applied sciences

MDPI

Article

Enhancing Security in Connected and Autonomous Vehicles: A Pairing Approach and Machine Learning Integration

Usman Ahmad [1,2], Mu Han [1,*] and Shahid Mahmood [3]

[1] School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China; usman715@gmail.com
[2] Department of Computational Science, The University of Faisalabad, Faisalabad 38000, Pakistan
[3] Intelligent Mobility and Software, FEV Iberia, 08039 Barcelona, Spain; mahmood@fev.com
* Correspondence: hanmu@ujs.edu.cn

Abstract: The automotive sector faces escalating security risks due to advances in wireless communication technology. Expanding on our previous research using a sensor pairing technique and machine learning models to evaluate IoT sensor data reliability, this study broadens its scope to address security concerns in Connected and Autonomous Vehicles (CAVs). The objectives of this research include identifying and mitigating specific security vulnerabilities related to CAVs, thereby establishing a comprehensive understanding of the risks these vehicles face. Additionally, our study introduces two innovative pairing approaches. The first approach focuses on pairing Electronic Control Units (ECUs) within individual vehicles, while the second extends to pairing entire vehicles, termed as vehicle pairing. Rigorous preprocessing of the dataset was carried out to ensure its readiness for subsequent model training. Leveraging Support Vector Machine (SVM) and TinyML methods for data validation and attack detection, we have been able to achieve an impressive accuracy rate of 97.2%. The proposed security approach notably contributes to the security of CAVs against potential cyber threats. The experimental setup demonstrates the practical application and effectiveness of TinyML in embedded systems within CAVs. Importantly, our proposed solution ensures that these security enhancements do not impose additional memory or network loads on the ECUs. This is accomplished by delegating the intensive cross-validation to the central module or Roadside Units (RSUs). This novel approach not only contributes to mitigating various security loopholes, but paves the way for scalable, efficient solutions for resource-constrained automotive systems.

Keywords: connected and autonomous vehicles (CAVs); cybersecurity; machine learning; support vector machine (SVM); TinyML; authentication; intelligent transportation systems (ITS)

check for updates

Citation: Ahmad, U.; Han, M.; Mahmood, S. Enhancing Security in Connected and Autonomous Vehicles: A Pairing Approach and Machine Learning Integration. *Appl. Sci.* **2024**, *14*, 5648. https://doi.org/10.3390/app14135648

Academic Editor: Yutaka Ishibashi

Received: 5 June 2024
Revised: 22 June 2024
Accepted: 25 June 2024
Published: 28 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Enhancing safety, comfort, and fuel economy, the integration of automated applications and digital connectivity in CAVs aim to supplement or replace human drivers [1]. These modern cars can contain up to 100 ECUs, which interact extensively with sensors, actuators, and other ECUs to obtain and evaluate data vital for making key decisions [2].

The backbone of communication inside CAVs is Inter-Vehicle-Networks (IVNs), formed using standard protocols such as Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay, Media Oriented System Transport (MOST), and Ethernet [3]. Apart from in-vehicle communication, CAVs rely on ad hoc networks employing short-range wireless technologies like Wi-Fi to enable vehicle-to-everything (V2X) communications, including vehicle-to-building (V2B) and vehicle-to-infrastructure (V2I) interactions [4]. These developments raise different security concerns [5–7], but also improve vehicle intelligence, connectivity, and safety. Artificial Intelligence (AI) has lately demonstrated considerable potential for improving automotive systems, including traffic prediction, 3D object detection, problem diagnosis, and cybersecurity. However, deploying AI systems on edge devices

poses challenges due to high computational demands and data requirements. To address these challenges, ongoing machine learning and TinyML initiatives strive to optimize AI algorithms for deployment on resource-constrained devices like microcontrollers.

1.1. Challenges

Despite the benefits of CAV interconnectivity, it introduces vulnerabilities. Critical ECUs managing vehicle operations are prime targets for attacks like frame injection, sniffing, falsification, and Denial of Service (DoS), posing risks to vehicle safety and occupant well-being. Wireless V2X communication channels are also vulnerable to signal jamming, spoofing, and data interception, enabling unauthorized access and control over vehicle systems.

Furthermore, the sensory layer comprising sensors and cameras is susceptible to manipulation and false data injection, potentially leading to erroneous decision-making by vehicle control systems. Addressing these formidable security challenges necessitates robust, adaptive solutions capable of safeguarding both in-vehicle networks and wireless communication interfaces.

1.2. Objective and Contributions

In addressing the evolving security landscape of CAVs, our current study introduces a novel dual-layer pairing methodology. Our current research extends our previous work documented in [8], which concentrated on evaluating the reliability of sensor data from IoT nodes through various machine learning techniques and inventive sensor pairing methods. Our research is driven by the hypothesis that employing this dual-layer approach will not only ensure accurate data validation, but also improve power efficiency and computational effectiveness, without imposing additional memory or network burdens on ECUs. This major contributions of this work are outlined below:

1. **Innovative ECU and vehicle pairing approaches**: introducing a dual-layer methodology that notably strengthens security by securing communication among ECUs within a vehicle and across vehicles, while also safeguarding sensor data from spoofing, injection, and similar attacks.
2. **The practical use and effectiveness of TinyML in embedded systems within vehicles**: Our research further contributes to the field by demonstrating the practicality and effectiveness of TinyML in embedded systems within vehicles. This implementation improves the accuracy and efficiency of security measures in CAVs, providing insights into power utilization and resource optimization. It highlights the practical value of TinyML in environments constrained by resources.
3. **Comprehensive evaluation**: we undertake comprehensive experiments to evaluate the effectiveness of our proposed method in mitigating various security threats across CAVs environments.
4. **Scalability and efficiency**: the study explores particular difficulties presented by limited resources in CAVs, providing insights into implementing security solutions customized for these settings, including strategies and considerations for optimizing memory usage.

By addressing the current deficiencies in security research for CAVs, this study aims to advance the development of automotive systems that are more secure, safe, and dependable, thereby enhancing operational integrity.

1.3. Structure of the Paper

The rest of the paper is structured as follows.

Section 2: This section reviews the relevant literature, providing insights into previous research on security attacks targeting CAVs. It discusses physical attacks, wireless communication attacks, sensing layer and camera attacks, as well as deep learning-based defense systems.

Section 3: Here, we delve into the security concerns pertaining to CAVs addressed in this study. We explore two distinct pairing strategies, discussing their implementation and potential implications.

Section 4: This section provides detailed insights into the electronic components required for the experiment. It outlines the steps for preprocessing the dataset and setting up the experimental configuration. Additionally, it explains how SVM and TinyML techniques are integrated into our solution for data validation.

Section 5: finally, the paper concludes by summarizing the key findings and contributions of the research.

2. Related Work

In response to the evolving landscape of CAVs, significant attention has been drawn to the vulnerabilities inherent in their essential communication protocols, including CAN, LIN, FlexRay, MOST, and Ethernet. Initially, automotive cybersecurity guidelines were formalized under SAE J3061 [9], focusing on rigorous risk assessment, threat modeling, and lifecycle validation. These guidelines laid a foundational framework for robust cybersecurity measures in the design and operational phases of vehicle development.

In recent years, the industry has transitioned to ISO/SAE 21434 [10], a globally recognized standard that builds upon SAE J3061's principles. ISO/SAE 21434 offers a more comprehensive framework for managing cybersecurity risks in automotive systems, emphasizing collaboration among stakeholders and integration with existing engineering processes. This standard ensures that cybersecurity measures are systematically integrated into vehicle development and manufacturing, ensuring alignment with global regulatory standards, including UNECE (WP.29) R155 regulation governing cybersecurity of the road vehicles [11].

Building upon these challenges and guidelines, the following sections provide a detailed overview of the primary security attacks and critically examine existing literature, highlighting the need for more robust defense mechanisms in CAVs. We present an overview of primary security attacks, for which we propose solutions (Figure 1). Our outline addresses the major surfaces of attack faced by CAVs and delves into the deep learning-based security measures utilized in the literature to defend against such breaches.

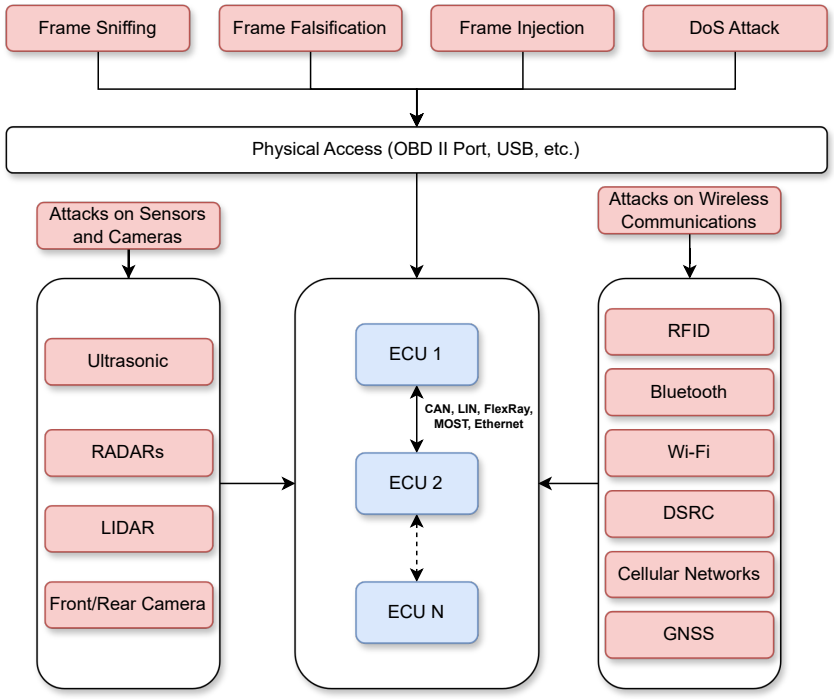


Figure 1. The attack surface for launching security attacks on ECUs.

2.1. Physical Attacks

Analyzing previously recorded CAN frames can reveal information about the system. Koscher et al. [12] developed CarShark, a program designed to intercept CAN frames, revealing their limited range of validity and potential vulnerabilities in specific ECUs. This study was groundbreaking in exposing the vulnerabilities within CAN bus systems. However, its primary limitation lies in the focus on isolated CAN bus attacks without addressing broader vehicle network architectures or the increasing complexity of modern vehicle systems. Additionally, it primarily targets older vehicle models, which may not reflect the current state of automotive technology.

Exploiting the OBD port, attackers can collect diagnostic data, access the in-vehicle network, and install malware [13]. Cai et al. demonstrated backdoor creation in BMW entertainment systems via USB port exploitation [14]. While Cai et al. effectively highlight specific vulnerabilities in BMW systems, their approach may not be generalizable to all vehicle makes and models. Moreover, the study focuses on infotainment systems, potentially overlooking other critical ECU functions. These vulnerabilities allow for the manipulation of fuel levels, speedometer values, or false failure notifications [12].

2.2. Wireless Communication Attacks

Weaknesses in the Hitag2 cipher enable manipulation of remote keyless entry systems [15]. Dibaei et al. exploited this vulnerability to manipulate a Mercedes-Benz vehicle's keyless entry system [16]. While Dibaei et al. provide valuable insights into specific wireless communication vulnerabilities, the narrow focus on a single vehicle model and keyless entry system limits the broader applicability of their findings. Rouf et al. caution against hobbyist manipulation of TPMS, leading to erroneous readings and alerts [17]. While Rouf et al. [17] effectively demonstrate the risks associated with TPMS, the study does not propose robust countermeasures or solutions to mitigate these vulnerabilities, leaving a gap in practical applications.

2.3. Sensing Layer and Cameras Attacks

In this section, we provide a summary of the two major categories of CAV sensing layer attacks.

2.3.1. Wireless Sensors Attacks

Yan et al. [18] and Petit et al. [19] demonstrated security attacks on various sensors, including LIDAR and cameras. Blinding attacks obscure cameras, while replay or relay attacks create artificial echoes to mislead LIDAR. While these studies provide comprehensive insights into potential attacks on vehicle sensors, they often lack detailed countermeasures or defense mechanisms, focusing primarily on the vulnerabilities rather than solutions. Attackers deploy stationary sensors along roadsides for these attacks. This method, while theoretically sound, may not reflect practical deployment challenges or the evolving nature of sensor technologies and their associated security measures.

2.3.2. Analog Sensors Attacks

Exploiting weaknesses in analog sensors, such as ABS wheel speed sensors, can lead to hazardous situations. Shoukry et al. [20] illustrated how attackers can provide malicious signals to cancel out legitimate measurements. While Shoukry et al. provide a clear demonstration of sensor vulnerabilities, the study is limited in scope, addressing only specific sensor types and attack scenarios, potentially overlooking other critical components of vehicle safety systems.

2.4. Deep Learning-Based Defense Systems

Deep learning offers effective control in dynamic environments [21–23]. Basavaraj et al. proposed a lightweight IDS using deep learning to detect anomalies in vehicle systems [24]. Basavaraj et al.'s work is a significant step towards real-time anomaly detection;

however, its effectiveness in diverse vehicular environments and against sophisticated attacks remains to be thoroughly evaluated. Kang and Kang's intrusion detection system utilizes low-dimensional features from IVN packets [25]. While innovative, the reliance on low-dimensional features may limit the system's ability to detect complex, multi-faceted attacks, potentially reducing its overall robustness.

Taylor et al. employed LSTM RNN-based anomaly detectors for intrusion detection [26], while Ahmad et al. used LSTM-RNN to identify drivers based on real-world driving data [27]. Both studies underscore the potential of LSTM-RNN models for enhancing vehicle security, but their dependency on extensive training data and computational resources may limit their applicability in real-time or resource-constrained environments.

3. Methodology

Our pairing approach effectively addresses numerous security issues affecting ECUs in CAVs. To test the hypothesis that the dual-layer pairing approach enhances the security of ECUs in CAVs, we developed two distinct pairing methods: ECU pairing within a vehicle and vehicle pairing between CAVs. These methods were integrated with SVM and TinyML techniques for effective data validation. This integration not only ensures high accuracy in data validation, but also maintains efficiency in power consumption and computation. In this section, we detail the specific security challenges that our research targets and explain how our pairing approach, incorporating SVM and TinyML, enhances the security and performance of CAVs applications.

3.1. Threat Model: Addressing Security Attacks on ECUs

CAVs extensively use a wide range of wireless and physical communication channels and multiple sensors. These communication channels are vulnerable to various security attacks [19,28,29]. ECUs establish connections with various sensors and actuators, communicate with other ECUs, and engage with communication interfaces, enabling them to acquire and interpret data, make informed decisions, and manage the vehicle's systems effectively. The attack surface of CAVs refers to the collection of entry points, vulnerabilities, pathways, and methods that an intruder could exploit to launch a cyberattack on ECUs. These entry points can be classified into three main types (shown in Figure 1): physical access attacks, sensing layer attacks, and wireless communication attacks. Below are descriptions of the attack surfaces associated with these categories that could be used to infiltrate ECUs and carry out malicious actions.

3.1.1. Physical Attacks

Due to numerous physical interfaces, Intelligent Transportation Systems (ITS) are highly susceptible to cyberattacks. These interfaces can provide direct access to the vehicle's IVNs, such as the CAN and LIN buses, as well as the ECUs, making it easier for an attacker to compromise the system. One such interface is the OBD-II port, commonly used by technicians to diagnose and program ECUs during inspections [12]. It should be noted that if an attacker already has physical access to the vehicle, especially in a service station context, executing an attack becomes significantly easier. Physical security measures are therefore crucial in such environments to prevent unauthorized access [30].

CAVs are also vulnerable to physical access through additional interfaces like USB ports and Bluetooth communication, which could potentially be exploited by attackers. For instance, an attacker with physical access to a USB port can upload malicious firmware or software directly to the vehicle's ECU. Similarly, Bluetooth interfaces can be exploited to gain unauthorized access if not properly secured.

Experimental investigations have been conducted on attacks targeting in-vehicle networks [31]. Various types of attacks on ECUs include [32,33]:

- **Frame sniffing**: Due to the broadcast-based communication protocol of CAN buses and the absence of designated sender and receiver addresses, every node in the network receives frames without any Message Authentication Code (MAC) or digital

signature, resulting in insecure transmission [34,35]. Frame sniffing is the starting point for all in-vehicle communication attacks. Since CAN frames are sent to all ECUs, a compromised ECU can intercept all frames transmitted through the CAN bus via interfaces like the OBD port, USB ports, and aftermarket devices [36,37].

- **Frame falsifying**: If attackers possess extensive knowledge of legitimate CAN frames, they can construct attacks by transmitting counterfeit frames via the CAN bus. These fraudulent frames may contain falsified data that can trick authentic ECUs [12,38].
- **Frame injection**: Attackers can initiate actions by exploiting a malicious node, such as a malware-infected laptop connected to the OBD port. In severe scenarios, the attacker may have reprogrammed the ECU and added wireless or remote communication capabilities such as Wi-Fi or Bluetooth [12].
- **Replay attack**: similarly to the attacks described earlier, attackers can use a malicious node to send commands resulting in legitimate frames being transmitted to the CAN bus at specific time intervals [12,39].
- **DoS attack**: attackers can direct the malicious node to continuously broadcast high-priority frames, preventing other nodes from transmitting their frames if a higher-priority frame is present in the network [12].

Given the ease with which physical access can facilitate these attacks, it is essential to implement robust physical security measures. This includes securing access to OBD-II ports and other interfaces, using tamper-evident seals, and conducting regular security audits to detect and mitigate potential vulnerabilities.

3.1.2. Sensing Layer and Camera Attacks

CAVs' wireless sensors, like LIDAR, RADAR, cameras, and remote keyless entry (RFID), can be exploited within short range. Depending on the channel distance, hackers can execute such assaults by placing a wireless transmitter next to the receiver (sensor) [40]. Sensing layer attacks can be classified into two categories: wireless sensor attacks (including RADARs, ultrasonic sensors, and cameras) [18,19] and analog sensor attacks, such as anti-lock braking system attacks [20].

3.1.3. Wireless Communication Attacks

ECUs in CAVs are equipped with wireless communication technologies such as radio communication, Bluetooth, Wi-Fi, DSRC, cellular networks, and GNSS, transforming CAVs into open systems and creating potential threats [41,42]. Wi-Fi has become an appealing target for a wide range of security threats, carrying more than 75% of the Internet traffic in the last mile [43]. Cybercriminals often engage in blocking and spoofing GNSS signals to compromise authenticity and integrity [44–47]. Cellular networks face security threats like sniffing and jamming attacks [48].

3.1.4. Application Attacks

Technologies enabling precise vehicle control have boosted driver safety and convenience, but have also become targets for exploitation. Surprisingly, open platforms such as the Android OS have been incorporated into vehicle systems without sufficient consideration for potential security issues [49].

3.2. Diving into Pairing Approach

In this section, we introduce two different pairing approaches aimed at improving the security of CAVs: intra-vehicle ECU pairing and vehicle-to-vehicle pairing. In our approach to ECU pairing, the emphasis is on linking individual ECUs within a vehicle. Establishing secure connections between ECUs helps to guarantee reliable and coordinated communication inside the vehicle network. On the other hand, the vehicle-to-vehicle pairing technique concentrates on linking whole vehicles, thereby allowing secure communication between them. Our pairing approaches are supposed to provide a whole, effective solution

covering the security concerns CAVs come with in internal and external communication environments.

3.2.1. Pairing of ECUs and Validation of Data

This section presents the details of our innovative ECU pairing approach designed to identify security threats without taxing the ECUs with extra memory or network traffic. Crucially important is the onboard central module, which uses machine-learning techniques for attack detection. Figure 2 shows how our ECU pairing model is implemented. The criteria we apply to choose the suitable ECUs for matching are briefly described below.

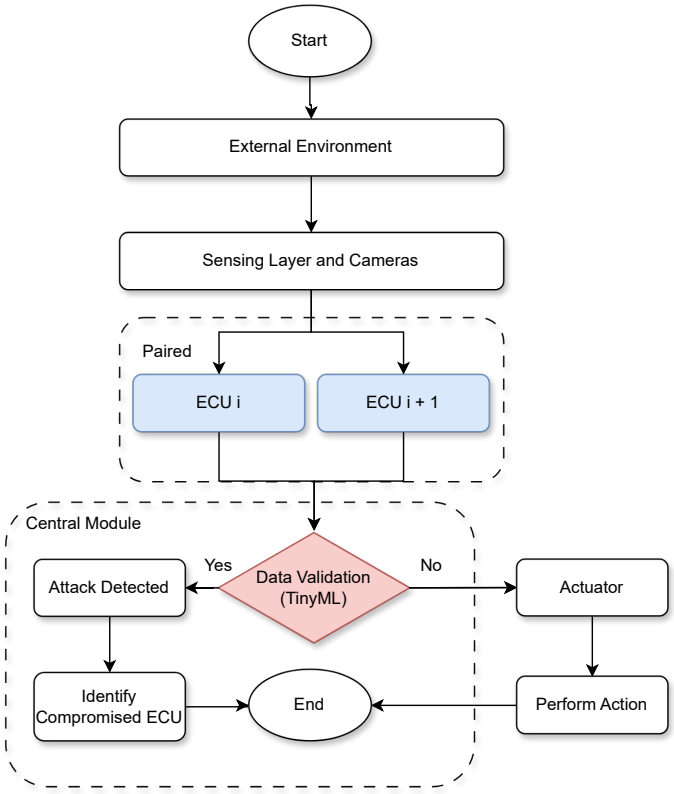


Figure 2. Flow of the proposed ECU pairing authentication model.

Selecting ECUs for Pairing and Data Validation

First, the choice of ECUs to pair starts with data compatibility. Each ECU is first evaluated to see whether it shares compatible, similar data with another ECU in the network, therefore ensuring smooth data flow between paired ECUs. The method guarantees that every ECU is unique, thereby prohibiting self-pairing, and takes ECU availability into account to prevent pairing units already engaged with others. Special pair IDs are assigned to stop simultaneous involvement in several pairings. ECUs are chosen for pairing in the design of the vehicle depending on their common use of particular sensor data, hence improving system security and dependability. For activities based on wheel speed sensor data, the Tyre Pressure Monitoring System (TPMS) might, for example, be combined with the Stability Control System (SCS) ECU. Aggregating redundant and complimentary sensor data from these linked ECUs allows for data cross-valuation.

Data from the SCC ECU and TPMS is forwarded to the central module for data cross-validation. Analyzing data from both connected ECUs ensures reliability and precision, therefore facilitating the detection of anomalies or differences implying security breaches. This well-crafted pairing approach seeks to preserve data integrity, therefore strengthening the system's resistance to possible hazards resulting from erroneous data or sensor manipulation by threat actors.

By means of a module, the central validation of data greatly improves the control system security of the vehicle. ECUs are coupled based on their mutual need to exchange certain information. This method guarantees strong data validation, even when operational effectiveness is maintained. Whereas Algorithm 1 fully describes the pairing and data validation procedure, Figure 3 offers an abstract depiction of our ECU pairing concept. Each ECU is allocated a unique pair ID and ECU ID to ensure robust security since only two ECUs are linked to the network. Effective detection accuracy is enhanced by the good data comparison between connected ECUs enabled by this design, so the module efficiently follows these pair IDs to support data comparison.

Algorithm 1 ECU pairing and data validation

Initialize an empty list called *pairedECUs* to store the paired ECUs.
Initialize an empty dictionary called *pairIDs* to store the pair ID for each ECU.
for each ECU_i in the network **do**
  Set a flag variable *isPaired* to False for ECU_i.
  for each ECU_j in the network **do**
    if ECU_i and ECU_j exchange the same kind of information **and** $ECU_i \neq ECU_j$ **then**
      if ECU_i and ECU_j are available for pairing **and** pairIDs[ECU_i] $\neq$ pairIDs[ECU_j] **then**
        Set the flag variable *isPaired* to True for ECU_i and ECU_j.
        Add ECU_i and ECU_j to the *pairedECUs* list.
        Assign a unique pair ID *pairID* to the pair (ECU_i, ECU_j).
        Associate the pair ID with ECU_i and ECU_j in the *pairIDs* dictionary.
      end if
    end if
  end for
  for each *pairedECU* in the *pairedECUs* list **do**
    Validate the second ECU data based on the first ECU's input in the pair using a validation function.
    if the data are valid **then**
      continue to the next pair
    else
      Attack is Detected. Remove the pair (ECU_i, ECU_j) from the *pairedECUs* list.
      Update the *pairIDs* dictionary accordingly.
    end if
  end for
end for

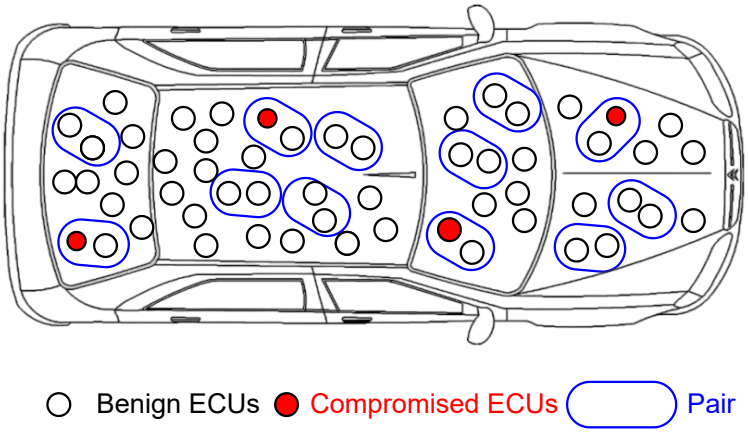


Figure 3. Abstract depiction of proposed ECU pairing model.

Algorithm 1: ECU Pairing and Data Validation

Algorithm 1 presents a technique for pairing ECUs inside a network so that they can analyze their data and exchange compatible information to find probable attacks. Initially, it initializes an empty list, *pairedECUs*, to keep track of successful pairings, and an empty dictionary, pairIDs, to store unique pair IDs for each ECU. For each ECU (ECUi), the algorithm sets a flag, *isPaired*, to False. It then iterates through each other ECU (*ECUj*). If *ECUi* and *ECUj* exchange compatible information, are distinct ECUs, and both are available for pairing with unique pair IDs, they are designated as paired. They are included in *pairedECUs* with a unique pair ID recorded in *pairIDs*. Once all pairings are established, each pair in *pairedECUs* undergoes validation. If one ECU's data validates the other's, the process continues. If not, an attack is identified, the pair is removed from *pairedECUs*, and *pairIDs* is updated. This ensures only valid and secure ECU pairings are maintained within the network.

3.2.2. Vehicle Pairing and Data Validation

In this section, we present a vehicle pairing model aimed at identifying security attacks on ECUs through inter-vehicle pairing, ensuring no extra memory or network load on the vehicles. Central to this security model is a centralized database integrated into the infrastructure, utilizing advanced machine learning methods to accurately detect and pinpoint attacks. Figure 4 shows the flow of our vehicle pairing model. We have paired vehicles of different types that engage in V2V communication, exchanging identical information such as road conditions. Each vehicle collects event data from its sensor network and sends it to a centralized database specialized in identifying potential attacks. Throughout these data exchanges, vehicles utilize vehicle-to-infrastructure communication protocols to interact with the central database. In V2V, vehicles have the capability to exchange crucial information to enhance security. Should Vehicle A and Vehicle B be connected, for instance, they can share real-time traffic updates. Each vehicle's data are forwarded to an RSU for confirmation. The RSU verifies its correctness and legitimacy by means of information analysis from both cars, therefore facilitating traffic condition monitoring, risk detection, and the prevention of fake data injection attacks.

Deciding Factors: Choosing Vehicle for Pairing

Confirming their involvement in V2V communication and providing suitable data, including road conditions, would help establish the related vehicle selection criteria. Every car has to be unique; self-pairing is unacceptable. Moreover, both vehicles have to be easily accessible for pairing purposes; consequently, they are not already paired. To allow exact validation and comparison of event data acquired from the matched vehicles, the central database generates unique pair IDs. This method ensures data confidentiality and integrity without incurring higher overheads on cars.

Each vehicle has a different pair ID and vehicle ID to help identify possible targets of security attacks. Carefully entered in the central database, these pair IDs enable a thorough comparison of event information gathered from every vehicle pair. By verifying the data integrity of every vehicle's information with its matching counterpart, this verification system guarantees that every vehicle runs autonomously and acts as a verifying agent for its matched vehicle, therefore guaranteeing the validity of sensed data without adding additional memory or communication overhead. While Algorithm 2 defines the procedures involved in vehicle pairing and supporting data validation, Figure 5 provides a conceptual overview of our vehicle pairing method.

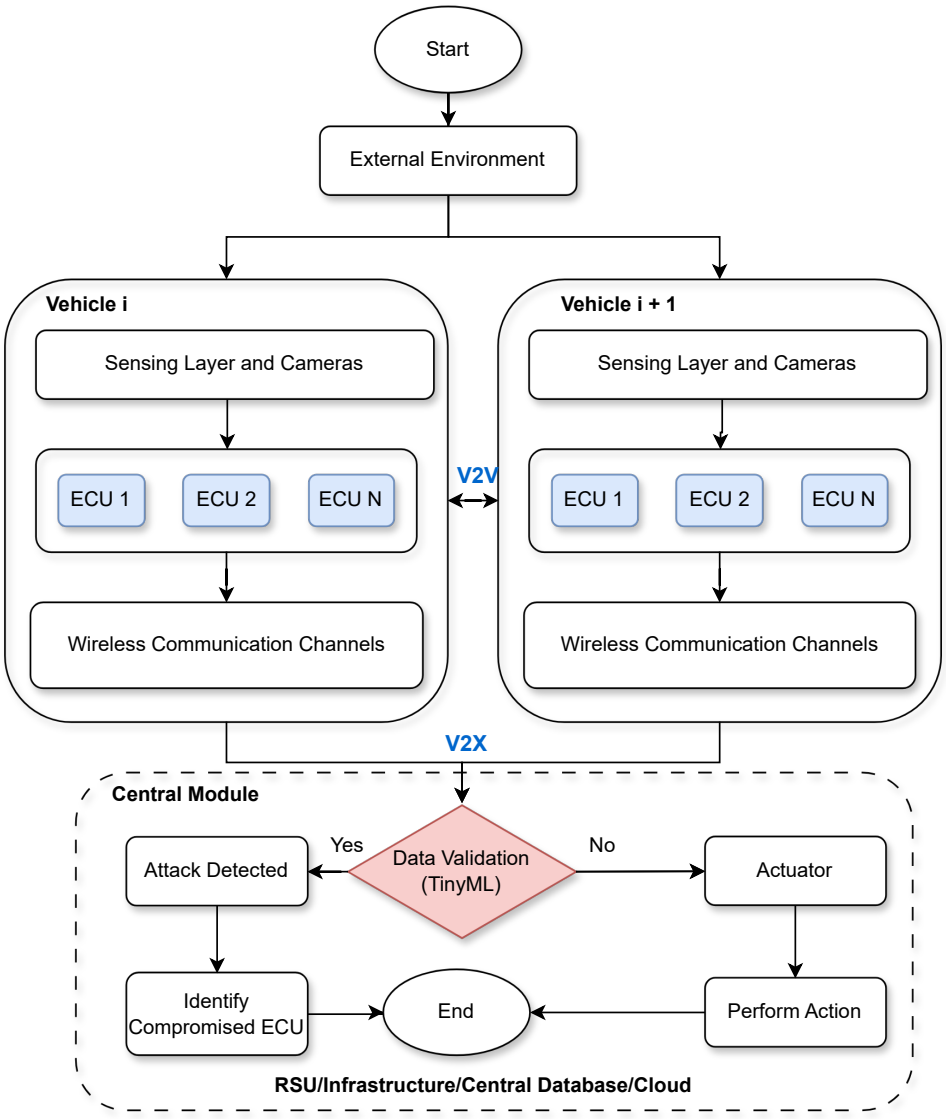


Figure 4. Flow of the proposed vehicle pairing model.

Algorithm 2: Vehicle Pairing and Data Validation

Algorithm 2 provides secure communication and shows our way of pairing the vehicles in a network. The approach begins with an empty list called *pairedVs* to document the vehicles that have been successfully paired and an empty dictionary called *pairIDs* to retain distinct pair IDs for every vehicle. Every vehicle (V_i) in the network starts with a false boolean variable known as isPaired. The machine then goes on to compare V_i with every single other vehicle (V_j). V_i and V_j are designated as paired if they engage in vehicle-to-vehicle communication (V2V), are different vehicles, and both have unique pair IDs for possible pairing. Each of the vehicles in the *pairedVs* collection has a unique pair ID; this ID is kept in the *pairIDs* collection. If an attack is detected, the pair is removed from the list of paired vehicles, and the pair IDs are updated accordingly. This procedure ensures that only legitimate and secure vehicle pairings remain within the network.

Algorithm 2 Vehicle pairing and data validation

```
Initialize an empty list called pairedVs to store the paired vehicles.
Initialize an empty dictionary called pairIDs to store the pair ID for each vehicle.
for each V_i in the network do
    Set a flag variable isPaired to False for V_i.
    for each V_j in the network do
        if V_i and V_j are communicating (V2V) and V_i ≠ V_j then
            if V_i and V_j are available for pairing and pairIDs[V_i] ≠ pairIDs[V_j] then
                Set the flag variable isPaired to True for V_i and V_j.
                Add V_i and V_j to the pairedVs list.
                Assign a unique pair ID pairID to the pair (V_i, V_j).
                Associate the pair ID with V_i and V_j in the pairIDs dictionary.
            end if
        end if
    end for
    for each pairedV in the pairedVs list do
        Validate the data of the second vehicles based on the input of the first vehicle in
        the pair using a validation function.
        if the data are valid then
            continue to the next pair
        else
            Attack is Detected. Remove the pair (V_i, V_j) from the pairedVs list.
            Update the pairIDs dictionary accordingly.
        end if
    end for
end for
```

3.2.3. Memory Considerations

In contrast to earlier methods like witness-based approaches [50,51], where each electronic component needed extra buffer storage as a witness, our new approach follows a different direction. In those previous methods, the confidential data from the source component had to be stored in buffer memory to verify its authenticity, leading to higher memory usage. Unlike previous approaches that burdened ECUs with storing confidential data for verification, our solution adopts a different strategy. Rather than relying on ECUs to store such data, our method utilizes a centralized module. This module utilizes machine learning techniques to authenticate event data collected from paired ECUs. Centralizing the verification process eliminates the necessity for ECUs to retain extra buffer storage for the private information of their counterparts. By removing the need for ECUs to store sensitive data, our approach notably reduces memory usage. Furthermore, this decrease in memory load does not compromise the ability to effectively detect security breaches. Our approach successfully detects security breaches without adding any additional strain on the memory or network resources of the ECUs.

4. Experimentation and Results

In this section, we detail the essential electronic components required for the experiment and describe the procedures for organizing and preparing the dataset. Following this, we present the results obtained from applying the SVM and TinyML algorithms to our data.

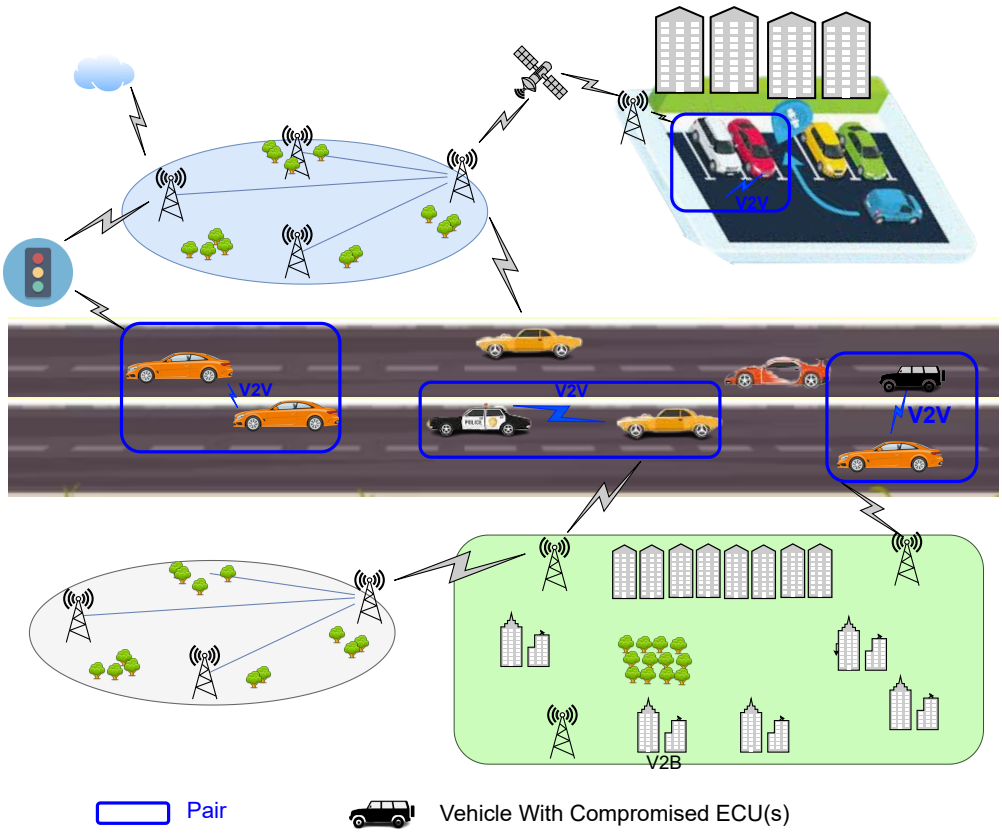


Figure 5. Abstract depiction of proposed vehicle pairing model.

4.1. Testing Environment Setup

When developing applications for edge devices like ECUs, many of the techniques and structures used in traditional machine learning approaches are still relevant. The key distinction lies in the optimization required for these models to function efficiently on smaller, resource-constrained devices. This is where TinyML comes into play, enabling machine learning models to perform tasks directly on edge devices with limited computational power and memory.

We have utilized TensorFlow Lite for Microcontrollers, a widely adopted framework tailored for deep learning on edge devices. TensorFlow Lite for Microcontrollers is specifically designed to implement machine learning models on embedded systems with severely limited resources, typically only a few kilobytes of RAM. This makes it an ideal choice for ECUs in automotive applications, where the available memory and computational capacity are constrained.

We selected Python as the primary language for developing our deep learning models, due to its extensive libraries and ease of use. However, when it comes to deploying these models on edge devices, TensorFlow Lite offers significant advantages. It is efficient, does not require an internet connection for model creation and deployment, and supports C/C++ for performance-critical applications.

To embark on a TinyML project using TensorFlow Lite, a compatible microcontroller board is essential. Below, we outline the primary steps involved in creating a TinyML project [52]:

- Utilize cloud computing resources to train the model, leveraging their high-performance capabilities for efficient and scalable machine learning training.
- Optimize the model for a lightweight environment by transitioning from TensorFlow to TensorFlow Lite, ensuring it is suitable for deployment on edge devices.

- Deploy the model on a specific microcontroller, such as using the Arduino library, to run the inference efficiently on resource-constrained hardware.

Additionally, we conducted the SVM experiments independently on a different machine, using the scikit-learn library to ensure robustness and reliability of the results.

Hardware Configuration

We utilized the Arduino Nano 33 BLE Sense, a microcontroller supported by the library. This microcontroller features a 32-bit ARM Cortex-M4F processor with 256 KB of RAM and 1 MB of program memory, running at 64 MHz. Its capabilities are sufficient to operate TinyML models effectively. The Arduino Nano 33 BLE Sense is equipped with a comprehensive suite of sensors, including those for color, motion, brightness, gesture, vibration, orientation, proximity, humidity, temperature, and pressure. Additionally, it includes a Bluetooth low-energy module and a digital microphone, making it highly versatile for various applications.

For our experiment, we connected three Arduino Nano 33 BLE Sense boards using the Universal Asynchronous Receiver-Transmitter (UART) wired communication protocol. Two of these boards function as ECUs, responsible for collecting and transmitting sensory data, while the third board serves as the vehicle's central module. The UART communication protocol facilitates seamless data transmission and sharing between the ECUs and the central module, ensuring efficient and reliable communication within the system. Given that the pairing approach fundamentally operates similarly in both ECU pairing and vehicle pairing scenarios, our experimentation focuses exclusively on the ECU pairing model.

4.2. Experiments and Results

This section covers the organization and preprocessing of the dataset, followed by the presentation of the experimental results obtained from applying SVM and TinyML techniques.

4.2.1. Dataset Organization and Preprocessing

Our dataset consists of eight attributes, with seven serving as input features from the two ECUs (non-target) and one acting as the label. The non-target features include ECU Serial Number, ECU Type, Sensor Type, Sensor Serial Number, Sensor Connectivity (wired or wireless), Sensor Data Timestamp, and Sensor Measurement/Output (motion detection). The label feature categorizes each record as either benign or malicious. Each dataset record corresponds to an individual observation. We have a total of 1000 records in our dataset with an equal distribution of benign and malicious that is 50/50 split. The dataset preparation follows the steps below.

Data Collection's Experimental Setup

Our experimental setup involved using two Arduino Nano 33 BLE Sense devices as ECUs for motion detection. We devised a systematic approach to detect motion and associate it with object detection. After continuously reading values from the Arduino ECU boards function as ECUs within a loop, we compared the values in the feature labeled "Sensor Measurement/Output (motion detection)" values of paired sensors at the same time stamp that is stored in the feature labeled "Sensor Data Timestamp". These values are compared at the third board, which serves as the vehicle's central module. If the values are identical, we classify this record as benign; otherwise, it is classified as a malicious record. Each data entry was then stored in a text file. We carefully considered the error margin inherent in sensor measurements and accounted for it during the comparison process to ensure accuracy. By adjusting the acceptable range of deviation, we mitigated the impact of minor discrepancies due to sensor noise or timing differences, thereby enhancing the reliability of our classification. Finally, we made an adjustment to the delay time using the delay() function in order to meet the required response time of the system.

Preprocessing

We deleted null and negative values from our produced dataset during the preparation stage. We also scaled the data using *Min-Max* normalization between 0 and 1. The formula applied for this normalizing is displayed here:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

We employ the above formula in the *Min-Max* normalizing process to standardize the values. X here is the original value; X_{min} and X_{max} correspondingly show the minimum and maximum values of the dataset. Denoted by X_{new}, the output is the normalized value. Dataset entries including null values were either deleted or controlled using mean, median, or mode imputation. Appropriate transformations were performed to manage negative values in the dataset, such as turning them into absolute numbers or changing them by adding the absolute minimum value, guaranteeing all values were non-negative.

4.2.2. Experimental Results

Using two alternative assessment approaches, we split the data into training and testing sets and perform 10-fold cross-validation. This part concentrates on the SVM and TinyML algorithm findings.

Support Vector Machine Results

Extensive research has been performed on SVM, a machine learning model well respected for handling both regression and classification tasks [53]. In this work, we compared the performance of SVM with the TinyML method using data classification. Rooted in statistical learning theory, SVM is preferred for its ability to avoid local optima and distinguishes itself from other classification algorithms. SVM operates by determining ideal hyperplanes to define different data classes. It achieves this with kernel functions, therefore allowing SVM to operate efficiently even in cases where linear separation of data are not possible. Specifically, the Radial Basis Function (RBF) kernel is adept at managing nonlinear scenarios.

The dimensions of the SVM plot are determined by the number of features in our dataset, where each data record is depicted as a point in a multi-dimensional space. SVM's objective is to identify a hyperplane, or a set of hyperplanes, that effectively separate the classes within this space. Importantly, SVM strives to maximize margins, which denote the distance between the classifier and the nearest training data points. Table 1 presents the results of the SVM learning model, showing how effective it was using two different evaluation methods; that is, train/test split and 10-fold cross-validation. In the train/test split section, the SVM model achieved an accuracy of 96.1%, indicating the percentage of correctly classified instances. Further information about the model's performance is provided by the precision and recall measures, which are 94.2% and 94.7%, respectively. While precision is the proportion of true positive predictions among all positive predictions, recall, on the other hand, is the proportion of real positives that are detected by the model properly.

Table 1. SVM and TinyML learning results using train/test split and 10 Fold cross-validation.

Model Validation Technique	Performance Metrics	SVM	TinyML
Train/Test Split	Accuracy	96.10%	93.40%
	Precision	94.20%	94.10%
	Recall	95.70%	95.40%
10 Fold Cross-Validation	Accuracy	97.20%	96.60%
	Precision	95.30%	95.20%
	Recall	96.90%	96.30%

When the SVM model was applied to a 10-fold cross-validation, its accuracy increased to 97.2%, showing strong generalization across various subsets of the dataset. The results of the SVM model are summarized in Table 1; the precision and recall are 95.3% and 96.9%, respectively. During 10-fold cross-validation, the SVM model achieved an accuracy of 97.2%, demonstrating robust performance across different subsets of the dataset. The detailed results are shown in Table 1, with precision at 95.3% and recall at 96.9%.

We conducted an analysis using a Shallow Neural Network (SNN), where we progressively added layers to improve its capabilities. Our evaluation criteria encompassed accuracy, precision, and recall. The outcomes of the SNN assessment are presented in Table 1. The model attained an accuracy of 93.4%, indicating how accurately the SNN predicted and classified instances. The results of the SNN analysis are shown in Table 1. Our model achieved an accuracy of 93.4%, showing the overall correctness of the SNN's predictions, or the percentage of correctly classified instances among the total. The precision rate results in 94.1%, indicating the accuracy of positive predictions, or the percentage of true positive instances among all instances predicted as positive by the SNN. Likewise, the recall rate, also called sensitivity or true positive rate, is 95.4%, showing the SNN's ability to correctly identify all relevant instances.

When employing a 10-fold cross-validation approach, the overall correctness of predictions yielded an accuracy rate of 96.6%. For positive predictions, the precision reached 95.2%, with a recall rate of 96.3%. A comparison between SVM and TinyML results is presented in Figure 6.

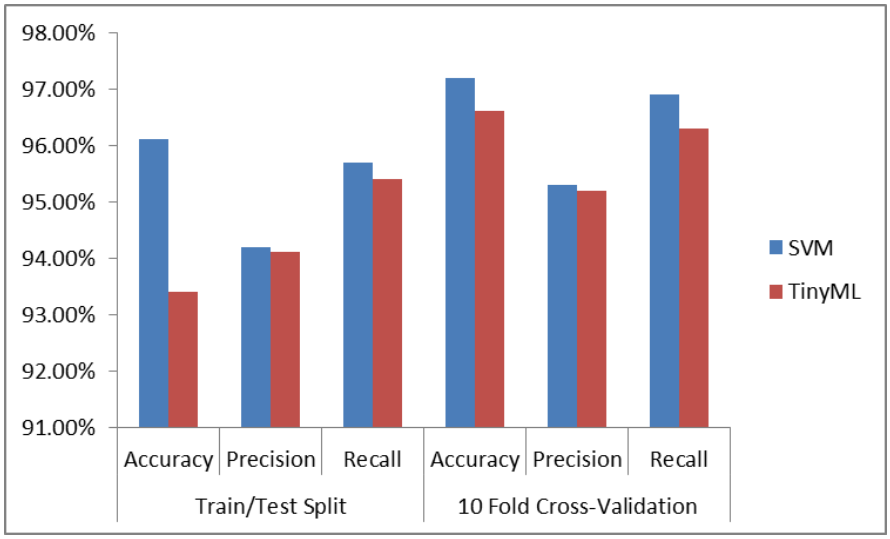


Figure 6. SVM and SNN learning results comparison.

4.3. Foiling Dual Attacks on Paired Devices

The robust security measures implemented significantly raise the bar for attackers attempting to target both devices within a pair. This heightened security is primarily attributed to the secure storage of vital pairing information, such as the unique pair ID assigned to ECUs (for ECU pairing) or vehicles (for vehicle pairing). This crucial data are securely stored within the processing unit where the TinyML algorithm operates. Consequently, malicious actors face formidable challenges in attempting to uncover the elusive pair ID and orchestrating coordinated attacks on both ECUs and vehicles. The results of our study validate the hypothesis that the dual-layer pairing approach enhances the security of ECUs in CAVs. Key findings include:

Accuracy: the integrated SVM and TinyML techniques achieved a remarkable accuracy rate of 97.2% in data validation.

Scalability: The proposed method did not impose any additional memory or network burden on the ECUs. Instead, it facilitated seamless data cross-validation by the central ECUs or RSUs.

These results substantiate our hypothesis, highlighting the effectiveness of the dual-layer pairing methodology in strengthening the security posture of CAVs against potential cyberattacks.

Heterogeneous ECU Pairing

In ECU pairing, the types of pairing can be broadly categorized as homogeneous and heterogeneous. Homogeneous pairing involves pairing electronic components or sensors of the same type. For example, a vehicle may feature multiple sensors of the same kind, such as various cameras (front-facing, side-view, rear-view) or LIDAR and radar sensors, aimed at collecting environmental data. Similarly, external sensors in ITS serve various functions like vehicle detection, monitoring road surface conditions, or assessing weather conditions.

Conversely, heterogeneous pairing involves pairing electronic components of different types that share the same information. For instance, within a vehicle, the navigation system, ADAS, and telematics components all rely on GPS for location data. Similarly, components like the tire pressure monitoring system, electronic stability control, and anti-lock brake systems utilize wheel speed sensors. Outside the vehicle, sensors serve multiple components such as supplying data to access control, toll collection, or parking management systems, as well as sharing weather and air quality information with traffic management systems, variable message signs, and road condition monitoring systems.

In our research, we focused on the heterogeneous pairing of ECUs due to limitations in vehicle architecture that may not accommodate multiple ECUs or sensors for a single task. However, in future studies, homogeneous pairing could be explored to address time constraints and congestion issues in transportation systems more effectively.

5. Conclusions

This study presents a novel pairing approach aimed at enhancing the security of CAVs. Emphasizing especially ECU vulnerabilities, the proposed method offers two innovative techniques to address different security challenges inside CAVs: ECU pairing and vehicle pairing. Using machine learning technologies such as SVM and TinyML for data validation, the solution raises CAVs' ability to identify and prevent probable security breaches. Our results confirm that adding SVM and TinyML methods to a dual-layer pairing scheme greatly improves ECU security inside CAVs. The high accuracy rate of 97.2% in data validation demonstrates the usefulness of this method in real-life situations. Moreover, the scalability of the system supports our hypothesis, since it allows the ECUs to run free from additional memory or network burden. These results not only help to reduce vulnerabilities connected to different security attacks, but also offer a customized solution relevant to the future ITS. Furthermore, the adaptability of the pairing methodology goes beyond enhancing security, offering exciting prospects to enhance the effectiveness and dependability of ITS. Future research endeavors might investigate broader uses of this method, potentially tackling non-security issues within transportation systems and pushing forward the domain of intelligent transportation. Ultimately, the proposed pairing approach signifies a substantial leap forward in fortifying the security and robustness of CAVs against evolving cyber threats.

Author Contributions: Conceptualization, U.A. and S.M.; Methodology, U.A.; Software, U.A.; Validation, S.M.; Formal analysis, U.A. and S.M.; Investigation, U.A.; Resources, M.H.; Data curation, S.M.; Writing—original draft, U.A. and S.M.; Writing—review & editing, S.M.; Visualization, M.H.; Supervision, M.H.; Project administration, U.A. and M.H.; Funding acquisition, U.A. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: The Jiangsu Province Excellent Postdoctoral Programme of China (Grant Number 2023ZB89) funded this research.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: The original contributions presented in the study are included in the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Montanaro, U.; Dixit, S.; Fallah, S.; Dianati, M.; Stevens, A.; Oxtoby, D.; Mouzakitis, A. Towards connected autonomous driving: Review of use-cases. *Veh. Syst. Dyn.* **2019**, *57*, 779–814. [CrossRef]
2. Charette, R.N. This Car Runs on Code-IEEE Spectrum. IEEE Spectrum: Technology, Engineering, and Science News. 2009. Available online: https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code (accessed on 26 June 2024).
3. Huang, J.; Zhao, M.; Zhou, Y.; Xing, C.C. In-vehicle networking: Protocols, challenges, and solutions. *IEEE Netw.* **2018**, *33*, 92–98. [CrossRef]
4. Li, B.; Peng, Z.; Hou, P.; He, M.; Anisetti, M.; Jeon, G. Reliability and capability based computation offloading strategy for vehicular ad hoc clouds. *J. Cloud Comput.* **2019**, *8*, 1–14.
5. Chowdhury, M.; Islam, M.; Khan, Z. Security of connected and automated vehicles. *arXiv* **2020**, arXiv:2012.13464.
6. Ahmed, A.; Iqbal, M.M.; Jabbar, S.; Ibrar, M.; Erbad, A.; Song, H. Position-based emergency message dissemination schemes in the internet of vehicles: A review. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 13548–13572. [CrossRef]
7. Mahmood, S.; Nguyen, H.N.; Shaikh, S.A. Systematic Threat Assessment and Security Testing of Automotive Over-The-Air (OTA) Updates. *Veh. Commun.* **2022**, *35*, 100468. [CrossRef]
8. Ahmad, U. A node pairing approach to secure the Internet of Things using machine learning. *J. Comput. Sci.* **2022**, *62*, 101718. [CrossRef]
9. *J3061_202112*; Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. SAE International: Warrendale, PA, USA, 2016.
10. *ISO/SAE 21434: 2021*; Road Vehicles: Cybersecurity Engineering. ISO: Geneva, Switzerland, 2021.
11. Costantino, G.; De Vincenzi, M.; Matteucci, I. A comparative analysis of unece WP.29 R155 and ISO/SAE 21434. In Proceedings of the 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 6–10 June 2022; IEEE: Hoboken, NJ, USA, 2022; pp. 340–347.
12. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; et al. Experimental security analysis of a modern automobile. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; IEEE: Hoboken, NJ, USA, 2010; pp. 447–462.
13. Carsten, P.; Andel, T.R.; Yampolskiy, M.; McDonald, J.T. In-vehicle networks: Attacks, vulnerabilities, and proposed solutions. In Proceedings of the 10th Annual Cyber and Information Security Research Conference, Oak Ridge, TN, USA, 7–9 April 2015; pp. 1–8.
14. Cai, Z.; Wang, A.; Zhang, W.; Gruffke, M.; Schweppe, H. 0-days & mitigations: Roadways to exploit and secure connected BMW cars. *Black Hat USA* **2019**, *2019*, 39.
15. Liu, H.; Ma, J.; Zhu, S.; Lu, Z.; Liu, Z. Practical contactless attacks on hitag2-based immobilizer and RKE systems. *DEStech Trans. Comput. Sci. Eng.* **2018**, 505–512. [CrossRef]
16. Dibaei, M.; Zheng, X.; Jiang, K.; Maric, S.; Abbas, R.; Liu, S.; Zhang, Y.; Deng, Y.; Wen, S.; Zhang, J.; et al. An overview of attacks and defences on intelligent connected vehicles. *arXiv* **2019**, arXiv:1907.07455.
17. Rouf, I.; Miller, R.D.; Mustafa, H.A.; Taylor, T.; Oh, S.; Xu, W.; Gruteser, M.; Trappe, W.; Seskar, I. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In Proceedings of the USENIX Security Symposium, Washington, DC, USA, 11–13 August 2010; Volume 10.
18. Yan, C.; Xu, W.; Liu, J. *Can You Trust Autonomous Vehicles: Contactless Attacks against Sensors of Self-Driving Vehicle*; DEF CON: Las Vegas, NV, USA, 2016.
19. Petit, J.; Stottelaar, B.; Feiri, M.; Kargl, F. Remote attacks on automated vehicles sensors: Experiments on camera and LiDAR. *Black Hat Eur.* **2015**, *11*, 995.
20. Shoukry, Y.; Martin, P.; Tabuada, P.; Srivastava, M. Non-invasive spoofing attacks for anti-lock braking systems. In Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2013: 15th International Workshop, Santa Barbara, CA, USA, 20–23 August 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 55–72.
21. Wang, Y.; Zhang, D.; Liu, Y.; Dai, B.; Lee, L.H. Enhancing transportation systems via deep learning: A survey. *Transp. Res. Part C Emerg. Technol.* **2019**, *99*, 144–163. [CrossRef]
22. Eskandarian, A. *Handbook of Intelligent Vehicles*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 2.
23. Minhas, A.A.; Jabbar, S.; Farhan, M.; Najam ul Islam, M. A smart analysis of driver fatigue and drowsiness detection using convolutional neural networks. *Multimed. Tools Appl.* **2022**, *81*, 26969–26986. [CrossRef]

24. Basavaraj, D.; Tayeb, S. Towards a lightweight intrusion detection framework for in-vehicle networks. *J. Sens. Actuator Netw.* **2022**, *11*, 6. [CrossRef]
25. Kang, M.J.; Kang, J.W. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE* **2016**, *11*, e0155781. [CrossRef] [PubMed]
26. Taylor, A.; Leblanc, S.; Japkowicz, N. Anomaly detection in automobile control network data with long short-term memory networks. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, 17–19 October 2016; IEEE: Hoboken, NJ, USA, 2016; pp. 130–139.
27. Ahmad, U.; Song, H.; Bilal, A.; Alazab, M.; Jolfaei, A. Securing smart vehicles from relay attacks using machine learning. *J. Supercomput.* **2020**, *76*, 2665–2682. [CrossRef]
28. Anisetti, M.; Ardagna, C.; Cremonini, M.; Damiani, E.; Sessa, J.; Costa, L. Security Threat Landscape, 2020. Available online: https://sesar.di.unimi.it/download/white-paper-security-threats/ (accessed on 26 June 2024).
29. Mahmood, S. Systematic Threat Assessment and Security Testing of Automotive over-the-Air Updates. Ph.D. Thesis, Systems Security Group, Centre for Future Transport and Cities, Coventry University, Coventry, UK, 2021.
30. Studnia, I.; Nicomette, V.; Alata, E.; Deswarte, Y.; Kaâniche, M.; Laarouchi, Y. Survey on security threats and protection mechanisms in embedded automotive networks. In Proceedings of the 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W), Budapest, Hungary, 24–27 June 2013; IEEE: Hoboken, NJ, USA, 2013; pp. 1–12.
31. Liu, J.; Zhang, S.; Sun, W.; Shi, Y. In-vehicle network attacks and countermeasures: Challenges and future directions. *IEEE Netw.* **2017**, *31*, 50–58. [CrossRef]
32. Philipsen, S.G.; Andersen, B.; Singh, B. Threats and attacks to modern vehicles. In Proceedings of the 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), Bandung, Indonesia, 23–24 November 2021; IEEE: Hoboken, NJ, USA, 2021; pp. 22–27.
33. Malik, K.R.; Ahmad, M.; Khalid, S.; Ahmad, H.; Al-Turjman, F.; Jabbar, S. Image and command hybrid model for vehicle control using Internet of Vehicles. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3774. [CrossRef]
34. Kleberger, P.; Olovsson, T.; Jonsson, E. Security aspects of the in-vehicle network in the connected car. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; IEEE: Hoboken, NJ, USA, 2011; pp. 528–533.
35. Nilsson, D.K.; Larson, U.E. Simulated attacks on can buses: Vehicle virus. In Proceedings of the IASTED International Conference on Communication Systems and Networks (AsiaCSN), Langkawi, Malaysia, 2–4 April 2008; pp. 66–72.
36. Zeadally, S.; Guerrero, J.; Contreras, J. A tutorial survey on vehicle-to-vehicle communications. *Telecommun. Syst.* **2020**, *73*, 469–489. [CrossRef]
37. Bharati, S.; Podder, P.; Mondal, M.; Robel, M.; Alam, R. Threats and countermeasures of cyber security in direct and remote vehicle communication systems. *arXiv* **2020**, arXiv:2006.08723.
38. Miller, C.; Valasek, C. *Adventures in Automotive Networks and Control Units*; DEF CON: Las Vegas, NV, USA, 2013.
39. Hoppe, T.; Dittman, J. Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy. In Proceedings of the 2nd Workshop on Embedded Systems Security (WESS), Salzburg, Austria, 30 September–5 October 2007; pp. 1–6.
40. Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S.; Koscher, K.; Czeskis, A.; Roesner, F.; Kohno, T.; et al. Comprehensive experimental analyses of automotive attack surfaces. In Proceedings of the USENIX Security Symposium, San Francisco, CA, USA, 8–12 August 2011; Volume 4, p. 2021.
41. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef] [PubMed]
42. Ye, H.; Li, G.Y. Deep reinforcement learning for resource allocation in V2V communications. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; IEEE: 2018; pp. 1–6.
43. Gao, D.; Lin, H.; Li, Z.; Qian, F.; Chen, Q.A.; Qian, Z.; Liu, W.; Gong, L.; Liu, Y. A nationwide census on wifi security threats: Prevalence, riskiness, and the economics. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, New Orleans, LA, USA, 25–29 October 2021; pp. 242–255.
44. Bittl, S.; Gonzalez, A.A.; Myrtus, M.; Beckmann, H.; Sailer, S.; Eissfeller, B. Emerging attacks on VANET security based on GPS time spoofing. In Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 28–30 September 2015; IEEE: Hoboken, NJ, USA, 2015; pp. 344–352.
45. Thing, V.L.; Wu, J. Autonomous vehicle security: A taxonomy of attacks and defences. In Proceedings of the 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 15–18 December 2016; IEEE: Hoboken, NJ, USA, 2016; pp. 164–170.
46. Stubberud, S.C.; Kramer, K.A. Threat assessment for GPS navigation. In Proceedings of the 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, Alberobello, Italy, 23–25 June 2014; IEEE: Hoboken, NJ, USA, 2014; pp. 287–292.
47. Thomas, M.; Norton, J.; Jones, A.; Hopper, A.; Ward, N.; Cannon, P.; Ackroyd, N.; Cruddace, P.; Unwin, M. *Global Navigation Space Systems: Reliance and Vulnerabilities*; The Royal Academy of Engineering: London, UK, 2011.
48. NIST; Cichonski, J.; Franklin, J.; Bartock, M. *Guide to LTE Security*; Special Publication (NIST SP), National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017. [CrossRef]

49. Jo, H.J.; Choi, W.; Na, S.Y.; Woo, S.; Lee, D.H. Vulnerabilities of android OS-based telematics system. *Wirel. Pers. Commun.* **2017**, *92*, 1511–1530. [CrossRef]
50. Khan, W.Z.; Hossain, M.S.; Aalsalem, M.Y.; Saad, N.M.; Atiquzzaman, M. A cost analysis framework for claimer reporter witness based clone detection schemes in WSNs. *J. Netw. Comput. Appl.* **2016**, *63*, 68–85. [CrossRef]
51. Dong, M.; Ota, K.; Yang, L.T.; Liu, A.; Guo, M. LSCD: A low-storage clone detection protocol for cyber-physical systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2016**, *35*, 712–723. [CrossRef]
52. Alati, M.F.; Fortino, G.; Morales, J.; Cecilia, J.M.; Manzoni, P. Time series analysis for temperature forecasting using TinyML. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2022; IEEE: Hoboken, NJ, USA, 2022; pp. 691–694.
53. Joachims, T. *Making Large-Scale SVM Learning Practical*; Technical Report; Universität Dortmund: Dortmund, Germany, 1998.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.