Intrusion Prevention System Against Spoofed Data Frames at the Electronic Control Unit Level

Zhengyuan Liu[1], Weidong Yang[2,3], Shuguang Wang[4,5], Hongwei Fan[1]
[1]College of Information Science and Engineering, Henan University of Technology, Zhengzhou,Henan,450000, China
[2]School of Artificial intelligence and Big Data, Henan University of Technology,Zhengzhou,Henan, 450000, China
[3]Hangzhou Institute of Technology, Xidian Universy, Hangzhou, Zhejiang,310000, China
[4]School of Computer Science and Technology, Xidian Universy, Xi'an, Shanxi,710126, China
[5]Shandong Institute of Standardization, Jinan, Shandong,250014, China

The Controller Area Network (CAN) serves as the backbone of modern vehicle networks, connecting multiple Electronic Control Units (ECUs) and providing an efficient data transmission environment for the entire vehicle control system. With the advancement of automotive intelligence, the methods for intruding upon in-vehicle CAN networks are becoming increasingly diverse, posing significant threats to driving safety. However, existing Intrusion Detection Systems (IDSs) often require considerable time to detect anomalies, potentially allowing malicious data frames to escape under current security mechanisms. Therefore, there is an urgent need for an efficient anomaly detection and defense mechanism to enhance the security of CAN networks. This paper proposes an ECU-level Intrusion Prevention System (IPS) based on statistical methods that does not require modifications to the existing ECU architecture. By analyzing matrix area features generated from CAN data frame payloads, the system can determine normal ranges for these feature parameters in an unsupervised manner. When detected data frame characteristics exceed predefined thresholds, the system identifies them as anomalies, thereby achieving effective detection and defense against potential attacks. Experimental results demonstrate that, under real attack scenarios and tampering attack scenarios, the system achieves detection rates of 99.76% and 96.5%, respectively, while maintaining a false positive rate of 0%. Additionally, the system is deployed on a low-cost STM32F407MINI development board simulating ECU functionality, with a detection process lasting only 64 μs.

***Index Terms*—Controller Area Network, Electronic Control Unit, Intrusion Prevention System, Unsupervised, Payloads.**

I. Introduction

THE Controller Area Network (CAN), serving as the backbone network of in-vehicle networks, connects multiple Electronic Control Units (ECUs) to facilitate functions such as vehicle body control, steering control, and engine management. Consequently, an attack on the CAN network poses a significant threat to driving safety. As early as 2010, researchers had demonstrated the capability to infiltrate CAN networks via the On-Board Diagnostics (OBD-II) interface. By employing reverse engineering techniques, they were able to inject false messages into the network and successfully take control of the vehicle's body[1].In the following years, researchers developed various methods for remotely infiltrating in-vehicle CAN networks. These methods included attacks via Wi-Fi, Bluetooth, in-vehicle infotainment systems, GPS, and Over-The-Air (OTA) technologies[2–6].

Due to the broadcast nature of data transmission in CAN networks and the lack of authentication mechanisms in CAN data frames, ECUs connected to the CAN network are vulnerable to attacks involving forged messages within the network.To address this issue, some researchers have proposed transmitting authentication frames to verify the legitimacy of the sending ECU [6, 7]. However, this approach undoubtedly imposes additional overhead on CAN network communications.Other researchers have suggested broadening the number of bits in a CAN frame to incorporate authentication mechanisms [8, 9]. However, designed that the CAN protocol specifies a fixed frame format, this method is impractical for real-world implementation.

Addressing vehicle security concerns, the automotive industry has implemented multiple layers of security measures in vehicles. These include protecting critical messages, ensuring the integrity of ECUs, utilizing gateways to isolate data transmission between different domains, deploying firewalls and external network security interfaces, as well as adopting Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) [10]. IDS and IPS differ from other security mechanisms. For instance, gateway isolation can complicate communication between different domains, and authentication mechanisms may impose additional burdens on the bus. Deployed within the CAN network, IDS and IPS continuously monitor and inspect data frames transmitted over the bus without affecting normal communication. This approach minimizes interference with existing systems and does not necessitate modifications to the existing CAN protocol.[11].

Currently, there has been extensive research on IDS for CAN networks. 1)Methods based on the physical characteristics of CAN bus communications create unique fingerprints for each ECU to detect anomalous frame transmissions [12, 13]. However, due to environmental factors such as temperature and electromagnetic interference, it is challenging to maintain high detection rates with this approach. 2)Anomaly detection methods based on the sequence of CAN data frames [14, 15] identify anomalies by detecting disruptions to the original frame sequence order or periodicity caused by malicious

Manuscript received March 24, 2025; revised July 7, 2025. Corresponding author: Shuguang Wang (email: 2869377@qq.com).

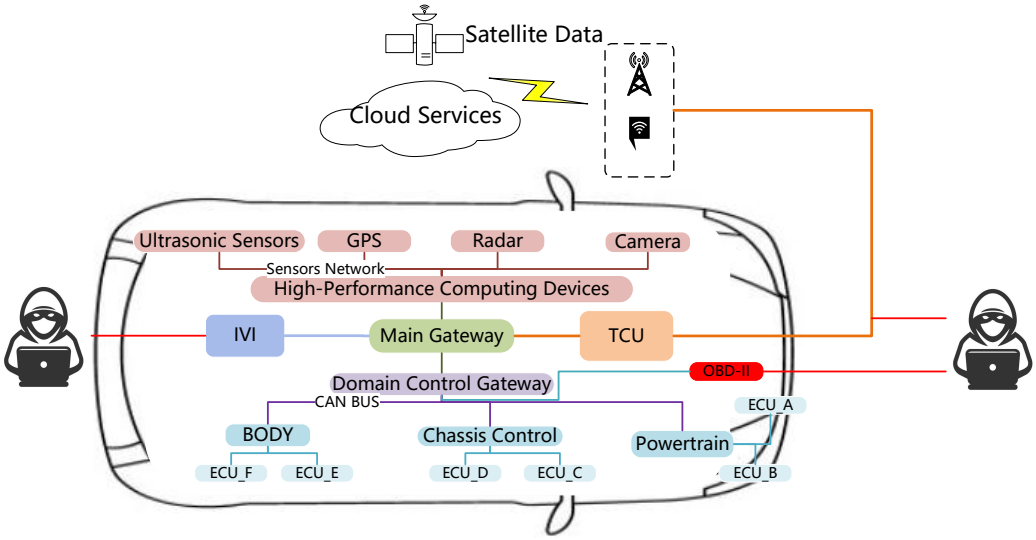


Fig. 1. Schematic Diagram of In-Vehicle Network Attacks.An attacker can infiltrate the in-vehicle network through the On-Board Diagnostics (OBD) interface, remote communication channels and In-Vehicle Infotainment(IVI) system. In severe cases, this can lead to the control of the vehicle, posing a serious threat to driving safety.

frames. Nevertheless, when attackers reduce the frequency of frame injection, the effectiveness of this method diminishes significantly. 3)Anomaly detection methods based on the intrinsic features of data frames [16, 17], which often employ deep learning techniques to extract features through image and matrix transformations, classify normal and anomalous frames. However, deep learning-based methods are difficult to deploy in resource-constrained onboard systems. These IDSs are trained using known attack models, making their efficiency in detecting unknown attacks unverifiable. To address this issue, unsupervised training-based IDS and online learning-based IDS have been proposed [18, 19]. Unsupervised IDS aims to identify common features of normal communication data to handle unknown attacks; online learning IDS seeks to modify existing model parameters when new types of attacks emerge, thereby adapting to new threats. However, the IDS proposed in [18] achieves a detection rate of only 91.7%. And, the IDS introduced in [19] requires significant time for online data sampling and model training.

Despite this, IDS can only detect attacks but lacks defensive measures, whereas an IPS can prevent network attacks even when other security mechanisms fail. Olufowobi et al. in [20] proposed an IPS that detects attacks before data transmission is completed, but it does not analyze the requirements between detection time and the time needed to stop the attack. Matsumoto et al. in [21] introduced an IPS utilizing nodes communicating on the CAN bus to monitor for abnormal behavior, with the idea of sending error frames to mask the transmission of anomalous data frames upon detecting an attack. However, these methods require modifications to ECUs, which undoubtedly increases the burden on the vehicle's system. Longari et al. in [22] proposed a method using CAN fault confinement mechanisms to detect the connection status between nodes and the bus, but it can only prevent attacks from disconnected nodes. Cheng et al. in [23] proposed a method providing shifted legitimate identifiers to each node, aiming to expose the IDs of false data frames. Nevertheless, this approach cannot defend against attacks that have already been injected into the bus. Paulo et al. in [24] proposed an IPS based on unsupervised learning targeting anomalous data frames with legitimate identifiers, intending to discard such frames before they can harm the vehicle system. However, unsupervised detection methods need to ensure a false positive rate of 0, which was not achieved in their work. Additionally, to meet detection time requirements, they only inspect the first 5-6 bytes of the payload. If attackers are aware of this, targeted attacks can still evade the IPS.

To address the aforementioned issues, this paper proposes an anomaly detection method based on offline-trained parameters to defend against forged data frames carrying legitimate identifiers injected by hackers after compromising the TCU, OBD-II or IVI. The core idea of this method is to isolate local similar regions within matrices generated from the actual data fields of CAN data frames with identical identifiers, treating these regions as features. By performing matrix operations to extract characteristic parameters of these regions as feature values, and employing statistical methods to determine the threshold range of feature values under attack-free scenarios, this method achieves anomaly detection. This enables the enhancement of system robustness in an unsupervised manner. Unsupervised anomaly detection based on statistical methods can significantly improve detection efficiency, reduce detection time, and maintain generalization capabilities against unknown attacks. Since our approach does not involve any machine learning or deep learning models, it can be easily deployed within ECUs, providing ECU-level security protection.

The contributions of this paper are as follows:

- A statistical unsupervised anomaly detection method is proposed, which achieves frame-by-frame detection of data frames communicated within the CAN network,

Fig. 2. Standard CAN Data Frame Format.A standard data frame with a DLC (Data Length Code) of 8 occupies a total of 108 bits. If considering the bit stuffing mechanism, it can be up to 125 bits.

maintaining high detection rates while ensuring a false positive rate of 0.

- A novel IPS is introduced, which is deployed on a low-cost STM32F407ZGT6 development board used to simulate the functionality of an ECU, with a detection time of only 64 μs for a single data frame and anomalous data frames can be proactively identified and discarded by the system.
- The method proposed in this paper effectively addresses the issue of model obsolescence due to ECU updates. Engineers need only adjust the statistical test parameters during the ECU update process.

II. Background

A. CAN frame

As shown in the Fig 2, CAN data frames come in two different formats: Standard Frame Format and Extended Frame Format. The difference between them lies in the number of bits reserved for the message identifier (ID). The ID in the standard frame format is 11 bits, while in the extended frame format, it can be up to 29 bits. The Start-of-Frame (SOF) segment indicates the beginning of the data frame and is used for bus synchronization; the Arbitration field (ID) indicates the priority of the frame, with smaller ID values having higher priority; the Control field defines the type of frame and the length of the data. The first bit in the Control field is the Identifier Extension bit (IDE), which indicates whether the frame uses a standard ID (set to 0) or an extended ID (set to 1). The second bit is the Reserved bit (RO), which is always set to 0. The next four bits are the Data Length Code (DLC), indicating the size of the transmitted data. The Data field contains the actual transmitted data, up to 64 bytes. The Cyclic Redundancy Check (CRC) field consists of 15 error-checking bits and a delimiter bit, using cyclic redundancy check methods to detect and correct transmission errors. The Acknowledgment (ACK) field indicates that an error-free message has been sent. Each ECU that receives the message correctly will overwrite the recessive bit in the original message with a dominant bit to indicate successful reception. If an ECU detects an error, the bit remains recessive, and the message is discarded. The End-of-Frame (EOF) field marks the end of the frame, typically consisting of seven recessive bits.

B. CAN communication mechanism

Nodes connected to the CAN bus send data while also receiving messages from the CAN bus. When a node intends to send a CAN message, it transmits the Start-of-Frame (SOF) within the CAN bus clock cycle, signaling that the node is about to send a data frame. When other nodes detect an ongoing data frame transmission before they start sending their own messages, they wait until the current data frame transmission is complete before proceeding. If multiple nodes transmit SOF simultaneously, arbitration begins from the ID bits. If a node sends a recessive bit (1) and receives a dominant bit (0) at the same time, it ceases transmission and waits. Thus, the larger the ID, the lower the priority. This is the process of CAN bus arbitration.

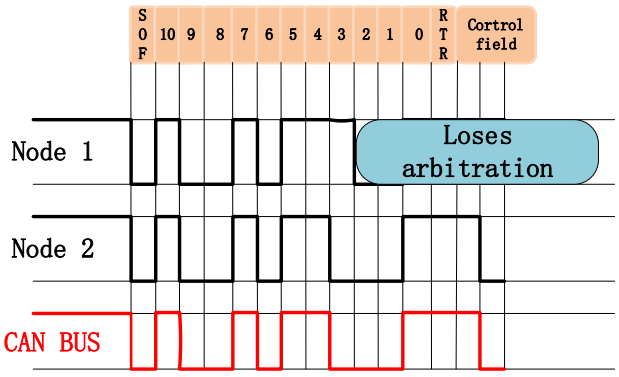


Fig. 3. CAN bus arbitration mechanism. When Node1 detects a dominant bit '0' on the CAN bus while transmitting the 8th bit, it immediately stops sending and switches to listening mode upon the completion of that bit transmission.

For the reception of CAN bus messages, the process is handled by the CAN controller within the ECU, also known as the hardware filter, without requiring intervention from the MCU. This reduces the processing load on the MCU. The process of data frame reception by an ECU is as follows: The ECU continuously monitors the transmission of data frames on the CAN bus. Upon detecting a Start-of-Frame (SOF), it immediately begins receiving the frame. The received identifier is then compared with the mask identifier field set in its own filters. If a conflict is detected, the frame is discarded, and the memory is released. The ECU then waits for the transmission process to end before continuing to monitor the CAN bus. If there is no conflict in the identifier field, the ECU continues to receive the data frame until the reception is complete and stores it in the reception mailbox.

The ECU's filters operate in two modes: Mask Bit Mode and Identifier List Mode.

- **Mask Bit Mode**: As depicted in Fig 4, this mode utilizes a predefined Identifier and a Mask to filter incoming data frames. The Mask specifies which bits of the identifier must match exactly. When a bit in the Mask is set to 1, the corresponding bit in the received frame's identifier must match the predefined ID, as indicated by the purple-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	STID 10	STID 9	STID 8	STID 7	STID 6	STID 5	STID 4	STID 3	STID 2	STID 1	STID 0	RTR	IDE	RSV	RSV	RSV
ID	0	0	0	0	1	1	1	1	1	1	1	0	0	0	1	0
Mask	1	1	1	1	1	0	0	1	0	1	0	1	1	1	1	1
	0	0	0	0	1			1		1		0	0	0	1	0

Fig. 4. ECU Shielder Working Principle Diagram.The bottom row in the figure shows the format of receivable identifiers. The blank areas can be either '0' or '1'.

shaded bits. Conversely, if a bit in the Mask is set to 0, the corresponding bit in the received frame's identifier can be either 0 or 1. This means that any value in the unmasked (blank) region will be accepted regardless of whether it matches the predefined ID.

- **Identifier List Mode**: In this mode, the filter contains a list of predefined Identifiers. The filter accepts only those incoming data frames whose identifiers match one of the predefined IDs in the list. If the identifier of an incoming frame does not match any of the predefined IDs, the frame is discarded. This mode provides a more stringent filtering mechanism compared to the Mask Bit Mode, as it requires an exact match with one of the specified identifiers.

C. Introduction to Detection Methods

This paper will adopt the Interquartile Range (IQR) test as the method for detecting anomalies in CAN data frames. The IQR detection method is a statistical approach used to identify outliers in the data. Assume there is an ordered dataset $D = q1, q2, q3, ..., qn$, where *Q1* is the 25th percentile of *D*, *Q2* is the median of *D*, and *Q3* is the 75th percentile of *D*. The IQR detection method determines the range of outliers by calculating the quartiles *(Q1, Q2, Q3)* and IQR. The *IQR* is defined as the difference between *Q3* and *Q1*, i.e.,

$$IQR = Q3 - Q1 \tag{1}$$

The inner fences are defined as:

$$M = [Q1 - K * IQR, Q3 + K * IQR] \tag{2}$$

Where *k* is the multiplier factor, typically set to 1.5. The lower boundary is defined as $Q1 - K * IQR$,and the upper boundary is defined as $Q3 + K * IQR$.If there is a data point *y* being tested, and if *y* falls within the range *M* (i.e., between the lower and upper boundaries), it is considered normal data. Otherwise, *y* will be classified as an outlier.

D. Introduction to Attack Scenarios

The CAN bus defense mechanism proposed in this paper is deployed within the ECU to defend against *Spoofing* attacks that evade other security mechanisms. It is assumed that the attacker has knowledge of the identifiers in the compromised CAN network and can forge CAN data frames using known identifiers, enabling these forged frames to be received and executed by the ECU. A spoofing attack involves an attacker injecting data frames with known identifiers into the CAN network. These data frames have identifiers that match the acceptance filter masks of certain ECUs, enabling them to be received by those ECUs. To achieve their malicious goals, the data fields of these injected frames often consist of random bits.

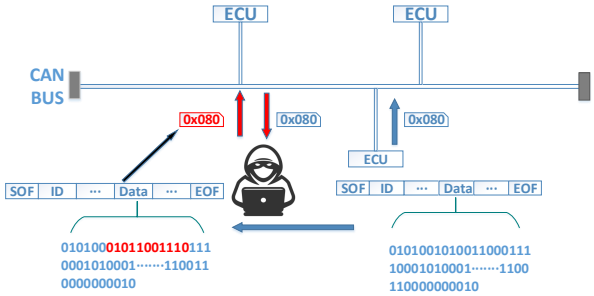


Fig. 5. Spoofing Attack Schematic. The attacker monitors the data frames transmitted on the bus and modifies a small portion of the bits, repackaging them into legitimate data frames that can be accepted by the ECU.

Lee et al. in [14] described an experiment where they injected CAN data frames with known identifiers and random bit data fields into a vehicle driving on a secure road. This resulted in noticeable vehicle jitter, demonstrating the potential impact of such attacks on vehicle safety and performance.As shown in Fig 5, if an attacker gains access to the CAN network and can capture data frames from the bus, they can reverse-engineer these frames to understand their data distribution. By injecting forged messages into the bus with selectively modified critical bits, the attacker can carry out targeted attacks. These types of attacks are often much harder to detect.

III. Methods

A. Anomaly Detection Method Based on Simple Matrix Grouping Features

Longari et al. in [25] mentioned that the CAN data frame's identifier, RTR, and data are converted into a 9x9 matrix in an attempt to identify features for classifying anomalous behavior. Paulo et al. propose a method that can detect anomalous behavior using only the first few bytes of the payload [24]. The purpose of the detection method proposed in this paper is to identify the characteristics of data frames carrying payloads for the same identifier. Therefore, only the payload is selected and converted into an 8x8 matrix by arranging it horizontally. This method treats the corresponding rows and columns of the matrix as a sensing region; for example, the region represented by parameters *b_2* and *b_6* in Fig 6 is referred to as the receptive field.

In the envisioned approach, when the bit values within this region change, the corresponding feature values *b_2* and *b_6* will also change. Since the data transmitted by CAN data frames is not static and can vary, using a single receptive field may not be sufficient to detect the exact position of the bits altered by a forged message.Therefore, by overlaying two

receptive fields, a new region called the overlapped receptive field is formed, denoted as *Z_26*. When the bit values within the overlapped region change, *Z_26* will also change. This way, a single feature value can represent the changes in 28 bits across two rows and two columns of the original matrix, significantly reducing the number of features needed.

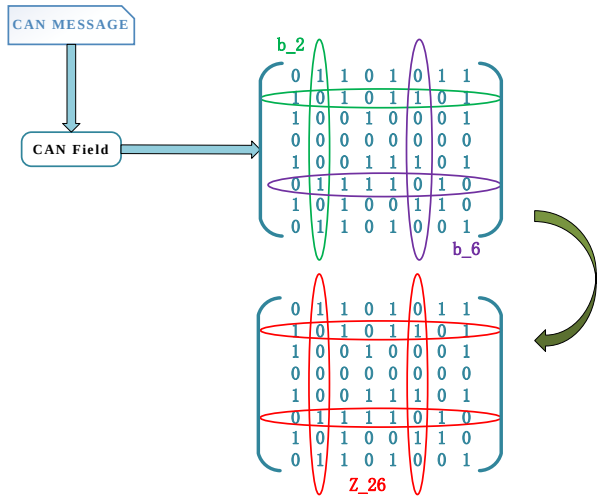


Fig. 6. Matrix Region Characterization Parameters Schematic. Parameter *b* is utilized to detect changes in data within a 1x1 cell. In contrast, Parameter *Z*, which aggregates the values of two Parameter *b* instances, serves to perceive feature variations and can thus monitor changes in data across a 2 x 2 cell range. Here, 'n x n' denotes a matrix area comprised of n rows and n columns.

To monitor the entire data matrix, at least four overlapped receptive fields are necessary, requiring a minimum of four *Z* parameters. However, multiple *Z* parameters can lead to overlapping areas, as illustrated in Fig 7. When bits in the red region change, two *Z* parameters will also change. We refer to this overlapping area as a strong receptive field. For anomaly detection, we focus on regions that seldom change or remain constant, as these have more stable bit values. If a forged message alters bits in these stable regions, the corresponding Z parameters will detect the changes, thus identifying anomalies.

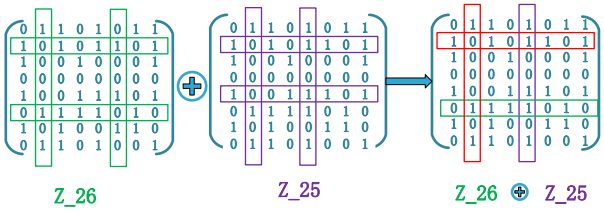


Fig. 7. Strong Sensory Field.The figure illustrates the result of two eigenvalues being overlaid. When bits within the overlaid region change, the eigenvalues can sensitively detect the occurrence of these changes.

To meet the requirements of embedded-level computing resources and the real-time nature of in-vehicle systems, we attempt to use mathematical methods to identify characteristic values that can represent changes in matrix regions. The decision was made to derive these characteristic values using simple arithmetic operations. Based on this principle, the following design was implemented to obtain the features:

step1: Arrange the data field of the CAN data frame (up to 64 bits) horizontally into an 8×8 matrix. If the number of bits is less than 64, pad the remaining positions with '0'.

step2: Compute the sum of each row and each column of the matrix, resulting in 16 parameters. Then, divide the parameter of each row by the corresponding parameter of each column to obtain the receptive field parameter b.

step3: To ensure that the characteristic parameter of the overlapped receptive field changes subtly, we sequentially divide the 8 b values obtained in step 2. The resulting quotient serves as the final characteristic value Z. Calculation Process of Perceptual Characteristic Values:
Define an 8×8 matrix filled with 0s and 1s, denoted as A.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,8} \\ a_{2,1} & a_{2,2} & \dots & a_{2,8} \\ \dots & \dots & \dots & \dots \\ a_{8,1} & a_{8,2} & \dots & a_{8,8} \end{bmatrix} \tag{3}$$

Define a matrix *C* where all elements are equal to 1, expressed as follows: $C = \begin{bmatrix}1,1,1,1,1,1,1,1\end{bmatrix}^T$
Let *P* and *Q* represent the sums of each row and each column of matrix *A*, respectively:

$$P = A * C, Q = A^T * C \tag{4}$$

Now, take the reciprocal of all elements in *Q* to form a new matrix *Q'*. Let *B* represent the quotient of the row sums and column sums of matrix *A*, i.e.:

$$B = [b1 \quad b2 \quad \dots \quad bn]^T = P \odot Q \tag{5}$$

Now, let the new matrix generated by taking the reciprocal of all elements in *B* be denoted as *B'*. The matrix composed of z-parameters is denoted as *Z*, i.e.:

$$Z = B * (B^T)' \tag{6}$$

The main diagonal of matrix *Z* consists entirely of '1'. Since the parameters in the upper triangular region represent the same overlapped receptive fields as their symmetric counterparts in the lower triangular region, we can simply take the parameters from the upper triangular region (excluding the main diagonal) as features. This results in a total of 28 feature values.

B. Design of Detection Model

1) Feature Selection Algorithm

In III-A, 28 features for anomaly detection were introduced. To optimize the detection model, it was decided to select four of these features as the focus for anomaly detection. According to previous analysis, regions in the data matrix with lower change frequency and smaller variation ranges should be targeted for anomaly detection. Following this principle, one can statistically analyze the distribution intervals of the features and select those with smaller interquartile ranges as the actual features for anomaly detection.
The simplest approach is undoubtedly to select the four features with the smallest interquartile ranges as classification

features. However, this still presents a challenge: the false positive rate must be controlled at 0%. If not, it means that some normal CAN data frames will always be flagged as attacks by the anomaly detection program, preventing these frames from being executed by the ECU and thereby disrupting the vehicle's normal driving process. If the four features with the smallest interquartile ranges are chosen, there may be outliers that do not fall within the inner fences. Although expanding the multiplier factor k can widen the inner fences, this might result in very small IQR values, making the selection of an appropriate k value particularly challenging.To address this issue, it is proposed to utilize an expanded k and the interquartile range. When selecting features, two conditions should be followed:

Algorithm 1 Feature Selection Algorithm

Input: $CAN\_Zlist[Z1, Z2, ..., Zn]$ ▷
A two-dimensional list is used to store the characteristic values of N data entries with the same identifier.
Output: $Bound$ ▷ store the inner fence threshold values.
Fin_Z ▷ store the indices of the selected features.
Initialize $P \leftarrow 0$ ▷ Sliding factor
Initialize $IQR \leftarrow [\ ]$ ▷ An empty list
Initialize $Fin\_Z \leftarrow [\ ]$ ▷ An empty list
Initialize $Bound \leftarrow [\ ]$ ▷ An empty list
IQR←Calculate-IQR(Zlist)▷ Calculate the IQR for each Z-value in Zlist.
Zlist.sort(IQR)▷ Sort the list in ascending order based on the IQR
while 1 **do**
  Initialize $Curlist \leftarrow [\ ]$ ▷ An empty list
  for $int\ \ i < 4$ **do** ▷ The sliding window size is 4.
    Fin_Z[i]←Zlist[P+i][0]▷ The first row of the list contains the feature indices.
    while j < Zlist.Length()-1 **do**
      Bound[i] ← Calculate-Inner-Field(Zlist[p+i])
      if The current inner fence region does not meet the requirement of a 0% false negative rate. **then**
        Update the inner fence region parameters
        if Inner fence parameter overflow **then**
          p+1
          Break
        Continue
      else
        Break
return Bound,Fin_Z

- **Condition1:** The inner fences of the selected features should be as small as possible;
- **Condition2:** The detection model created using the selected feature set must ensure a false positive rate of 0%.

Algorithm 1 describes the feature selection process. First, it calculates the characteristic values of CAN data frames with the same identifier and stores them in a two-dimensional list Zlist, which serves as the input for the algorithm. Then, it sets up a sliding window of size 4 for feature selection. Initially, the first four features from the sorted (in ascending order) Zlist are selected as input, and the algorithm checks if the false negative rate is 0%. If this condition is not met, the sliding window is shifted to the right, and the process is repeated until a set of features that satisfies the condition is found.

In line 14 of Algorithm 1, the process for updating the inner fence parameters is depicted in Fig 8. The initialization sets Q1 to the 25th percentile, Q3 to the 75th percentile, and k=1.5. Under these initial conditions, if the computed inner fence fails to meet the requirement of a 0% false negative rate, the algorithm first adjusts the value of k. Specifically, k is incremented by 0.5 in each iteration to progressively widen the inner fence region. If the condition remains unmet after 10 iterations, the algorithm then shifts Q1 and Q3 outward by 5% of their respective ranges and revalidates the results. Should the condition still not be satisfied even when Q1 and Q3 reach the 10th and 90th percentiles, respectively, this suggests that the selected features include a considerable number of outliers. Consequently, the features within the current window are deemed unsuitable for effective anomaly detection. However, if the parameters corresponding to the inner fence satisfy the condition, they are output as valid results.

Algorithm 2 Optimized Feature Selection Algorithm

Input: $CAN\_Zlist[Z1, Z2, ..., Zn]$ ▷
A two-dimensional list is used to store the characteristic values of N data entries with the same identifier.
Output: $Bound$ ▷ store the inner fence threshold values.
Fin_Z ▷ store the indices of the selected features.
Initialize $P \leftarrow 0$ ▷ Sliding factor
Initialize $IQR \leftarrow [\ ]$ ▷ An empty list
Initialize $Fin\_Z \leftarrow [\ ]$ ▷ An empty list
Initialize $Bound \leftarrow [\ ]$ ▷ An empty list
IQR←Calculate-IQR(Zlist)▷ Calculate the IQR for each Z-value in Zlist.
counter ← 0
Zlist.sort(IQR)▷ Sort the list in ascending order based on the IQR
while counter < 5 **do**
  while 1 **do**
    Bound[i] ← Calculate-Inner-Field(Zlist[p+i])
    if The current inner fence region does not meet the requirement of a 0% false negative rate. **then**
      Update the inner fence region parameters
      if Inner fence parameter overflow **then**
        p+1
        Break
      Continue
    else
      Fin_Z[i]←Zlist[P+i][0]▷ The first row of the list contains the feature indices.
      counter←counter+1
      Break
return Bound,Fin_Z

The guiding principle for feature selection is to ensure that the inner fence region is as small as possible. However, Algorithm 1 has a limitation: the results it produces may represent only a local optimum rather than a global optimum. To illustrate this, consider an example where $Z1, Z2, Z3, Z4, Z5$

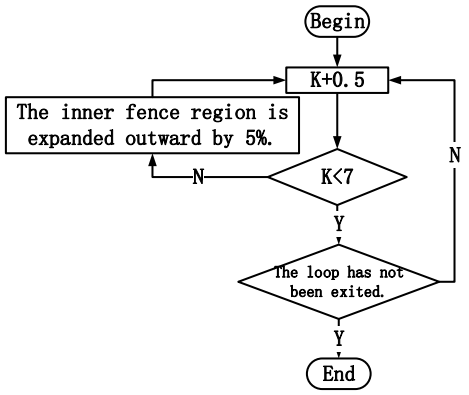


Fig. 8. Inner fence Parameter Update Flowchart.By dynamically adjusting the parameter *k* and the boundaries of the inner region, it ensures that the selected features meet the requirements of the intrusion defense mechanism.

are the five features with the smallest inner fence regions. If *Z2* contains outliers and its IQR is sufficiently small, the set that includes *Z2* may fail to satisfy the selection conditions. In such a scenario, the optimal feature set would instead consist of $Z1, Z3, Z4, Z5$. To resolve this issue, we propose an optimization to Algorithm 1.

In contrast to Algorithm 1, Algorithm 2 eliminates the use of a sliding window and instead employs a sliding factor. The algorithm sequentially evaluates the features in the Zlist, which is sorted in ascending order. If a particular feature fails to meet Condition 2 even within its extreme inner fence region, the sliding factor p is increased by 1, and the evaluation proceeds to the next feature. By applying Algorithm 2, the process ensures the selection of features that simultaneously satisfy both Condition 1 and Condition 2.

C. ECU Anomaly Detection Process Design

The detection method proposed in this paper can be applied to an STM32F407 development board that simulates ECU functionality. For a specific class of CAN data frames identified by their identifier, Algorithm 2 can provide the indices of the detection features and their corresponding inner fence boundary values. These parameters will be stored in the inspection program as validation parameters. When the ECU detects a data frame and stores it in the receive mailbox, the inspection program will initiate during the ECU processor's idle time.

Given the limited computational resources of the ECU, in addition to deploying the necessary inspection parameters in the detection program, the design should aim to minimize memory usage as much as possible. Algorithm 3 describes the process of converting the data field of the current data frame into a matrix. Initially, a zero matrix is defined, and the grid initialization is completed through a loop traversal. Since the data received by the ECU will be represented in hexadecimal format, the process involves iterating over the data array. During each iteration, bytes are further broken down into bits. If a bit is found to be 1, the corresponding

Algorithm 3 Detection Program Design

Input: RxLength, RxData, Fin_Z, Bound
Output: True Or False
Initialize grid ▷ 8*8 matrix with all elements set to zero
Initialize $Feature \leftarrow [\,]$ ▷ An empty list
Initialize $B, Q, P \leftarrow [\,]$ ▷ Three empty lists
for i **do** in Traver the RxData array ▷ Matrix Creation Process
    byte←RxData[i]
    for j **do** < byte.Length()
        bit ← $byte\&(1 << j)$
        if bit ! = 0 **then**
            grid[i][7-j] ← '1'
        else
            grid[i][7-j]←'0'
for x in 8 **do**
    for y in 8 **do**
        P[x]←P[x]+grid[x][y]
        Q[x]←Q[x]+grid[y][x]
    B[x]←P[x]/Q[x]
for z in Bound.Length() **do**
    Feature[z]←B[Fin_Z[z][0]]/B[Fin_Z[z][1]]
    if Feature[z]∉Bound[z] **then** ▷ The feature value falls within the anomaly domain.
        return False
return True

position in the grid is set to 1; otherwise, it remains unchanged. This approach not only completes the creation of the matrix but also ensures that the matrix is properly filled when the data length is less than 8 bytes.

Subsequently, according to the feature value calculation process mentioned in III-A, the feature values are computed. To simplify the program, no additional feature matrix is generated during this computation. Instead, the feature values are directly calculated using the parameters *b* obtained from the feature indices derived in Algorithm 2.

D. Attack Model

It is assumed that an attacker can infiltrate the internal CAN network and intercept data frames communicated on the bus. The attacker modifies the data fields of these frames and injects forged messages into the CAN bus, thereby executing a spoofing attack. Additionally, the attacker employs strategic methods to evade other security mechanisms. For instance, the attacker may inject forged frames at the same communication frequency as the original data frames or hijack an ECU to send forged frames, thereby concealing their malicious activities.

This paper used anomalous data from the OTIDS[14] and Car-hacking[26] real-world datasets to simulate the CAN data frames forged by attackers.Table I demonstrates the experimental results of extracting forged messages from the fuzzing attack and spoofing attack data tables in the OTIDS dataset. During this process, 45 distinct CAN IDs were analyzed, and forged messages corresponding to 8 unique IDs were successfully extracted. The distribution of these messages for

detection purposes was further analyzed. The results indicate that the number of attack messages is distributed with high uniformity.

TABLE I
OTIDS DATASET DESCRIPTION

ID	Normal Message	Attack Message
0x164	51079	6859
0x220	14517	6817
0x4b0	28093	6794
0x153	51245	6877
0x1f1	33058	7051
0x5a0	10384	7247
0x5a2	10381	6856
0x4b1	33058	6819

Table II illustrates the data distribution of fuzzing attacks and spoofing attacks in the Car-hacking dataset. Through preliminary analysis of the messages, it was found that the Spoofing attacks in the OTIDS dataset were carried out by injecting CAN messages with random data, whereas the fuzzing attacks in the Car-hacking dataset involved random injection, and the spoofing attacks were performed by periodically injecting messages with fixed data. Further analysis of the fuzzing attack data revealed 33 legitimate identifiers involved in fuzzing attacks. Among these, data frames with identifiers '0x43f' and '0x316' were found to be present in both fuzzing and spoofing attacks.

TABLE II
CARHACKING DATASET DESCRIPTION

Dataset	Normal Message	Attack Message
Fuzing	3347013	491847
Driver Gear Spoofling	3845890	597252
RPM Gauge Spoofling	3966805	654897

To evaluate the robustness of the proposed detection method, we decided to modify the attack scenarios. Using the method illustrated in Fig 9, we tampered with the Data fields of the data frames in both datasets to simulate targeted spoofing attack scenarios. During this process, the number of bits modified was a random number between 3 and 10. The modified attack scenarios were named OTIDS-X and Car-hacking-X. Through this approach, we created three distinct attack scenarios: Scenario A (no attack), Scenario B (random bit injection), and Scenario C (targeted attack). In these scenarios, OTIDS-X contains 8 legitimate identifiers, while Car-hacking-X contains 33 legitimate identifiers.

IV. EXPERIMENTS AND ANALYSIS

The analysis of anomaly detection in this experiment is based on Python 3.9.0 and PyTorch 12.0, conducted in an environment with a 12th Gen Intel(R) Core(TM) i7-12700H processor running at 2.30 GHz and equipped with 16GB of RAM. The ECU simulation and validation process will be carried out on the STM32F407 mini development board, which features an ARM Cortex-M4 32-bit RISC core that supports floating-point operations and DSP instruction sets, along with

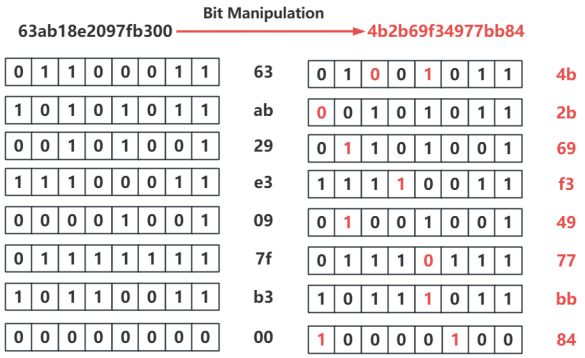


Fig. 9. Process of Spoofing Data Frames. Multiple bits in the 64-bit data field are randomly selected and flipped. This modification aims to make the tampered data field remain somewhat similar to the original data field, thereby evading simple anomaly detection mechanisms.

1MB of built-in Flash memory and 192KB of SRAM. In this section, we first analyze the detection efficiency of the proposed detection method; then, we introduce the process of simulating ECU protection and provide an analysis of the time consumption.

A. Anomaly Detection Analysis

The analysis here focuses on the CAN data frames with identifier '0220' from OTIDS and identifier '043f' from Car-hacking as examples. As shown in Fig 10, for the CAN data frame with identifier '0220', the data in bytes Data_1 and Data_3 are relatively scattered under Scenario A, while the distributions of other bytes converge. The byte distribution of targeted attack data frames obtained in Scenario C is close to that of non-attack messages, whereas the byte distribution of random bit injection attacks is more dispersed. For the CAN data frame with identifier '043f', only Data_6 exhibits a scattered distribution in the no-attack scenario (Scenario A), while the distributions are more scattered in Scenarios B and C. This demonstrates that the majority of bits in the data frames transmitted by the ECU remain fixed, further supporting the logical basis of the method proposed in III-A.

From the Fig 10(c), both *Z_1* and *Z_6* can be considered as effective classification features, but *Z_6* performs better than *Z_1*. The reason is that under the no-attack scenario, the distribution of *Z_6* is more concentrated compared to *Z_1*. Fig 10(d) shows the random feature distributions of the CAN data frame with identifier '043f' across three different scenarios. As can be seen from the figure, it is almost impossible to identify effective classification features from these eight features. This is because although the feature distributions are concentrated under the no-attack scenario, there is an overlap of features in the targeted attack scenario, leading to a lower detection rate. Therefore, relying solely on a single feature cannot maintain a high detection rate.

In this section, we will use the CAN data frame with identifier '0x220' as an example to analyze the efficiency of the proposed detection method. Fig 11 presents the execution results of Algorithm 2 under Scenario B. Under the condition that the false positive rate is 0, an increase in the number of

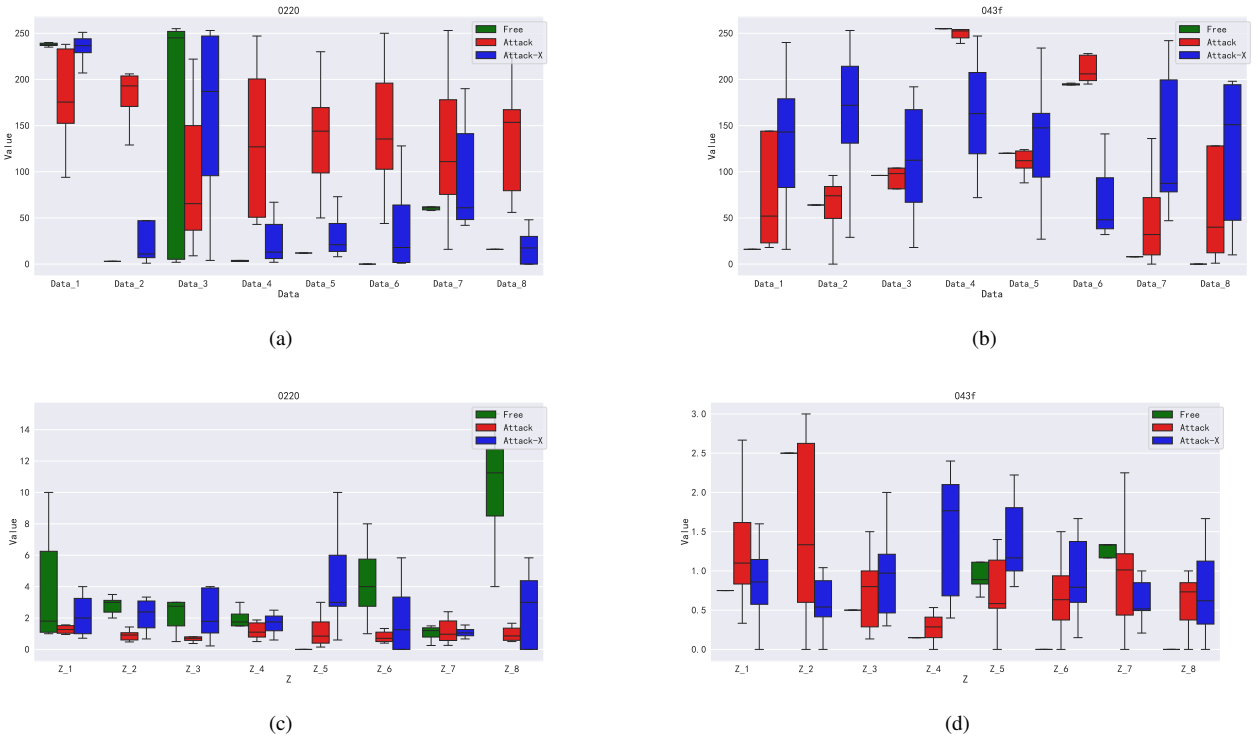


Fig. 10. Distribution of CAN Frame Data Fields and Features in Different Scenarios.Figures a and b show the distribution of data fields, presented in decimal format. Figures c and d display the distribution of 8 randomly selected features.

```
Found optimal combination with K=1, features: ['d26'], and percentiles: [25, 75]
Confusion Matrix:
[[56194     0]
 [ 4033 52161]]
Found optimal combination with K=1, features: ['d26', 'd16'], and percentiles: [25, 75]
Confusion Matrix:
[[56194     0]
 [ 2791 53403]]
Found optimal combination with K=1, features: ['d26', 'd16', 'd5'], and percentiles: [25, 75]
Confusion Matrix:
[[56194     0]
 [ 2442 53752]]
Found optimal combination with K=1, features: ['d26', 'd16', 'd5', 'd20'], and percentiles: [25, 75]
Confusion Matrix:
[[56194     0]
 [ 2362 53832]]
Final selected features: ['d26', 'd16', 'd5', 'd20']
```

Fig. 11. Execution Results of Data Frame with ID '0x220' in Algorithm 2

features leads to a significant improvement in the detection rate.

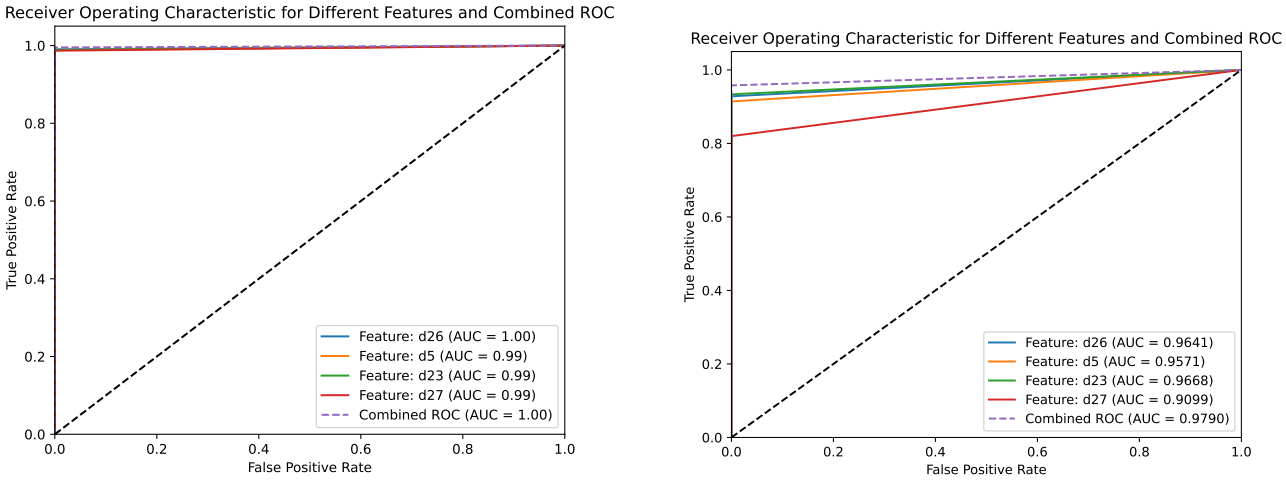
Fig 12 presents the ROC curve when the selected features are [d26, d5, d23, d27]. As shown in the Fig 12(a), the false positive rate remains consistently at 0, and the AUC of each individual feature is above 0.99, indicating that a single feature can maintain a high detection rate even under Scenario B. The AUC for the feature set reaches as high as 0.9976, demonstrating that this feature set outperforms any single feature in detecting anomalous data. We tested the obtained feature set under Scenario C, with results depicted in Fig 12(b). Although the detection performance of individual features decreased, it still remained above 0.95, indicating that the features proposed for anomaly detection possess a high classification effectiveness. The AUC for the feature set was still as high as 0.979. This proves that, even in an unsupervised setting, the feature selection resulting from the execution of Algorithm 2 maintains a high detection rate under targeted attack scenarios.

The messages corresponding to the 8 identifiers proposed in III-D, along with the messages of the 2 identifiers shared between fuzzing and spoofing attacks in the Car-hacking dataset, were tested across a total of 10 identifiers. The test results are presented in Table III. The proposed detection method achieves a detection rate of over 99.6% for attacks under Scenario B and over 96.5% for attacks under Scenario C. These findings clearly demonstrate the robustness and high efficiency of the proposed detection method.

TABLE III
ANOMALY DETECTION TEST RESULTS UNDER DIFFERENT ATTACK SCENARIOS

ID	OTIDS	OTIDS-X	Car-hacking	Car-hacking-X
0x164	100%	96.84%		
0x220	99.76%	97.97%		
0x4b0	100%	98.4%		
0x153	100%	98.1%		
0x1f1	99.87%	97.26%		
0x5a0	99.79%	99.45%		
0x5a1	100%	96.5%		
0x4b1	99.89%	97.36%		
0x43f			99.87%	99.32%
0x361			99.93%	98.7%

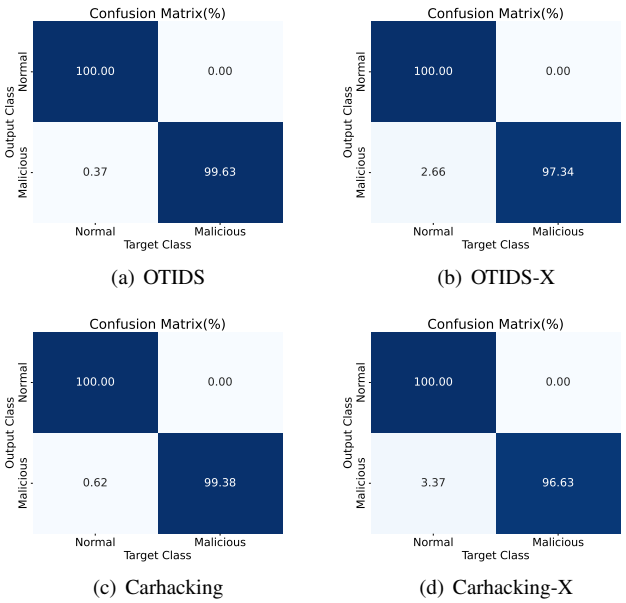
To further examine the robustness of the proposed method, we designed an experiment where 70% of the non-attack data was extracted to compute the corresponding detection parameters for each identifier's CAN messages. The attack scenario data was then fused with the remaining 30% of non-attack



(a) (b)

Fig. 12. ROC Curve of Anomaly Detection for Data Frame with ID '0220'.Figure a shows the detection analysis results in Scenario B, while Figure b shows the detection analysis results in Scenario C. These figures are used to evaluate the anomaly detection performance of this data frame under different scenarios.

data to validate the datasets. As shown in Fig 13, the detection model maintains a 0% false detection rate across all scenarios. For forged messages in the OTIDS dataset and the Car-hacking dataset, the detection rates are 99.63% and 99.38%, respectively. In tampered attack scenarios, the targeted attack detection rates are 97.34% and 96.63%, respectively. These results further demonstrate the effectiveness and reliability of the proposed method.



(a) OTIDS (b) OTIDS-X

(c) Carhacking (d) Carhacking-X

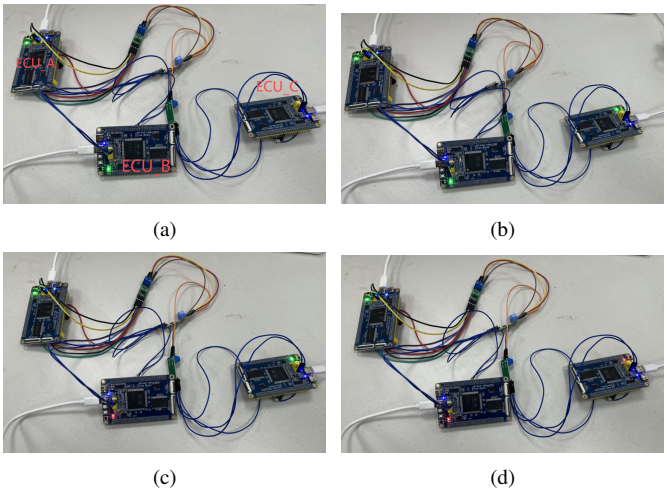
Fig. 13. Confusion Matrix of Detection Results.

B. Verification of ECU Protection Mechanism

The validation process will utilize an STM32F407ZGT6 embedded development board to simulate the ECU, employing three identical development boards, three CAN transceivers, and several Dupont wires to simulate the CAN bus communication process. Fig 12 illustrates the connection setup for the communication lines. The STM32F407ZGT6 has a built-in CAN controller, and during communication, only an additional CAN transceiver (TJA1050 used in this case) needs to be connected. The STM32 configures the CAN communication-related registers, clock, and pins using standard library functions.

Deploying the detection system inside the ECU poses two main challenges: 1) The ECU's computing and storage resources are limited, so the deployed IPS must be lightweight; 2) The number of receive mailboxes inside the ECU is limited, so the anomaly detection process should be completed as quickly as possible before the next data frame arrives. Currently, the baud rate of most automotive internal buses is 500 kbps, meaning that transmitting one bit takes 2 ms. In the worst-case scenario, multiple consecutive data frames with the same identifier may be received. For non-extended frames, a data frame consists of 108 bits, which means the IPS must complete the detection within 216 ms to prevent data frame buildup.

In IV-A of this paper, it has been verified that the detection method achieves a high detection rate. In this subsection, the experiment validates the feasibility of the ECU protection mechanism by measuring the size of the detection program deployed within the ECU and the time required for detection. According to the description of Algorithm 3, the time complexity of converting the Data field into an 8×8 matrix is O(8 × Rx_Length), meaning that the maximum time complexity at this stage is O(64). During the calculation of the eigenvalues, the time complexity for parameter b is $2 \times O(n^2)$, where n represents the dimension of the matrix. Thus, the time complexity here is fixed at $2 \times O(64)$. Through verification on the development board, the execution time of Algorithm 3 was found to be 0.58 ms, which is significantly lower in efficiency compared to some existing methods. Subsequently, we opti-



(a) (b)

(c) (d)

Fig. 14. Simulated ECU Intrusion Detection Validation Diagram.ECU_A is responsible only for sending data frames, while ECU_B and ECU_C are responsible for receiving data frames. Among them, ECU_C is configured to receive only data frames with the identifier '0x220'. The diagram demonstrates the response mechanisms of different ECUs when faced with forged messages. The experimental results show that the system can effectively identify abnormal frames and visually display the detection status through LEDs.

mized Algorithm 3 by replacing floating-point operations with integer operations and shift operations. After optimization, the processing time for a single data frame was reduced to just 64 μs. This result demonstrates that the optimized algorithm not only meets real-time requirements but also effectively adapts to the limited computing resources of the ECU.

The detection program designed using the method described in III-A has a size of 3.03 KB. We experimented with different ways of storing the inspection parameters. When using macro definitions, the parameter size for a single identifier occupies 179 bytes. In contrast, when using structures, the structure definition takes up 75 bytes, and the parameters corresponding to the identifier occupy 119 bytes. This demonstrates that using structures can significantly reduce memory usage when the ECU masking field contains multiple identifiers. Storing the inspection parameters of all 10 CAN data frame identifiers mentioned in III-D using structures results in a total memory usage of 1280 bytes, with the entire detection model totaling 4.28 KB in size. In conclusion, the detection method proposed in this paper not only achieves a high detection rate but also effectively utilizes the limited computing resources of the ECU, making it fully suitable for scenarios requiring ECU-level computing resources.

V. Conclusion

This paper proposes an Intrusion Prevention System deployed inside the ECU, aimed at addressing the vulnerability of CAN networks to attacks and defending against potential control disruptions caused by spoofing attacks. The system can detect data frames within the ECU's receive mailbox, thereby preventing the execution of anomalous frames. Furthermore, our detection method is based on unsupervised learning, ensuring a false detection rate of 0% while maintaining a detection rate of 96.5% against targeted attacks in both spoofing and tampering attack scenarios on real datasets. The designed IPS was deployed on an STM32F407ZGT6 development board simulating ECU functionality, with a processing time of just 64 μs per single CAN data frame.

Acknowledgment

This paper was supported by the National Natural Science Foundation of China (Project No. 62172141), Analysis and optimization of network capacity under V2V/V2R hybrid communication mode in Internet of Vehicles.

References

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

[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015.

[3] L. L. Sen Nie and Y. Free-fall, "Hacking tesla from wireless to can bus," *Black Hat USA*, 2017.

[4] S. Nie, L. Liu, Y. Du, and W. Zhang, "Over-the-air: How we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars," *Briefing, Black Hat USA*, vol. 91, pp. 1–19, 2018.

[5] A. Guzman and A. Gupta, *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices*. Packt Publishing Ltd, 2017.

[6] B. Groza and S. Murvay, "Efficient protocols for secure broadcast in controller area networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2034–2042, 2013.

[7] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (can) communication protocol," in *2012 International Conference on Cyber Security*. IEEE, 2012, pp. 1–7.

[8] B. Carnevale, F. Falaschi, L. Crocetti, H. Hunjan, S. Bisase, and L. Fanucci, "An implementation of the 802.1 ae mac security standard for in-car networks," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 24–28.

[9] M. Wolf and T. Gendrullis, "Design, implementation, and evaluation of a vehicular hardware security module," in *Information Security and Cryptology-ICISC 2011: 14th International Conference, Seoul, Korea, November 30-December 2, 2011. Revised Selected Papers 14*. Springer, 2012, pp. 302–318.

[10] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Network*, vol. 31, no. 5, pp. 50–58, 2017.

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

[12] J. Ning and J. Liu, "An experimental study towards attacker identification in automotive networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[13] Y. Zhao, Y. Xun, and J. Liu, "Clockids: A real-time vehicle intrusion detection system based on clock skew," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 593–15 606, 2022.

[14] H. Lee, S. H. Jeong, and H. K. Kim, "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.

[15] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1727–1736, 2020.

[16] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan, and R. Agrawal, "Intrusion detection using payload embeddings," *IEEE Access*, vol. 10, pp. 4015–4030, 2021.

[17] S. B. H. Samir, M. Raissa, H. Touati, M. Hadded, and H. Ghazzai, "Machine learning-based intrusion detection for securing in-vehicle can bus communication," *SN Computer Science*, vol. 5, no. 8, p. 1082, 2024.

[18] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3369–3388, 2018.

[19] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, pp. 5597–5621, 2019.

[20] Y. Wei, C. Cheng, and G. Xie, "Ofids: online learning-enabled and fingerprint-based intrusion detection system in controller area networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 4607–4620, 2022.

[21] T. Matsumoto, M. Hata, M. Tanabe, K. Yoshioka, and K. Oishi, "A method of preventing unauthorized data transmission in controller area network," in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*. IEEE, 2012, pp. 1–5.

[22] S. Longari, M. Penco, M. Carminati, and S. Zanero, "Copycan: An error-handling protocol based intrusion detection system for controller area network," in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, 2019, pp. 39–50.

[23] K. Cheng, Y. Bai, Y. Zhou, Y. Tang, D. Sanan, and Y. Liu, "Caneleon: Protecting can bus with frame id chameleon," *IEEE Transactions on Vehicular technology*, vol. 69, no. 7, pp. 7116–7130, 2020.

[24] P. F. De Araujo-Filho, A. J. Pinheiro, G. Kaddoum, D. R. Campelo, and F. L. Soares, "An efficient intrusion prevention system for can: Hindering cyber-attacks with a low-cost platform," *IEEE Access*, vol. 9, pp. 166 855–166 869, 2021.

[25] S. Longari, C. A. Pozzoli, A. Nichelini, M. Carminati, and S. Zanero, "Candito: Improving payload-based detection of attacks on controller area networks," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 2023, pp. 135–150.

[26] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th annual conference on privacy, security and trust (PST)*. IEEE, 2018, pp. 1–6.

[27] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "Cacan-centralized authentication system in can (controller area network)," in *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, 2014, p. 10.

[28] E. Kristianto, P.-C. Lin, and R.-H. Hwang, "Sustainable and lightweight domain-based intrusion detection system for in-vehicle network," *Sustainable Computing: Informatics and Systems*, vol. 41, p. 100936, 2024.