Securing the Controller Area Network: A Multi-Layer Defense Framework Against Automotive Cyberattacks

Aidan Shaheen
Computer Engineering
ashahee1@terpmail.umd.edu

Aleena Ajayakumarbini
Computer Engineering
aleenaab@terpmail.umd.edu

Brian Wu
Computer Engineering
bwu32@terpmail.umd.edu

Calvin Berlin
Electrical Engineering
cberlin@terpmail.umd.edu

Kira Thompson
Computer Engineering
kthomp24@terpmail.umd.edu

Abstract

Modern vehicles rely on the Controller Area Network (CAN) bus for critical communication between Electronic Control Units, yet this protocol lacks fundamental security features. We propose and evaluate a three-layer security framework incorporating encryption, access control, and integrity verification to mitigate injection, replay, and flooding attacks. Our simulation-based methodology uses synthetic attack scenarios based on documented real-world patterns to measure defense effectiveness while considering automotive constraints on timing, memory, and power consumption.

CCS Concepts

• **Security and privacy → Systems security**; • **Computer systems organization → Embedded systems**.

Keywords

CAN bus, automotive security, intrusion detection, encryption, HMAC, embedded systems, real-time systems

ACM Reference Format:
Aidan Shaheen, Aleena Ajayakumarbini, Brian Wu, Calvin Berlin, and Kira Thompson. 2025. Securing the Controller Area Network: A Multi-Layer Defense Framework Against Automotive Cyberattacks. In *Proceedings of ENEE457 Computer Systems Security (CSS '25).* ACM, New York, NY, USA, 7 pages.

1 Introduction

The Controller Area Network (CAN) connects different Electronic Control Units (ECU) like brakes, doors, and engine modules to communicate seamlessly. When Bosch created it in the 1980s, the focus was on reliability and timing, not security. Because of this, the network is inherently vulnerable to malicious access. A compromised connected device or ECU is one way issues can arise, since any node can broadcast a message in the CAN system. This would allow attackers to send signals to control systems they were never meant to have access to. Flooding attacks can also disrupt the CAN bus by overwhelming it.

The simulation that we implemented provides a visual representation of how signals pass through the CAN bus and affect latency, arbitration, and ECU behavior. This way, we don't need an actual vehicle to run tests on and interfere with its systems. We want to focus on the outcome effects so that we can apply Computer Systems Security (CSS) principles. By observing these effects, we can identify which has the greatest effect, causing the most disruption. This would also point out vulnerability in the CAN's priority-based arbitration, revealing what points we'll target to strengthen.

Due to a vehicle's limits on timing, memory, and power, we have to factor these into our simulation. Our goal is to simulate the in-vehicle CAN communications and show how attackers can inject, replay, or flood messages to manipulate ECUs and then apply computer security principles to strengthen those vulnerabilities. The resulting findings can assist in upgrading the standard for future CANs.

2 Related Work

The Controller Area Network (CAN) bus is a communication system that was developed by Robert Bosch in the 1980s. CAN Bus allows different electronic control units (ECUs) within a vehicle to communicate with each other. Before it was developed, vehicles often relied on complex wiring systems that were very expensive, heavy, and easily prone to errors. When Bosch released the first CAN protocol, it quickly became popular in the automotive industry because of its robustness and reliability.

The CAN Bus is essentially a decentralized communication protocol. This means that any ECU can send data at any time without needing a central controller. In a CAN network, all ECUs, sensors, and actuators (CAN nodes) are connected by twisted pair wiring or optical fiber cables. Each node has its own microcontroller that processes incoming messages and sends outgoing messages. When a node transmits data on the shared bus, the message is broadcasted to all other nodes that are connected to the CAN network.

2.1 CAN Bus Types

There are three main types of CAN buses:

- **Low-Speed or fault-tolerant (ISO 11898-3) CAN:** This CAN operates at speeds up to 125 kbps and is used in non-critical systems like door locks, windows, and body control modules.

CSS '25, University of Maryland, College Park

- **High-Speed CAN (ISO 11898-2):** This CAN operates at speeds up to 1 Mbps and is used in time-sensitive and safety-related systems like engine control, transmission, and braking.
- **CAN FD (Flexible Data Rate):** This CAN operates at data speeds up to 5 Mbps and message sizes up to 64 bytes. This version is used in modern vehicles that need to handle large amounts of data from sensors, ECUs, and infotainment systems.

The CAN Bus has several advantages such as being simple and inexpensive. Since CAN Bus allows ECUs to communicate through a shared network instead of separate wiring for each connection, it reduces weight, complexity, and costs. Also CAN Bus makes diagnostics and updates easier because technicians or systems can access all connected ECUs from one point, allowing for quick troubleshooting, data logging, and even over-the-air updates.

2.2 CAN Bus Limitations

However, as vehicle technology continues to advance rapidly, the limitations of the CAN Bus have increasingly become more evident. Originally designed for simple, low-bandwidth tasks such as window operations and engine control, the CAN Bus now struggles to meet the complex and data intensive demands of modern vehicles. One major limitation of CAN is its bandwidth constraint. While CAN FD offers faster data transfer speeds up to 5 Mbps, this is still not enough to efficiently support the huge data exchange required by advanced driver assistance systems (ADAS) and autonomous driving features, which rely heavily on high-resolution sensors, cameras, radar, LiDAR, and V2X communication.

Furthermore, when the CAN protocol was first developed, it was a closed in-vehicle network and did not include any built-in security functions such as encryption, authentication or access control. However, modern vehicles today often have the ability to connect to external networks such as smartphones or telematics units. These connections make it possible to remotely access the in-vehicle CAN and expose it to cyberattacks. Due to the limited bandwidth and computation capacity of CAN Bus, implementing cryptographic methods such as Public Key Infrastructure (PKI) and Message Authentication Codes (MACs) are difficult in in-vehicle CAN networks.

As a result, this leaves CAN based systems highly vulnerable to cyberattacks such as spoofing, message injection and denial-of-service (DoS) attacks. For example, in a message flooding attack an attacker can continuously broadcast a large number of CAN messages to overwhelm the network. This type of DoS attack is very dangerous because it does not require the attacker to identify or manipulate critical messages. Instead the attacker just needs to keep sending a huge number of CAN messages until the system stops working properly. Once the network is compromised, attackers can control or disable safety functions such as the brakes, steering, or engine.

2.3 Intrusion Detection and Prevention

To address these vulnerabilities, researchers have developed Intrusion Detection Systems (IDS) as a defense mechanism for in-vehicle networks. IDS can detect abnormal CAN traffic patterns by analyzing physical attributes such as frequency, voltage and timing patterns and by analyzing digital attributes such as CAN identifiers, and message frequencies. Various IDS models have been proposed, including message frequency-based, clock skew-based, electrical signal-based, and entropy-based approaches.

Alongside IDS, Intrusion Prevention Systems (IPS) have been introduced to restrict access from compromised ECUs by disabling their communication with the CAN bus. Together, IDS and IPS can detect and prevent many types of attacks, however, some challenges still remain. For example, if an attacker performs a flooding attack using CAN identifiers that belong to actual ECUs, current IDSs still struggle to distinguish between normal and malicious messages. This can result in false positives, where normal messages are misclassified as attacks, and in turn, the IPS may mistakenly block a functioning ECU. Such false positives can be exploited by attackers to trigger defective systems against the vehicle itself. This further demonstrates the ongoing need for a more advanced and secure CAN Bus network.

3 Proposed Method

The method we are proposing involves adding a security layer that would help alleviate security issues of the CAN bus. The very first step is to come up with a simulated model of a CAN-driven vehicle that we can use for our project and then proceed to create a security solution which will integrate encryption, access control, and integrity verification measures.

3.1 Encryption Layer

The initial point in our strategy is to attach an encryption mechanism whose purpose will be to secure data that is sent over the CAN bus to avoid interception and reading of the data by unauthorized parties. Since the CAN protocol was not originally designed with security in mind, by implementing encryption we make sure that even if someone manages to access the network, the data they grab will still be incomprehensible without the correct encryption keys. Besides that, if the attacker doesn't possess the same encryption keys, then it is not easy for them to send false messages. The encryption system should be implemented in such a way that it is lightweight and does not affect the speed of communication between different parts of the vehicle. This is vital not only for cars but for all vehicles where safety depends on speed and reliability of the system.

3.2 Access Control Layer

The second part is an access control system. Only allowed devices will be able to send or receive specific types of messages. Every node on the network is to check its identity before communication. Unauthorized devices that have been hacked or compromised can be very dangerous in terms of injecting false info into the network, but by enforcing these permissions and using role-based access control (RBAC), it will be difficult or impossible for them to do this. System integrity is the core concept that the system uses to support and guarantee that every component shall carry out its function only and do not interfere.

3.3 Integrity Verification Layer

As for the third layer, it is to do with integrity verification. The purpose of this level is to guarantee that the communication messages between the different units have not been interfered with in any manner or altered. We are also looking to include message authentication codes or digital signatures, which will allow the recipient device to verify that the data came from the legitimate sender. Apart from that, it can be realized that there is a sudden and unexpected change in the behavior of messages thus making it possible to detect unusual activity or unexpected message patterns or timing that could possibly point toward an attack.

This system will act as three-layer protection, establishing a more secure base upon which the CAN bus system can be built. The adoption of these security measures of encryption, access control, and integrity verification will result in much lower chances of successful cyberattacks on the vehicle communication network.

4 Implementation

To begin our implementation, we started by focusing on specific fundamental security mechanisms that the CAN Bus lacks, such as encryption, authentication, and access control. Then we considered different cyberattacks that could occur, like spoofing, message injection, and denial-of-service (DoS) attacks. This was followed by applying a variety of solution systems to the simulation, including the Intrusion Detection Systems (IDS) and the Intrusion Prevention Systems (IPS).

4.1 Virtual CAN Network

To implement these systems, we simulated a virtual CAN network coded in Python. Python was used as it has the best integration with cybersecurity libraries. Our main focus when creating the simulation was to ensure that we included the most important features of CAN particularly the features attacks use to carry out their intended purpose.

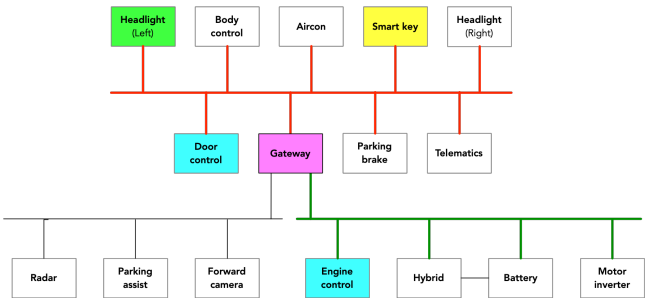

Figure 1: A chart representation of how the CAN Network's ECU systems communicate with one another.

4.1.1 CAN Arbitration. In CAN, every message has an id and this id is used for arbitration where lower ids have higher priority. A CAN bus essentially consists of just two wires, a high and low, where the output is just the two wires logically ANDed. This leads to 0's dominating so for example if the transmission has id 0x0100 and anti-lock brakes has id 0x0001, both will send 0 then the transmission will send 1 where the brakes will send 0. The transmission will recognize this and stop transmitting giving the ABS systems priority. Now this system is the backbone of a lot of attacks as the attacker can ensure their message has highest priority preventing any valid messages from being seen. The simulation of the CAN arbitration was implemented through a minimum heap ensuring that the lowest id was always the next message served.

4.1.2 Message Acknowledgment. Another important feature of CAN is the acknowledgement tag of the CAN message where at the end of each CAN message a bit sent to 1 is put on the bus. Now if any other node (each ECU is a node) receives this message it will acknowledge receipt of the message by putting a 0 on the bus. The sender will wait for this acknowledgment and if it sees the bus is now 0 it knows the message has been received. However if the bus still has the 1 on it then the sender will retry until the message is acknowledged or the max number of retries has been met.

In our simulation this behavior is handled by the CAN controller and an `ack_success` variable which is just a boolean. If `ack_success = true` the message is popped off the minimum heap, if `ack_success = false` the controller will resend the message until success or max_retries is met. One key difference from an actual CAN network is that in our simulation each message is intended for a certain node and the ack is only set to true if the actual node intended to receive the message receives it. In an actual CAN network any node can send acknowledgement, so just because the sender sees the message was received does not mean the intended node received it. This is a drawback to the CAN network, but not something we exploited in our attacks so just having a general acknowledgement feature was enough.

4.2 Network Architecture

The general layout of a car's CAN includes multiple sub CAN networks that can communicate through a gateway controller. Systems a part of the powertrain like the engine and transmission, might be on its own bus running at a fixed rate then systems like the dashboard, and infotainment could be on another bus running at a different rate. Often the gateway just serves to translate and filter messages sending them to the correct bus, but has no security. This means even if something like the car's headlight is on a bus completely separate from the engine, the headlight ECU can still send messages to the engine ECU.

To make the simulation easier we removed the complexity of multiple buses and a gateway focusing on a network of 4 nodes (ECUs). For the purpose of the attacks going through the headlight or going directly to the engine makes little difference in what we're testing. The simulation does allow many nodes to be created but for simplicity we decided on creating the engine ECU, brake ECU, transmission ECU, and body control ECU. The function of each ECU is relatively basic for this simulation as the engine ECU just sends RPMs and temp, the brake ECU sends pressure and status, the transmission sends the gear it's in, and the body ECU controls doors and lights. Also each node is set to listen for a particular node so in our case the engine listens to the brake ECU, and the transmission listens to the engine RPM to decide when to shift.

4.3 Concurrent Execution

With the basic CAN network structure, we then wanted to ensure that like a car's CAN network each ECU is running concurrently. This required the use of threads where each ECU runs on its own thread and the CAN bus which serves to send the messages runs on its own thread. To ensure that threads didn't access certain data at the same time locking was used mainly to prevent more than one thread trying to update the min heap.

4.4 Latency Measurement

With a fully functioning simulation we just needed a way to test the latency of messages as that would be key to evaluating how attacks and encryption affects the latency. To implement this when a message was sent the time would be recorded and once received it would compare the two times to find the total time and then store it in a CSV file which could be analyzed at a later time. Also since we're using a simulation we purposely added some sleep time to mimic the time it would actually take for a message to be sent on an actual CAN bus. The complete implementation can be viewed on GitHub at https://github.com/bwu32/canbus which includes the front end work allowing easy control over the simulation.

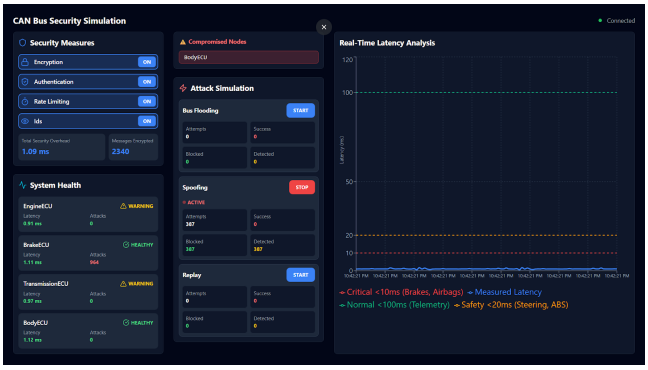

Figure 2: The visual aspect of our virtual CAN network proof-of-concept demonstrating 4 security measures being deployed against a spoofing attack.

4.5 Attack Implementation

Possible attacks we observed after implementing the CAN were Bus Flooding, which is an attack that sends many high-priority messages. We also observed how spoofing could occur, which is an attack that sends modified or malicious messages. And finally, we saw how replay could be injected, which is an attack that copies legitimate messages and sends them later to attack the system.

4.6 Security Measures

To protect against these attacks, 4 security measures were implemented:

(1) **AES-128 Encryption** using CBC mode with random IV per message, in order to protect the confidentiality of the system.
(2) **HMAC-SHA256 Authentication**, which is a Message Authentication Code using a shared secret to protect the integrity of the system.

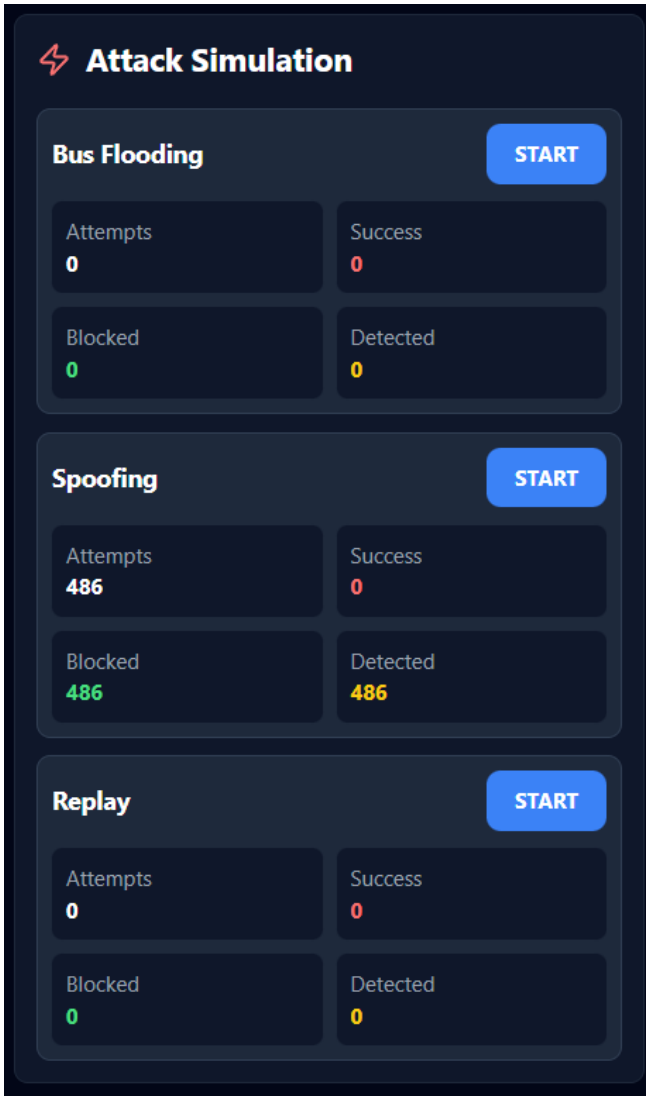

Figure 3: The 3 attack methods of our simulation that can be toggled on and off, with the measured statistics being total attempts, successes, attacks blocked, and attacks detected.

(3) **Rate Limiting** using a sliding window counter to protect against attacks sending many high-priority messages.
(4) **Intrusion Detection System** using a statistical baseline of normal message frequencies in order to protect the system from being flooded by new or replay messages from an attacker.

5 Evaluation

5.1 Original Evaluation Plan

Our original plan was to use a dual-platform approach which combines software simulation and physical testing of hardware. By leveraging SocketCAN, an open source linux networking subsystem for virtual CAN networking, in conjunction with python libraries

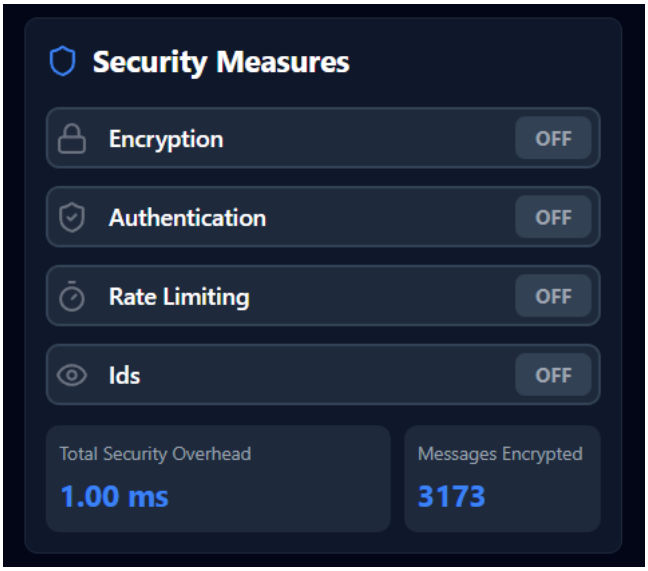

Figure 4: The 4 encryption methods of our simulation that can be toggled on and off, with the total latency and total messages encrypted being measured.

(python-can, cantools, scapy-can), we planned to facilitate attack scripting, defensive mechanism implementation, protocol analysis, and controlled message injection. If we decided to take it a step further, we could use Wireshark with CAN dissector to provide a graphical interface for traffic analysis and anomaly detection. Our hardware would consist of Raspberry Pi or Arduino boards equipped with MCP2551 CAN transceivers. This would allow us to create a small-scale ECU network to replicate automotive electrical and timing characteristics.

The primary dataset planned to use was the NDSS 2024 CAN Bus Attack Dataset [5]. It features standard CAN messages captured from real vehicles in operation as well as labeled attack events, which include injection attacks, replay attacks, flooding scenarios, and fuzzing attempts. We wanted to use a real-world dataset because it would capture authentic timing characteristics, message frequency patterns, and network load conditions that laboratory simulations may fail to properly replicate. Extra supplementary datasets from CSS Electronics and other specific traffic capture research could be used if deemed necessary.

5.2 Evaluation Metrics

The evaluation plan would be quantified on six primary metrics:

- **Detection and Prevention Rate** measures the percentage of attack messages successfully detected or blocked, calculated separately for each attack type to identify which threats are most effectively addressed.
- **False Positive Rate** quantifies legitimate messages erroneously flagged or blocked; automotive systems require rates below 0.1% to avoid disabling critical functions.
- **Latency Impact** measures additional delay introduced by security controls, recognizing that safety-critical messages tolerate only microsecond delays while convenience features accommodate longer processing times.
- **Network Throughput** quantifies changes in frames processed per second, ensuring security mechanisms do not create bottlenecks given CAN's fixed bandwidth (500 kbit/s or 1 Mbit/s for CAN-FD).
- **Resource Usage** profiles CPU cycles and memory consumption.
- **Resilience** evaluates recovery time and network stability during and after attacks, measuring whether systems maintain operation without vehicle restart.

We planned to first load the dataset to characterize the traffic patterns and corresponding timing relationships, then run various attack scenarios against the unprotected baseline, documenting any compromised functions and performance degradation. Then we would secure the system and run the scenarios again, and analyze the data as received.

5.3 Modified Implementation

Based on our original plan above, we modified the evaluation for our updated implementation. We did not use Raspberry Pi, and rather ran the simulation through a backend built in Python and a frontend built in React and hosted with Vite. Our evaluation metrics were largely the same, but we additionally broke them down into more specifics based on the attack used.

6 Key Results

Using the evaluation framework described in the previous section, a series of controlled experiments were conducted to assess both the security effectiveness and real-time performance of the proposed CAN bus security architecture. The experiments were designed to determine whether the added protections are practical for deployment in real vehicle environments while maintaining strict timing and resource constraints.

6.1 Methodology

To evaluate the Detection and Prevention Rate, the CAN bus simulation was subjected to flooding, spoofing, and replay attacks. The simulation recorded the number of malicious messages that were successfully blocked and the number of malicious messages that were able to pass through the system. The False Positive Rate was measured by checking whether normal ECU messages were incorrectly blocked by the security mechanisms. To meet automotive reliability requirements, the false positive rate was kept below 0.1%, with rate limiting configured at 20 messages per second. This preserved normal ECU behavior.

The Latency Impact metric, which is very important for safety functions in a vehicle, was measured as end-to-end message delay. Latency was evaluated against strict automotive thresholds of under 10 ms for braking messages, under 20 ms for steering messages, and under 100 ms for telemetry data. Network Throughput was assessed by monitoring message processing rates on a simulated 500 kbps CAN bus, which can theoretically support approximately 3,900 frames per second.

The Resource Usage metric was evaluated by simulating realistic processing overhead introduced by each security mechanism,

including 0.5 ms for encryption, 0.3 ms for HMAC verification, and 0.1 ms each for rate limiting and intrusion detection processing. Finally, Resilience was assessed by observing ECU stability and measuring how quickly the system returned to normal operation after attack traffic stopped.

6.2 Security Effectiveness

The experimental results showed that the proposed security architecture provides strong protection while maintaining real-time performance. To test this, the CAN bus was evaluated under unsecured, partially secured, and fully secured configurations.

When no security mechanisms were applied, all attack types succeeded, allowing malicious messages to freely travel across the CAN bus. While latency remained low in this setup, the lack of protection clearly showed that a standard CAN network is vulnerable to message injection, spoofing, and denial-of-service attacks.

When HMAC-based authentication was enabled as the only security mechanism, spoofing attacks were fully prevented, resulting in a 0% success rate. This shows that message authentication is effective at stopping unauthorized ECUs from impersonating legitimate nodes. Furthermore, this protection caused very little performance impact, with total added latency staying below 1 ms. However, flooding and replay attacks were still partially effective in this configuration, showing that additional security layers are required.

Adding rate limiting greatly improved system reliability by blocking flooding and replay attacks. By limiting how many messages could be accepted, the CAN bus was protected from overload while continuing to operate normally. Even with this added control, total processing overhead remained close to 1 ms, demonstrating that rate limiting can be used without affecting real-time performance.

6.3 Comprehensive Protection

When all security mechanisms were enabled simultaneously, including encryption, authentication, rate limiting, and intrusion detection, all attack attempts were successfully prevented. Spoofing attacks were blocked through authentication, flooding and replay attacks were mitigated through rate limiting, and all malicious activity was consistently identified by the intrusion detection system. This layered defense approach ensured that even if one mechanism were bypassed, additional protective mechanisms continued to operate.

Despite the added security layers, total end-to-end latency remained approximately 2 ms, which is well below the strict real-time requirements for safety-critical vehicle functions. Normal ECU communication continued without disruption, and no false positives were observed during standard operation. Network throughput remained within acceptable limits for a 500 kbps CAN bus, and resource usage did not exceed the processing limits of embedded automotive ECUs. After the attacks were stopped, the system returned immediately to stable operation without requiring resets or manual intervention, demonstrating strong resilience and operational stability.

6.4 Optimal Configuration Analysis

After analyzing all of the possible security configurations, all four measures with 2.0 ms of overhead are the best overall solution due to its ability to provide full protection with encrypted transmissions, message authentication, DoS prevention, etc. However, if we compare the overall speed, efficiency, and cost effectiveness to overall security, we found that HMAC + Rate Limiting is the most practical solution, since it offers protection from three attack types with only 0.7 ms of overhead. For safety-critical systems such as brakes and airbags, which cannot exceed a 10 ms reaction time, this is the best choice that will optimize overall security and have the lowest impact on performance and latency.

Overall, these results show that the proposed multi-layer CAN bus security architecture effectively protects against common automotive cyberattacks while maintaining the timing, reliability, and resource limits required in real vehicles. The findings confirm that strong security mechanisms can be added to CAN-based systems without reducing performance, making the proposed approach both effective and practical for real-world automotive use.

7 Conclusion and Future Work

Based on our results, our simulation and security measures have shown that the CAN bus system is very susceptible to a variety of attacks. Implementing simple security measures in the system can help prevent these attacks. We also found that lightweight security measures increase latency; this would need to be heavily considered by the automotive industry to find the right balance between security and low latency. We would not want to add any security measures that might prevent important legitimate messages from being received in a timely manner, such as braking. Overall, this is a topic that is important, especially in the upcoming years of advancing technology, but requires a great deal of careful and thoughtful consideration.

7.1 Future Directions

There is a great deal of future work to be done in this area. As the automotive industry continues to move towards using computer systems in vehicles, it will be increasingly important to secure these systems, as more parts of a car, such as steering/braking systems by wire are becoming commonplace. Additionally, services like Waymo have self-driving vehicles that come with many more control surfaces and connectivity to computer networks. Waymo riders would want to be sure that attackers could not remotely hijack their vehicle and put them in a dangerous situation. The automotive industry needs to put careful and thoughtful work into further developing CAN security.

Alternatively, our project could be used as evidence to show that the limitations of CAN convey a need for a replacement some time down the road. This would be a significant endeavor, as CAN has been in use for decades, and there would be many issues, technical and regulatory, to work out. The automotive industry would need to research the needs of consumers, specifications of developing computer and automotive systems, as well as the potential risks, attacks, vulnerabilities, and other concerns when developing a new system. Another major challenge to consider would be standardizing a new system, as the automotive industry exists worldwide,

involving engineering and manufacturing from different countries and regulations around the world.

7.2 Network Architecture Improvements

Another option that wouldn't be as difficult of an endeavor is to redesign the structure of a car's CAN network and build on what's already there. With a car's CAN already being split up into sub CAN networks, it's possible to separate these more and through the CAN's central gateway connecting all these networks implement the majority of the security there. This allows important systems on the same bus to communicate with little latency and only introduces high latency when going through the gateway. Attacks through the means of accessing a car's headlight would be less effective as instead of being able to access the engine or locks, they could only really just take down one bus on the car. This kind of approach could make cars CAN much harder to compromise without adding too much complexity.

Overall, this is a topic that will require careful consideration and technological research and development going forward. CAN bus is currently a very important aspect of modern vehicles, and this project serves as a proof-of-concept to highlight that security measures for these systems are becoming increasingly important as new attacks emerge.

Acknowledgments

We would like to thank Professor Li for his instruction throughout this course.

References

[1] S. Alhelou. CAN bus limitations in modern connected vehicles. Automotive Connectivity Provider | Horizon Connect, May 2025. https://horizonconnect.de/can-bus-limitations-connected-vehicles/

[2] CSS Electronics. Can bus explained - a simple intro [2025]. January 2025. https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial

[3] J. Huang. Can bus uncovered: Basics and applications in vehicles. www.emqx.com, March 2025. https://www.emqx.com/en/blog/can-bus-how-it-works-pros-and-cons

[4] S. B. Park, H. J. Jo, and D. H. Lee. Flooding attack mitigator for in-vehicle CAN using fault confinement in CAN protocol. *Computers & Security*, 126:103091, 2023. https://doi.org/10.1016/j.cose.2023.103091

[5] NDSS 2024 CAN Bus Attack Dataset. A comprehensive CAN bus attack dataset from moving vehicles. 2024.