Contents

[Introduction 3](#_Toc181648658)

[Functional Summary 4](#_Toc181648659)

[1. **Development**: 4](#_Toc181648660)

[Assumptions 5](#_Toc181648661)

[Functional Requirements 6](#_Toc181648662)

[System Context 6](#_Toc181648663)

[Requirements List 6](#_Toc181648664)

[**I.** **Data Management:** 6](#_Toc181648665)

[**II.** **Tamper Detection:** 6](#_Toc181648666)

[**III.** **Data Integrity:** 7](#_Toc181648667)

[**IV.** **Security:** 7](#_Toc181648668)

[Data Requirements 8](#_Toc181648669)

[**I.** **External Data Sources** 8](#_Toc181648670)

[**II.** **Local Data** 8](#_Toc181648671)

[Technical Investigation 9](#_Toc181648672)

[CAN Bus Security and Odometer Data Authentication 9](#_Toc181648673)

[Vulnerabilities in the CAN Bus Protocol 9](#_Toc181648674)

[SecOC 10](#_Toc181648675)

[Encryption on the CAN bus 12](#_Toc181648676)

[Telematics Systems and Vehicle-to-Cloud Communication 14](#_Toc181648677)

[Communication & Protocols 14](#_Toc181648678)

[Large-scale data management 17](#_Toc181648679)

[Interface Specifications 19](#_Toc181648680)

[Onboard 19](#_Toc181648681)

[Offboard 19](#_Toc181648682)

[Non-functional Requirements 22](#_Toc181648683)

[1. Performance Requirements 22](#_Toc181648684)

[1.1. Response Time: 22](#_Toc181648685)

[1.2. Data Processing Speed: 22](#_Toc181648686)

[1.3. Network Bandwidth: 22](#_Toc181648687)

[1.4. Latency for Critical Alerts 22](#_Toc181648688)

[2. Security Requirements 22](#_Toc181648689)

[2.1. Data Encryption: 22](#_Toc181648690)

[2.2. Access Control: 22](#_Toc181648691)

[2.3. Message Integrity: 22](#_Toc181648692)

[2.4. Audit Logging: 22](#_Toc181648693)

[3. Scalability Requirements 23](#_Toc181648694)

[3.1. Horizontal Scaling: 23](#_Toc181648695)

[3.2. Data Storage 23](#_Toc181648696)

[Quality Assurance Provisions 24](#_Toc181648697)

[Software Test procedures 24](#_Toc181648698)

[Software Validation Procedures 24](#_Toc181648699)

[Risk and Contingencies 25](#_Toc181648700)

[Methodology Overview 25](#_Toc181648701)

[Project Plan 27](#_Toc181648702)

[References 29](#_Toc181648703)

# Introduction

The goal of my project is to investigate mileage tampering and create a solution on how it can be prevented or minimised.   
  
The recent increase of complexity within vehicle electronics and systems has led to new opportunities for vehicle exploitation. One of the most popular forms of automotive fraud in the automotive market is odometer tampering also known as mileage fraud. Odometer tampering is the illegal practice of altering the mileage displayed on a vehicle’s odometer which results in the vehicle displaying a lower mileage than the vehicle has actually travelled.   
  
Mileage fraud effects many in the automotive space such as buyers in the second-hand market, vehicle manufactures and vehicle leasing companies. Mileage fraud can be used against vehicle manufacturers who have a warranty in place for their vehicles which includes a maximum mileage limit before the vehicle is out of warranty. This is a very relevant topic as more manufactures begin to sell electric cars that come with a warranty cover on the battery itself, such as Hyundai Ireland who grant warranty on their batteries for 8 years or up to 160,000 kms. Similarly with leased vehicles they will come with an annual mileage limit which typically falls between 8,000 and 15,000 kilometres and mileage fraud can be used to exceed this limit to avoid addiction charges.  
  
To combat the issues of mileage fraud there has been several regulations been put in place at both national and international levels. In the European Union one of the key directive addressing the issue is [EU Directive 2014/45/EU](https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32014L0045#:~:text=(25),by%20the%20Commission.). This Directive mandates that Member States are required to conduct periodic vehicle inspections along with recording the odometer readings at the time of the inspection. This solution is not a 100% fix to the problem and includes some gaps

* **Cross Border Sales:** One major problem with this solution is that mileage records are typically only accessible with a single Member State, this meaning once a vehicle is exported to a different state the records for that vehicle are often not transferred.
* **Technological Limitations:** With the current regulation in place it heavily relies on manual inspections which is not a sufficient solution to combat mileage blockers as they are and active device fitted to the vehicle which blocks mileage or blocks a percentage of mileage from being recorded.

The aim for this project is to address the problems with mileage tampering and to develop a solution to include a large amount of vehicle manufactures. I will attempt this by using Vector standards along with using Vector tools to develop my solution along with using Vector CANoe to demonstrate the solution to prevent mileage tampering.

# Functional Summary

What I plan to do to accomplish this project idea includes the following but that may be subject to change:

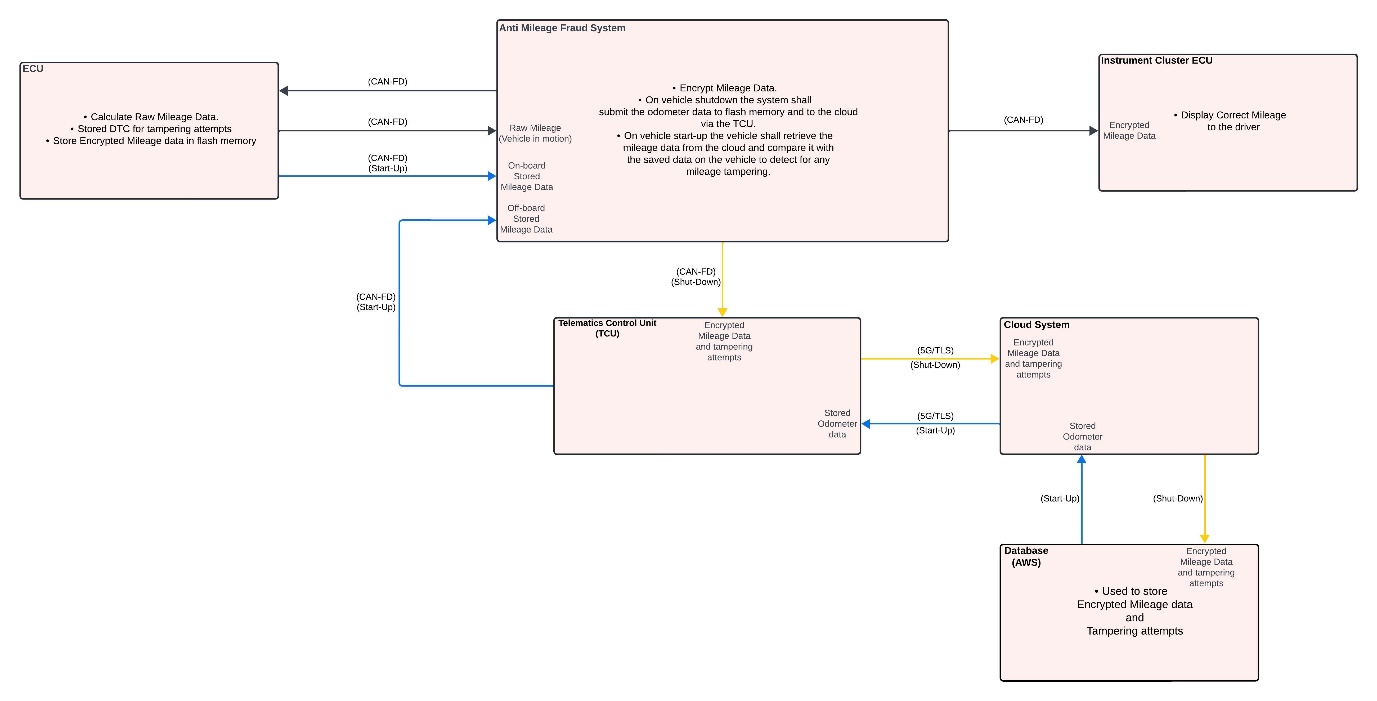
1. **Development**:
   1. **Mileage Tracking:** The system will continuously monitor and display accurate odometer readings using data collected from the vehicle’s ECUs.
   2. **Tamper Detection:** The system will detect any attempt to modify or block the odometer data. It will log all tampering attempts and store these logs securely.
   3. **Secure Data Storage:** Odometer data will be stored both onboard and offboard, ensuring that the mileage data is tamper proof and can be verified at any time.
   4. **Data Integrity Verification:** The system will provide mechanisms to verify the authenticity of odometer data. In case of a discrepancy, the offboard stored data will serve as a reference to verify legitimate mileage.
   5. **Reporting of Tampering Attempts:** Any tampering or inconsistency in mileage data will trigger alerts and generate logs that can be accessed for reporting.

# Assumptions

* **Vehicle Compatibility:** This project assumes that the vehicle models targeted for testing have standardised communication protocols such as OBD-II and CAN bus, which will allow for easy access to odometer data.
* **Regulatory Compliance:** It is assumed that the system will be designed within the bounds of current automotive regulations regarding data logging and tamper protection.
* **Network Connectivity:** The offboard storage system assumes stable internet connectivity for transmitting odometer data securely.
* **Tamper Detection Mechanisms:** It is assumed that mileage tampering can be detected based on either change in the communication patterns of the vehicle’s ECUs or the failure of the system to log mileage data at regular intervals.
* **Onboard Storage Capacity:** It is assumed that the onboard storage capacity in the vehicle’s hardware is sufficient to store encrypted odometer data locally.

# Functional Requirements

## System Context



## Requirements List

1. **Data Management:** 
   1. The system shall read odometer data from the vehicle's ECU periodically.
   2. The system shall store odometer data onboard in encrypted form to prevent unauthorized modifications.
   3. The system shall transmit odometer data to offboard storage after each vehicle shutdown or at regular intervals.
2. **Tamper Detection:** 
   1. The system shall detect any attempt to manipulate odometer readings or disrupt the communication flow between the ECU and the odometer.
   2. The system shall log all tampering attempts in a secure and non-tampering format.
   3. The system shall detect any tampering attempts and log such attempts via onboard diagnostics and offboard storage.
3. **Data Integrity:**
   1. The system shall use encryption and cryptographic signatures to verify the integrity of odometer data stored onboard and offboard.
   2. The system shall compare onboard odometer data with offboard stored data to verify its authenticity upon each vehicle startup.
4. **Security:**
   1. The system shall use industry standard encryption algorithms for storing data.
   2. The system shall employ user authentication to access mileage data logs e.g. data is only accessible to main dealers.

## Data Requirements

1. **External Data Sources**

This project requires data to be stored offboard via a database. Information such as vehicle model, make, year and UUID are all requirements to be stored externally.   
  
Mileage data will also be stored externally and will require odometer data, time and the vehicle UUID. Mileage will also need to be logged with a date and time along with the mileage and vehicle UUID. This is crucial for tracking mileage history.  
  
Alerts will also be stored externally when mileage tampering is detected. Which will include a status (active or resolved) to allow the alert to be tracked.

1. **Local Data**

Vehicle data such as vehicle UUID, model, year and current mileage will all need to be stored locally.  
  
Diagnostics data will be used to store onboard tampering attempts in the form of a DTC.

# Technical Investigation

## CAN Bus Security and Odometer Data Authentication

### Vulnerabilities in the CAN Bus Protocol

The CAN (Controller Area Network) bus is the primary communication protocol used in modern vehicles to allow communication between various ECUs (Electronic Control Units). CAN is lightweight, efficient and cheap the protocol has many vulnerabilities, particularly when it comes to security. CAN was designed in the 1980s with reliability and real time communication in mind so now as modern vehicle technology grows CAN lacks the built in support for cybersecurity features.

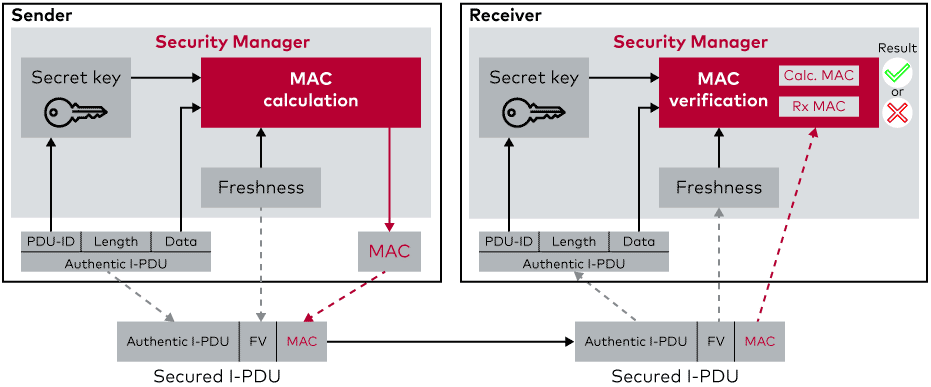
In this investigation I will focus on the key vulnerabilities of the CAN protocol that can be exploited for mileage fraud.

* **Lack of Authentication:** The CAN protocol does not authenticate messages that are being send around the CAN bus, meaning any ECU on the bus can send and receive messages without verifying their where they came from. This is how false data can be injected such as false mileage readings.
* **Message broadcast on CAN:** Any ECU that is connected to the CAN bus will receive every message broadcasted over the network. This opens the opportunity for tapping into the CAN bus and would allow for eavesdropping, message spoofing and injection of false data on the CAN bus. For example, an attacker could broadcast manipulated mileage data that is accepted by the odometer without the system noticing.
* **Message Tampering:** Since CAN messages lack any form of integrity checks, it is possible to modify CAN messages while they are in transit. An attacker could intercept a message containing odometer data and modify it to reflect a lower mileage before it reaches the destination ECU.
* **Replay Attack:** CAN is vulnerable to replay attacks, this is when a previously recorded message is rebroadcasted onto the network to manipulate something like the odometer reading.

(Understanding CAN Bus Vulnerabilities and How Blockchain Can Amplify Security, 19-09-24 )

### SecOC

SecOC (Secure Onboard Communication) was first introduced by AUTOSAR in 2015 to provide secure onboard communication for automotive systems that use CAN and CAN-FD networks. Over the years the AUTOSAR (Automotive Open System Architecture) standard has continued to refine SecOC to improve its functionality and integration into modern vehicles along with aligning with continuously evolving automotive cybersecurity standards such as ISO/SAE 21434.

SecOC is a security protocol designed to ensure the integrity and authenticity of messages exchanged between ECUs on the CAN bus network. SecOC doesn’t include encryption but it is highly effective at protecting data from tampering and replay attacks, both of which are critical concerns when it comes to securing odometer data.  
  
  
Figure 1 (Secure Onboard Communication, no date)

SecOC uses MACs (Message Authentication Codes) to detect any modification of messages. Each CAN message is appended with a MAC which is calculated using a secret key shared between ECUs. These keys must be distributed among each ECU that intends to receive the data from the sender. The necessary input values such as secret key and freshness values are stored in the Security Manager (Secure Onboard Communication, no date).

SecOC verifies the authenticity of the data by ensuring that it originates from a legitimate ECU, this prevents any attempt at injecting incorrect mileage values into the system.

Another security method implemented into SecOC is the use of freshness values. This helps stop the use of a replay attack. A replay attack occurs when a valid data transmission is fraudulently repeated to mislead the system, for example in the case of this project it would be used to send an older and lower mileage reading.  
*This is done by the SecOC module on the sender side creating a secured I-PDU by adding Authentication Information to the outgoing Authentic I-PDU. When using the Freshness counter, the Freshness counter should be incremented by the freshness manager prior to providing the Authentication Information to the receiver side.*

*On the receiver side, the SecOC module checks the freshness and authenticity of the authentic I-PDU by verifying the Authentication information that has been appended by the sending side SecOC Module. To verify the authenticity and freshness of an Authentic I-PDU, the secured I-PDU provided to the receiving side SecOC should be the same secured I-PDU provided by the sending side SecOC and the receiving side SecOC should have knowledge of the Freshness Value used by the sending side SecOC during creation of the Authenticator.* (Specification of Secure Onboard Communication Protocol, 30-11-2020 ).

SecOC is a lightweight solution to providing data integrity across ECUs where bandwidth and processing power is limited. (Specification of Secure Onboard Communication Protocol, 30-11-2020 ).

### Encryption on the CAN bus

The CAN bus network was designed without considering the need for cybersecurity, it also lacks native encryption leaving it vulnerable to data interception such as tampering and replay attacks. I will explore the feasibility and challenges of introducing encryption into CAN communications with a focus on protecting data like mileage readings.

1. **CAN Bus Architecture and Constraints:**The CAN protocol is highly efficient for real time communication with its simple architecture designed to prioritise message delivery as quick as possible. However this also introduces constraints:

* **Frame Size:** The CAN frames are limited to 8 bytes of data which makes for a significant challenge to introduce encryption. Encryption algorithms generally add overhead in the form of initialisation vectors, padding or keys which may result in exceeding the available payload of a single message. This would then introduce the need for message fragmentation.
* **Low Latency:** CAN is designed for low latency communication between ECUs. Encryption introduces overhead which could delay the transmission and processing of the data.

1. **Lightweight Cryptographic Algorithms and Integration:**

In regard to the limitations of the CAN bus, lightweight cryptographic algorithms can be a better solution than standard cryptographic algorithms such as AES-256. AES-128 would be a good candidate due to its lower computational complexity compared to AES-256 but still being able to provide a high level of security.  
  
Encryption on the CAN bus requires careful integration to avoid breaking the fundamental principle of the protocol. The following aspects need to be considered:

* **Key Management:** Securely distributing and managing encryption keys is an essential. Key distribution schemes are also important to decide such as Symmetric key encryption for AES.
* **Encryption Placement:** Determining wherethe encryption should be applied in the CAN message is also important. Deciding whether the whole message should be encrypted or just the portion that included sensitive data such as mileage data.

1. **CAN-FD as an Alternative:**

Using CAN-FD as an alternative to standard CAN would allow for a lot more flexibility on the encryption side. CAN-FD allows for a larder frame size which is up to 64 bytes. This larger frame size would give more room for encryption overhead making it more suitable for implementing encryption while still maintaining compatibility with legacy CAN systems.

1. **The Impact of Encryption on the CAN Bus Performance:**

Implementing encryption in the CAN bus system will without a doubt have an impact on performance such as transmission latency, CPU load on ECUs and network bandwidth.

* **Latency and Real time Communication:**

Encryption increased message processing time and since CAN networks prioritise real time communication encryption may potentially lead to some delays. It will be important to investigate the affect that the added latency may have on the network.

* **Simulation of encrypted CAN traffic:**

Using software simulation I will evaluate how different encryption algorithms impact latency on the network.

* **Bandwidth Overhead:**

Encryption will add overhead in terms of addition data fields like initialisation vectors and padding which will reduce the effective payload capacity of CAN frames. This means that more frames may be required to transmit the same amount of data which may increase the overall bandwidth usage.

* **ECU Resource Utilisation:**

ECUs in vehicles have limited computing power, memory and power resources. Encrypting and decrypting messages can be expensive for these systems particularly for light weight ECUs.

## Telematics Systems and Vehicle-to-Cloud Communication

The Telematics systems in modern vehicles play a crucial role in vehicle to cloud communication by allowing data collection and transmission between the vehicles onboard systems and offboard networks. These systems are perfect for sending real time data about vehicles status, location and diagnostics. I will explore the structure and functionality of Telematics systems to understand how data is securely collected and transmitted to the cloud.

### Communication & Protocols

Telematics systems rely on specific communication protocols to transmit data between the vehicle and external networks.

* **Cellular Communications (4G/5G):**

Cellular networks are often used to connect the vehicle to the cloud. Both 4G and 5G provide different capabilities each with unique advantages for telematics including bandwidth, latency and coverage.  
  
4G: is the most widely deployed network for telematics applications today which offers a good balance of speed, latency and coverage which is crucial for transmitting vehicle data.  
  
5G**:** The next generation cellular technology, 5G offers significantly higher bandwidth, reduced latency and stronger connectivity enabling more complex telematics applications. As 5G networks continues to expand they will bring added benefits to vehicle data transmission.

* **Data Integrity in Vehicle-to-Cloud Communication:**

Data integrity makes sure that information stays correct and unaltered from the time its created to when it reaches where it’s supposed to go. There are plenty of ways to keep data accurate and secure from the moment its produced in the vehicle all the way to when it’s saved in the cloud. These include cryptographic methods, message authentication and tamper detection systems.

Message Authentication Code or MAC can be used to verify integrity and authenticate a message that is being shared between the vehicle and cloud. MAC uses a secret key that is shared between both the vehicle and the cloud system, it works by adding a cryptographic tag to each message being sent to the from each side of the communication network.

* **Transport Layer Security:**

Transport Layer Security or TLS is a commonly used protocol in modern vehicles for securing data in transit between a client and server. TLS provides encryption, authentication and data integrity in vehicle to cloud communication.

TLS works by creating a secure tunnel between the vehicle and the cloud server which protects all data transmitted within this connection.

1. **TLS Handshake:**
   1. The TCU on the vehicle initiates a connection to the cloud server by requesting a secure connection.
   2. The server then responds with a digital certificate which is verified by the TCU to ensure that the server is legitimate.
   3. Both the TCU and the cloud server generate and exchange session keys which are used for encrypting the data.
2. **Session Encryption:**
   1. Once the secure session is established all data sent from the TCU to the cloud are encrypted with symmetric encryption such as AES-128. The encryption prevents eavesdropping to anyone intercepting the data in transit.
3. **Integrity Checks with MAC:**
   1. TLS also ensures that each data packet is signed with a MAC or message authentication code which allows the recipient to verify that the data has not been altered while in transit.
   2. Once the packet arrives the recipient recalculates the MAC and compares it with the MAC attached to the packet. If both MACs match the integrity of the data is confirmed.
4. **Session Termination:**
   1. Once data transmission is complete the TLS session is terminated. Each session has its own unique key which prevents future transmission form using the same encryption key.

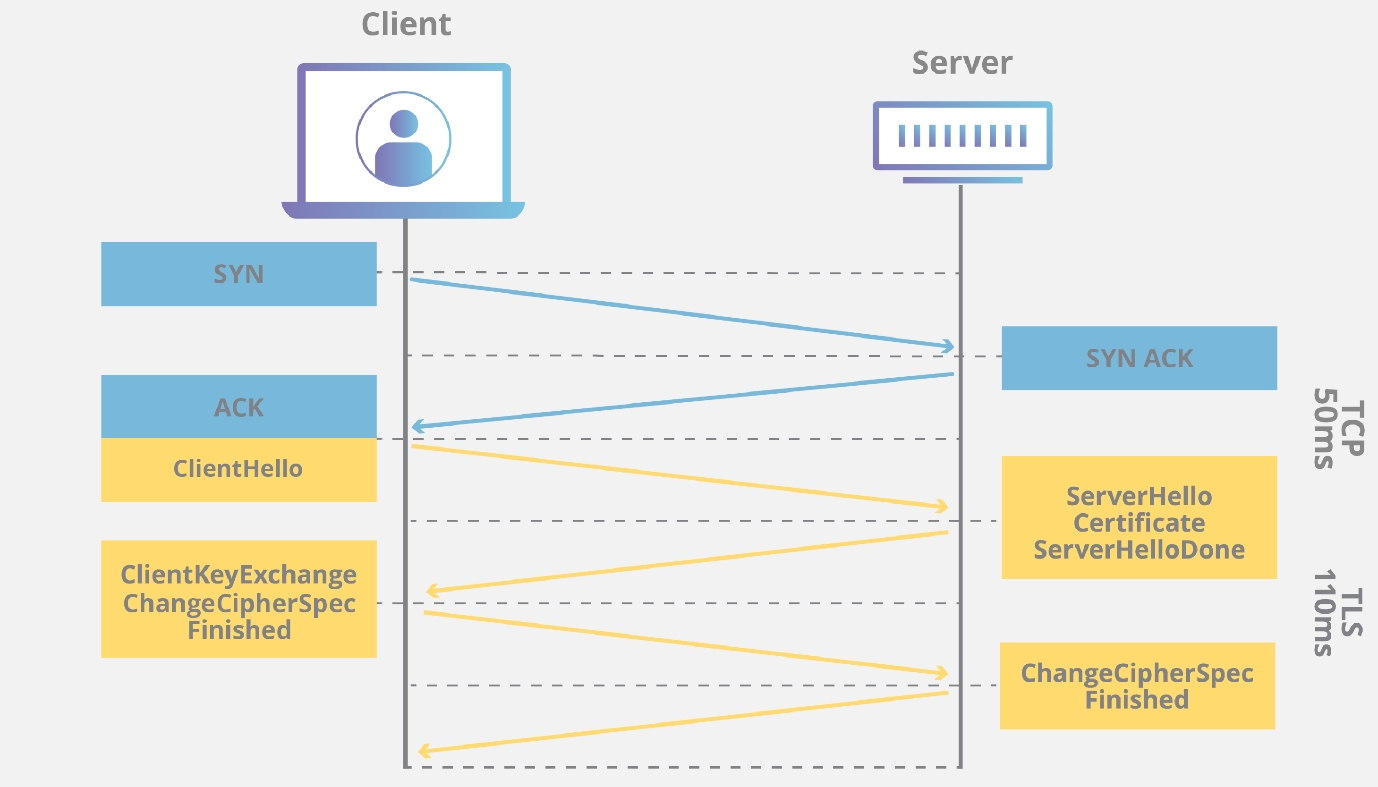


Figure 2 (What is TLS (Transport Layer Security)?, no date)

### Large-scale data management

As telematic systems in modern vehicles continue to evolve vehicles begin to generate and transmit large volumes of data. Managing this data effectively across a large fleet of vehicles requires a large-scale data management system to ensure efficient data collection and secure storage. For this investigation I will be focusing on the key points of data management in relation to the project goal.

1. **Key Components:**
   1. **Cloud Storage Architecture:**It is important to have a well designed cloud storage architecture for handing vast amounts of data being sent through the telematics systems. I will look at some of the key storage options that will mostly relate to what this project requires.
      1. **Distributed Storage:**A good option of storage for a telematics system is distributed storage such as Amazon s3 buckets, Microsoft Azure Blob storage or Google cloud storage to handle the large volume of data. Distributed storage works by dividing data across multiple servers which provides high availability and scalability.
      2. **Cold and Hot storage:**In the case of this project I will look into both cold and hot storage. Hot storage is used for frequently accessed data and allows for faster access. While cold storage is used for data that requires less frequent access and can be used to reduce cost.
   2. **Data Retrieval:**It is also important to setup efficient data retrieval that will ensure that data like recent mileage is quickly available.
      1. **Accessing Data:**In cases where data will need to be accessed globally something like AWS CloudFront would be very useful as it can cache and distribute data across multiple geographic locations. This would help improve accessibility for data like mileage globally.
   3. **Database Solution:**Before beginning the project its important that I pick the correct database for the projects needs.
      1. **Time Series Database:**I believe that the best suited database for storing mileage data is a time series database using AWS Timestream. These databases are optimised for storing and querying time indexed data which makes them ideal for monitoring change in mileage data.

# Interface Specifications

## Onboard

For this project I plan to have a simulated odometer in canoe that will be used to show the vehicles mileage. This will allow me to show that the simulated vehicle is gaining mileage without a tampering device it shall also show that a tampering device can spoof/block the mileage and lastly it should also demonstrate that the system that I develop to prevent mileage tampering is effective.

The onboard interface shall not show any warning to the driver that mileage tampering has been detected as this will be logged using a DTC.

## Offboard

The offboard interface such as a UI will be designed to be used by fleet management or technicians only.

1. **User Interface:**I will create a web-based dashboard which will be designed to provide the user with a clear platform to view and manage the vehicles data such as mileage and tampering alerts.  
   1. **Screen Layouts:**
      1. **Login Screen:** A login screen will be created which will allow the restriction on who can access the data.
      2. **Dashboard Home:** The home page will be used to display key metrics such as total mileage and active tampering alerts.
      3. **Vehicle Details:** There will also be another section that will show detailed data for the selected vehicle which will include mileage history and all past tampering alerts.
      4. **Alerts:** Tampering attempts will be listed as alerts which will also have the option to mark them as resolved.
   2. **Control Actions:**
      1. **View Mileage Logs:** Allows access to detailed mileage data history.
      2. **Acknowledge Alerts:** Markalerts as resolved if required.
      3. **Generate Report:** Ability to export data for analysis or compliance purposes.
   3. **Navigation:**
      1. Login > Home > Vehicle Details.
2. **Database Interface:**I will the database interface to define how the data is stored, retrieved and managed within the cloud system.  
   1. **Database System:**I will use AWS Timestream as my database to store data as it is ideal for time series data.
   2. **Main Tables:**
      1. **Mileage Data:** Stores odometer readings with timestamps and vehicles UUID.
      2. **Alerts:** Contains tampering alerts, timestamps and response status.
      3. **Vehicle Information:** Stores details like vehicle UUID, model and year.
   3. **Data Access:**
      1. **RESTful APIs:** Will be used for direct HTTP requests to the Timestream database for data ingestion and querying.
3. **Network Protocols:**The network protocols specify how data is transmitted between the vehicles Telematics Control Unit, CAN bus, cloud server and database.  
   1. **CAN Bus:**
      1. **Protocol:** CAN-FD and CAN for in vehicle communication between ECUs.
      2. **Data Exchanged:**
         1. **Odometer Readings:** Regularly updated mileage data from source ECU (such as BCM).
         2. **Diagnostic Codes:** Transmits diagnostic fault codes such as DTCs.
      3. **Communication Type:** Broadcasts messages to all ECUs on the CAN network.
      4. **Security:** SecOC protocol and encryption for integrity and authenticity of data.
   2. **Vehicle to Cloud:** 
      1. **Protocol:** MQTT and TLS for lightweight and secure data transmission.
      2. **Data Exchange:**
         1. **Mileage Data:** Transmitted at scheduled intervales (start-up/shutdown) for central logging.
         2. **Tampering Alerts:** Instant notifications/logging sent when tampering is detected.
      3. **Communication Type:** Publish-subscribe for MQTT, client-server communication for TLS.
      4. **Security:** TLS 1.3 encryption to secure data in transit and prevent unauthorised access.

# Non-functional Requirements

## Performance Requirements

### Response Time:

The should ensure that the vehicle data is transmitted from the vehicle to the cloud in a reasonable time.

### Data Processing Speed:

The cloud system shall be able to adequately process data from each vehicle and update dashboards accordingly.

### Network Bandwidth:

The system should support efficient data transferring with reasonable network speeds on 4G and 5G networks.

### Latency for Critical Alerts

When tampering is detected and alert should be sent to the cloud as soon as the detection occurs.

## Security Requirements

### Data Encryption:

All data related to mileage should be encrypted, using AES for CAN network messages and TLS for vehicle to cloud messages.

### Access Control:

The data collected from each vehicle relating to mileage should be protected via Role-based access control. This shall allow only fleet management and technicians access to mileage data.

### Message Integrity:

All data should include some sort of integrity check to verify that each message is from a verified source. This shall be done by using SecOC for CAN and TLS for vehicle to cloud communication.

### Audit Logging:

The system shall log all access and data modification activities ( eg. Marking an alert as resolved). This should include timestamps, user IDs and the action.

## Scalability Requirements

### Horizontal Scaling:

The cloud infrastructure should support horizontal scaling to handle an increase in the number of connected vehicles without a performance effect.

### Data Storage

The database should be able to store vehicle data history for several years with also including scalable storage solutions as data volumes continue to grow.

# Quality Assurance Provisions

## Software Test procedures

For the software test procedures of this project, I plan on using Vectors tool called Vector CAST. “Vector CAST is a highly automated unit and integration test solution to validate safety and business critical embedded systems.” (VectorCAST/C++, no date ). This will allow me to ensure that individual functions work as intended.

Integrating the system will also be another form of my testing as it will verify how the different components such as the TCU, CAN bus and cloud interact with each other. Using CANoe I will be able to test the flow of data between ECUs and the cloud. I will also be able to use Postman to test APIs for the database.

## Software Validation Procedures

The software validation procedures will help me confirm that the system will be able to meet the requirements that I outlined in the requirements specification. I will use Vector CANoe to validate CAN message transmissions and interactions between vehicle components while Postman will help me test APIs and the user interface. I will use realistic data to confirm that important functionalities like mileage updating and tampering alerts work correctly.

# Risk and Contingencies

This project includes many potential risk. It is important that I identify these risks early and have a solution ready.  
  
It is very possible that I may encounter integration challenges due to the need to connect diverse components such as CAN bus, TCU and a cloud system. My way of managing this will be to catch any problem with this during prototyping while using tools like CANoe which will help identify any issues early.   
  
With the idea of encryption in this project it is very possible that I may encounter performance issues especially in real world applications that are under heave data loads. The limited computational resources of ECUs could pose challenges for data processing and security task like encryption and decryption. I will conduct load testing and using a lightweight algorithm like AES-128 should help me balance performance and security.

# Methodology Overview

For the development of this project I chose to follow the V-Cycle development methodology. This process is greatly suited for my project as it requires verification and validation at each phase. The ensures that each as the project is being developed it is also being tested as the project continues.

1. **Requirements Analysis**

The first step of my project was to gather and analyse all requirements for the system. I was mainly focusing on functional but also keeping non functional requirements in mind. This involved me defining specific requirements for data handling, security, performance and user interface specification.

1. **System Design**

The system design phase was there I began to outline the architecture and design of the system, detailing how components such as the CAN bus, Telematics Control Unit and cloud communication will interact with each other. I designed an inept system context diagram which helped visually see what was going where in the project. This will also greatly help when it comes to the development stage of the project. This phase is also where I began to look at what tools I required and after being introduced to some of VECTOR tools such as DaVinci and CANoe. I was able to visualise how my project may come together and how the tools would help me complete my project.

1. **Design Investigation**

In this phase I would begin to investigate more into the specific design of each moduel within the system such as the Cloud communication and CAN network. Investigating each module helped define each function and interface between modules. Understanding each component of this project was very important along with trying to get a good understanding before the development began. I was introduced to some AWS features in one of my modules which I believe will be key when it comes to designing the database for the offboard storage. I had to ability to explore some of the feature’s AWS has to offer and I also had the opportunity to use an assignment for this module to get a start on developing the API and database for this project.

# Project Plan

1. The first step I took for my project was to begin thinking about what research I would need to conduct in order to understand the problem and also be able to develop a solution. I had already got somewhat of an understanding of the problem as this project was a task that I covered while I was on work placement. I began by creating a blank word document and began to create bullet points on what the project would need and involve. I visualised the process like building a house in terms of the steps taken.
2. Using the “rough work” and after discussions with my supervisor we came up with a project plan which was created on TeamGantt. This allowed me to have each task written out on show and also allowed me to insert a time period for which each task should be completed by.

A screenshot of a project gantt chart

Description automatically generated

1. Once my project plan was completed, I then created the report document, finalising my idea and setting research priorities. The functional summary was the starting point detailing core system functions such as mileage tracking, tamper detection, and secure data storage. This summary narrowed down important system features which in turn helped with later stages and minimised me expanding the project scope later in the project. This kept the project aligned with its goals and reduced the potential for project to veer off course.
2. Following the functional summary, I created some system assumptions such as standard protocols for target vehicles and reliable network connectivity for offboard storage. These assumptions were important for the design phase where I outlined data handling, security requirements and interface specifications.
3. After setting the functional summary and system assumptions, I moved onto the functional requirements. This phase involved identifying and all functional and data requirements for the system. I outlined key functionalities such as real time mileage updates, tampering detection and secure data logging to meet the projects core objective. I also defined non-functional requirements later which included system performance, security standards and user interface usability. By specifying these requirements early on I established clear goals and benchmarks for each stage helping to keep the project on track.
4. With a clear understanding of requirements, I progressed to the technical investigation phase, where I would conduct in depth research for each component of the system. This involved investigating suitable protocols such as CAN and CAN-FD for in vehicle communication and MQTT and TLS for secure vehicle to cloud data transfer. Once I have completed the investigation I had a solid technical foundation to guide the development process.

# References

Understanding CAN Bus Vulnerabilities and How Blockchain Can Amplify Security   
( 19-09-24 ) | Medium. <https://medium.com/@chaincom/understanding-can-bus-vulnerabilities-and-how-blockchain-can-amplify-security-a58388bf1fb4>. ( Accessed: 18-10-2024 )

Specification of Secure Onboard Communication Protocol ( 30-11-2020 ) | AUTOSAR. Available at: <https://www.autosar.org/fileadmin/standards/R20-11/FO/AUTOSAR_PRS_SecOcProtocol.pdf>. ( Accessed: 18-10-2024 )

Secure Onboard Communication ( no date ) | Vector. Available at: <https://www.vector.com/us/en/products/solutions/safety-security/automotive-cybersecurity/security-manager/#c212393>. ( Accessed: 18-10-2024 )

What is TLS (Transport Layer Security)? ( no date ) | Cloudflare. Available at: <https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/>. ( Accessed: 22-10-2024 )

VectorCAST/C++ ( no date ) | Vector. Available at: <https://www.vector.com/us/en/products/products-a-z/software/vectorcast/vectorcast-c/#:~:text=VectorCAST%2FC%2B%2B%20is%20a%20highly,%2C%20railway%2C%20and%20financial%20industries>. ( Accessed: 28-10-2024 )