D5.4 Appendix A

ATE – Tests – Vertical 1 Scenario 1 (TEC Demonstrator)

Project number	830892
Project acronym	SPARTA
Project title	Strategic programs for advanced research and technology in Europe
Start date of the project	1st February, 2019
Duration	36 months
Programme	H2020-SU-ICT-2018-2020

Deliverable type	Report
Deliverable reference number	SU-ICT-03-830892 / D5.4 / V1.0 / Appendix A
Work package contributing to the deliverable	WP5
Due date	Jan 2022 – M36
Actual submission date	2nd February, 2022

Responsible organisation	TECNALIA
Editor	Angel López
Dissemination level	PU
Revision	V1.0

Abstract	This document is part of the ATE activities performed during task 5.4 of the CAPE program of the SPARTA project. This document contains test procedures and report for the TEC demonstrator in Scenario 1 named Basic Scenario
Keywords	Assessment, security test, platoon, safety, security, connected cars

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892.

Editor

Angel López (TEC)

Contributors (ordered according to beneficiary numbers)

Martínez Cristina, Amparan Estibaliz (TEC)

Mirko Malacario, Claudio Porretti, Nicoletta Imperatori (LEO)

Reviewers (ordered according to beneficiary numbers)

Maximilian Tschirschnitz (TUM)

Rimantas Zylius (L3CE)

Disclaimer

Table of Content

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Document Overview

This document provides a description of the test procedure and report for the TEC demonstrator in the Basic Scenario (also known as Scenario 1) of the "Connected and Cooperative Car Cybersecurity" vertical (also knowsn as Connected Car Vertical or Vertical 1).

The goal of Scenario 1 is to evaluate the process, from security analysis, requirements to implementation, verification, and validation, for increasing the security of vehicle platooning when assuming a malicious intruder that can manipulate the communication channels.

Figure 1 shows the life-cycle of the requirements that have been explicited for the Connected Car Vertical. First of all, the requirements have been stated in the Protection Profile (PP) document [3] Secondly, the requirements have been addressed in Secure Design, as it was documented in D5.2 [1] and D5.3 [2]. Finally, the requirements have been tested and assessed, and this process has been documented in the current document, thus supporting the ATE test phase of the evaluation procedure based on Common Criteria [4], [5] and [6].

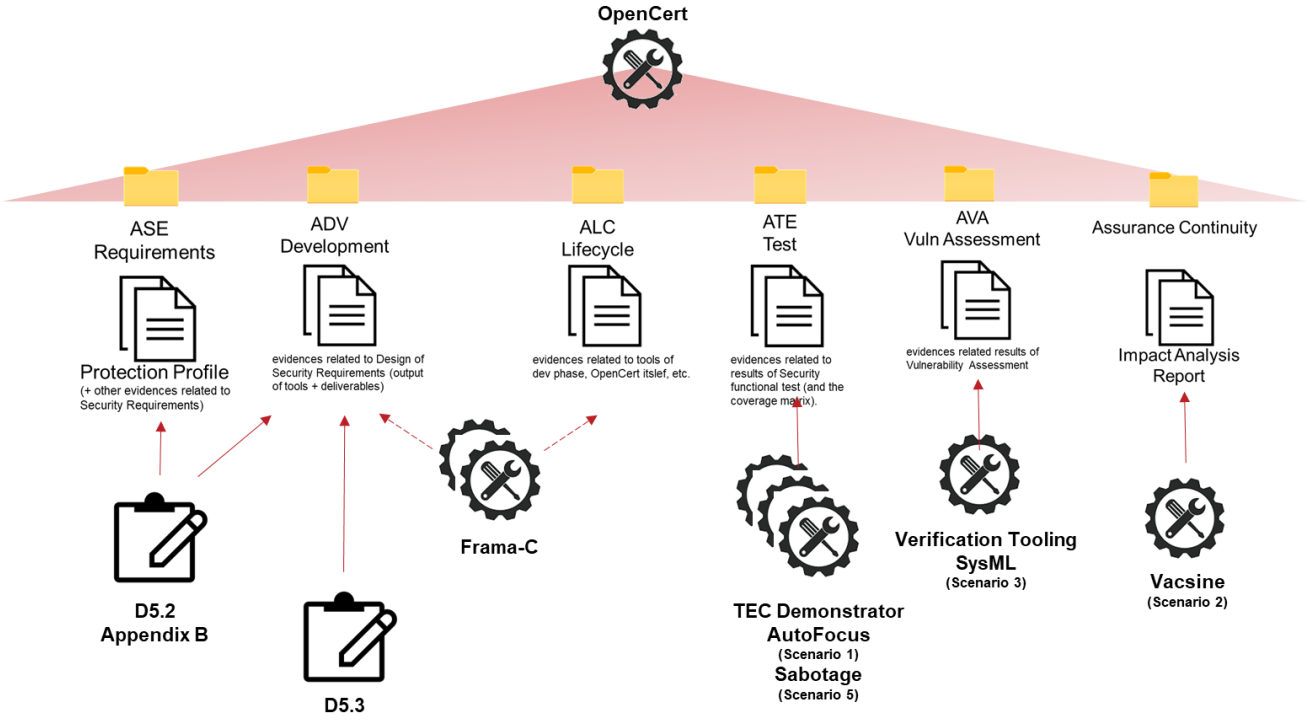

Figure 1: Evaluation strategy for the Connected Car Vertical

The structure of the document is organized as follows:

- **Chapter 1 Introduction** is the current section presenting the objectives, scope and structure of the document.
- **Chapter 2 Test preparation** presents the hardware and software used for testing.
- **Chapter 3 Test descriptions** details the different test cases to be executed and their results.
- **Chapter 4 Test Summary Coverage** shows the completeness of tests coverage.

Chapter 2 Test preparation

2.1 System overview

The Connected Car Vertical (Vertical 1) has been fully described in D5.2 [1]. In this section, we provide an overview of the case study description, focusing on the first scenario, named “Basic Scenario”.

The goal of the Connected Car Vertical is to advance the cyber-security of connected vehicles driving in platoon mode. A platoon is a sequence of vehicles as depicted by Figure 2, that it is composed by a leader vehicle and a sequence of followers.

Each vehicle in the platoon communicates using dedicated communication channels. Moreover, each vehicle in the platoon possesses sensors, such as cameras, distance sensors, enabling a highly automated mode of operation. Indeed, when formed, the platoon requires only driver supervision.

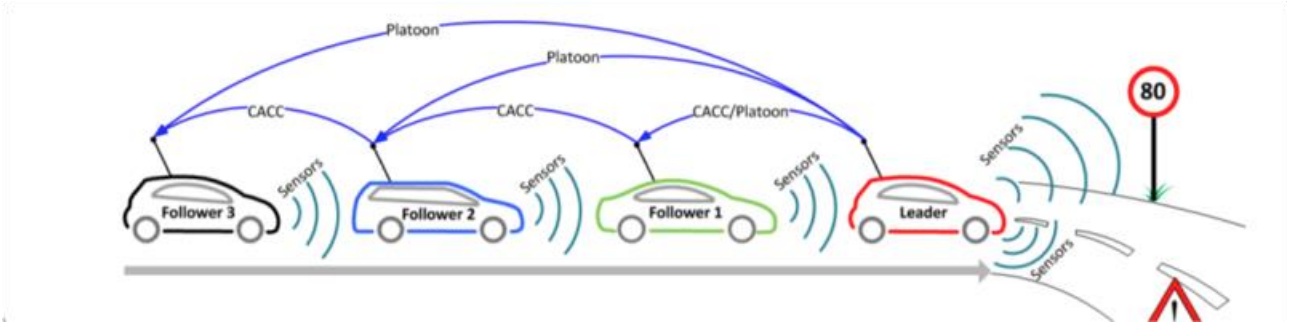

Figure 2: Platooning scenario

We consider a platoon of three members, with one leader and two followers using Cooperative Adaptive Cruise Control (CACC). All cars have exactly the same hardware and the same platooning software but with different configurations.

The platoon vehicles navigate on the circuit designed.

The vehicles can communicate each other thanks to a WiFi 802.11n access point.

2.1.1 Hardware preparation

Figure 3 shows the hardware components of TEC rovers, a more detailed description can be read in D5.2 [1]:

- ADAS-ECU. An Odroid-XU4 board that runs all the autonomous-driving-functions and the CACC functions developed by FORTISS. This board is also in charge of managing the rover communications with other rovers, through a WiFi network, and with the Vehicle-ECU, in order to manage the camera and the ultrasonic sensor.
    - All these functions together with the developments made to cover the requirements defined by the Protection Profile, which are being verified in this document, are what we, call “platooning software”.
- Vehicle-ECU, to manage the speed and direction of the rover.
- Camera to capture images for the detection of the road to calculate the direction.
- Ultrasonic Sensor, to obtain the distance of the preceding rover or the distance to any obstacles in the path of a rover.
- Encoders in each of the back wheels to measure the current wheel speed.
- Wifi Module to allow communications between rovers.

- Remote Control. The vehicle remote control system is used as a deadman-switch to enable the movement of the rover when the platooning software is running.

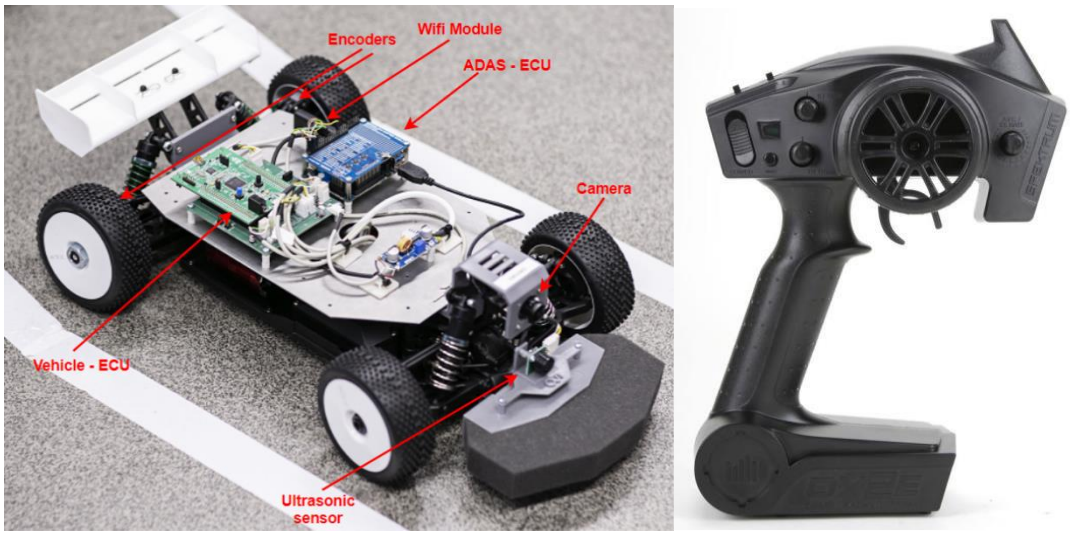

Figure 3: TEC Rover + Remote Control

The test will be executed on a single lane circuit of 3m x 2,5m (see Figure 4), marked with white lines separated approximately by 45cm.

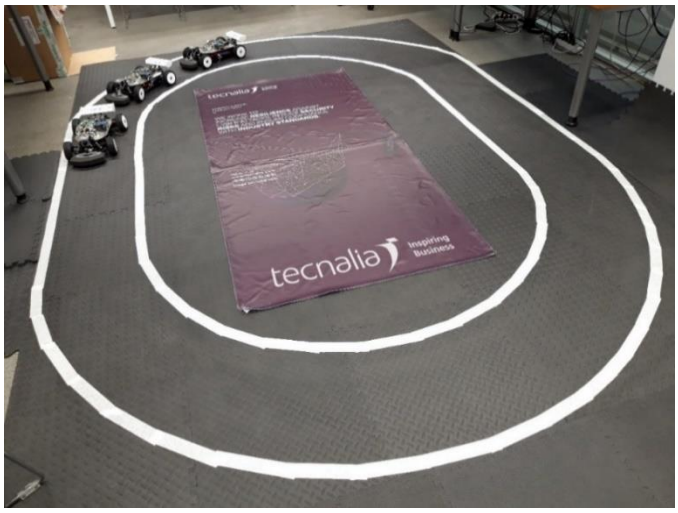

Figure 4: TEC Rovers moving on the circuit

The following conditions must be met for the testing:

- The car batteries must have a charge of more than 30%.
- The WiFi access point to create the platoon private WiFi network will be turned on.
- A testing laptop will be available with useful tools installed to connect to the cars via SSH, debug the developed code, capture network traffic, manage databases and any other utility needed to evaluate test cases.

2.1.2 Software preparation

All the cars connect automatically to the platoon private WiFi network during the start-up.

All cars have exactly the same platooning software but with different configurations, therefore, there is no need to test the same requirements in every car.
Every car is identified in the platoon system as follows:

- car1 always is configured as *leader*, and its id is 1.
- car2 is configured as *follower 1*, and its id is 2.
- car3 is configured as *follower 2*, and its id is 3.

The Connect Car basic scenario is complemented by the incorporation of a Dashboard (see Figure 5), a web page that allows to check the status of the platoon and launch cyber-attacks to the platoon members.

The Dashboard shows data for each platoon member, such as, the car unique identifier, the current mode of the car in the platoon, the current speed of the car, the distance gap with any obstacle or car behind detected by the distance sensor, the identifier and speed of the preceding vehicle, and the number of messages received that failed any of the plausibility checks.

Since all this real-time information about the cars displayed by the Dashboard is related to many of the test cases described in Chapter 3, we consider the Dashboard as a tool to evaluate the result of those tests cases.

The Dashboard will also have capabilities to inject false speed messages from one car to another, this is a requirement needed in some of the test cases.

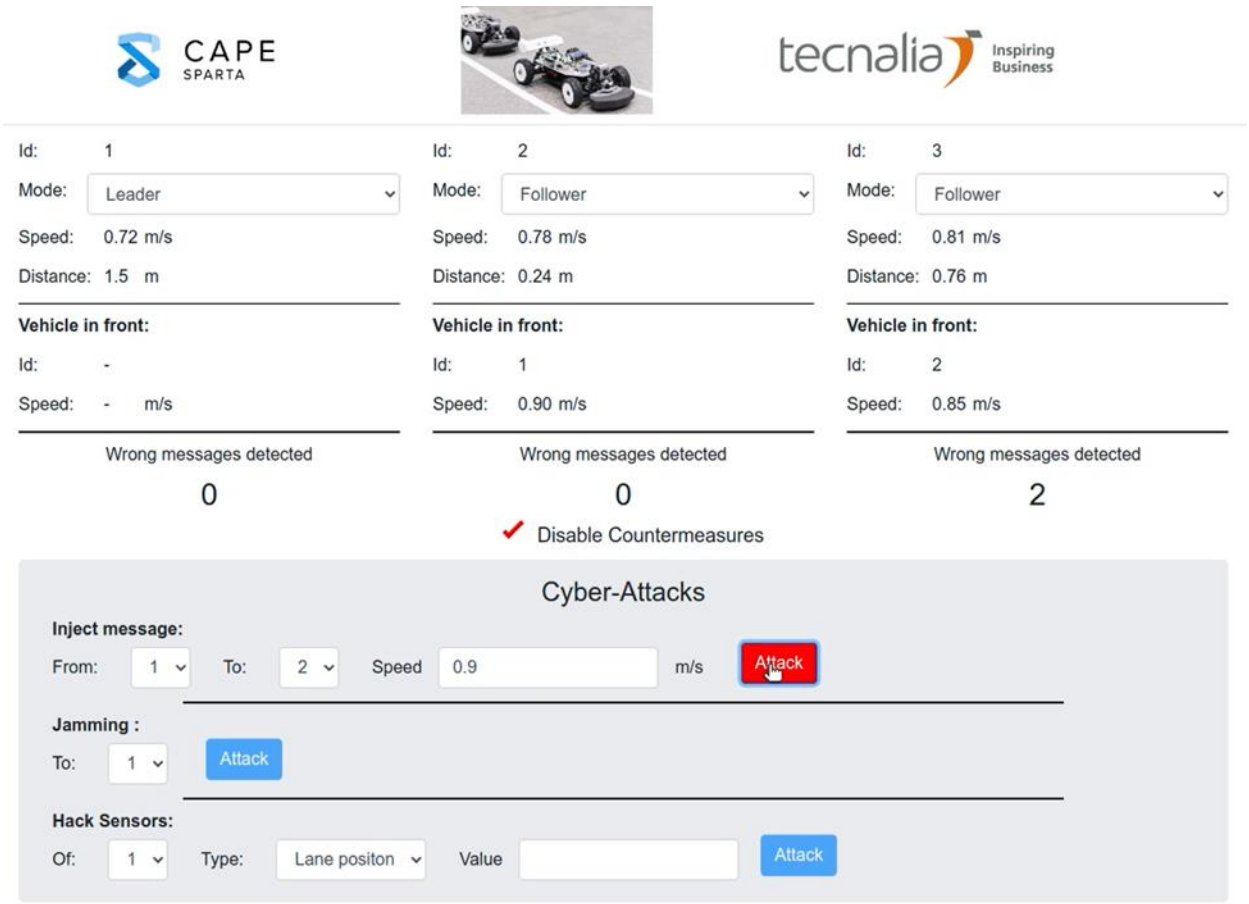

Figure 5: TEC Dashboard tool, developed for testing the Scenario 1

Chapter 3 Test descriptions

Table 1 shows the Security Functional Requirements (SFRs) of the PP [3] that have been implemented for the Scenario 1 in TECNALIA's demonstrator. In the next sections, we describe the test descriptions that have been elaborated to support the test of these requirements.

Req. Id (PP [3])	Short Description
PMM_IF.1	Maintain heart-beat data (vehicle identifier, speed, direction, geo-position, timestamp) to VCS
PMM_IF.2	Maintain heart-beat data from VCS
PMM_IF.3	Maintain incoming emergency brake information flow from other vehicles
PMM_IF.4	Maintain outgoing emergency brake information flow to other vehicles
PMM_IF.5	Maintain data from VCM
PMM_IF.6	Maintain data to VCM
PMM_PC.1	Data passes all VCS plausibility checks
PMM_PC.3	Inform on Failed Plausibility Checks
PMM_VCS-HPC.1	Maintain heart-beat data history
PMM_VCS-HPC.2	Heart-beat message consistent to the history
PMM_VCS-SPC.1	Maintain distances history
PMM_VCS-TPC.1	Consult the TOE vehicle internal clock
PMM_VCM-HPC.1	Maintain sensor data history
PMM_VCM-TPC.1	Consult the TOE vehicle internal clock
FPT_ITC.1	Inter-TSF confidentiality during transmission
FPT_ITI.1	Integrity of exported TSF data

Table 1: Requirements covered by the TEC demonstrator in the Scenario 1

3.1 PMM_IF.1_TC1

Test case to validate the generation and the composition of a heartbeat (HB) data message.

3.1.1 Security Requirements addressed

PMM_IF.1

3.1.2 Test preconditions

Car1 and car2 are on. Car3 is off.

There are no obstacles in the circuit.

3.1.3 Expected test results

Car1 generates HB messages every 0,5 seconds, with the following (non-null) information: the vehicle unique identifier, the vehicle speed, direction, Geo-position and timestamp.

3.1.4 Criteria for evaluating results

The test duration is 1 minute.

The data interchanged is stored in the car1 local database.

A HB message is generated at least every 0,5 s.

3.1.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role, with 50 cm of separation between each other.
- Set a 0,5 m/s speed to car1 (leader).
- Run the platooning software in each car involved in the test by using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open the histo_1.db copied
- Verify the HB messages generated by car 1, their frequency, and their contents using the SQL:

```
Select * from intercar_data where source = 1 and destination = 2
```

3.1.6 Test Results

Car1 generates HB messages every 0,5 seconds and the messages contain the information abovementiended with not null values.

The HB messages also contain the distance to any object detected by the ultrasonic sensor, a flag to indicate is the car is in *emergency_brake* state (with a 0 value) and the *emergency_brake* identification number (with a 0 value).

Status: *PASSED WITH DEVIATIONS*

Deviations from test procedure

The cars do not have GPS; HB messages do not contain Geo-Position.

3.2 PMM_IF.1_TC2

Test case to validate the sending of HB messages.

3.2.1 Security Requirements addressed

PMM_IF.1

3.2.2 Test preconditions

All the cars are on.

There are no obstacles in the circuit.

The contents of the "*intercar_data*" table have been deleted in every car's database.

3.2.3 Expected test results

All the HB messages generated by car1 are sent to car2 and car3.

3.2.4 Criteria for evaluating results

The test duration is 1 minute.

A HB message is generated at least every 0,5 s.

The data interchanged between car1, car2 and car 3 can be seen in the Dashboard.

All the HB messages interchanged between cars are stored in each car in a local database, so we will check in car's local database that the contents of the "*intercar_data*" table contain all the HB messages sent.

3.2.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to car1 (leader).
- Run the platooning software in each car involved in the test by using the SSH connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Verify in the dashboard, in the car2's section labelled "Vehicle in front" that shows a 0,5m/s speed.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open the database copied
- Verify the HB messages sent by car 1 using the following SQL sentence:

```
Select * from intercar_data where source = 1 and (destination = 2
                        or destination = 3)
```

3.2.6 Test Results

All the HB messages generated every 0,5 s by car1 are sent to car2 and car3.

Status: *PASSED*

3.3 PMM_IF.2_TC1

Test case to validate the reception of HB messages with the following (non-null) information: the vehicle unique identifier, the vehicle speed, direction, Geo-position, timestamp and digitally signed certificates.

3.3.1 Security Requirements addressed

PMM_IF.2

3.3.2 Test preconditions

All the cars are on.

There are no obstacles in the circuit.

The contents of the "*intercar_data*" table have been deleted in every car's database.

3.3.3 Expected test results

HB messages with the following (non-null) information: the vehicle unique identifier, the vehicle speed, direction, Geo-position, timestamp and digitally signed certificates are received.

All the HB messages sent by car1 are received by car2 and car3.

All the HB messages sent by car2 are received by car1 and car3.

All the HB messages sent by car3 are received by car1 and car2.

3.3.4 Criteria for evaluating results

The test duration is 1 minute.

The data interchanged between car1 and car2 can be seen in the Dashboard.

The data interchanged between car2 and car3 can be seen in the Dashboard.

All the HB messages interchanged between cars are stored in each car in a local database, so we will check in every car's local database that the contents of the "*intercar_data*" table contain the same data.

3.3.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 speed to car1 (leader).
- Run the platooning software in each car involved in the test by using the SSH connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.

- Verify in the dashboard, in the car2's section labelled "Vehicle in front" that shows a 0,5m/s speed.
- Verify in the dashboard, in the car3's section labelled "Vehicle in front" that shows a speed different to 0,0m/s speed.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop
- Copy the car3 database in /odroid/sparta/ flask_server/histo_3.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open histo_1.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 1
```

- Run another instance of the tool DB Browser for SQLite in the testing laptop and open histo_2.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 2
```

- Run another instance of the tool DB Browser for SQLite in the testing laptop and open histo_3.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 3
```

- To verify that the HB messages sent by car1 are received by car2 and car3 check that the results of the sentence

```
Select * from intercar_data where source = 1
```

against the three databases are the same.
- To verify that the HB messages sent by car2 are received by car1 and car3 check that the results of the sentence

```
Select * from intercar_data where source = 2
```

against the three databases are the same.
- To verify that the HB messages sent by car3 are received by car1 and car2 check that the results of the sentence

```
Select * from intercar_data where source = 3
```

against the three databases are the same.

3.3.6 Test Results

The content of "*intercar_data*" table is the same in every car.

The HB messages contain the information abovementioned with not null values.

Status: *PASSED WITH DEVIATIONS*

Deviations from test procedure

The cars do not have GPS, HB messages do not contain Geo-Position.

The digitally signed certificate is stored in every car, HB messages do not contain the digitally signed certificate.

3.4 PMM_IF.3_TC1

Test case to validate the reception of emergency brake messages (EB) with the following (non-null) information: identifier of the vehicle to which the emergency brake has been issued, emergency brake identifier, timestamp and digitally signed certificate.

3.4.1 Security Requirements addressed

PMM_IF.3

3.4.2 Test preconditions

All the cars are on.

The contents of the "*intercar_data*" table have been deleted in every car's database.

There are no obstacles in the circuit.

3.4.3 Expected test results

All the EB messages sent by car1 are received by car2 and car3.

EB messages with the following (non-null) information are received: identifier of the vehicle to which the emergency brake has been issued, emergency brake identifier, timestamp and digitally signed certificate

When car2 also enters in EB state, all its EB messages sent are received by car1 and car3.

When car3 also enters in EB state, all its EB messages sent are received by car1 and car2.

3.4.4 Criteria for evaluating results

The test duration is the time needed by car1 to reach the end of the circuit.

Car EB messages must be generated at least every 0,5 seconds from when car1 breaks the safety distance. The EB messages

All the EB messages interchanged between cars are stored in each car in a local database, so we will check in every car's local database that the contents of the "*intercar_data*" table contain exactly the same data.

The Dashboard shows a zero speed for each car in EB status, and for its follower a zero speed is displayed in the field that shows to the speed received from the preceding vehicle.

3.4.5 Test Procedure

The following steps will be carried out:

- Put an obstacle (e.g. a cardboard box) 20 centimeters behind car3.
- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Check on TEC dashboard that car1 speed shown is 0,0 m/s when breaks the safety distance with the box and in the car2's section labelled "Vehicle in front" that shows a 0,0m/s speed.

- Check on TEC dashboard that car2 speed shown is 0,0 m/s when breaks the safety distance with car1 and in the car3's section labelled "Vehicle in front" that shows a 0,0m/s speed.
- Check on TEC dashboard that car3 speed shown is 0,0 m/s when breaks the safety distance with car2.
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop
- Copy the car3 database in /odroid/sparta/ flask_server/histo_3.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open histo_2.db copied to verify the information of the EB messages received using the following sentence:

```
Select * from intercar_data where source=1 and destination = 2 and
                        emergency_brake=1;
```

- In the tool DB Browser for SQLite open histo_3.db copied to verify the information of the EB messages received using the following sentence:

```
Select * from intercar_data where source=1 and destination = 3 and
                        emergency_brake=1;
```

3.4.6 Test Results

Car1 generates EB messages every 0,5 seconds from when it breaks the safety distance, the messages are received by car 2 and car3. Later, car2 enters in EB state and starts generating emergency break messages for car1 and car3. Finally, car3 also enters in emergency state.

The EB messages contain the information abovementioned with not null values but also more information as speed, distance and direction.

When a car is in EB state, the Dashboard shows a zero speed. Also, for its following car, the Dashboard shows a zero speed in the field that shows the speed of the preceding vehicle.

Status: *PASSED WITH DEVIATIONS*

Deviations from test procedure

The digitally signed certificate is stored in every car, EB messages do not contain the digitally signed certificate.

3.5 PMM_IF.4_TC1

Test case to validate the generation and the composition of EB messages.

3.5.1 Security Requirements addressed

PMM_IF.4

3.5.2 Test preconditions

Car1 and car2 are on and car3 is off.

The contents of the "*intercar_data*" table have been deleted in every car's database.

There are no obstacles in the circuit.

3.5.3 Expected test results

Car1 generates emergency break messages at least every 0,5 seconds. The messages are composed of the vehicle unique identifier, EB identifier and timestamp with not null values.

When car1 is in emergency break state, a zero speed is shown in the Dashboard for it.

3.5.4 Criteria for evaluating results

The test duration is 20 seconds.

The data interchanged between car1 and car2 are displayed in the Dashboard.

An EB message is generated at least every 0,5 seconds.

We will check in car1's local database that the contents of the "*intercar_data*" table contain the emergency break messages sent.

3.5.5 Test Procedure

The following steps will be carried out:

- Put an obstacle (e.g. a cardboard box) 2 metres ahead of car1.
- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to the car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open the car1 database copied
- Verify the EB messages generated by car 1, their frequency, and their contents using the SQL:

```
Select * from intercar_data where source = 1 and destination = 2
                    and emergency_brake=1;
```

3.5.6 Test Results

Car1 generates EB messages every 0,5 seconds from when it breaks the safety distance and the messages contain the required information (id of the vehicle, EB identifier and timestamp) but also have more information as speed, distance and direction.

When car1 is in emergency break state, a zero speed is shown in the Dashboard for it, and for car2 a zero value is displayed in the field that shows the speed received from car1.

Status: PASSED

3.6 PMM_IF.4_TC2

Test case to validate the sending of EB messages.

3.6.1 Security Requirements addressed

PMM_IF.4

3.6.2 Test preconditions

All the cars are on.

The contents of the "*intercar_data*" table have been deleted in every car's database.

3.6.3 Expected test results

All the EB messages generated by car1 are sent to car2 and car3.

3.6.4 Criteria for evaluating results

The test duration is the time needed by car1 to reach the end of the circuit.

The data interchanged between car1 and car2, and between car2 and car3, are displayed in the Dashboard.

Car EB messages must be generated at least every 0,5 seconds from when the leader breaks the safety distance.

We will check the contents of the "intercar_data" table in car1's local database that should contain the EB messages sent.

The Dashboard shows a zero speed for each car in EB status, and for its follower a zero speed is displayed in the field that shows to the speed received from the preceding vehicle.

3.6.5 Test Procedure

The following steps will be carried out:

- Put an obstacle (e.g. a cardboard box) 20 cm behind car3.
- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to the car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop
- Copy the car3 database in /odroid/sparta/ flask_server/histo_3.db to the testing laptop

- Run the tool DB Browser for SQLite in the testing laptop and open histo_1.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 1 and
                emergency_brake=1;
```

- Run another instance of the tool DB Browser for SQLite in the testing laptop and open histo_2.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 2 and
                emergency_brake=1;
```

- Run another instance of the tool DB Browser for SQLite in the testing laptop and open histo_3.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 3 and
                emergency_brake=1
```

- To verify that the EB messages sent by car1 are received by car2 and car3 check that the results of the sentence
  `Select * from intercar_data where source=1 and emergency_brake=1`
  against the three databases are the same.
- To verify that the EB messages sent by car2 are received by car1 and car3 check that the results of the sentence
  `Select * from intercar_data where source=2 and emergency_brake=1`
  against the three databases are the same.
- To verify that the EB messages sent by car3 are received by car1 and car2 check that the results of the sentence
  `Select * from intercar_data where source=3 and emergency_brake=1`
  against the three databases are the same.

3.6.6 Test Results

All the EB messages generated every 0,5 seconds by car1 are sent to car2 and car3.

When car1 is in emergency break state, a zero speed is shown in the Dashboard for it, and for the car2 a zero value is displayed in the field that shows the speed received from car1.

Status: *PASSED*

3.7 PMM_IF.5_TC1

Test case to validate the reception of incoming messages from VCM and its composition.

3.7.1 Security Requirements addressed

PMM_IF.5

3.7.2 Test preconditions

Only car1 is on.

There are no obstacles in the circuit.

The contents of the "*car_update*" table and "*distance*" table have been deleted.

3.7.3 Expected test results

The TOE receives the data from the sensors.

Messages transmitted contain the following (not null) data: Speed, Direction, Geo-Position, and Gap to the next vehicle and distance to the edges of the lane.

3.7.4 Criteria for evaluating results

The test duration is 1 minute.

Data from the sensors must be received at least 0,2 seconds.

Car1 speed and distance are displayed in the Dashboard.

All the data coming from sensors are stored in a local the database in each car. We will check in car1's local database that the contents of the "*car_update*" table contain the speed and direction received from the ECU and the contents of the "*distance*" table to check the distances to any object received from the ultrasonic sensor.

3.7.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Place car1 on the circuit.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to car1.
- Run the platooning software in car1 using the *SSH* connection.
- Press the acceleration trigger of the remote control of the car1 and do not release it for the duration of the test.
- Check in the Dashboard the speed and distance of the car1
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open the histo_1.db copied
- Verify the sensor messages of car1, their frequency, and their contents using these SQL:

```
Select * from car_update" and "Select * from distance;
```

3.7.6 Test Results

Data from the sensors is received every 0,2 seconds

Status: *PASSED WITH DEVIATION*

Deviations from test procedure

The cars do not have GPS, VCM messages do not contain Geo-Position.

The distance to the edges is not necessary for the car's lane keeping algorithm; therefore VCM messages do not neither contain this data.

3.8 PMM_IF.6.1_TC1

Test case to validate the outgoing information from the TOE to VCM and its content.

3.8.1 Security Requirements addressed

PMM_IF.6

3.8.2 Test preconditions

Car1 and car2 are on. Car3 is off.

There are no obstacles in the circuit.

The contents of the "*car_update*" table and "*distance*" table have been deleted.

3.8.3 Expected test results

Car2 changes its speed according to the HB messages received from car1.

3.8.4 Criteria for evaluating results

Car2 speed and distance to car1 are displayed in the Dashboard and also stored in its own local database

3.8.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Place the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,3 m/s speed to car1.
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Increase 0,1 m/s the speed of car1 every 5 seconds, using the command to set speeds, until a speed of 1,0 m/s is reached.
- Verify in the Dashboard that the car2 speed also increases.
- Decrease 0,1 m/s the speed of car1 every 5 seconds, using the command to set speed, until it a speed of 0 m/s is reached.
- Verify in the Dashboard that the car2 speed also decreases.
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop.
- Run the tool DB Browser for SQLite in the testing laptop and open the histo_2.db copied.
- Check that the shift pattern of the speed of car2 is the same as that of car1 using these SQL:

```
Select date_inserted, speed from car_update order by date_inserted
ASC;
```

3.8.6 Test Results

Car2 changes its speed according the speed received from car1 to keep the safety distance to car1. This means that the TOE sends a message to the VCM, which acts on the car engine.

Status: PASSED WITH DEVIATIONS

Deviations from test procedure

The direction is calculated by the car's lane keeping algorithm and its only can be read, therefore the TOE does not send direction information to VCM.

3.9 PMM_PC.1_TC1

Test case to validate that the TOE accepts data incoming from the VCS only if the data passes all plausibility checks defined.

3.9.1 Security Requirements addressed

PMM_PC.1

3.9.2 Test preconditions

Car1 and car2 are on. Car3 is off.

There are no obstacles in the circuit.

3.9.3 Expected test results

Car2 accepts all the data incoming from the VCS.

3.9.4 Criteria for evaluating results

The test duration is 2 minutes.

The data interchanged between car1 and car2 are displayed in the Dashboard.

The number of wrong messages detected by car2 that are displayed in the Dashboard must be 0 at the end of the test.

3.9.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Place the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to the car1 (leader).

- Run the platooning software in each car involved in the test using the SSH connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Increase in 0,1 m/s the speed of car1 every 10 seconds, using the command to set speeds, until a speed of 1,0 m/s is reached.
- Verify in the Dashboard that the car2 speed also increases.
- Decrease in 0,1 m/s the car1 speed every 10 seconds, using the command to set speeds, until a speed of 0,3 m/s is reached.
- Verify in the Dashboard that the number of wrong messages detected by car2 is 0.

3.9.6 Test Results

All the messages sent by car1 pass the plausibility checks defined and enter in car2 CACC.

At the end of the test, the number of wrong messages detected by car2 shown in the Dashboards is 0.

Status: *PASSED.*

3.10 PMM_PC.1_TC2

Test case to validate that the TOE accepts data incoming from the VCS only if the data passes all the plausibility checks defined.

3.10.1 Security Requirements addressed

PMM_PC.1

3.10.2 Test preconditions

Car1 and car2 are on. Car3 is off.

There are no obstacles in the circuit.

3.10.3 Expected test results

Car2 refuses the data incoming from the VCS when the incoming speed value deviates 10% from the average of the last 3 speed values received from car1.

3.10.4 Criteria for evaluating results

The test duration is 1 minute.

The data interchanged between car1 and car2 are displayed in the Dashboard.

The number of wrong messages detected by car2 that are displayed in the Dashboard must be greater than 0 at the end of the test.

3.10.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Place the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- At second 15, start a message injection attack using the Dashboard from car1 to car2 with 0,9 m/s speed value.
- Verify in the Dashboard that the car2 speed does not change when car1 is under attack and the number of wrong messages detected by car2 is increasing.

3.10.6 Test Results

All the messages sent by car1 pass the plausibility checks defined and enter in car2 CACC before the attack starts. After the attack starts, all the messages are refused by car2 and car2 does not change its speed.

The number of wrong messages detected by car2 that are displayed in the Dashboard is greater than 0 at the end of the test.

Status: *PASSED.*

3.11 PMM_PC.3_TC1

Test case to validate that the Safety and Security Platooning Management Module informs about the Failed Plausibility Checks detected.

3.11.1 Security Requirements addressed

PMM_PC.3

3.11.2 Test preconditions

Car1 and car2 are on. Car3 is off.

There are no obstacles in the circuit.

3.11.3 Expected test results

The Dashboard shows the number of wrong messages detected by car2.

3.11.4 Criteria for evaluating results

The test duration is 1 minute.

The data interchanged between car1 and car2 are displayed in the Dashboard.

The number of wrong messages detected by car2 are displayed in the Dashboard.

The number of wrong messages detected must increase since the message injection attack (at second 15) is started until the end of the test (at second 60).

3.11.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the the testing laptop to establish *SSH* connections to the cars involved in the test.
- Place the cars on the circuit in the right order according their role with 50 cm of separation between each other.
- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,5 m/s speed to car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- At second 15, start a message injection attack using the Dashboard from car1 to car2 with 0,9 m/s speed value.
- Verify in the Dashboard that the number of wrong messages detected by car2 is increasing when car1 is under attack.

3.11.6 Test Results

During the attack, the number of wrong messages detected by car2 displayed in the Dashboard increases constantly 1 by 1 until the end of the test.

Status: *PASSED.*

3.12 PMM_VCS-HPC.1_TC1

Test case to validate the maintenance of a heart-beat data history.

3.12.1 Security Requirements addressed

PMM_VCS-HPC.1

3.12.2 Test preconditions

All the cars are on.

There are no obstacles in the circuit.

The contents of the "*intercar_data*" table have been deleted in every car's database.

3.12.3 Expected test results

All the HB messages generated by car1 are sent to car2 and car3 and are stored in their own local database in the table "*intercar_data*" table. Also, all the messages received by car1 from car2 and car3 are stored in the "*intercar_data*" table.

All the HB messages generated by car2 are sent to car1 and car3 and are stored in their own local database in the "*intercar_data*" table. Also, all the messages received by car2 from car1 and car3 are stored in "*intercar_data*" table.

All the HB messages generated by car3 are sent to car1 and car2 and are stored in their own local database in the "*intercar_data*" table. Also, all the messages received by car3 from car1 and car2 are stored in "*intercar_data*" table.

3.12.4 Criteria for evaluating results

The test duration is 2 minutes.

A HB must be generated every 0,5 seconds.

We will check the contents of the "**intercar_data**" table in car1's local database. The table should have a register for each HB message sent from car1 to the other two platoon members, a register for each HB message sent from car2 to car1 and a register for each HB message sent from car3 to car1.

The contents of the "*intercar_data*" table in every car's local database must be the same.

3.12.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Set a 0,4 m/s speed to the car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop
- Copy the car3 database in /odroid/sparta/ flask_server/histo_3.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open histo_1.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 1;
```

- Run another instance of the tool DB Browser for SQLite in the testing laptop and open histo_2.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 2;
```

- Run another instance of the tool DB Browser for SQLite in the testing laptop and open histo_3.db copied to verify the information of the HB messages received using the following sentence:

```
Select * from intercar_data where destination = 3;
```

- To verify that the HB messages sent by car1 are received by car2 and car3 check that the results of the sentence

```
Select * from intercar_data where source = 1
```

against the three databases are the same.

- To verify that the HB messages sent by car2 are received by car1 and car3 check that the results of the sentence

```
Select * from intercar_data where source = 2
```

against the three databases are the same.

- To verify that the HB messages sent by car3 are received by car1 and car2 check that the results of the sentence

```
Select * from intercar_data where source = 3
```

against the three databases are the same.

3.12.6 Test Results

Each vehicle stores in a local database, in a table called table “intercar_data”, all the HB messages sent to the other vehicles and received from the other vehicles.

Status: *PASSED*

3.13 PMM_VCS_HPC.2_TC1

Test case to validate that the TOE accepts HB messages consistent to the history.

3.13.1 Security Requirements addressed

PMM_VCS_HPC.2

3.13.2 Test preconditions

Car1 and car2 are on. Car3 is off.

There are no obstacles in the circuit.

3.13.3 Expected test results

car2 accepts all the HB messages coming from car1 when the incoming speed value deviates less than 10% the average of the last 3 speed values received from car1, otherwise it rejects them and the number of wrong messages detected by car2 that are displayed in the Dashboard increases.

3.13.4 Criteria for evaluating results

The test duration is 2 minutes.

The data interchanged between car1 and car2 are displayed in the Dashboard.

The number of wrong messages detected by car2 is displayed in the Dashboard.

3.13.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Place the cars on the circuit in the right order according to their role with 50 cm of separation between each other.

- Open the Dashboard (https://car1.sparta.org) in the testing laptop.
- Set a 0,4 m/s speed to car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Increase in 0,1 m/s the speed of car1 every 5 seconds until a speed of 0,8 m/s is reached.
- Verify in the Dashboard that the car2 speed increases and the number of wrong messages detected by car2 remains in 0.
- When the speed of car1 is 0,8m/s, start a message injection attack using the Dashboard from car1 to car2 with 1,2 m/s speed value for 15 seconds.
- Verify in the Dashboard that the car2 speed does not change (should be 0,8m/s approx.) when car1 is under attack and the number of wrong messages detected by car2 is increasing.
- Stop the attack using the Dashboard.
- Verify in the Dashboard that the car2 speed does not change (should be 0,8m/s approx.) when the attack is stopped, and the number of wrong messages detected by car2 stops changing.
- Decrease in 0,1 m/s the speed of car1 every 5 seconds until a speed of 0,3 m/s speed is reached.
- Verify in the Dashboard that the car2 speed decreases and the number of wrong messages detected by car2 remains fixed.

3.13.6 Test Results

All the messages sent by car1 are consistent to the history before the message injection attack is started.

All the messages sent by car1 are inconsistent to the history during the message injection attack and the number of wrong messages detected increases.

All the messages sent by car1 are consistent to the history after the message injection attack is stopped and the number of wrong messages detected does not change until the end of the test.

Status: *PASSED.*

3.14 PMM_VCS-SPC.1_TC1

Test case to validate the maintenance of a history of gaps to the vehicle in front measured by the sensors data history.

3.14.1 Security Requirements addressed

PMM_VCS-SPC.1

3.14.2 Test preconditions

All the cars are on.

There are no obstacles in the circuit.

The contents of the “distance” table have been deleted in every car’s database.

3.14.3 Expected test results

Each car stores in its local database, in table “distances”, all the distance measures obtained by its ultrasonic sensor and the data and time of the measure.

A measure should be taken every 0,2 seconds.

3.14.4 Criteria for evaluating results

The test duration is 1 minute.

We will check the contents of the “*distances*” table in every car’s local database. The table should have a register for each distance measured obtained by the ultrasonic sensor.

Measured distance data is in cm. A value of -2 will be returned if the measured distance is below the minimum specified sensor distance (20cm) and a value of -3 will be returned if the measured distance is above the maximum specified sensor distance (150cm).

The distance value registered for car 1 must be -3 and for car2 and car3 around 50.

3.14.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Set a 0,0 m/s speed to the car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop
- Copy the car3 database in /odroid/sparta/ flask_server/histo_3.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open every copied to verify the information about distance registered by each car in its database using the following sentence:

```
Select * from distance;
```

3.14.6 Test Results

Each vehicle stores all the distance measures obtained by the ultrasonic sensor in a local database, within the “*distances*” table.

A distance value is stored every 0,1 seconds in the distance table.

The distance value registered in the DB for car1 is -3 and for car2 and car3 is around 50 cm.

Status: *PASSED*

3.15 PMM_VCX-TPC.1_TC1

Test case to validate that the TOE shall be able to consult the TOE vehicle internal clock and the internal clock is synchronized with the clocks of other vehicles.

3.15.1 Security Requirements addressed

PMM_VCS-TPC.1, PMM_VCM-TPC.1

3.15.2 Test preconditions

All the cars are on.

3.15.3 Expected test results

All the cars have the same time in their Odroid boards.

3.15.4 Criteria for evaluating results

We will check in car1's local database that the contents of the "*intercar_data*" table contain the emergency break messages sent.

3.15.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Run the command:

  ```
  date +"%Y-%m-%d %H:%M:%S,%3N
  ```

  in the three cars at the same time using the *SSH* connections and verify that clocks are aligned.

3.15.6 Test Results

All the cars have the same date.

Status: *PASSED*

3.16 PMM_VCM-HPC.1_TC1

Test case to validate the maintenance of a history of VCM messages with data related to car speed and direction.

3.16.1 Security Requirements addressed

PMM_VCM-HPC.1

3.16.2 Test preconditions

All the cars are on.

There are no obstacles in the circuit.

The contents of the "*car_update*" table are deleted in every car's database.

3.16.3 Expected test results

Each car stores in its local database, in the "*car_update*" table, data coming from the car ECU about the current speed, the current direction (steering angle) and the time of the data.

A VCM message should be stored every 0,2 seconds.

3.16.4 Criteria for evaluating results

The test duration is 1 minute.

We will check the contents of the "*car_update*" table in every car's local database.

3.16.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Set a 0,5 m/s speed to the car1 (leader).
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.
- Copy the car1 database in /odroid/sparta/ flask_server/histo_1.db to the testing laptop
- Copy the car2 database in /odroid/sparta/ flask_server/histo_2.db to the testing laptop
- Copy the car3 database in /odroid/sparta/ flask_server/histo_3.db to the testing laptop
- Run the tool DB Browser for SQLite in the testing laptop and open every copied to verify the information about speed and direction registered by each car in its database using the following sentence:

```
Select * from car_update;
```

3.16.6 Test Results

Each vehicle stores in its local database, in the "*car_update*" table, every 0,2 seconds its current speed and steering angle and the time of the data.

Status: *PASSED*

3.17 FPT_ITC.1_ITI.1_TC1

Test case to validate the protection and integrity of all TSF data transmitted from the TSF to another trusted IT product from unauthorised disclosure during transmission. TSF data transmitted from the TSF to another trusted IT product is the following:

- timestamp: message time
- source: car sender id
- destination: car receiver id
- speed: sender car speed
- distance: distance to previous car of the car sender
- direction: steering angle of the car sender
- emergency_brake: boolean indicating if the car sender is in emergency mode
- emergency_break_id: emergency break id:

3.17.1 Security Requirements addressed

FPT_ITC.1, FPT_ITI.1

3.17.2 Test preconditions

All the cars are on.

There are no obstacles in the circuit.

3.17.3 Expected test results

All the HB messages generated by car1 are sent to car2 and car3 establishing a secure TLS 1.3 connection with each of them.

3.17.4 Criteria for evaluating results

The test duration is 1 minute.

We will check all the wifi traffic collected to verify that the HB messages interchanged between car1 and car2 and car3 are encrypted.

TLS ensures that between the processes of encrypting, transmitting, and decrypting the data, no information is lost, damaged, tampered with, modified or falsified.

3.17.5 Test Procedure

The following steps will be carried out:

- Connect the testing laptop with the debugging tools to the platooning Wifi network.
- Use the testing laptop to establish *SSH* connections to the cars involved in the test.
- Put the cars on the circuit in the right order according to their role with 50 cm of separation between each other.
- Set a 0,5 m/s speed to car1 (leader).
- Capture Wi-Fi traffic using the testing laptop's Wireshark and save it in a pcap file.
- Run the platooning software in each car involved in the test using the *SSH* connection.
- Press, at the same time, the acceleration trigger of the remote control of the cars involved in the test and do not release it for the duration of the test.

- Open the pcap file with Wireshark to verify that the messages send from car1(192.168.43.101) to car2 (192.168.43.101) are encrypted using the following filter:

```
ip.src= 192.168.43.101 and ip.dst= 192.168.43.102 and
        http.request.method == 'POST'
```

- Verify also that TLSv1.3 is displayed in the "Protocol" column

3.17.6 Test Results

All the HB messages generated by car1 are sent to car2 and car3 and their content cannot be seen from the testing laptop because they are encrypted.

TLS 1.3 connections are working properly, therefore, in addition to encryption.

Requirements FPT_ITC.1.2 and FPT_ITI.1.1 are tested by analysis as following: we have data integrity and unmodifiable data implicitly by using TLS and no test case is needed to verify the integrity or data modification detection of all TOE Security Functionality data transmitted between the cars.

Status: *PASSED*

Chapter 4 Test Summary Coverage

This chapter shows the completeness of tests coverage: each test covers at least one requirement, and every requirement has been tested at least by one test.

The following Table 2 demonstrates that each test cover at least one requirement.

Test ID	Requirement code	Results (including section reference)	Notes
PMM_IF.1_TC1	PMM_IF.1	PASSED WITH DEVIATIONS (3.1.6)	Geo-Position missing due to the absence of GPS in the cars
PMM_IF.1_TC2	PMM_IF.1	PASSED (3.2.6)	--
PMM_IF.2_TC1	PMM_IF.2	PASSED WITH DEVIATIONS (3.3.6)	The cars do not have GPS, HB messages do not contain Geo-Position. The digitally signed certificate is stored in every car, HB messages don't contain the digitally signed certificate
PMM_IF.3_TC1	PMM_IF.3	PASSED WITH DEVIATIONS (3.4.6)	The digitally signed certificate is stored in every car, HB messages don't contain the digitally signed certificate
PMM_IF.4_TC1	PMM_IF.4	PASSED (3.5.6)	--
PMM_IF.4_TC2	PMM_IF.4	PASSED (3.6.6)	--
PMM_IF.5_TC1	PMM_IF.5	PASSED WITH DEVIATIONS (3.7.6)	Geo-Position missing due to the absence of GPS in the cars
PMM_IF.6_TC1	PMM_IF.6	PASSED WITH DEVIATIONS (3.8.6)	The direction is calculated by the car's lane keeping algorithm and its only can be read therefore the TOE doesn't send direction information to VCM
PMM_PC.1_TC1	PMM_PC.1	PASSED (3.9.6)	--
PMM_PC.1_TC2	PMM_PC.1	PASSED (3.10.6)	--
PMM_PC.3_TC1	PMM_PC.3	PASSED (3.11.6)	--
PMM_VCS-HPC.1_TC1	PMM_VCS-HPC.1	PASSED (3.12.6)	--
PMM_VCS-HPC.2_TC1	PMM_VCS-HPC.2	PASSED (3.13.6)	--
PMM_VCS-SPC.1_TC1	PMM_VCS-SPC.1	PASSED (3.14.6)	--

Test ID	Requirement code	Results (including section reference)	Notes
PMM_VCX-TPC.1_TC1	PMM_VCS-TPC.1, PMM_VCM-TPC.1	PASSED (3.15.6)	--
PMM_VCM-HPC.1_TC1	PMM_VCM-HPC.1	PASSED (3.16.6)	--
FPT_ITC.1_ITI.1_TC1	FPT_ITC.1, FPT_ITI.1	PASSED (3.17.6)	FPT_ITC.1.2 and FPT_ITI.1.1 have been tested by analysis

Table 2: Test Summary Coverage (Tests vs Requirements)

The following Table 3 demonstrates that each requirement has been verified at least through one test.

Requirement code	Test ID	Results (including section reference)	Notes
PMM_IF.1	PMM_IF.1.1_TC1,	PASSED WITH DEVIATIONS (3.1.6)	The HB messages contain extra data
	PMM_IF.1.1_TC2	PASSED (3.2.6)	--
PMM_IF.2	PMM_IF.2.1_TC1	PASSED WITH DEVIATIONS (3.3.6)	The cars do not have GPS, HB messages do not contain Geo-Position. The digitally signed certificate is stored in every car, HB messages don't contain the digitally signed certificate
PMM_IF.3	PMM_IF.3.1_TC1	PASSED WITH DEVIATIONS (3.4.6)	The digitally signed certificate is stored in every car, HB messages don't contain the digitally signed certificate
PMM_IF.4	PMM_IF.4.1_TC1,	PASSED	--
	PMM_IF.4.1_TC2	PASSED (3.6.6)	--
PMM_IF.5	PMM_IF.5.1_TC1	PASSED WITH DEVIATIONS (3.7.6)	Geo-Position missing due to the absence of GPS in the cars
PMM_IF.6	PMM_IF.6.1_TC1	PASSED WITH DEVIATIONS (3.8.6)	The direction is calculated by the car's lane keeping algorithm and its only can be

Requirement code	Test ID	Results (including section reference)	Notes
			read therefore the TOE doesn't send direction information to VCM
PMM_PC.1	PMM_PC.1.1_TC1,	PASSED (3.9.6),	--
	PMM_PC.1.1_TC2	PASSED (3.10.6)	--
PMM_PC.3	PMM_PC.3.1_TC1	PASSED (3.11.6)	--
PMM_VCS-HPC.1	PMM_VCS-HPC.1.1_TC1	PASSED (3.12.6)	--
PMM_VCS-HPC.2	PMM_VCS-HPC.2.1_TC1	PASSED (3.13.6)	--
PMM_VCS-SPC.1	PMM_VCS-SPC.1.1_TC1	PASSED (3.14.6)	--
PMM_VCS-TPC.1	PMM_VCX-TPC.1.1_TC1	PASSED (3.15.6)	--
PMM_VCM-HPC.1	PMM_VCM-HPC.1.1_TC1	PASSED (3.16.6)	--
PMM_VCM-TPC.1	PMM_VCX-TPC.1.1_TC1	PASSED (3.15.6)	--
FPT_ITC.1	FPT_ITC.1_ITI.1_TC1	PASSED (3.16.6)	--
FPT_ITI.1	FPT_ITC.1_ITI.1_TC1	PASSED (3.17.6)	--

Table 3: Test Summary Coverage (Requirements vs Tests)

The following matrix (Table 4) shows the complete coverage between Security Functional Requirements and tests.

	PMM_IF.1	PMM_IF.2	PMM_IF.3	PMM_IF.4	PMM_IF.5	PMM_IF.6	PMM_PC.1	PMM_PC.3	PMM_VCS-HPC.1	PMM_VCS-HPC.2	PMM_VCS-SPC.1	PMM_VCS-TPC.1	PMM_VCM-HPC.1	PMM_VCM-TPC.1	FPT_ITC.1	FPT_ITI.1
PMM_IF.1_TC1	X															
PMM_IF.1_TC2	X															
PMM_IF.2_TC1		X														
PMM_IF.3_TC1			X													
PMM_IF.4_TC1				X												
PMM_IF.4_TC2				X												
PMM_IF.5_TC1					X											
PMM_IF.6_TC1						X										
PMM_PC.1_TC1							X									
PMM_PC.1_TC2							X									
PMM_PC.3_TC1								X								
PMM_VCS-HPC.1_TC1									X							
PMM_VCS-HPC.2_TC1										X						
PMM_VCS-SPC.1_TC1											X					
PMM_VCX-TPC.1_TC1												X		X		
PMM_VCM-HPC.1_TC1													X			
FPT_ITC.1_ITI.1_TC1															X	X

Table 4: Matrix of test coverage

Chapter 5 List of Abbreviations

Abbreviation	Translation
ACC	Adaptive Cruise Control
BSI	Bundesamt für Sicherheit in der Informationstechnik
CACC	Cooperative Adaptive Cruise Control
ECU	Electronic Control Unit
GPS	Global Positioning System
HB	Heartbeat
IT	Information Technology
NTP	Network Time Protocol
PCAP	Packet Capture
PMM	Platoon Management Module
SSH	Secure Shell
TC	Test Case
TLS	Transport Layer Security
TOE	Target Of Evaluation
TSF	TOE Security Functionality
VCM	Vehicle Communication System
VCS	Vehicle Control Module

Chapter 6 Bibliography

[1] SPARTA D5.2 Demonstrators specifications. January 2021.

[2] SPARTA D5.3 Demonstrator prototypes. January 2021.

[3] SPARTA D5.2 Appendix B Protection Profile for a Safety and Security Platooning Management Module, January 2021

[4] Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 5, April 2017. Part 1: Introduction and general model.

[5] Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 5, April 2017. Part 3: Assurance security components.

[6] Bundesamt für Sicherheit in der Informationstechnik (BSI) Guidelines for Developer Documentation according to Common Criteria Version 3.1 Version 1.0