, 0xCC, 0xDD] tricks the vehicle into thinking the engine temperature is normal when it is overheating.

3.7 Suspension attack

- *Definition*: Frames are manipulated to stop a specific ECU from communicating by using higher-priority messages.

- *Impact on CAN*: Disables the targeted ECU's functionality, potentially affecting critical vehicle systems.

- *Example*: Continuous injection of high-priority frames such as ID: 0x050 prevents the airbag ECU (ID: 0x120) from transmitting safety-critical messages.

3.8 Fabrication attack

- *Definition*: A fabrication attack occurs when an attacker uses a compromised ECU to inject entirely fake messages into the CAN network. These forged messages typically mimic legitimate ones but include maliciously altered payloads.

- *Impact on CAN*: Misleads ECUs by introducing conflicting or non-existent data, potentially causing safety-critical malfunctions or erratic behavior in the vehicle.

- *Example*: An attacker compromises ECU A to inject fake speedometer messages with ID: 0x500, Payload: [0x00, 0x01, 0x00, 0xFF] at a higher frequency than the legitimate transmitter, ECU B. This causes receiving ECUs to predominantly process the attacker's fabricated messages, leading to false speed readings on the dashboard.

3.9 Masquerading attack

- *Definition*: A masquerading attack combines suspension and impersonation, where the attacker disables a legitimate ECU and injects messages at the original frequency from a compromised ECU, mimicking the legitimate one.

- *Impact on CAN*: Bypasses protection mechanisms by maintaining the expected timing of messages, deceiving receiving ECUs into treating the forged messages as authentic.

- *Example*: An attacker uses ECU A to inject forged RPM messages with ID: 0x600 and malicious payloads, while suspending ECU B (the legitimate sender). This prevents contradictions in timing or signal patterns, causing the dashboard to display false RPM values.

The impacts of various attacks on the Controller Area Network (CAN) manifest as anomalies in the ID sequence, payload sequence, and time gap of transmitted messages. Attacks such as Denial of Service (DoS) and Suspension disrupt the ID sequence by overwhelming the bus with high-priority frames, thereby preventing legitimate messages from being processed. Replay, Impersonation, and Masquerading attacks manipulate the payload sequence by injecting previously captured, spoofed, or maliciously crafted data to mimic legitimate ECUs and execute unauthorized actions. Fuzzing attacks corrupt the payload by injecting random or malformed frames across a range of low to high-priority IDs, while Malfunction attacks specifically target CAN IDs to alter data fields, inducing abnormal ECU behavior. Fabrication attacks introduce entirely fake messages that mislead ECUs into processing non-existent or harmful commands. Additionally, attacks such as DoS and Replay significantly alter the time gap between consecutive frames, either by flooding the network with excessive messages or delaying the transmission of legitimate ones. These disruptions in ID sequencing, payload integrity, and timing underscore the necessity of robust IDS to effectively identify and mitigate malicious activities in CAN networks, ensuring the security and reliability of in-vehicle communications.

3.9.1 Simulation in loop

The quality of an attack dataset significantly influences the effectiveness of an IDS. However, collecting attack data from a real vehicle involves substantial safety risks, particularly when the vehicle is in motion. To address these challenges, we developed a controlled testbed environment utilizing components from real vehicles, as illustrated in **Figure 7**. This setup enables safe and controlled data collection while effectively simulating various vehicle states.

3.9.1.1 Testbed configuration

The testbed is constructed using actual vehicle parts, following detailed circuit diagrams and manuals. These components are interconnected to create a functional simulation of a vehicle's communication system, providing a safe platform for attack dataset generation. By following these manuals and circuit schematics, we assemble and configure the testbed to replicate the vehicle's environment, ensuring the accuracy and reliability of the collected data. To simulate the vehicle, we need to inject the dataset that is collected from the vehicle into the testbed. In order to simulate the environment when the dataset is collected, the video is also recorded as well. To synchronize those CAN messages and videos, the first injection should be in the testbed. The injection tool is built in Python using the PCAN library, which is supported by the peak system. At the same time, the video should be played at the same time with the injection tool. All those tasks are handled by a thread since the thread is the method through which two different tasks run at the same time. CAN message datasets are inherently influenced by the vehicle's model and manufacturer. Different vehicle types exhibit unique CAN message characteristics, requiring tailored datasets for each type. To address this, we built separate testbeds for each vehicle model and manufacturer in our collection. This enables us to simulate various vehicles accurately and inject attack messages during the simulation to generate a diverse range of attack datasets. To simulate real vehicle conditions, we utilize data collected

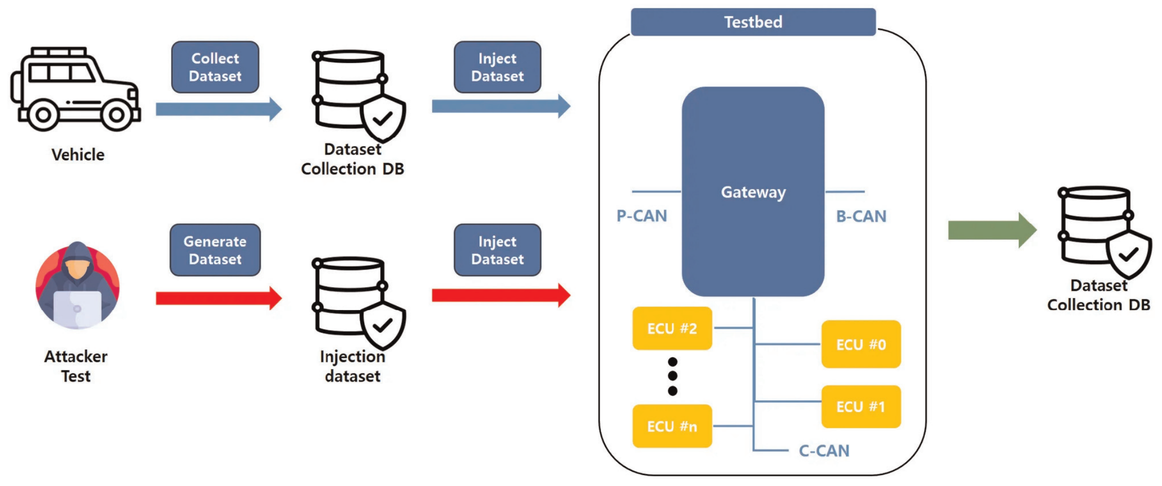


Figure 7.
Data collection overview.

from actual vehicles and replicate these conditions within the testbed. Each testbed corresponds to a specific vehicle type, ensuring that we capture the necessary variation in-vehicle data. CAN message dataset generation is different based on vehicle model and manufacture. However, we need different types of vehicle datasets. In order to do that, we have built different vehicle types of testbeds. For simulation purposes, the laboratory has a total of six different testbeds from six different vehicles, which are KIA SOUL, KIA SOUL BOOSTER, Genesis G80, etc., as shown in **Figure 8**. Each testbed corresponds to a specific vehicle type, ensuring that we capture the necessary variation in-vehicle data. Additionally, to recreate the environmental context, a synchronized video recording from the data collection process is also incorporated. This synchronized playback ensures accurate simulation of real-world conditions. The CAN message dataset is injected into the testbed using a custom Python-based injection tool. This tool is built using the PCAN library, a reliable framework provided by Peak Systems [6]. The injection replicates the exact sequence and timing of the original CAN messages, ensuring fidelity in simulation. Simultaneously, the recorded video is played back alongside the injection process. This synchronization is crucial to provide a comprehensive understanding of the vehicle's operational state during the original data collection. Multithreading is employed to manage the concurrent execution of CAN message injection and video playback. This approach ensures that both tasks run simultaneously and are synchronized effectively. Threads enable efficient handling of these independent yet interrelated processes, maintaining consistency throughout the simulation.

3.9.2 V2X data

LISA aims to develop a system capable of collecting and transmitting real-time vehicle data (**Figure 9**), including sensor (e.g., LiDAR) and in-vehicle network (IVN) data, to a cloud server using wireless communication technologies such as LTE-V2X and DSRC. Two approaches were explored: the first involves using an LTE-enabled Road-Side Unit (RSU) to collect and transmit data to the cloud, while the second involves injecting sensor data into custom V2X messages transmitted via DSRC and received by an RSU for cloud upload. Challenges include hardware limitations of On-Board Units (OBUs) that lack C-V2X capabilities, outdated SDKs, and incomplete

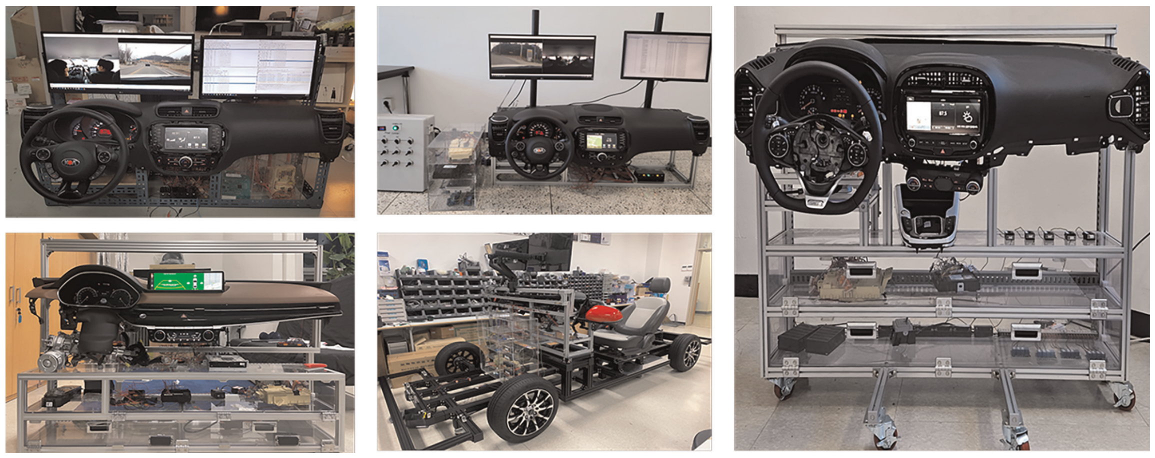


Figure 8.
Various testbeds.

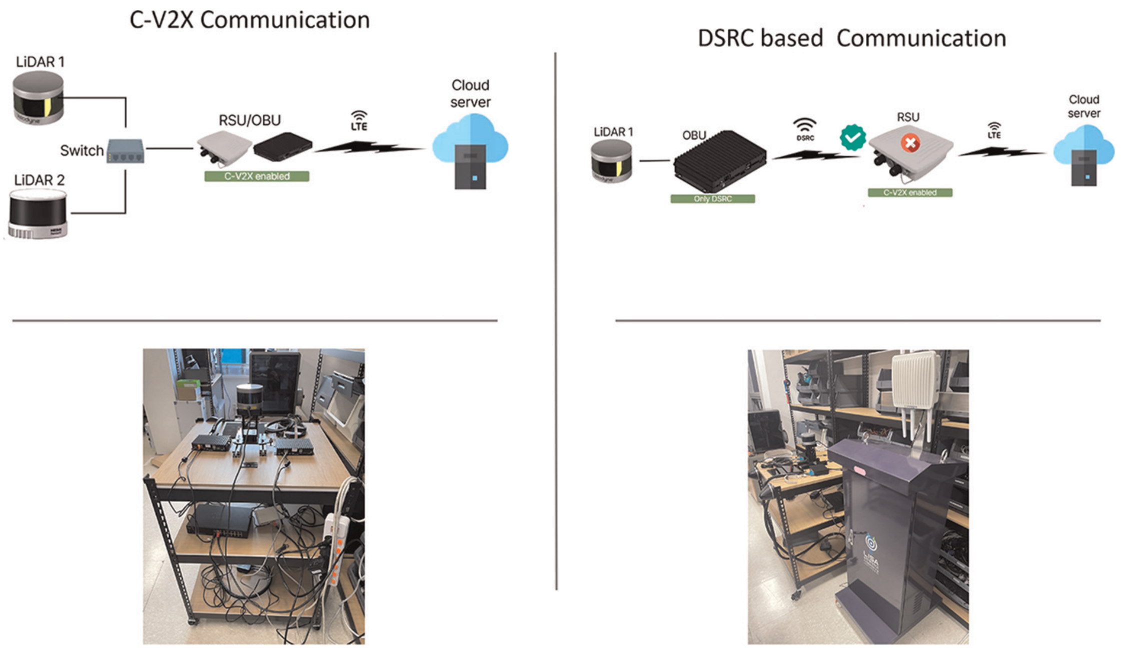


Figure 9.
V2X collection environment.

product documentation from Commsignia. The LISA overcame these hurdles by modifying existing software, cross-compiling binaries for the ARMv7 architecture, and dynamically linking libraries extracted from the devices.

In the first approach, an intermediary switch connects LiDAR sensors to the RSU, which uses an LTE connection to send data to a cloud server. A second approach utilizes OBUs to create custom V2X messages with sensor data. These messages are sent through DSRC to an RSU, which forwards them to the cloud. The team successfully injected sensor data into V2X messages using modified software and verified data transmission through packet analyzers. However, the issue of extracting sensor data at the RSU for cloud transmission remains unresolved. The project also includes developing methods to sniff WSM packets directly from DSRC antennas and enhancing LTE connectivity by integrating new hardware or using alternative LTE-enabled devices.

Future developments focus on refining data collection and synchronization for high-performance cloud processing. The team plans to explore server frameworks such as Flask or asyncio, real-time data processing tools like Apache Kafka, and database solutions like MySQL or MongoDB for data storage and retrieval. Solutions to address current limitations, such as obfuscated packet capture apps and hardware constraints, are also under consideration. These efforts aim to create a robust, scalable system for real-time vehicular data analysis and integration into intelligent transportation systems.

4. Data analysis and feature selection

4.1 Data analysis

Intrusion detection in CAN networks can be approached through various methods, primarily rule-based systems and machine learning-based models [2]. Both approaches rely heavily on extensive data analysis to ensure effectiveness. Rule-based methods require thorough analysis to define precise rules that can identify abnormal behaviors, while machine learning-based systems depend on detailed feature extraction and accurate data labeling to train models capable of distinguishing between normal and attack scenarios. This foundational data analysis is critical for identifying patterns, establishing baselines, and enabling robust detection mechanisms, making it a vital step in developing any reliable intrusion detection system.

In analyzing CAN data to distinguish between normal and attack scenarios, several key factors highlight the impact of malicious activities. Under normal conditions, the data generation frequency for specific vehicles remains consistent [10], determined by the manufacturer's ECU functionalities. For specific CAN IDs, there is a well-defined time gap interval in which these IDs generate messages or share status updates with other ECUs, maintaining system integrity [11]. However, during an attack, this frequency becomes irregular, and the time gap intervals for specific CAN IDs deviate from their expected ranges [2]. Additionally, sequential patterns in CAN IDs, which reflect the interactions between ECUs, are disrupted when an attack is initiated, breaking the order crucial for functionality. Furthermore, the payload, containing four types of information as specified by the CAN DBC [12], shows significant anomalies in its distribution during an attack, further indicating system compromise. These disruptions across frequency, time gaps, sequential patterns, and payload distribution collectively provide clear indicators of malicious activities within the CAN network.

Data analysis for CAN networks often relies on platforms such as Python and Pandas, which are widely utilized in existing studies for their efficiency and flexibility [11]. To effectively analyze CAN data, which is represented in hexadecimal format, it is essential to first transform the data into numerical form. This transformation can involve converting hexadecimal values into decimal or binary formats or labeling the data with specific numerical categories to facilitate analysis [13, 14]. Among the commonly used techniques, comparative analysis, statistical analysis, and correlation analysis stand out. In comparative analysis, patterns and characteristics are extracted from normal data and compared to attack data to identify anomalies. Statistical analysis focuses on evaluating data distributions and trends to detect deviations introduced by attacks [1]. Correlation analysis, typically employing Pearson correlation, examines the relationships between variables to uncover irregularities [15]. By integrating these transformation and analysis methods, researchers can build a robust framework for detecting and understanding abnormal behaviors within CAN networks.

4.2 Feature selection

4.2.1 Feature selection processes

Designing an IDS for IVNs presents unique challenges due to the limited power resources inherent in such systems. High computational demands can significantly hinder the feasibility of deploying IDS on IVNs, making efficiency a critical factor. At the same time, response time and model generalization are crucial requirements for effective intrusion detection, as delayed responses or overfitting can compromise vehicle safety and system reliability. Consequently, the design of an IDS must achieve a delicate balance between computational power demands, response time, and detection accuracy. A key step in attaining this balance is minimal feature selection, which ensures the IDS operates efficiently while maintaining high detection performance.

Feature selection is pivotal in reducing computational overhead and improving response time without sacrificing accuracy. Among various techniques, wrapper methods [16] are particularly effective due to their ability to iteratively evaluate and select the most relevant feature subsets for a given model. Wrapper methods are categorized into three main types:

- *Forward selection*: This method begins with an empty feature set and iteratively adds features one by one. At each step, the feature that most improves the model's performance is included in the set. The process continues until adding more features no longer enhances performance or meets the desired trade-offs. Forward selection is computationally efficient and well-suited for scenarios where lightweight models are required.

- *Backward elimination*: In this method, the process starts with the full feature set, and features are removed one at a time based on their significance. At each step, the least significant feature (i.e., the feature contributing the least to the model's performance) is excluded. This approach is useful when the dataset contains irrelevant or redundant features, but it can be computationally demanding, especially for larger feature sets.

- *Exhaustive search*: This method evaluates all possible combinations of features to identify the subset that yields the best performance. Unlike the other two methods, exhaustive search guarantees the optimal feature set by exploring every possible combination. While it is computationally expensive and often impractical for datasets with a large number of features, it becomes viable for smaller datasets, such as CAN data, which consists of manageable feature sets. Exhaustive search ensures that the selected subset delivers the highest possible accuracy, making it ideal when computational resources are available or when the feature set is limited.

In CAN frames, there are a total of eleven feature columns, including CAN ID, Data Length Code (DLC), eight payload bytes, and time intervals between frames [13]. Given this relatively small feature set, an exhaustive search emerges as the most suitable method for selecting the optimal combination of features. Based on exhaustive evaluation, the best combination includes CAN ID and time intervals [2]. These features are particularly effective in detecting anomalies, as they reflect the expected communication patterns, timing behaviors, and interaction sequences of ECUs. For example, the CAN ID sequence and time gap intervals are highly useful for identifying abnormalities. However, for ECU-specific attacks, relying solely on these features might lead to an increase in false positives, necessitating additional refinement to improve accuracy.

4.2.2 Chunk selection for categorical data in CAN networks

There are two ways to input data into the model: first, by inputting data in single instance input, and second, by using time series input by chunks. Inputting data in a single instance, input makes model deployment more complex and has a higher potential for false positive rates. On the other hand, time series input with an optimal chunk size makes the model more effective and generalized. Chunk selection for categorical data in CAN networks, particularly for analyzing CAN ID sequences, can be effectively performed using methods like Sliding Window with Overlap, Frequent

Pattern Mining, Mutual Information Analysis, and Change Point Detection [17–19]. Each of these methods serves a unique purpose in identifying meaningful chunks within the data.

- *Sliding window with overlap*: This method divides the CAN ID sequence into overlapping windows, ensuring that transitions between IDs are captured. For example, consider a sequence of CAN IDs: 0x100, 0x200, 0x300, 0x400, 0x500. Using a sliding window of size 3 with an overlap of 1, the chunks would be [0x100, 0x200, 0x300], [0x200, 0x300, 0x400], and [0x300, 0x400, 0x500]. This overlap ensures that transitional relationships between IDs, such as 0x200 to 0x300 and 0x300 to 0x400, are not missed.

- *Frequent pattern mining*: This method identifies recurring CAN ID subsequences within the data. For instance, in a dataset where the sequences 0x100, 0x200, and 0x300 frequently appear, it can be inferred that these IDs are often generated together due to the interaction of specific ECUs. This helps group chunks of IDs that represent repeated operational patterns, like a periodic status check between ECUs.

- *Mutual information analysis*: This method quantifies the dependency between adjacent CAN IDs to create meaningful chunks. For example, if CAN_ID 0x100 frequently precedes CAN_ID 0x200, this strong relationship will result in grouping them into the same chunk, such as [0x100, 0x200, 0x300]. This ensures that IDs with interdependent operations are analyzed together.

- *Change point detection*: This method identifies significant shifts in the CAN ID sequence that may signify a transition between operational states or an abnormal event. For example, if a normal sequence like 0x100, 0x200, 0x300, 0x400 suddenly changes to 0x100, 0x200, 0x999, 0x400, the presence of 0x999 (a potentially anomalous ID) could indicate a change point. The sequence can then be chunked as [0x100, 0x200] and [0x999, 0x400], isolating the abnormal region for further analysis.

Among these methods, Mutual Information Analysis is particularly effective for CAN data, as data generation often depends on dynamic driving activities and active functions. For instance, in the case of the Kia Soul, a chunk size of 10 to 20 is optimal for attack classification, capturing enough contextual information for effective intrusion detection. The optimal chunk size, however, varies by manufacturer, reflecting differences in communication patterns and ECU interactions. If the chunk size is too small, the IDS may lack sufficient information for analysis, leading to frequent activations and increased power consumption. Conversely, overly large chunks can adversely affect response time, delaying the detection of critical anomalies. Thus, selecting an appropriate chunk size is essential for achieving a balance between power efficiency and response time, ensuring an optimal IDS model tailored to the specific characteristics of the vehicle's CAN network.

4.3 Intrusion detection

There are two primary approaches to building an IDS: rule-based systems and machine learning-based systems. In a rule-based detection system, predefined rules are extracted and implemented as logic to identify anomalies [20]. However, these systems are highly prone to false positive rates due to the nature of in-vehicle network (IVN) attacks, which are often data- and frequency-agnostic [2]. Attackers can creatively inject data in unpredictable ways, making it challenging for rule-based systems to cover all dynamic situations effectively. In contrast, machine learning-based detection systems excel in adapting to dynamic and complex attack scenarios. These systems can analyze large datasets, identify hidden patterns, and learn from diverse scenarios to generalize effectively across various conditions. This adaptability enables machine learning models to handle the evolving nature of IVN attacks and significantly reduce false positive rates. Among the various machine learning approaches, deep learning stands out due to its superior performance in terms of efficiency, generalizability, and accuracy. Deep learning models leverage advanced architectures like neural networks to automatically extract high-level features from raw data, eliminating the need for extensive manual feature engineering. Additionally, their ability to learn hierarchical representations makes them highly effective in recognizing subtle patterns and correlations in complex datasets, which are critical for accurate intrusion detection. Furthermore, deep learning models are highly scalable, allowing them to handle large amounts of data and adapt to new attack vectors, making them the most robust option for intrusion detection in IVNs.

Deep learning offers several approaches, including supervised, unsupervised, semi-supervised, and reinforcement learning [2, 13, 21, 22], each suited for different types of problems. Below is a brief overview of these methods and their advantages and disadvantages:

- *Supervised learning*: This approach involves training a model on labeled data, where the input-output relationships are explicitly defined. The model learns to map inputs to the correct outputs, making it highly effective for classification and regression tasks. It is the most reliable and effective method due to its ability to achieve high accuracy when sufficient labeled data is available.

- *Unsupervised learning*: In this approach, the model is trained on unlabeled data to discover patterns or groupings within the dataset. Techniques like clustering and dimensionality reduction fall under this category. However, the lack of labeled data makes it challenging to validate the results, leading to potential inaccuracies and difficulty in interpreting the outcomes.

- *Semi-supervised learning*: This combines a small amount of labeled data with a large amount of unlabeled data to improve performance. While it bridges the gap between supervised and unsupervised methods, its disadvantage lies in the dependency on the quality and representativeness of the labeled data, which can limit its effectiveness.

- *Reinforcement learning*: This approach trains models to make sequential decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Although it is powerful for tasks like control systems and decision-making, reinforcement learning is computationally expensive, requires extensive trial-and-error, and can be unstable in complex environments.

Among these approaches, supervised learning stands out as the most reliable and effective for intrusion detection in IVNs. Its ability to leverage labeled data for precise training ensures high accuracy and robustness, making it the preferred choice for building dependable IDS models.

Neural networks can be broadly categorized into Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs), each designed for specific tasks. DNNs are the most basic form of neural networks, consisting of fully connected layers used for general-purpose tasks. RNNs are specialized for sequential data, as they have the ability to remember previous sequence patterns, making them highly effective for time series data. CNNs are designed for spatial data, particularly image and video processing, using convolutional layers to extract hierarchical features.

Among these, RNNs and CNNs have proven to be the most effective for various applications. RNNs excel at processing sequential data due to their memory capabilities, which allow them to capture patterns over time. Popular RNN variants include Long Short-Term Memory (LSTM), which addresses long-term dependency issues, and Gated Recurrent Unit (GRU), which offers a simpler and lighter architecture while maintaining similar performance. GRU is often preferred over LSTM due to its efficiency and reduced computational overhead.

CNNs, on the other hand, are highly effective for spatial data and have numerous architectures, such as AlexNet, ResNet, Inception, and MobileNet. Among these, MobileNet stands out for its lightweight design, making it ideal for resource-constrained environments like edge devices or IVNs. MobileNet uses depthwise separable convolutions, significantly reducing the computational complexity while maintaining high accuracy, making it the best choice for scenarios requiring efficiency.

In terms of architecture, both RNNs and CNNs can incorporate Autoencoders, which are designed to learn efficient representations of input data by compressing it into a lower-dimensional space and reconstructing it back. This is particularly useful for tasks like anomaly detection, where autoencoders learn to identify deviations from normal patterns.

Finally, while advanced architectures like CNNs and RNNs are effective, even a simple model can yield high accuracy when trained on precisely labeled data that covers dynamic situational variations, ensuring both efficiency and reliability in real-world applications.

Acknowledgements

The author acknowledges the use of QuillBot and ChatGPT for language polishing of the manuscript.

Author details

Mahdi Sahlabadi[1]†, Md Rezanur Islam[2]†, Munkhdelgerekh Batzorig[1]† and Kangbin Yim[1]*†

1 Department of Information Security Engineering, Soonchunhyang University, Asan-si, Korea

2 Department of Software Convergence, Soonchunhyang University, Asan-si, Korea

*Address all correspondence to: yim@sch.ac.kr

† These authors contributed equally.

References

[1] Lokman S-F, Othman AT, Abu-Bakar M-H. Intrusion detection system for automotive controller area network (CAN) bus system: A review. EURASIP Journal on Wireless Communications and Networking. 2019;**2019**(1):1-17

[2] Islam M, R et al. CF-AIDS: Comprehensive frequency-agnostic intrusion detection system on in-vehicle network. IEEE Access. 2023

[3] Koh Y et al. Real Vehicle-based attack dataset for security threat analysis in a vehicle. In: International Conference on Broadband and Wireless Computing, Communication and Applications. Cham, Switzerland: Springer; 2022. pp. 137-146

[4] Batzorig M et al. A novel attack scenario dataset collection for intrusion detection system in CAN network. In: International Conference on Network-Based Information Systems. Vol. 183. Springer; 2023. pp. 130-141

[5] Fugiglando U et al. Driving behavior analysis through CAN bus data in an uncontrolled environment. IEEE Transactions on Intelligent Transportation Systems. 2018;**20**(2): 737-748

[6] Aliwa E et al. Cyberattacks and countermeasures for in-vehicle networks. ACM computing surveys (CSUR). 2021;**54**(1):1-37

[7] Lee H, Jeong SH, Kim HK. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In: 2017 15th Annual Conference on Privacy, Security and Trust (PST). IEEE; 2017. pp. 57-5709

[8] Han ML, Kwak BI, Kim HK. Anomaly intrusion detection method for vehicular networks based on survival analysis. Vehicular Communications. 2018;**14**: 52-63

[9] Nowdehi N et al. CASAD: CAN-aware stealthy-attack detection for in-vehicle networks. In: arXiv preprint arXiv: 1909.08407. 2019

[10] Thien-Nu H et al. CANPerFL: Improve in-vehicle intrusion detection performance by sharing knowledge. Applied Sciences. 2023;**13**(11):6369

[11] Islam M, R, Insu O, Yim K. CANTool an in-vehicle network data analyzer. In: 2022 International Conference on Information Technology Systems and Innovation (ICITSI). IEEE; 2022. pp. 252-257

[12] Choi W et al. An enhanced method for reverse engineering CAN data payload. IEEE Transactions on Vehicular Technology. 2021;**70**(4): 3371-3381

[13] Hoang T-N, Kim D. Detecting In-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders. Vehicular Communications. 2022;**38**: 100520

[14] Jo H, Kim D-H. Intrusion detection using transformer in con-troller area network. IEEE Access. 2024

[15] Tomlinson A, Bryans J, Shaikh SA. Using internal context to detect automotive controller area network attacks. Computers & Electrical Engineering. 2021;**91**:107048

[16] Maldonado J, Riff MC, Neveu B. A review of recent approaches on wrapper feature selection for intrusion detection. Expert Systems with Applications. 2022; **198**:116822

[17] Miao R et al. Real-time defect identification of narrow overlap welds and application based on convolutional neural networks. Journal of Manufacturing Systems. 2022;**62**: 800-810

[18] Peng W et al. Global motion filtered nonlinear mutual information analysis: Enhancing dynamic portfolio strategies. PLoS One. 2024;**19**(7):e0303707

[19] Truong C, Oudre L, Vayatis N. Selective review of offline change point detection methods. Signal Processing. 2020;**167**:107299

[20] Bozdal M, Samie M, Jennions IK. WINDS: A wavelet-based intrusion detection system for controller area network (CAN). IEEE Access. 2021;**9**: 58621-58633

[21] Narasimhan H, Ravi V, Mohammad N. Unsupervised deep learning approach for in-vehicle intrusion detection system. IEEE Consumer Electronics Magazine. 2021; **12**(1):103-108

[22] Xiao L et al. Reinforcement learning-based physical-layer authentication for controller area networks. IEEE Transactions on Information Forensics and Security. 2021;**16**:2535-2547