POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell' informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master in

Computer Science and Engineering

On the Security of Connected Vehicles

Supervisor:

PROF. STEFANO ZANERO

Assistant Supervisor:

DAVIDE QUARTA

Master Graduation Thesis by:

STEFANO LONGARI

Student Id n. 859130

Academic Year 2017-2018

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

2g Second Generation

ack Acknowledgment

aes Advanced Encryption Standard

afdx Avionics Full-Duplex Switched Ethernet

asil Automotive Safety Integrity Level

at Attention

asl Automotive Security Level

avb Audio Video Bridging

c2x Car to Everything

can Controller Area Network

can-fd Controller Area Network with Flexible Data rate

ccu Communication Control Unit

cia Confidentiality Integrity Availability

csma ba Carrier Sense Multiple Access with Bitwise Arbitration

csma cd Carrier Sense Multiple Access with Collision Detection

dos Denial of Service

dsrc Dedicated Short Range Communication

ecu Electronic Control Unit

evita E-safety Vehicle Intrusion protected Applications

gecu Gateway Electronic Control Unit

gprs General Packet Radio Service

gsm Global System for Mobile communication

hsm Hardware Security Module

ids Intrusion Detection System

iot Internet of Things

lin Local Interconnect Network

mac Message Authentication Code

mitm Man In The Middle

most Media Oriented Systems Transport

obd-ii On Board Diagnostic

open One-Pair EtherNet

pid Parameter ID

pki Public Key Infrastructure

rpm Repetitions Per Minute

sdr Software Defined Radio

sms Short Message Service

v2v Vehicle to Vehicle

v2x Vehicle to Everything

ABSTRACT

Modern vehicles cannot be defined mere mechanical devices anymore. They are composed of up to two hundreds microcontrollers and computers, sensors and electronic actuators interconnected through on-board networks. The development of these units and their connection with the outside world through cellular data, bluetooth and other communication protocols led to major advancements in the safety and efficiency of the vehicle and will enhance these two property even more in the next years, but on the downside it introduced a range of new issues and risks derived from the reachability of the vehicle from remote. This led, not more than fifteen years ago, to the birth of the automotive security field, whose goal has been and still is to secure vehicles and their passengers.

We propose a survey of the whole automotive security field which will tackle all the main known issues of the new and developing automotive systems: we start from a threat model that helps us presenting the main security requirements that have to be enforced, and through those requirements we highlight the current state of the art of both the academic and market solutions that have been proposed, focusing on the feasibility of those which have not yet been implemented.

Finally, to complete the overview on the whole field, we propose a case study unit currently on the marketto analyze the issues relative to its design.

SOMMARIO

L'automobile è decisamente il mezzo di trasporto più usato al mondo e dalla sua nascita si è visto cambiare continuamente: da un lato ci sono state e sono tuttora in corso le migliorie continuamente apportate ai nuovi motori sia in termini di prestazioni che di efficienza, in aggiunta a tutte le nuove miscele di carburanti che portano a emissioni minori e più ecologiche. Considerando i materiali utilizzati, i nuovi copertoni e le nuove carrozzerie sono stati appositamente pensati per portare ad un' evoluzione della sicurezza che ha salvato migliaia di vite evitando o contenendo i numerosi incidenti.
Di sicuro oltre a tutte queste migliorie tradizionalmente correlate con il concetto di automobile negli ultimi 40 anni un altro ambito che ha fortemente influenzato lo sviluppo dei veicoli su ruote è stato quello dell'integrazione nel veicolo di hardware e software informatico. In un moderno veicolo possiamo infatti trovare milioni di righe di codice eseguite su decine di diverse centraline elettroniche di controllo (ECU). Queste ECU, nate come mezzi per rendere più sicura la guida tramite l'attivazione di misure quali airbag e ABS ormai sono utilizzate per ogni aspetto dell'automobile, dall'apertura elettronica alla frenata assistita all'assistenza nel mantenimento della corsia in autostrada.
Queste centraline di conseguenza sono direttamente responsabili di strumenti che proteggono l'incolumità dei passeggeri del veicolo, il che li rende elementi estremamente pericolosi se utilizzati per scopi malevoli. Nello sviluppo di tali centraline e delle varie reti che le connettono tra di loro all'interno dell'automobile ovviamente questo fattore è stato considerato come elemento di grande importanza e ci si è sempre assicurati che non potessero essere fonte di danno invece che di aiuto alla guida.
Lo sviluppo del settore delle telecomunicazioni che ha permesso,

negli ultimi trent'anni, la produzione di massa di tecnologie per connessioni senza fili quali GSM, Bluetooth, WiFi e così via, è stato poi applicato, approssimativamente dagli anni 2000, anche al settore automobilistico: Le case produttrici di automobili e centraline hanno potuto grazie all'ormai irrisorio costo di tali tecnologie implementare direttamente nella rete interna dell'auto dei mezzi di comunicazione con l'esterno. Queste nuove tecnologie hanno aumentato nuovamente sia la sicurezza fisica che le comodità accessibili dal veicolo, basti pensare a quanto sia comodo avere un navigatore GPS invece che dover studiare le mappe cartacee prima di intraprendere un viaggio.
Sfortunatamente tutte queste comodità hanno portato con loro anche un importante lato negativo che tuttora non è stato risolto: permettere connessioni da remoto alla rete interna dell'automobile che ne preserva l'affidabilità cambia completamente il modello di attacco relativo al veicolo. Se una volta un malintenzionato, vuoi per furto o per nuocere a qualcuno, doveva essere fisicamente a contatto con il veicolo ora non è più così. Elemento aggravato dal fatto che le reti interne dell'automobile non sono state progettate per essere sicure contro un attaccante e di conseguenza da questo punto di vista piene di falle intrinseche derivanti da protocolli ormai diventati standard.

Le minacce derivanti dai fattori appena descritti sono fortunatamente già state definite e sono stati presi in parte provvedimenti al riguardo, aumentando i livelli di sicurezza delle reti interne e usando un occhio di riguardo nell'implementazione delle varie centraline, cercando di renderle il più possibile sicure. Negli anni sono stati dimostrati vari tipi di attacchi che possono nuocere direttamente all'incolumità del passeggero del veicolo e sono state proposte varie soluzioni per poter bloccare tali attacchi, ma la strada da fare è sicuramente ancora molto lunga, soprattutto a causa del fatto che lo sviluppo del campo automotive non si è decisamente fermato negli ultimi anni, proponendo nuovi mezzi di comunicazione in particolare tra due veicoli adiacenti, in modo da potersi comunicare a vicenda incidenti, code, pericoli e informazioni di vario tipo.

Nello studio di questo ambito della sicurezza informatica uno degli elementi che rende più difficile informarsi e approfondire le conoscenze su specifici elementi che lo compongono è la mancanza, ad oggi, di uno studio a 360 gradi che permetta al lettore di avere un idea di quali sono tutti i problemi ad oggi conosciuti di questo campo e quali proposte sono già state fatte, verso cosa si stanno facendo nuovi passi e come e se si sta prendendo in considerazione la sicurezza informatica nel farli. Questo è ciò che ci proponiamo di fare in questa tesi:

Il primo passaggio è produrre un threat model che ci permetta di mettere in luce i requisiti di sicurezza dell'ambiente. Abbiamo riproposto un threat model precedentemente prodotto nel 2009 relativamente al campo V2X e lo abbiamo esteso considerando anche le reti interne del veicolo, concludendo che ci sono tre requisiti di sicurezza fondamentali a seconda dell'elemento considerato all'interno dell'automobile che sono autenticazione, anonimità e disponibilità dei dati.

Seguendo questi requisiti di sicurezza ci siamo quindi concentrati sullo studio delle proposte fatte sia dal mondo accademico che quello aziendale relative all'ambito della sicurezza automotive, e siamo arrivati a proporre uno studio completo delle varie proposte analizzando se e come riescono ad assicurare tali requisiti.
Lo studio è molto incentrato sulla fattibilità delle varie proposte, sopra tutte quelle derivanti dal mondo accademico: per ogni proposta vengono presi in considerazione, oltre ai benefici, la complessità di applicazione, l'effettiva sicurezza che la soluzione può apportare e i costi da sostenere per applicarla. Non ci siamo permessi di dare giudizi finali sull'effettiva validità di una proposta, benché abbiamo espresso, dove necessario, i dovuti dubbi.

Per concludere la tesi abbiamo proposto un caso di studio: in tutta l'analisi precedente manca un elemento"umano", ovvero la possibilità di applicare in modo scorretto un metodo che di per sé non dovrebbe avere falle. Abbiamo voluto fare la valutazione della sicurezza di una centralina elettronica di controllo per studiarne il codice, l'hardware e

le connessioni. Ovviamente di per sé lo studio della singola centralina non porta a dare conclusioni generali riguardo ai livelli di sicurezza applicati da tutte le aziende ma può fare da esempio per sapere su cosa concentrarsi nella ricerca di elementi fallaci.

1

INTRODUCTION

1.1 THE CONNECTED CAR ENVIRONMENT

Cars and road vehicles in general are by far the most common mean of transportation in our society: In 2017 97,3 million cars and commercial vehicles were built worldwide[1] and in 2015 there were 1,78 billion vehicles circulating on Earth.
The automotive industry has massively developed in the last 40 years, transforming what once was merely a mechanical mean of transportation composed of an engine and wheels into a complex instrument in which networks of electronic devices have become of standard and broad use. These electronic devices, which are commonly known as Electronic Control Unit (ECU)s, were first born in the late seventies with the objective to control the safety systems like airbags, but since then they have become much more widespread: automotive manufacturers started to design different units through which they developed new safety systems, like lane assists and collision avoidance systems, or what helps and enjoys the driver like cruise control and electronic control of the engine settings to enable different kinds of driving experiences. The development of this units led to the implementation, in newest high end cars, of up to two hundred ECUs.
Among these devices many required to be connected with each other and with sensors or actuators, which were not physically adjacent. To enable communication multiple network protocols were proposed over the years, but one in particular have become the de facto standard in the whole automotive field, which is the Controller Area Network (CAN) protocol. The motivations behind this choice were mainly three: It was compliant with the strict requirements derived by

Figure 1.1: A Chevrolet Reverse 2016 implementing the new version of the OnStar system by General Motors

the safety critical messages it had to transmit, it was of extremely simple implementation and it was cheap, as it is implemented through a simple bus composed of two lines.

Later, with the development of new cellular networks and cheaper devices for wireless communication, the automotive industry found new applications for these technologies applying them inside vehicles: The first remote connection device installed in a vehicle was the OnStar system by General Motors which in 1996 was installed on high end cars to request emergency services through a call directly to the call centers of General Motors. In the years the OnStar system developed to enable bidirectional data transfer over cellular connections and similar systems are installed also in smaller and cheaper cars.

With the opening of GPS to the public there was a boom in the development of navigation devices, which have been implemented both for the end user experience and both by fleet management services for companies. Navigation devices are nowadays also paired with cellular connections through which new maps can be installed and paths can be calculated in remote. In the latest years GPS devices are being replaced with bluetooth connection to the driver's smartphone that through services like Android auto and Apple CarPlay preinstalled on the car enable the use of the navigation system, music and mes-

sages of the smartphone directly on the front screen of the vehicle.
The new communication means enabled also the new car sharing services that are nowadays moving toward a fully smartphone-based car access which requires cellular connection on the vehicle to open the car and enable driving. It has been estimated that the amount of car sharing services users are currently around 15 millions and that will increase by another 11 millions by 2020[2].
Finally, we are moving into the era of Vehicle to Everything (V2X) where vehicles will be able to send and receive information about the status of the roads, of accidents and queues to other vehicles and to infrastructures specially built for them. In such an environment almost every vehicle will possess one or more devices that will enable transmission of data.

1.2 CONSEQUENCES OF THE INTERCONNECTED VEHICLE

The on-board networks of the car, like CAN, have not been designed to be secure networks. They were though to be mere isolated networks that did not have necessity to be secure and attack proof, as the potential attacker of a car, either if his aim was to steal or to damage the car or the passengers, would have needed to be physically in the car to tamper the on-board networks and if access to the car was already obtained there were easier ways for the attacker to both steal the car or endanger the passengers.
With the birth of the connected vehicle the attacker model changed as he currently have the possibility to connect to devices inside the car from remote using one of the connected ECUs as a mean to reach the internal network of the car, not requiring him to be physically in or near the vehicle.
All the examples proposed above of connected devices inside the on-board networks have therefore become new attack surfaces of the car, exponentially increasing the potential attacks that can be performed.

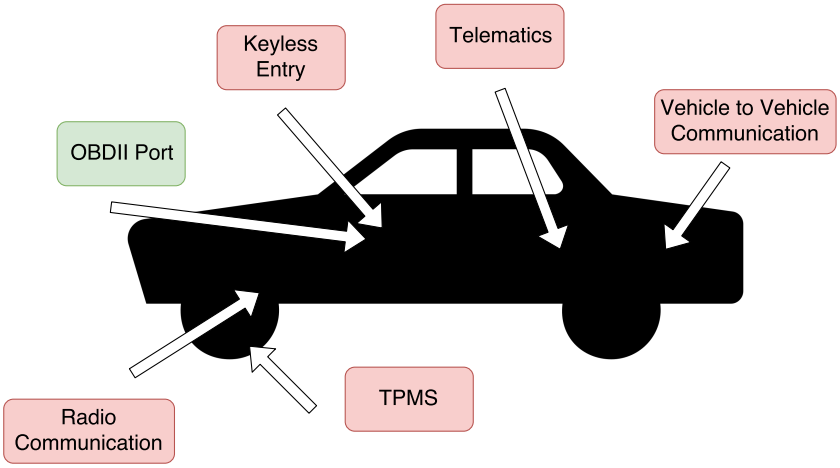

Figure 1.2: The attack surfaces of a connected car: in green the OBD-II port which once was the only possible entrance to the network, in red all the new surfaces deriving from the new connected devices

The danger behind connected automotive networks is already known and recognized as research-worthy. The first important analyses of the feasibility of these attacks have been made in 2011 by Checkoway et al.[3] which listed the different kinds of attacks feasible by remote and exploited the protocols used for the cellular connection of a real car, creating a remote access to the CAN bus. After them different researchers tried to exploit real cars, which lead to automotive companies trying to cope by enhancing security on connected ECUs. in 2014 Miller and Valasek published a report which listed more than twenty car models with their respective attack surfaces[4]. This paper was not well received by automotive companies, in fact it was answered saying that it was not possible to hack new high end cars thanks to the new security mechanisms that were implemented. In 2015 Miller and Valasek published another report[5] in which they showed how, treating the car ECUs as computers and attacking them with different means it was possible to gain full control of the CAN bus of a new high end car from remote. The report showed that the exploits could have potentially been done on every car that had the same connected ECU with little more effort.

Figure 1.3: Image of Miller and Valasek on the display of their test 2014 Jeep Cherokee after they hacked the infotainment system.

1.3 THE CONTRIBUTION OF THIS WORK

Since the first papers by Checkoway et al. the interest in the automotive security field has increased, leading automotive companies to consider as much as possible security in the new developed systems, but up until now no in depth reviews of the overall solutions have been produced. As the subfields of automotive security are various and differ significantly in the environment they are considering, our objective is to create an in depth review of all the automotive security field with focus on giving the reader a clear idea of the different directions in which the field is developing and why.

As this is an extremely new field it is hard to have a clear idea of what is the current status of the automotive environment and what is instead the status of the research around it, which elements of the already implemented systems fall short in relation to security and for what reason, and which proposals may be useful to solve these shortcomings.

To structure the paper we first, in chapter II, describe the notions that are not of common knowledge but that are required to understand the paper. Then in chapter III present the threat model of the automotive security environment describing which are the security

requirements that can be derived from it, and in chapter IV we use those requirements to present the actual survey on the automotive security field.
Chapter V is a case study: We propose the security assessment of an ECU to study the elements that were not considered in the survey. Finally in Chapter VI we give our conclusions, proposing the future works on the topic.

In summary, this thesis gives the following contributions:

- It proposes an adaptation of an old threat model for the current automotive environment, proposing three security requirements for the field.
- It describes in depth the current automotive security field, considering all its different facets and analyzing similar solutions to highlight the elements in which they differ and why.
- It describes the security assessment on a specific ECU to highlight risks discussed in the above sections.

2

BACKGROUND

Before stepping in the actual subjects of the thesis we propose a brief description of the set of topics that are required to understand the next chapters and that may not be of common knowledge in the computer science field.

2.1 CONTROLLER AREA NETWORK

The CAN is an asynchronous real-time bus standard released in 1986 by Robert Bosch Gmbh and designed for allowing communication of multiple microcontrollers specifically for the automotive market. It is currently the de facto standard for communication between safety related ECUs in vehicles as here is at least one CAN bus almost in every car on the market.

Its main properties for which it has been chosen as the current standard are:

- prioritization of messages;
- guaranteed latency times;
- broadcasting of the messages;
- multimaster;
- error detection and signaling;
- autonomous switching off of defect nodes;

CAN refers to the Physical and Data Link layers of the OSI model. In relation to the specification of the Physical layer CAN uses a differential signal with two states that can be written on the bus, dominant

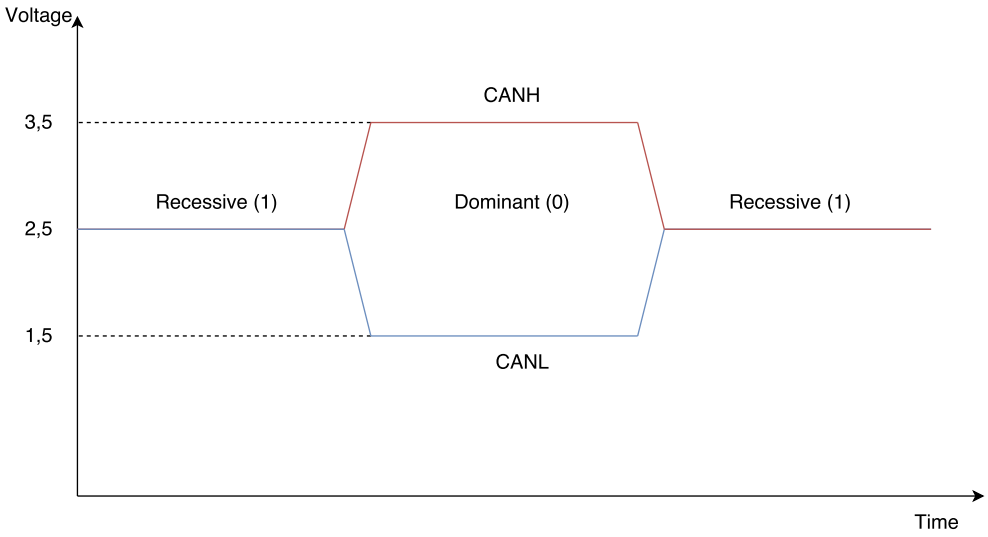

Figure 2.1: The signaling states from CAN electrical levels.

and recessive: the dominant one always overwrites the recessive. The difference the two states depends from the differential voltage between the two CAN lines,CAN-H and CAN-L as in figure2.1.

The maximum transfer rates supported by the standard (ISO 11898-2[6]) reach 1Mbps but due to the physical required time for the signal to travel the transmission medium from one end to the other and the requirement for each node to be synchronized transfer speed are restricted by cable lengths, which in standard cars are less than 100 meters long, leading to 500kbps of achievable transfer rates.

Regarding the Data Link layer the data frame can be either of standard or extended format and is composed of:

- Start of Frame: 1 dominant bit to allow hard synchronization of all nodes;
- Arbitration Field (standard): 11 bits of frame identifier, 1 dominant bit of remote transmission request;
- Arbitration Field (extended): 11 bits of base frame identifier, 1 recessive bit of substitute remote request, 1 recessive bit of identifier extension, 18 bits of extended frame identifier, 1 dominant bit of remote transmission request;

Figure 2.2: The CAN data frame with an 11 bit identifier.

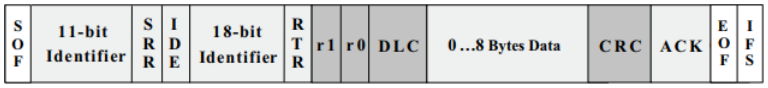

Figure 2.3: The CAN data frame with an 29 bit identifier.

- Control Field (standard): 1 bit of identifier extension, 1 dominant reserved bit, 4 bits of data length code;
- Control Field (extended): 2 dominant reserved bits, 4 bits of data length code;
- Data Field: up to 8 bytes carrying the actual data;
- CRC Field: 15 bits of CRC sequence, 1 recessive bit of CRC delimiters;
- ACK Field: 1 bit of ACK slot, 1 recessive bit of ACK delimiter;
- End of Frame: 7 recessive bits;

The error frame is sent whenever a bit, stuff, CRC, form or acknowledgment error is detected. It is composed of:

- Error Flag: 6 bits;
- Error Delimiter: 8 recessive bits;

The error frame overwrites the data frame as soon as the error is detected, alerting of the invalidity of the data frame.

The bus arbitration derives from the choice of a publish and subscribe mechanism instead of the traditional sender and receiver one (as it is visible in the data frame there is no indication of sender and receiver). When two or more nodes start transmitting in the same idle bus time they generate a collision which is solved in the following way: all nodes, even when transmitting a packet, read the signal

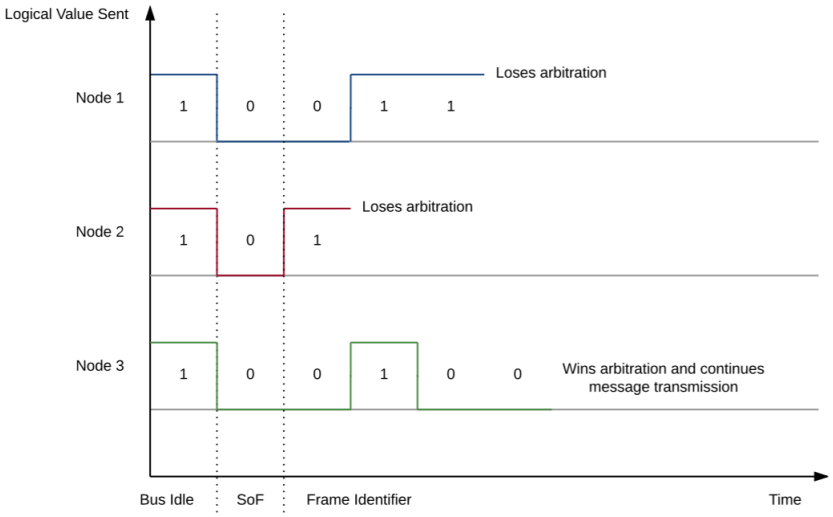

Figure 2.4: An example of arbitration process: node2 loses first at the third bit, while node 1 loses at fifth.

from the bus and eventually compare it to the packet they are sending. As the dominant bit (0) overwrites the recessive one (1) if the identifier of the read packet is different from the one being sent it means that another node is writing an identifier that has an earlier dominant bit and won the arbitration, therefore the first node stops sending its packet and retries it as soon as the bus turns idle. Therefore the identifiers can be set so that packets that should be sent with higher urgency have precedence on the others.

2.2 V2X COMMUNICATION

V2X also known as Car to Everything (C2X) communication is the transmission of information from a vehicle to an entity that may affect the vehicle and vice versa. In the latest years the developing of the automotive industry is leading toward interconnected vehicles that are able to communicate with each other (V2V), with infrastructures (V2I), with networks (V2N) and with pedestrians (V2P) through different means.

The focus of V2X is to increase the autonomy of the vehicle to increase safety, decrease traffic and lower emissions. To obtain these

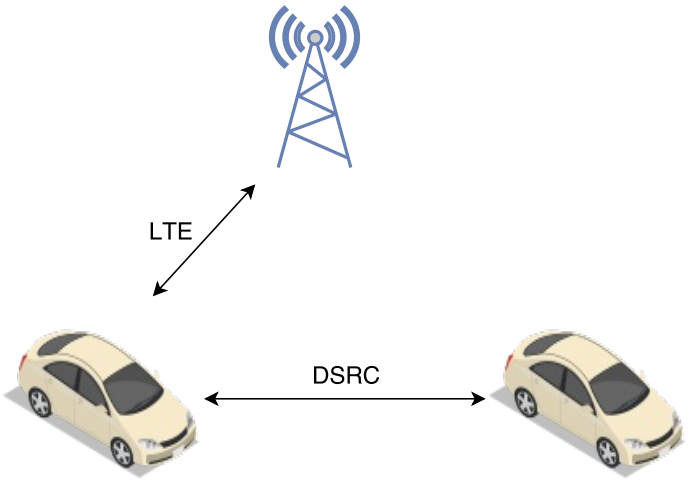

Figure 2.5: A representation of the communication between cars and with infrastructures.

objectives all vehicles in the next years will be using different kinds of means of communication to interact with the environment around it increasing the awareness of the driver over his line of sight, notifying of collisions, accidents, intersections, emergency vehicles, queues, parking lots and others.

Communicating with other cars will employ short range communication means like DSRC (IEEE 1609 family of standards“WAVE"[7]) or LTE direct that work even without cellular signal for the communication between adjacent vehicles, while for communication with infrastructures or remote vehicles traditional high speed cellular data like LTE or the new standards of 5G will be used.

2.3 AT COMMANDS

Attention (AT) Commands, also known as“Hayes Command Set”are a command language originally developed for Hayes Smartmodem 300 in 1981. They consist of short text strings which can be combined to create commands for dialing, hanging up and changing parameters of the connection which are used in the vast majority of dial-up

AT+CPIN=1234	Enter PIN Code
AT+CLCK="SC",0,"1234"	Remove PIN Code
AT&V	Status
AT+COPS=?	List available networks
AT+CSQ	Get signal strength

Table 2.1: Example of GSM related AT Commands

modems, often with extensions that were built in later years to control additional functionalities.
They have been created to fulfill the necessity of an automatic way to switch the modem between data mode (in which a modem sends data to another remote one) and command mode (in which data have to be considered as commands for the local modem and should not be sent to the remote one).

Specifically in relation to mobile communication the ETSI GSM 07.07[8] specifies the AT Commands for controlling a Global System for Mobile communication (GSM) phone or modem and the ETSI GSM 07.07[9] specifies the AT Commands for controlling the Short Message Service (SMS) feature of GSM. In table2.1 some examples of GSM related AT Commands.

3

THREAT MODEL OF THE AUTOMOTIVE CYBERSECURITY ENVIRONMENT

Creating a survey on the security of the automotive environment would be quite hard to understand if we didn't first outline which are the threats that the environment should defend from and why. We are going to present a threat model starting from already produced automotive models[10][3] but updating it to the current status of the automotive environment. To do so we proceed with the following steps: first we describe a general automotive on-board network to outline which is the system we are analyzing, then we consider the threats to that system and the known or considered attacks we need to defend from and finally processing these information we derive the security requirements for the automotive network. These steps are extremely useful to outline the reasons behind the solutions proposed in the next chapter.

3.1 SYSTEM UNDER ANALYSIS

The automotive on-board system is composed of one or multiple internal networks, each one connected with the others. If more than one network is present, usually each one is designated to a specific set of functions: one for safety critical ECUs, one for the instrumentation, dashboard, lights, one for the infotainment systems. To connect these networks there usually is one single main module (composed of the head unit and the communication control unit), but in newest cars it is possible to find one main module for each of them which are then connected to the main controller. For each ECU in the internal networks it is possible to find actuators, like the lights, and

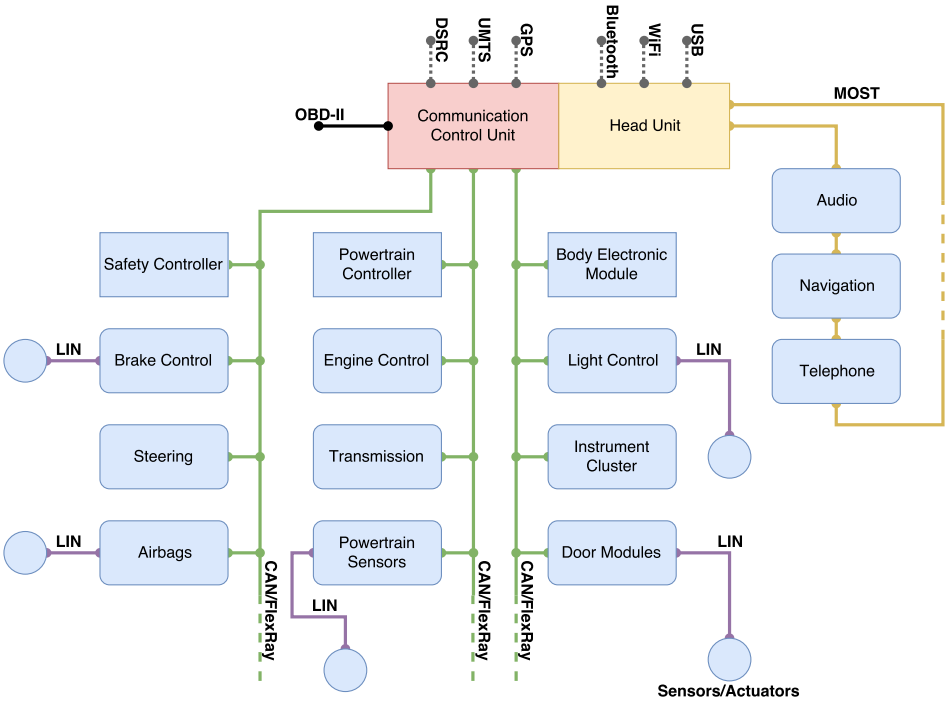

Figure 3.1: structure of the on-board networks of the car. OBD-II is the physical diagnostic port present in every car, DSRC (Dedicated Short Range Communications) will be soon used for V2V interactions.

sensors, that could be either activated by the driver, like the lever to turn lights on, or activated by the environment, like the wheel movement sensors that engage the ABS. Usually there are at least two main units that have available interfaces with the outside of the on-board networks: The Head Unit can be equipped with USB ports, Bluetooth and WiFi devices, while the Communication Control Unit (CCU) is designated to connect to Cellular networks (LTE, 3G, etc.), GPS, Dedicated Short Range Communication (DSRC) for V2X communication, and is usually equipped with a diagnostic port (On Board Diagnostic (OBD-II)) which enables vendors and repair shops to check for the diagnostic messages sent by the different ECUs to detect errors, upload firmwares and change settings, but is also used to connect aftermarket devices of different kinds like fleet control modules.

3.2 THREATS

The first step is to define a list of comprehensive threats: we describe the three more realistic threats to an automotive system explaining the reason behind them, ordered by dangerousness.

3.2.1 *Safety Threats*

In the latest years new safety systems have been installed in cars with the objective to help the driver in normal and dangerous situations, from park assist to emergency brake systems to cruise control. All these systems have access to safety related controls like steer, engine speed and brakes, which may makes it realistic for an attacker to tamper them and threaten the safety of the people inside the car.

3.2.2 *Economic Threats*

The most common threat we usually consider while thinking about the objective of the attacker in an automotive environment is theft, either of the vehicle or of something inside it. In both these cases the technologies installed on new cars like keyless ignition or keyless door unlocking help the attacker giving him a broader surface to work on.

Ransoms are also becoming a significant threat, both to the end user and to the producing company: To the end user it may be possible to exploit the car to force it to stop working and request him a ransom to receive back the full functionality of the car. Although this may seem overly cautious the theoretical attack has already been proposed.

3.2.3 *Privacy Threats*

A campaign promoted by ADAC (Allgemeiner Deutscher AutomobilClub), a german automobilclub, named My Car My Data[11] has been done to sensitize owners of cars about the data shared by their vehicles. The information transmitted by cars is starting to relate not only to the status of the car itself but also to the driver personal information, mostly through data received by smartphones through bluetooth connections that are nowadays almost always present.
Also, with the developing of new V2X systems, the position of the car is becoming sensible information: as the car sends a sort of ID to alert others of events on the road if it is possible to track it there is the possibility to discover information about the driver like his habits, where he lives and so on.

3.3 IDENTIFICATION OF SECURITY REQUIREMENTS

To identify security requirements we create a set of models based on known and potential attacks published by researchers in the last ten years and how they would be implemented and generalizing them to provide a broader set of possibilities.

3.3.1 *Attacks*

We selected a set of different attacks, both implemented or simply proposed, that help us define the security requirement models. We do not claim to provide the complete research over automotive cyber attacks but the resulting models highlight all the security requirements for the connected car.

In 2011 **Checkoway et al.**[3]for the first time proposed a set of attacks abusing of the external interfaces of the car, abusing of its connectivity: They managed to send arbitrary data packets to the internal

CAN Bus through the Bluetooth, USB and Cellular interfaces, finding flaws in each of the dedicated units. Through these flaws they were able to easily request the car to unlock its doors or to activate the microphone inside the head unit, or to receive GPS data.
Continuing in the same direction **Miller and Valasek** described in 2015 a new set of attacks[5] on a 2014 Jeep Cherokee. The Infotainment system of the car had two major flaws, one enabling the connection to the WiFi hotspot without authentication and the second enabling access to a service through a cellular connection (an unauthenticated open port), both of which enabled, abusing of other code vulnerabilities to access the internal networks of the car from anywhere. As they were able to completely reflash an ECU with new firmware they had complete control over the on-board networks, that enabled them to modify speeds, steering and other safety critical elements of the car.
Garcia et al. and Francillon et al.[12][13] both made different analyses on the security around Keyless Entry Systems and Keyless Start Systems, demonstrating how it was possible to bruteforce the algorithms as they are too weak, and that signals from Passive Keyless Systems (the one that enable door opening if the key is nearby) can be replied from a significant distance enabling door opening even without the owner being nearby.

Considering instead the internal networks of the car it is possible to underline a set of intrinsic vulnerabilities (partially the ones that enabled the attacks of Checkoway and Miller) in the CAN buses that can be exploited in various ways: In CAN, packets do not have a sender and a receiver field. The way a ECU decides if it has to read a packet is by checking its ID: Each packet has an ID that defines its function, but there is no way to know which ECU sent it, which becomes completely impossible to guess as the network has a bus topology and therefore every ECU writes on the same bus. Last but not least, to arbitrate packets sent on the bus CAN uses Carrier Sense Multiple Access with Bitwise Arbitration (CSMA BA): when two nodes start

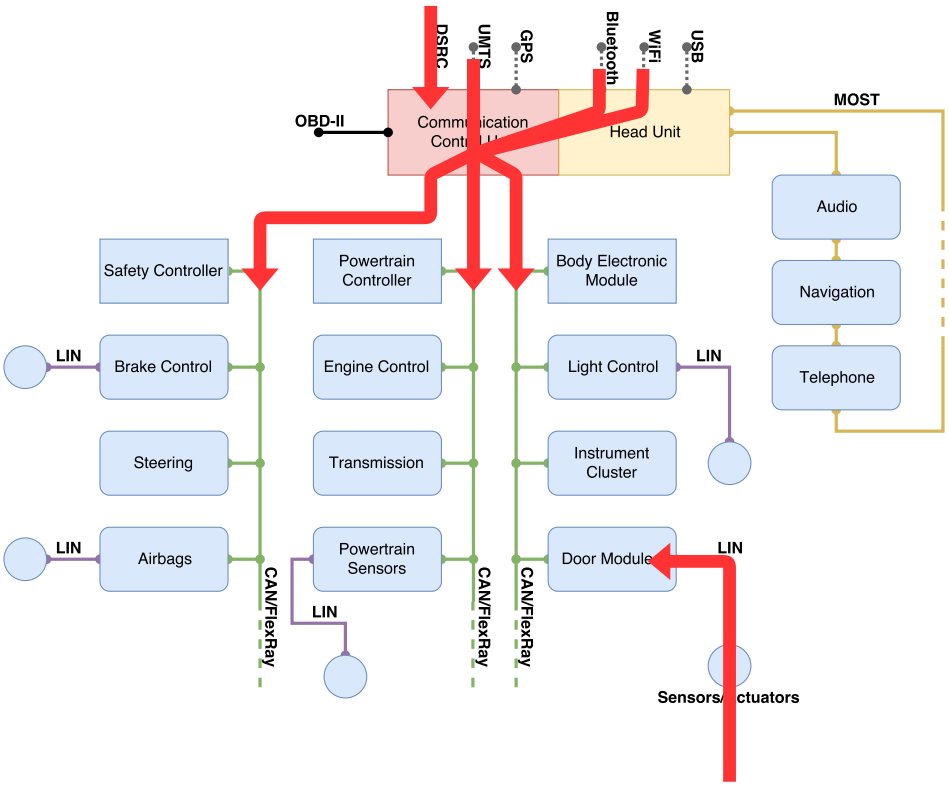

Figure 3.2: The paths of the described attacks. the end of the red line indicates the final objective of the attacker

writing their IDs at the same time on the bus, since every node always reads what is being written even when it is writing, the ID with higher priority wins the arbitration thanks to the recessive bit overwrite (the bit 0 overwrites the 1, therefore if an ID starts with 0 and the other with 1 the latter will not be written). This last flaw enables an attacker to apply a Denial of Service (DoS) attack as showed by **Palanca**[14] by targeting a specific ECU IDs and modifying them every time it tries to write, forcing it to lose the arbitration and, at a certain point, to reach a threshold over which it completely stops to communicate.

Considering the external interfaces and connection of the car we want to highlight the issue related to V2X although there are not already studied attacks as V2X is yet to be implemented: in the next years cars will start to have ways to communicate with each other to signal dangers and other information, which has already been

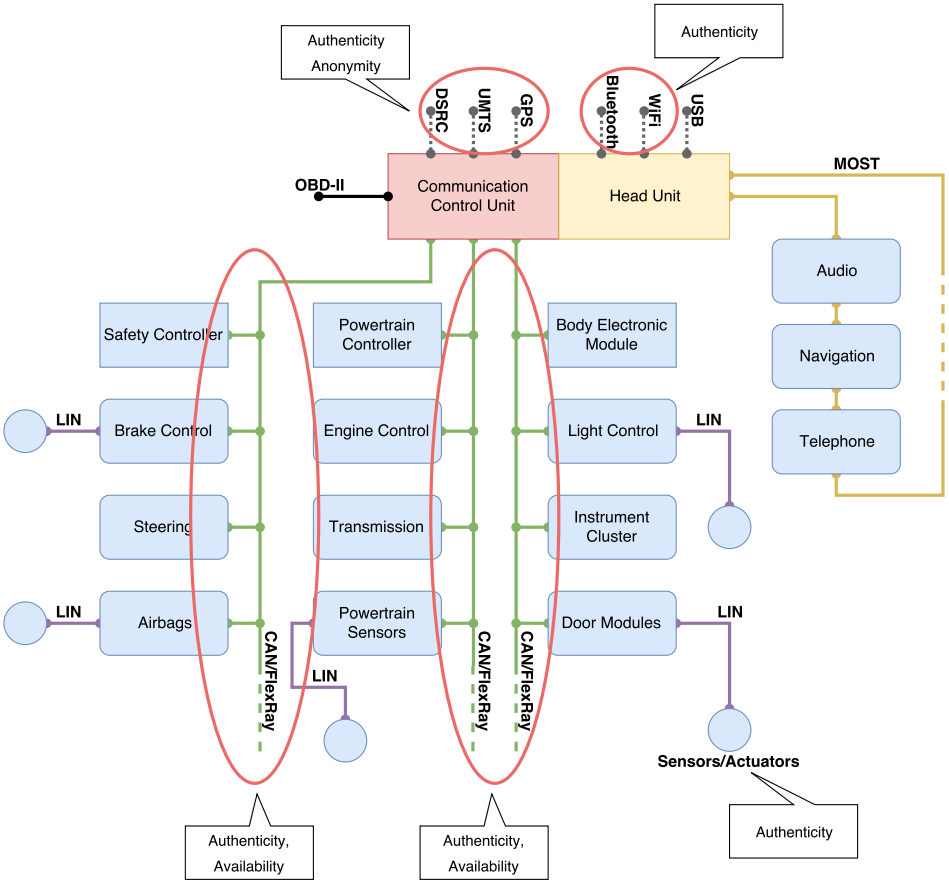

Figure 3.3: Requirements of each section of the network of the vehicle

flagged as a security issue for two major reasons. The first is safety related, as if it is possible to send a forged message to the car alerting of an incoming crash it would be possible to activate the emergency brakes without reason and therefore to endanger the passengers, while the second is a privacy one as already stated earlier: cars will have to authenticate each other in some way, which will lead them to send some sort of identification. If through this identification it becomes possible to track the car owner a set of sensible information can be derived: Where the owner lives according to the most common paths, the usual times in which he leaves home and so on.

3.4 SECURITY REQUIREMENTS

We now present the security requirements derived by the previously described attacks and analyses. Here we do not focus on proposing solutions relative to them as the next chapter revolves around that issue, we only want to highlight the constraints that has to be considered while designing a vehicle network.

- Authentication: It is evidently the most recurrent requirement, as the set of issues arising from not implementing it is the most consistent. We can further subdivide it in authentication of information originating from external sources, in terms of origins, content and time (such as firmware update requests), authentication of information originating within the vehicle, still in terms of origins, content and time (such as emergency braking).

- Availability: Although this requirement would be always useful, also in connection with external sources or in streams of sensor data, it is not strictly required in these two cases as the vehicle behavior is not strictly affected by them. In the case of unavailability of the internal networks instead, mostly the ones related to safety, powertrain and body of the car the vehicle does not act like it should. Safety issues may arise as the brakes do not work as they are supposed to or the engine may not start, not considering the unavailability of all the instrumentation like lights and wipers.

- Anonymity: This requirement for the external networks is becoming necessary as the V2X infrastructures are going to become popular.

In the next chapter our focus revolves around proposing solutions relative to these security requirements, analyzing the state of the art of both the research and the market, with solutions coming from the academic and industry field.

4

APPLIED AND PROPOSED SOLUTIONS

We proceed to describe the solutions that have been proposed to overcome the issues analyzed in the previous chapter. Each solution or set of similar solutions is accompanied by a brief analysis of feasibility and usefulness to understand how valid it is. All the proposed solutions are at least feasible, but not all have the same impact on security, therefore where possible while analyzing them we consider:

- The level of security brought by the solution: which requirements it is going to cover and its strength in doing so.
- The complexity required to bring the solution into the real world: Although some solutions may seem interesting their basic requirements are too high to be acceptable in the actual automotive field.
- The costs necessary to implement the solution: as the automotive field is not a theoretic one often costs are a bottleneck, although we are not going to give an actual cost of the implementation we compare different similar solutions considering costs.

The solutions cover a wide range of cases, therefore we need to divide them into smaller categories. To follow the structure created in the previous chapter we start with the solutions related to the most internal networks (specifically CAN as it is currently the network used for safety critical systems) describing the proposed authentication methods and protocols. We then describe the solutions relative to the overall internal networks: current network structures and proposed ones and hardware security modules, which include Intrusion Detection Systems. Making a step further we discuss the possibility of using new protocols for the on-board networks, first describing the

current ones and then which and why ones may take their place and why. Finally we give an overview of the newborn V2X security field: we describe the main issues that have already been highlighted and show which are the trends currently hypothesized to solve them.
Not all solutions are related only to one category, often there were multiple choices but we want to create a structure that is easy to follow.

4.1 AUTHENTICATION METHODS IN CONTROLLER AREA NETWORKS

Currently the communication inside CAN bus is completely unsecured and unsupervised. All packets transmit clear messages with no authentication or cryptography applied, therefore as anticipated in the previous chapter any kind of replay, forge, sniffing and DoS attack is completely unpreventable and undiscoverable.
In the threat model we considered Authenticity and Availability as the two main requirements for the internal networks. Sadly as in many other networks Availability is extremely hard to ensure due to DoS attacks, in this case in particular the one proposed by Palanca[14] which abuses of the protocol itself to force a node to go offline (even Palanca in the conclusions of its work is not able to propose a feasible solution). Authenticity on the other hand has been thoroughly studied, finding various solutions to replay, forge and masquerade attacks in particular.
But how do we apply some forms of authentication in a bus on which messages do not even have a sender and a receiver address? A simple way to solve this comes directly from the CAN standards, in which is stated that only one node can send packets with a specific ID, even if multiple ones can read them: virtually the ID can be considered as the sender address. This is what **Matsumoto et al.**[15] take advantage of when proposing their method to provide authenticity over CAN: Every node has to have a flag in the CAN controller for each ID that

node can send. Each time the node starts sending a packet it raises the relative flag up until the message is received. As CAN nodes are always reading from the bus as they have to check for incoming messages when idle and control arbitration when are in the send status, whenever a node reads an ID that is among the ones it is supposed to send, checks for the relative flag: if it is raised it means that the sender of the message is itself, therefore there is no security breach, while if the flag is not raised someone is trying to forge one of its messages and the node sends an error frame over the packet to nullify it.
This method, which does not actually enable real authentication but can block forged and replayed packets from being received and accepted by the network, has the main advantages to be extremely simple and easy to implement but is only able to recognize the IDs that are already part of the network which is a significant limitation. Costs of the solution are not as low as it may seem, as new hardware (a modified CAN transceiver) has to be installed in each node that has to implement the method.

The following proposals all share the same base which is the **TESLA**[16] broadcast authentication protocol, a protocol designed for Internet of Things (IoT) purposes that is not directly applicable to automotive networks due to the strict real time requirement which require extremely small delays. To prevent replay attacks what is proposed in this protocol is a counter that increases at every exchanged message between two devices so that a replay should know the new counter to pass through. To avoid forging some kind of cryptography is required and the most common method for this specific field is Message Authentication Code (MAC), which is like a digital signature as it is a cryptographic checksum of the message that provides integrity and authentication but it is extremely faster than digital signatures at the cost of not being able to ensure non repudiation. Considering the lifetime of the car the counter has to be reset after a while and to avoid replay of messages after said reset sessions are another common factor. Dependently on the proposal after a given time or number of

messages the session changes, resetting all the counters but removing the possibility to replay messages.

The first proposals that have been produced that fulfilled timing requirements have been **CANAuth and LiBrA-CAN**[17][18]. These two authentication protocols both are based on the same concept which is **CAN+**, a method to write up to 15 bytes of data in between two CAN messages. What is needed to do so is an hardware modification in the ECUs so that the transceiver is able to write and read in that times between CAN messages. CANAuth, which is the simpler of the two, is based on a pre shared key for each group of IDs coming from the same unit. For each CAN message a CAN+ one is sent with the MAC signature composed by the session key, the counter and the message itself, alongside with the counter in clear value so that each unit can immediately define if it is replayed. This protocol is distributed, that is each unit has to store multiple keys one for each ID it has to receive. LiBrA-CAN oppositely is a centralized protocol which uses CAN+ the same way to send the authenticating messages in between the traditional CAN ones, but differently to CANAuth the keys are stored in a centralized unit which authenticates the messages(which in practice means that if a message fails authentication an error frame is sent by the central unit over the message to avoid it being accepted). The distinguishing characteristic of LiBrA-CAN is also a division of the keys in groups of ECUs so that the overhead due to key exchange lowers significantly, but at the same time it has the main disadvantage that if one unit gets infected the whole group of other units sharing a key with it becomes unreliable.

The main positive side of these two protocols is the fact that they are completely backward compatible, as using the CAN+ timing to send the authentication messages other units which may not use one of the two protocols wouldn't even notice that more data are passing through the network. The reason though for which these protocols are hardly feasible in real world applications is the cost of the hardware required to support CAN+: as we stated earlier ECUs need to

be cheap enough to not increase eccessively the price of the vehicle, reason for which probably these two protocols are not suitable.

To overcome the necessity of significant hardware requirements in 2012 **LCAP**[19] was presented. This protocol proposes a"magic number"that has the objective to authenticate the sender which is hashed with MAC into the messages. To define and authenticate this magic number though a significant overhead is required: Before the start of the communication a setup phase is necessary in which each couple of ECUs that will communicate with each other sends six messages to define the keys. After that, if a synchronization is required, other N messages (where N is the number of requests for synchronization) are sent on the bus. More than this, for each couple of ECUs five new IDs have to be created and saved in each unit to define the specific messages that they need to use to setup the communication, specifically 5 x (number of senders) x (number of receivers) new IDs in total have to be created and shared. The last issue of this protocol derives from the necessity of adding the magic number to the data field of the packet. As the maximum length of the packet is eight bytes and the magic number uses two of them, every message that requires more than six bytes needs to be reformatted.
Although it does not present high costs, the sum of the aforementioned issues leads to the infeasibility of this protocol in real world scenarios as the amount of changes that are required to implement it do not make it the best choice for manufacturers.

To overcome the enormous overhead required by LCAP **CaCan**[20] was proposed by Kurachi et al. They simplify the overhead by using a monitor node installed in the network, which should be the only one with a modified controller to overwrite messages that failed authentication with an error frame. The advantage as seen in LiBrACAN of using centralized authentication is that the amount of keys required at each end point is much lower. Each unit makes a challenge-response authentication with the monitor node at the beginning of the session, after which they do not need to communicate with it di-

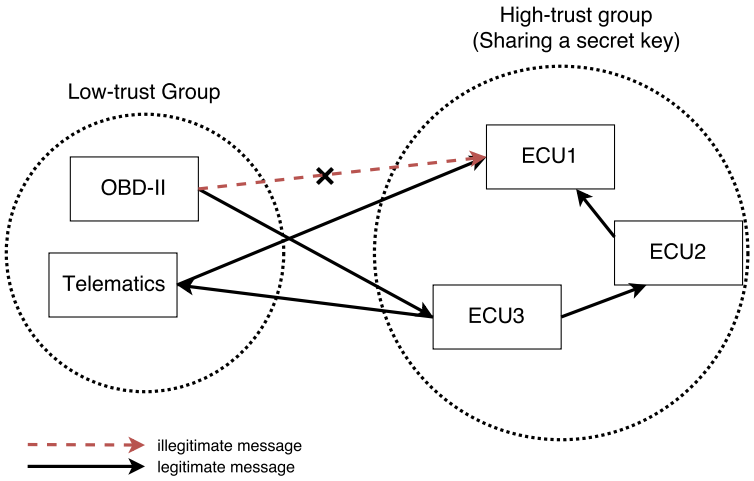

Figure 4.1: An example of groups of trust in VeCure, the units on the left cannot send messages freely to the ones on the right as they do not have the key.

rectly and the monitor will just check the correctness of the MAC and the counter. The advantage of this protocol is also derived by the fact that if more than one CAN networks coexist in a single vehicle, which is a recurrent setup nowadays, if the monitor is installed inside the gateway between the networks it can check both sides, replaying the message when necessary. This proposal is not extremely expensive as the only node that has to be modified in hardware is the central monitor, and overall is feasible and not extremely complex. The main disadvantage derives from the centralization which generates a single point of failure.

To remove the issue of centralization in 2014 Quiang and Sawhney proposed **VeCure**[21] which takes the best concepts out of the other protocols to produce a distributed authentication method whose main element are trust groups: groups of ECUs with the same level of trust share keys, which will be used to create two messages, the first is the normal CAN packet while the second is the MAC of the packet and the counter. Although there is some overhead it only doubles the amount of sent data and only for the units that are in a trust group, but in exchange for that overhead there is no need for hardware modifications and it is completely backward compatible.

	Security	Simplicity	Cheapness
Matsumoto et al.	Low	High	Low
CANAuth	High	High	Low
LiBrA-CAN	Medium	Medium	Low
LCAP	High	Low	High
CaCan	Medium	Medium	Medium
VeCure	Medium	High	High
LeiA	High	High	High

Figure 4.2: A graphical schema of the proposed authentication algorithms (the higher the better).

VeCure is not expensive and simple, making it a good candidate to be actually implemented, but its use of trust groups makes that if one node is compromised its whole trust group, which shares its key, becomes unreliable.

The last proposal we present is **LeiA**[22] proposed in the ESORICS 2016 conference that should be a good tradeoff between the low overhead brought by VeCure and more security: with this protocol each node stores a tuple composed of a symmetric key, an epoch, which is similar to a second counter, a session key and the counter for each authenticated ID.
With these information every session start each couple of nodes that need to authenticate each other increase their epoch instead of sharing data and use the new epoch to create session keys without any need to communicate apart for when a message has to be sent. In this case, similarly to VeCure, two CAN frames are sent, the first with the real data, the second with the MAC of data and counters. This proposal has very small downsides: the overhead is minimum, there is not a single point of failure as in centralized protocols, and the costs are very low as there is no need for a hardware modification or new modules.

As shown in4.3 we abstracted an idea of how the proposed algorithms can fit the three elements we proposed to take into consideration in the beginning of the chapter. It is evident how the last pro-

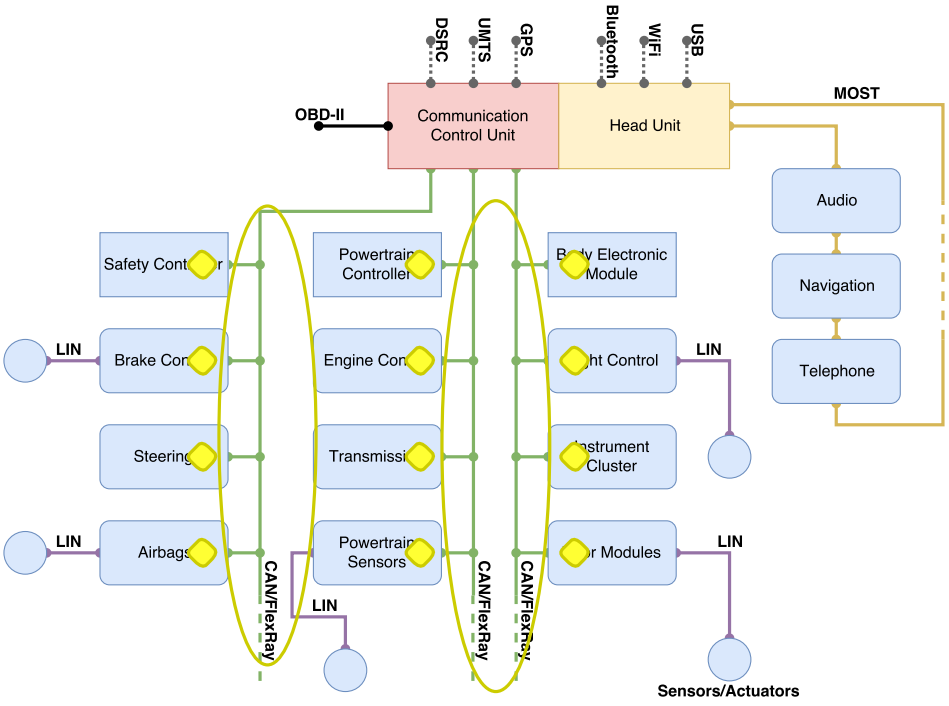

Figure 4.3: The automotive system if authentication methods are applied: yellow diamonds indicate where the changes have to be applied, while the yellow ellipses the elements affected by the applied solutions

posals are overall better than the first ones. To analyze these results we have to consider that the automotive security field is extremely new, therefore the five years of difference between the first and the last proposal do make the difference.

To clarify what the three sections mean, Security refers to the level of security the proposed algorithm ensures, considering limits of the model, level of security of the algorithms used and centralization versus distribution. Simplicity does not refer to simplicity of understanding of the proposal but to the implementation simplicity, considering how much has to be changed to implement the proposal. Cheapness refers to the prices required to implement the proposal.

Finally, to conclude the section we propose a representation of the automotive system if one of the methods proposed above are applied: these methods have to be applied either inside each ECU or inside one

for each network (dependently if they are distributed or centralized) and affect the security of the CAN buses around them.

4.2 NETWORK STRUCTURES

Current automotive internal networks can be represented as we did up until now: they are composed of a main central unit connected to the different CAN, FlexRay and Media Oriented Systems Transport (MOST), while sensors are connected directly to the ECU that require their information through Local Interconnect Network (LIN). The main security issue arising from such structure is the single point of failure of the main unit concurrently with the fact that all external interfaces are in that same unit. The current future steps, primarily proposed by the manufacturers[23], are moving toward a more decentralized structure in which each subnetwork has its own gateway or switch and all of them are connected together by a main network which should be an ethernet one (We discuss automotive ethernet in a successive section of this chapter) and LIN is partially substituted by a new ethernet subnetwork from which data are sent to the receiver ECU, as a significant amount of cameras and other bandwidth reliant sensors are being implemented, requiring a network much faster than LIN to transmit.

These new steps are interesting from a security point of view due to the decentralization factor that enables security specific gateways for each network and more control over the single network. Also, structures like the one in4.5 should not have latency issues as the single CAN subnetwork remains unchanged.

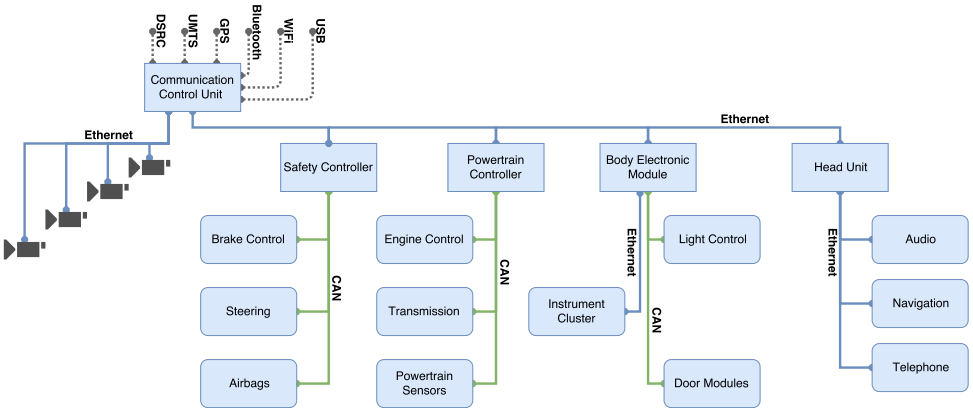

Figure 4.4: A graphical representation of the new automotive network structures.

4.3 HARDWARE SECURITY MODULES AND INTRUSION DETECTION SYSTEMS

As just seen CAN Bus is not the only network inside the car: LIN, MOST, FlexRay and in the near future probably Ethernet will compose the whole complex network inside the vehicle. To interconnect these networks currently there is at least one gateway to which all buses are connected and that redirects only a little fraction of the messages. For example, the only messages redirected from the CAN Bus to the OBD-II port are the diagnostic ones. Currently the gateway is not considered as a security module as its main focus is to connect the different networks but security solutions are already being designed by manufacturers.

The first noteworthy proposal relative to a hardware security module has been produced by the **E-safety Vehicle Intrusion protected Applications (EVITA) project**[24] which has been an european project with focus on prototyping secure automotive on-board networks that was run from 2008 to 2011. Although it is a project older than most of the ones we presented it still gives an insight on a different way to protect the internal networks of the car. What was proposed for the on-board network is a **Hardware Security Module (HSM)** which is a security dedicated hardware module that enables the implementation

of the cryptographic operations and dedicated security mechanisms. To try to mitigate the costs of a new piece of hardware for each ECU in the car three different models were designed, which had different costs and fulfilled different security requirements and different functional requirements: the EVITA HSM Full version was the one that has to be responsible for V2X Applications by verifying and signing messages for external communications, therefore is the one with the maximum levels of functionality, security and performances. It adds to the main ECU with its own application core another block which realizes all cryptography hardware side, composed of an hash engine, a symmetric cryptography engine and an asymmetric one and also its own CPU that can be used to directly access the memories. The application core CPU can access the EVITA security hardware only through its secure interface. The EVITA HSM Medium version is extremely similar to the Full one, has its own CPU and symmetric cryptography module but does not have hash and asymmetric cryptography modules, therefore it is perfect to use in the internal networks to send encrypted messages, but it is not feasible to use it alongside a V2X ECU as it is too slow in asymmetric encryptions. At last the EVITA HSM Light version is composed only of an optimized symmetric Advanced Encryption Standard (AES) hardware accelerator, everything else is left to the core application. This module have been proposed for sensors, actuators and ECUs that provide or process critical information.

This project surely fits the automotive environment as it has been done appositely for it and the definition of different modules permits to lower costs, but these costs remain high anyway, due to the implementation of one module for each ECU and sensor, which would lead to dozens if not hundreds of security modules per vehicle.

The next proposals are hardware security modules in the sense that they are new modules applied to the network, but they are more specifically Intrusion Detection System (IDS). Their scope is that of discovering attacks in one of the networks analyzing the stream of

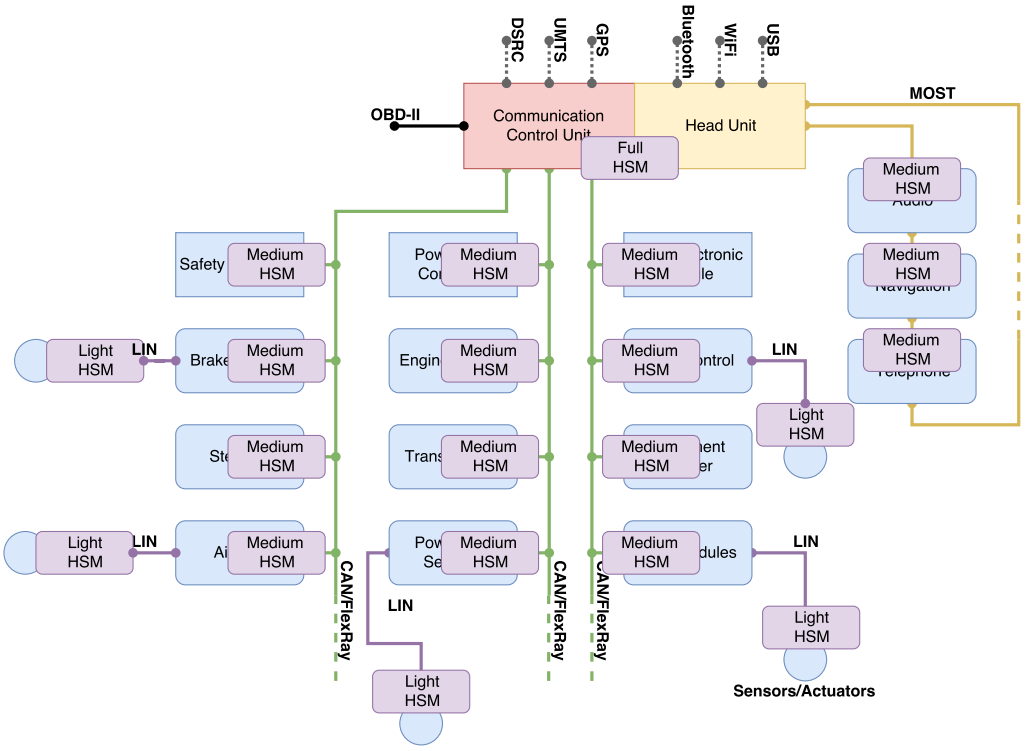

Figure 4.5: A description of all the positions in which EVITA Hardware Security Modules would be positioned.

data passing through them and raise an alert.

There is an obstacle in doing such kind of analysis in the CAN bus though: the majority of packets are transporting proprietary messages which meaning is known only to the producer, so how is it possible for an IDS to analyze data if it does not know what it means? The four systems described below use different features none of which relies on the knowledge of the meaning of data inside the packet.

Marchetti and Stabili[25] proposed an anomaly detection system that analyzes the ID sequences passing on the bus. After a period of necessary training while running in different situations the systems composes a matrix of all IDs and the potential following ones. After the training period it analyzes the data stream on the bus and if an ID is not in the matrix the IDS raises an alarm. The experimental evaluation on realistic attacks gave interesting results if the packets were not replayed, as in that case there is a significant possibility that these packets arrive at the right time. It is instead extremely effective

on forged packets with different IDs in respect to the ones usually sent on the network.

Waszecky et al.[26] propose instead an IDS based on in-Vehicle traffic monitoring: it analyzes and uses arrival timing curves to understand if there has been some kind of change in the patterns. This should work as timing curves may change during an attack due to the attacker usually trying to force an ECU to receive the altered packets over the right ones. In a little more depth in the first stage the car network specification is analyzed considering communication patterns, architecture and parameters, through these information arrival curves are derived. In the second stage these arrival curves are used to analyze the stream of data passing through the bus to detect attacks or malfunctions.

Kang and Kang[27] propose to apply deep neural networks to the CAN Bus: The results they produce are interesting, about 98% accurate detection ratio, although their results have not been proven by other researchers.

Finally, **Seifert and Obermaisser**[28] propose a"security gateway"that is connected to all networks and is used as a security module for the transmission of data inside the single network and from one to the other. The idea behind the gateway is that in any specific mode of operation (driving, static car with engine on/off etc.) only a subset of all the available messages is valid on a specific interface. The gateway has access to two databases, one is the log database that stores the latest messages from each interface and the other is the"Intentional Application Behavior Specification"which stores the valid application behaviors basing on the protocols specifications. The gateway has to control different kinds of networks and the actions to check for the validity of messages are personalized for each one: For event-triggered communication like CAN a minimum and maximum interarrival time is defined so if a message violates its time constraints it is flagged as faulty. For time-triggered communication like the static segments of

FlexRay a time-triggered definition is defined for each message and a cycle length, cycle repetition and slot is set. The gateway can from this information derive if the messages arrived in the correct slot and at the correct time. At last for stream communication it is possible to add dependencies between messages to the model, including the possibility to detect and mitigate buffer overflows knowing the amount of streaming data incoming to the receiver ECU.
After the initial setup tests have been done both during the normal behavior of the car and during attacks that flooded the CAN network of request to lower the window. In both cases the results were correct, although the specified attack is one of many and other types of attacks have not been tested.

The applicability of IDS in the automotive field is not as broad as the authentication systems proposed earlier, for various reasons: First, what if an attack is discovered? how do we act on the threat when it is already being performed? It is possible to alert the driver or an external entity but even if something like this is done, usually the attacks to the automotive environment are not meant to steal data, for which the alert may be a solution, they are usually focused on disrupting drive or steal the car, cases for which is quite useless to alert someone. Even if the dashboard has an indicator of an attack being in action the driver would already be in danger if driving.
Second it is, at least currently, not possible to ensure the complete reliability of these IDS and if there is even a little chance of a false positive, it is extremely dangerous to act over it removing the message from the network or shutting down the relative ECU as the automotive system is safety critical, so if the message was instead a safety related one the detection system may worsen the situation instead of solving an issue.
Third, all the proposed IDS are not suitable to every situation but only to small sections of attacks, mostly the ones in which new packets are being forged and forced inside the network. Replay attacks appear to be extremely hard to recognize, alongside with DoS attacks like the

one proposed by Palanca[14] which does not change the flow of the messages inside the network but only changes one bit per packet of the victim ECU.
For this reasons we would not say that IDS are ready to be deployed on cars at the moment as the developing process is not mature enough. Considering that CAN messages are proprietary and the restrictions of the requirements, in particular the low cost of the devices, this is predictable: what is probably needed is a standardization of some kind of the messages so that the IDS can be trained on the meaning of the packets and not their structure or timing.

4.4 COMMUNICATION PROTOCOLS

Through the years different communication protocols have been proposed to come to a solution for the various shortcomings of the internal networks in the automotive field but none directly aims at solving security issues. Even less communication protocols have been proposed to substitute CAN due to the strict requirements in relation to latency and reliability. The proposed protocols are, though, enabling other security solutions currently unavailable due to the specific network protocol they are relying on, therefore we describe them analyzing their strength in relation to the current ones to highlight where it would be possible to act to strengthen security.
The only current confirmed successor of CAN is Controller Area Network with Flexible Data rate (CAN-FD), proposed by BOSCH in 2011. The main problem it focuses on is the low amount of data available per packet in CAN: in CAN-FD the data field can carry up to 64 bytes compared to the 8 ones of CAN, which is possible thanks to the change of transmission rate of the data field from 1 to 8Mb/s. The limitation to 1Mb/s of CAN comes from the necessity to synchronize the arbitration phase of the message and the Acknowledgment (ACK) one, as during these phases there may be more than one ECU writing

on the bus, therefore changing the speed during the data phase in which only one node is transmitting becomes feasible.

The possibility to transmit 64 data bytes per packet enables to implement many more features on the packets and is leading to new solutions both in relation to the applications of CAN-FD both in relation to the development of security systems. Specifically **Woo et al.**[29] have already proposed an authentication protocol like the ones seen in section one but applicable to CAN-FD: they considered ISO 26262[30] (Automotive Safety Integrity Level (ASIL)) which is a standard enacted to prevent accidents caused by the malfunction of automotive systems and applied its safety goals to an authentication protocol based on groups (divided by ASIL levels). The protocol is partially centralized with a Gateway Electronic Control Unit (GECU) which is the only unit with high computational power and which is the only connection with the external networks, while other ECUs require only the computational power to encrypt and decrypt 47 bytes of Automotive Security Level (ASL) packets: the initial session key creation is done by the ECU with the GECU only if the ECU is in a high ASIL group, while during normal communication each ECU has to encrypt the data with AES (which is usually implemented hardware side on processors for embedded systems and therefore not too computational heavy). The receiver ECU, that has to own the session key produced by the GECU, decrypts the message at its arrival without external support. If the ECU is in a low ASIL group and therefore requires less security the data are not encrypted but a 16 bytes MAC is still inserted in the message.

Although experimental results show that the computation time of the encryption algorithm remains acceptable only with 300 MHz processors, which is usually too much for an embedded cheap device, it is evident how the use of a 128 bits MAC (which leaves 47 bytes available for data) in relation to the MAC proposed by previous authentication protocols in CAN (up to two bytes, leaving 6 bytes for data) leads to an exponentially increased difficulty in decrypting it.

Apart CAN-FD which is going to become one of the confirmed next standards, there are some other proposals that are being considered to substitute CAN and other protocols commonly used in less safety critical sections of the car network. In particular due to fact that networks have always had the tendency to converge to non proprietary solutions the main current proposal for less critical networks is Ethernet.

Traditional Ethernet like the ones used in home networks (100Base-T, 1000Base-X etc.) have a specific element that make them unfeasible for real time networks like the automotive one: It is extremely fast if there are no collisions, but when collisions happen th arbitration is done with Carrier Sense Multiple Access with Collision Detection (CSMA CD) in which each node that needs to send data after the collision waits for a random time and then tries again, hoping that no one starts sending in the meantime, as in that case the collision process is repeated with a longer random waiting time. This process is not acceptable for safety critical and in general real time networks as it generates too much overhead if the network becomes busy, therefore new Ethernet protocols were specifically produced. We proceed to describe three specific real time Ethernet protocols that are suitable for the automotive environment although we do not claim that they are all the possible ones, in this case we only want to describe them as an example of the general pros and cons of Ethernet over CAN or CAN-FD. We refer to this documents for the three protocols we chosen to describe[31][32]

Avionics Full-Duplex Switched Ethernet (AFDX) is an already used standard in the aeronautics which supports rate constrained traffic providing an upper bound on latency and determinism through modifications to the Ethernet protocol. This is done with bandwidth partition, as maximum data packet size plus scheduling of packets allow guaranteed bandwidth, using a defined packet order, as packets are always received in the same order they are sent, and using dual redundancy transmitting the same packet out of two ports.

Ethernet Audio Video Bridging (AVB) is an IEEE proposal for a real-time Ethernet solution. It supports rate constrained traffic through standards IEEE 802.1Qat and IEEE 802.1Qat, but also includes a 3-bit priority field which indicates frame priority that would be able to ensure that safety critical packets have priority and ensure their timing to be low.

The latest standard from IEEE and One-Pair EtherNet (OPEN) Alliance is **BroadR-Reach**[32] which enables longer distances, up to 700 meters, with connectivity at 100Mbps. The technologies used and the echo cancellers applied permit full duplex over a simple single pair cable. This feature, which lowers prices in relation to traditional Ethernet protocols, enables to create a star network and avoid buses without leading to excessive costs, and through the star network removes the problem related to CSMA CD as collisions are not going to happen if there is only one device sending and receiving data for each full duplex cable. The timing turns therefore to be a problem related to switch delay but it is possible to implement priority of safety critical packets to ensure an upper bound on latency.

4.5 V2X

V2X is an extremely new field and for this reason the security of the field has not yet been completely defined, but there are already known challenges. Indeed, it has been recognized as a priority for this field to be developed with security as a prerequisite. Specifically the requirements are authentication of the incoming messages from other vehicles or infrastructures and at the same time privacy of the sender of a given message.

The authentication requirement is currently not related to safety of the vehicle: A yet to be published risk assessment on the topic shows how, as the messages from other vehicles will not directly force safety related reactions from the car, accepting a malicious message will not

affect the safety of the passengers. Although the lack of security related reactions may change in future, enforcing the necessity of authentication, the current reason derives from the possibility, in case of reception of too many false messages and consequent alerts from the vehicle, of the driver deactivating the V2X features of its car.

Due to the authentication requirement each vehicle should own some sort of proof of it being a valid source of information like a public/private key couple, which leads to the privacy requirement: if each vehicle is distinguishable from the others it becomes possible to track the vehicle movements and through them reasonably assume sensible information of the driver or owner, like his address, the usual time in which he leaves home and comes back, where he works and many other sensible patterns.

There is then the necessity to develop a security system that ensures both authentication and privacy of the single user. The solution chosen both in Europe and in the United States and that will probably become a global standard is to use a Public Key Infrastructure (PKI) where each end user is provided with pseudonym certificates instead of one single certificate. Pseudonym certificates have a short lifetime after which they are changed with another one to maintain the privacy of the user.

The necessity to frequently change pseudonyms leads to high complexity in the managing of certificates: it has been estimated in the paper presenting of the leading proposals for the United States PKI that such PKI will be required to issue around 300 billion certificates per year[33].

Another new and consistent issue derives from the necessity to retrieve the certificates and validate the messages from other vehicles: to validate the messages, as in a traditional PKI, the sender has to sign them with the private key of a valid pseudonym, while the receiver has to verify it by decrypting the signature with the public key of the pseudonym certificate and the sender authenticity will be accepted only if verification of the respective certificate is valid up to a Root

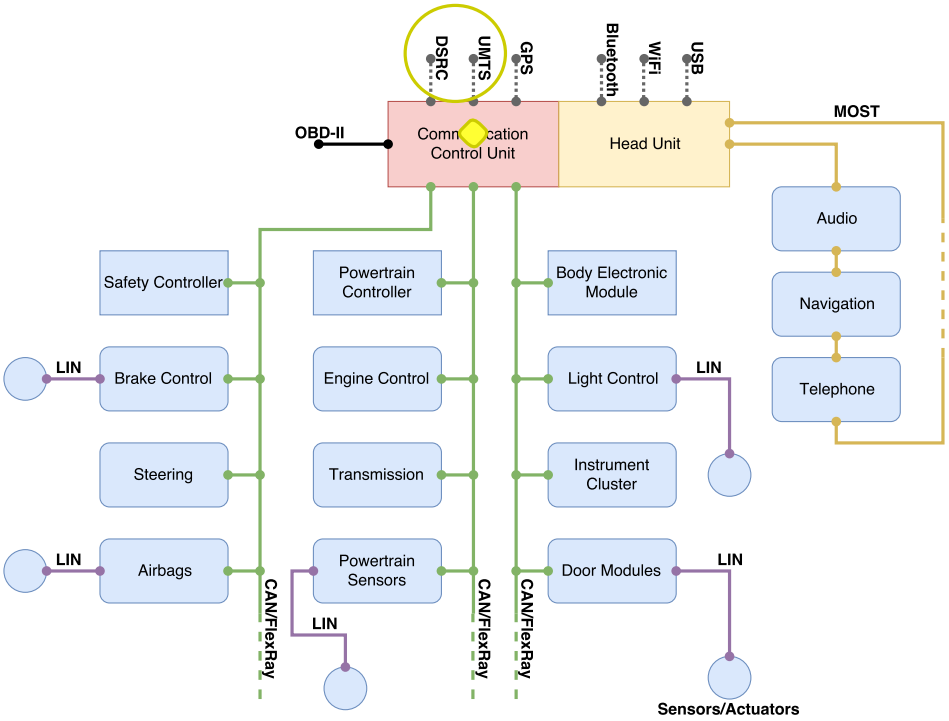

Figure 4.6: The automotive system if v2x PKIs are applied: yellow diamonds indicate where the changes have to be applied, while the yellow ellipses the elements affected by the applied solutions

CA. If the certificate authority is not available on the receiver device in a traditional PKI the usual procedure is to connect to one of the certificate authorities known to check for its existence. In the automotive case though it is not that simple as some of the received messages are real time information that become useless in a short period of time, therefore there are two options available, proposed to be used at the same time: The required certificate authority is requested through a connection to an infrastructure, if available, but up until the moment in which it will be validated the message is not fully trusted and only minimal actions can be done by the vehicle in the meanwhile.

Relatively to the pseudonyms they have to be requested to authorities, which are supposed to be either state driven or directly the manufacturers, to which vehicle will send periodically their long term public key (which the authority has to know) and will receive their new pseudonym certificate in return. The remaining question is how

and how often said pseudonym certificates should be changed to be sustainable and at the same time privacy ensuring. Both the european and united states proposals suggest that the single vehicle should own more than one valid certificate at a time (the american one in particular proposes 20 certificates minimum to ensure privacy) and that these certificates should be valid for around a week of time.
These proposals are going to be deployed in the next years and the automotive industry will slowly follow, with more and more cars being able to communicate with each other and with infrastructures. The early development of security measures directly embedded in the design of the overall structures should be a significant step of improvement as opposed to the issues with the internal networks of the car like CAN which even after ten years from the first security assessments are not definable as secure.

4.6 CONCLUSIONS AND OBSERVATIONS

In this chapter we proposed an overview of the current automotive security environment, describing the proposed solutions for all subsets of the field, from the authentication methods for the CAN to the new network protocols for the internal networks to the newborn field of V2X and its already known issues.
Before giving our conclusions we present a specific study case of the only element we did not consider in the survey, which is the single ECU, of which we propose the security assessment.

5

CASE STUDY

5.1 INTRODUCTION AND MOTIVATION

In the previous chapters we have proposed a survey on the state of the art of the automotive security field describing the methods that are going to be applied, but even if the applied methods are actually enhancing the security of the whole system what has to be ensured also is that devices should be designed with security as one of the main elements to ensure. If this concept is not applied protocols can be bulletproof and all kinds of security measures can be used but the flaws in the device could lead to exploitation of the whole network as“the chain is as strong as its weakest link". For this reason this chapter shows a case study relative to the security assessment of an ECU to present which can be the flaws on the single device that can endanger the whole system. Obviously one single unit cannot lead to conclusions on the overall status of the security of these devices and it is not our purpose to claim it, what we want to show is how these devices should be analyzed and which elements should be the main focus of the assessment.
The need to structure the work leads to five different sections:

- First we describe the studied ECU;
- Before presenting the actual work we briefly describe the instrumentation we used;
- The third section is an analysis of the hardware of the ECU. We opened it, visually analyzed which elements inside it may help us in the assessment and studied their manuals. At the same

time we searched for physical ports through which access the data inside the ECU;

- The fourth section regards the recovery and analysis of the software inside the unit. This step, in particular the reverse engineering of the code, has definitely been most time consuming part of the whole paper;
- The fifth section focuses on the communication of the device through the external interfaces, on one side CAN and on the other side the cellular connection through the modem;

Before getting into the actual analysis we highlight two different situations that occur throughout the chapter. As first, after the description of the main steps there are some italic written paragraphs that are the actual security considerations we did after the various analyses.
The second consideration regards the complexity of the description: to ensure a smooth read of the whole section and to avoid to create simply a list of what we did our analysis is simplified and not always explained in complete depth as we focus on describing the reasons behind the security concepts explained.

5.2 CHARACTERISTICS OF THE STUDIED UNIT

The ECU we studied is an aftermarket (as it does not come along the vehicle when bought but has to be installed in a second moment) unit developed to assist fleet management systems. Although we did not investigate on the prices and the methods for acquiring one we suppose that these units are sold as packs alongside the service to track them, which is an online website through which is possible to receive data from the various units and divide them into fleets to track each one with different parameters.
Alongside the unit we received the cables and antennas to install it

Figure 5.1: The ECU, still closed.

and a second smaller unit that is described later. Without further investigation it is possible to assume that the way this device works is to first receive data from the on-board network of the car and after some kind of processing of the data to send them to the servers of the company that offers the fleet management service. To avoid confusion with names that company is referred to as *Company* through the rest of the chapter.

5.3 INSTRUMENTATION OVERVIEW

Before passing to the actual description of the assessment process we describe the instrumentation that we used throughout the whole analysis which is visible in image5.2.

1. **Power Supply**: As the assessment has been done in a laboratory environment we had the necessity to power up the device

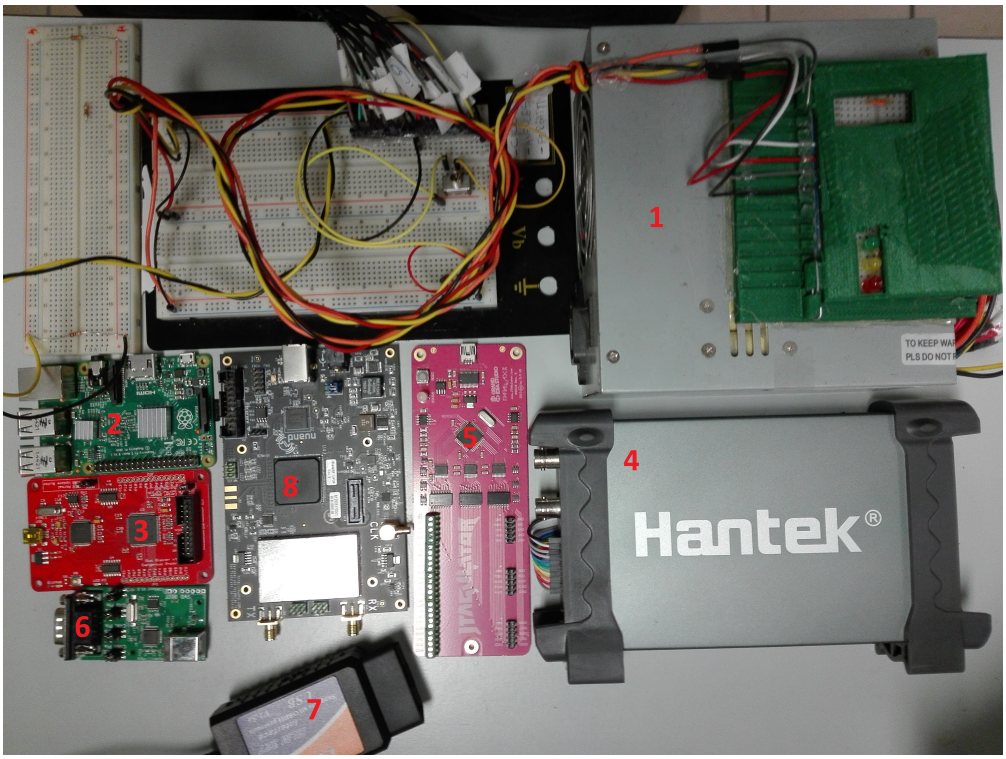

Figure 5.2: The instrumentation used to perform the security assessment

without connecting it to a vehicle. To do so we needed a power supply that resembled the battery of the car, therefore we dismounted an old Desktop PC and recovered its power supply, cut the cables to use only the ones we actually needed and connected them to a breadboard. The final product was a tabletop power supply able to generate 12, 5 and 3.3 Volts;

2. **Raspberry Pi 2**: It is a small and cheap computer that by default runs a light version of linux called Raspbian. We found that it would have been useful to be able to work on the device by remote, so instead of installing the software and drivers to connect the various boards to the device directly on one of our personal computers we set up a Raspberry Pi 2 connected to the local network of the laboratory through which we had access to the boards;

3. **Bus Blaster V4.1**: It is one of the low cost boards able to act as a JTAG debugger. Although its communication speed is not high

(we tested it up to 500 khz) it was suitable for our purpose of downloading not too many bytes of data. It has been connected on one side to the Raspberry Pi 2 and on the other to the JTAG interface of the ECU;

4. **Hantek 6022NBL**: In the initial steps of the assessment process we clashed mostly with the analysis of hardware parts. The Hantek 6022NBL is an oscilloscope and logic analyzer (which we used mostly for its latter function) that enabled us to study the signals coming from the various physical ports of the ECU and to create an initial idea of what they were;

5. **Jtagulator**: It is an hardware tool that assists in identifying on-chip debug interfaces from the available test points on a board. We used it specifically to understand the pinout of the JTAG interface of the ECU by connecting it to the Raspberry Pi 2;

6. **CANtact**: It is a CAN to USB open source interface working with Python libraries used to receive and send CAN packets to the board;

7. **ELM327**: It is a diagnostic OBD-II to USB interface which requests information through Parameter ID (PID)s to the on-board networks of the car and sends them to a diagnostic software on a personal computer to display them;

8. **NUAND BladeRF**: Software Defined Radio (SDR) platform that acts as a radio receiver for, amongst various choices, cellular connections. We required it to test the cellular connection of the device;

9. **IDA Pro**: disassembler and debugger we used to read, disassemble and reverse engineer the code recovered from the board.

10. **openOCD**: an on-chip debugger program to read and write to the JTAG interface of the board. We installed it on the Raspberry Pi 2.

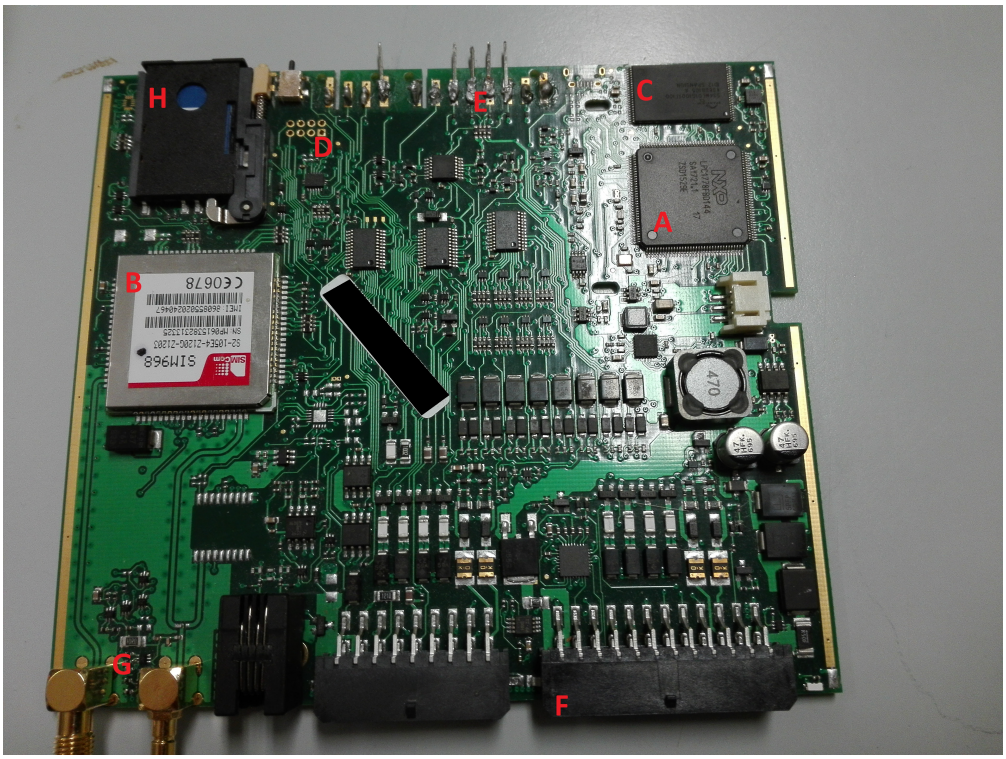

Figure 5.3: The device as it appeared when we removed the case (soldered parts were added by us later in the analysis).

5.4 HARDWARE ANALYSIS

The first step to understand what the ECU was able to do and how it did it was to discover which chips was it composed of. To obtain this information the easier way was to simply open the case of the device and recover the board. After, we visually analyzed which elements seemed to be the most important ones and looked up what they were. Although simple this step has been fundamental to understand better what we were facing. In image5.3 it is possible to see which elements attracted our attention, which we proceed to describe.

5.4.1 *The Microcontroller*

Chip A in figure5.3 is an ARM Cortex-M3 based microcontroller produced by NXP[34] under the code LPC1778FBD144 with the following characteristics:

- ARM Cortex-M3 processor running at 120MHz with Thumb-2 instruction set.
- 512 kB of on-chip flash memory.
- 64 kB of Static RAM.
- Multiple UARTs.
- an SD card interface that supports MMC cards.
- a standard JTAG test interface.

Inside the user manual we found different interesting sections and tables relative to memory addresses that would have become extremely useful later. image5.4 represents the internal structure of the microcontroller: the elements we focused on were the flash memory, RAM, and boot ROM positioned at the beginning of the memory. We used also CAN peripheral addresses and UART peripheral addresses for the reverse engineering of the code.
Where necessary we provide the required information about the microcontroller while analyzing the next sections.

5.4.2 *The Modem*

Chip B in figure5.3 is a SIM968 by SIMcom[35] GSM and GPS modem with the following characteristics:

- Quad-band GSM 850, EGSM 900, DCS 1800, PCS 1900.
- GPRS connectivity with uplink transfer up to 42,8 kbps and downlink transfer up to 85.6 kbps.

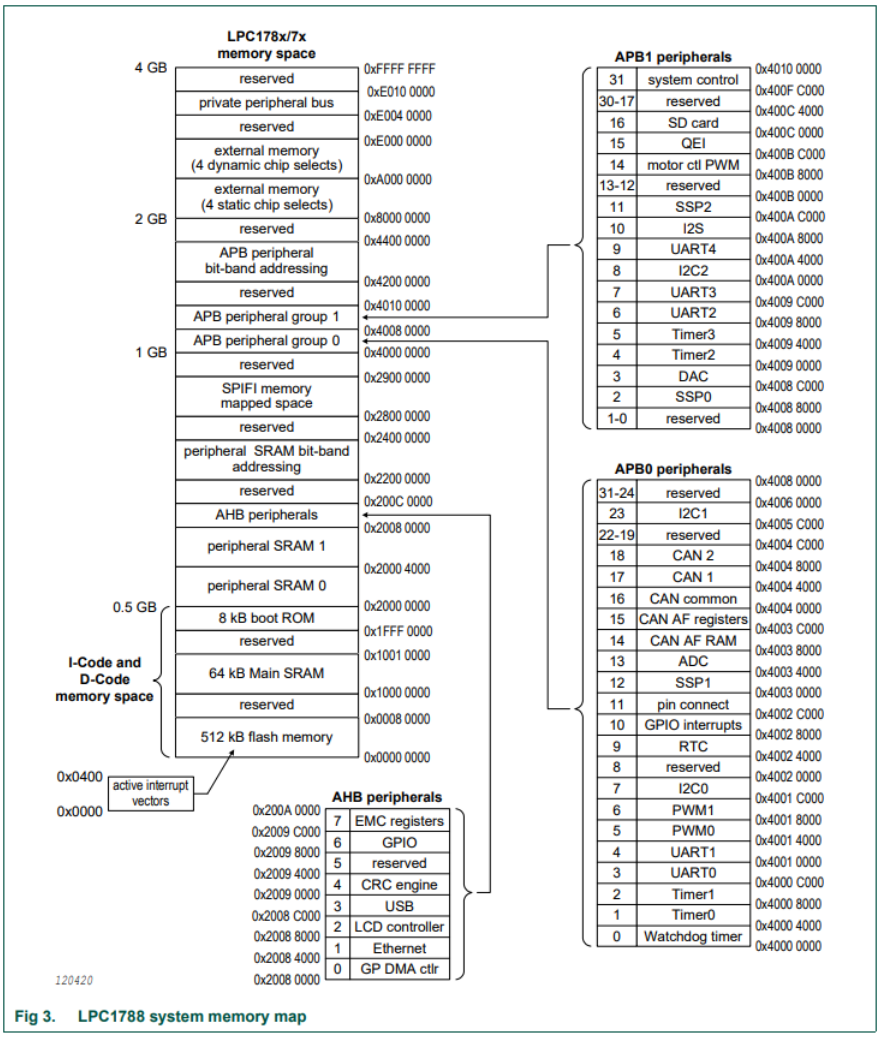

Fig 3. LPC1788 system memory map

Figure 5.4: The memory map of the LPC177 family of microcontrollers

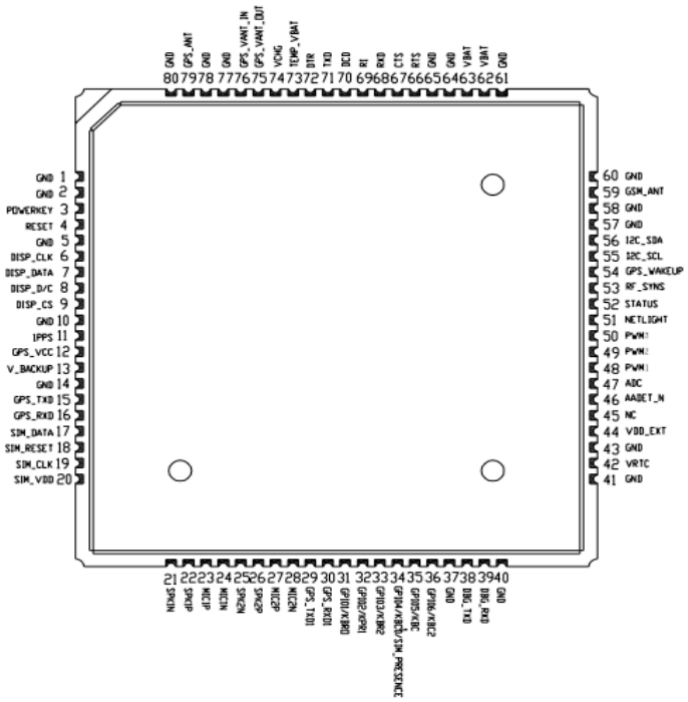

Figure 5.5: The disposition of pins on the surface of the modem

- Serial debugging port available for AT Commands or data streams.

In image5.5 we can see the pinout of the modem.
We did not analyze the physical connections of the modem due to the difficulties found in connecting to the pins without unmounting the modem from the board, which we preferred not to try as it could have easily lead to break it and at the same time it never appeared to be a useful step, if not for reverse engineering the modem itself which was not in the scope of the assessment.

5.4.3 *The External Memory*

Chip C in figure5.3 is a NAND Flash for embedded devices produced by Cypress[36] under the code S34ML01G1 with the following characteristics:

- 1Gb memory space.
- Open NAND Flash Interface (ONFI) 1.0 compliant.

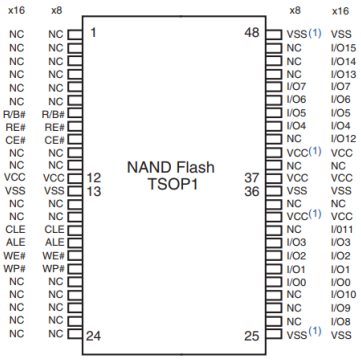

Figure 5.6: The disposition of pins on the surface of the external memory

- One Time Programmable Area.

In image5.6 we can see the pinout of the NAND memory.
As with the modem we did not proceed with physical connections directly to the memory due to the high risk of breaking it. We later discovered that the microcontroller has a direct interface to the memory (the SD card interface quoted earlier).

5.4.4 *The (two) UART Ports*

Pins D in figure5.3 are two different UART ports. To discover it we connected to the ports first a multimeter to discover the voltage of the various ports and then a logic analyzer, tried some of the standard baud rates to find the correct one (115200 bps) and at the same time checked the graphs in figure5.7 to understand which pins were ground, data out, data in and power.
The results were interesting: the ports were used to send AT commands from the microcontroller to the modem and in the opposite way. We used this information later while reverse engineering the code of the microcontroller knowing that we could find said strings.
Through the stream of AT commands we also discovered the PIN of the SIM installed on the device.5.7

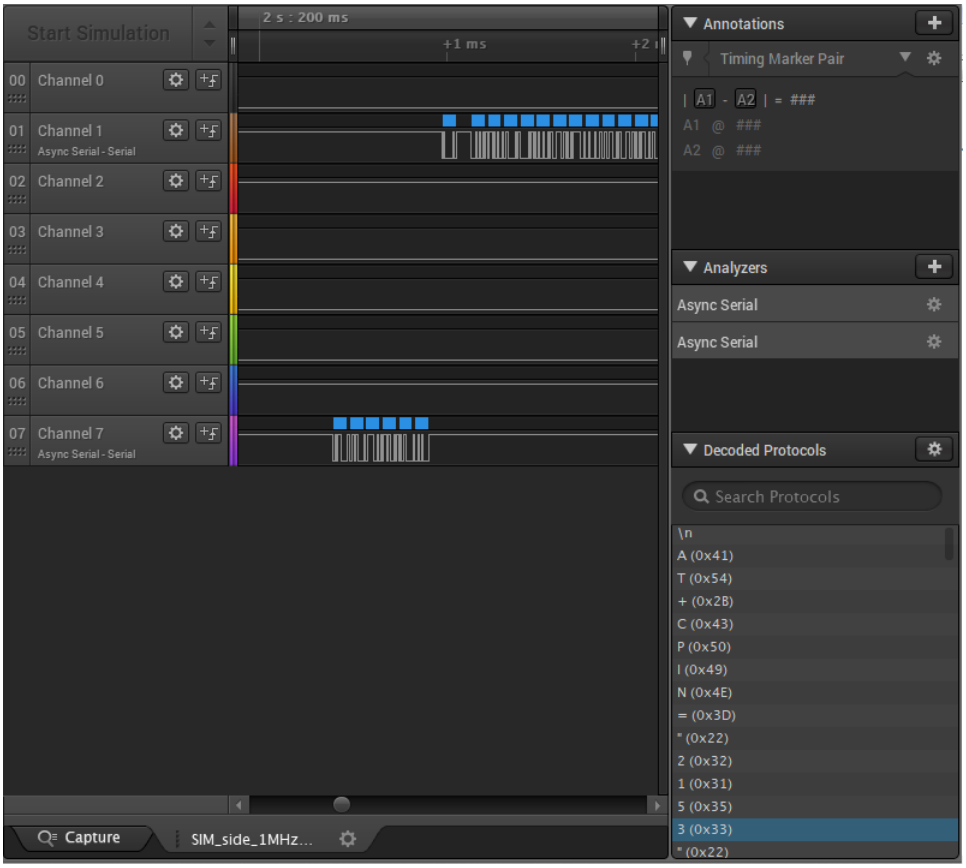

Figure 5.7: the logic analyzer output reading data passing through the UART ports. on the lower right corner it is possible to see the command“AT+CPIN =“2153”which sends the request to the modem to access the SIM card with pin 2153

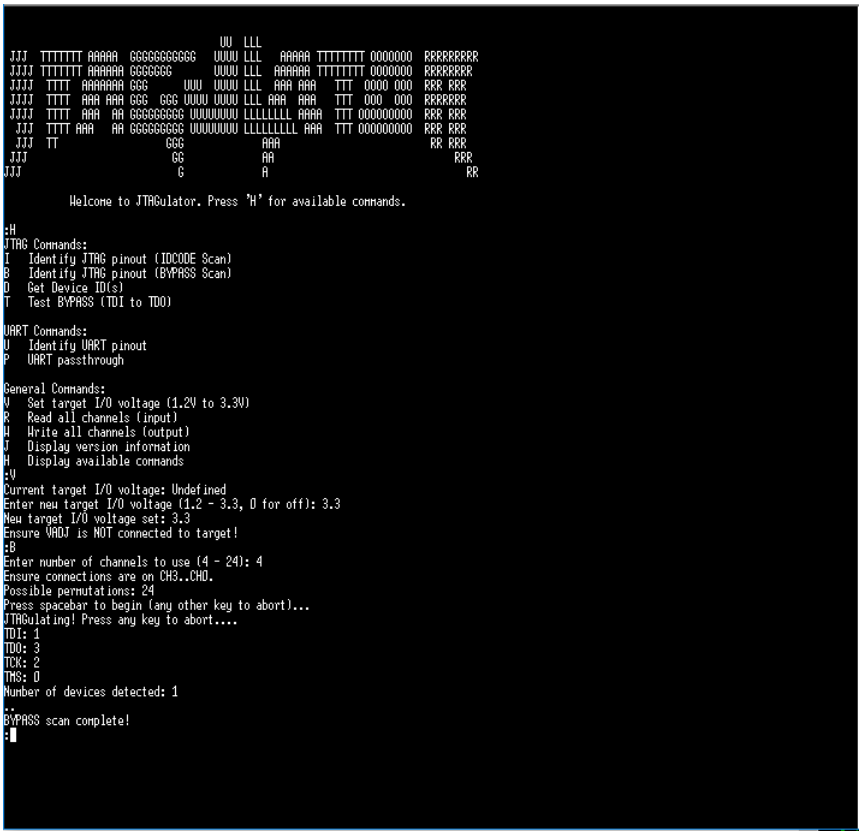

Figure 5.8: The jtagulator software sending the pinout of the JTAG port of the board through USB console.

5.4.5 *The JTAG interface*

Pins E in figure5.3 are a JTAG interface. Discovering this was not as easy as for the UARTs as we needed new instrumentation, in particular a Jtagulator[37], as finding the pinout of a JTAG interface is complex to do by hand: a JTAG interface is composed of at least five pins, ground, data in, data out, a clock and a reset. Many of these pins are not obvious to recognize with a logic analyzer as they do not have a fixed value, therefore the Jtagulator tries all combinations ensuring the safety of the device and outputs the correct pinout5.8. This was the step that gave us the possibility to retrieve the code of the microcontroller.

5.4.6 *Other Analyzed Elements*

We briefly describe the other elements of the board.

- F: Connector for power and external interfaces like CAN and UARTs. The Instruction manuals we received enabled us only to find CAN ports and power ones.
- G: GMS and GPS Antennas.
- H: SIM slot. The ECU was provided with a SIM card to provide GSM and GPRS connection.

5.5 CODE RECOVERY AND ANALYSIS

As soon as we discovered the microcontroller memory map we focused on trying to recover the code and data inside it. As the manuals highlighted a JTAG interface and we already found one on the board we started to set up the connection. We have been lucky in this case as both the Bus Blaster and the family of LPC17xx processors were already supported by openOCD, so we did not need to create new drivers for them. We proceeded to connect to the board and dump the data we needed: as stated above we already knew the starting and ending addresses of the flash memory, RAM and ROM so we simply dumped their images. We then used IDA Pro to disassemble these still unknown images, knowing that whatever they were they were running on an ARM-Cortex M3 processor.

IDA Pro is a powerful tool that amongst other functions can disassemble the code and automatically tries to decompile it to obtain a c-like pseudocode, but requires setup first to try to find functions, pieces of code and data. The setup phase of IDA required about a week: considering image5.9 the window on the left shows what IDA presents when opening the image of the flash of the board for the first time (usually the flash is used to store both code and data), the

window in the middle shows an intermediate step in which parts of the code have been recognized and the instance of IDA on the right is our final code. The bar on top of the three windows uses four colors: green represents where there are“unknown data", red is assembly code, while blue is a recognized function. The objective is usually to generate a completely blue bar, which we couldn't do as after a lot of tests small parts were still not recognized. The last green piece on the right of the finished analysis is instead just full of zeroes, as it was not used by the program.

To structure the analysis of the code, which took us a couple of months but could have gone on for much more time if we did not set some constraints to our work, we decided to focus on two main macro sections: First, the interfaces of the microcontroller like UART and CAN to understand the flow of the code from one to the other (we made an assumption at the beginning of the analysis which was that there should have been a flow of data from the CAN interface to the UART one which sent the AT commands to the modem, as the main objective of the ECU was to report information on the status of the car to the servers), second, the standard functions to read and manage strings, as AT commands were received from the modem as strings, with the objective to discover if some of them were coded badly and could lead to exploitation.

Through the documentation we discovered the UART and CAN Buffer Registers addresses and used them to define the read, write and interrupt functions of both the interfaces. In5.1 we show part of the code used for the interrupt handler of one of the UART interfaces (names of the variables were added by us while studying the code). To understand this code we check for MEMORY[0x4000C000] which is the UART0 register initial address. in 0x4000C0014 we find the Line Status Register. The first bit of the LSR indicates is the Receiver Data Ready flag (if raised, data are being stored in 0x4000C000 and waiting to get saved). Therefore this piece of code is the read from UART0 (it is implemented in a bigger function), which receives the data and

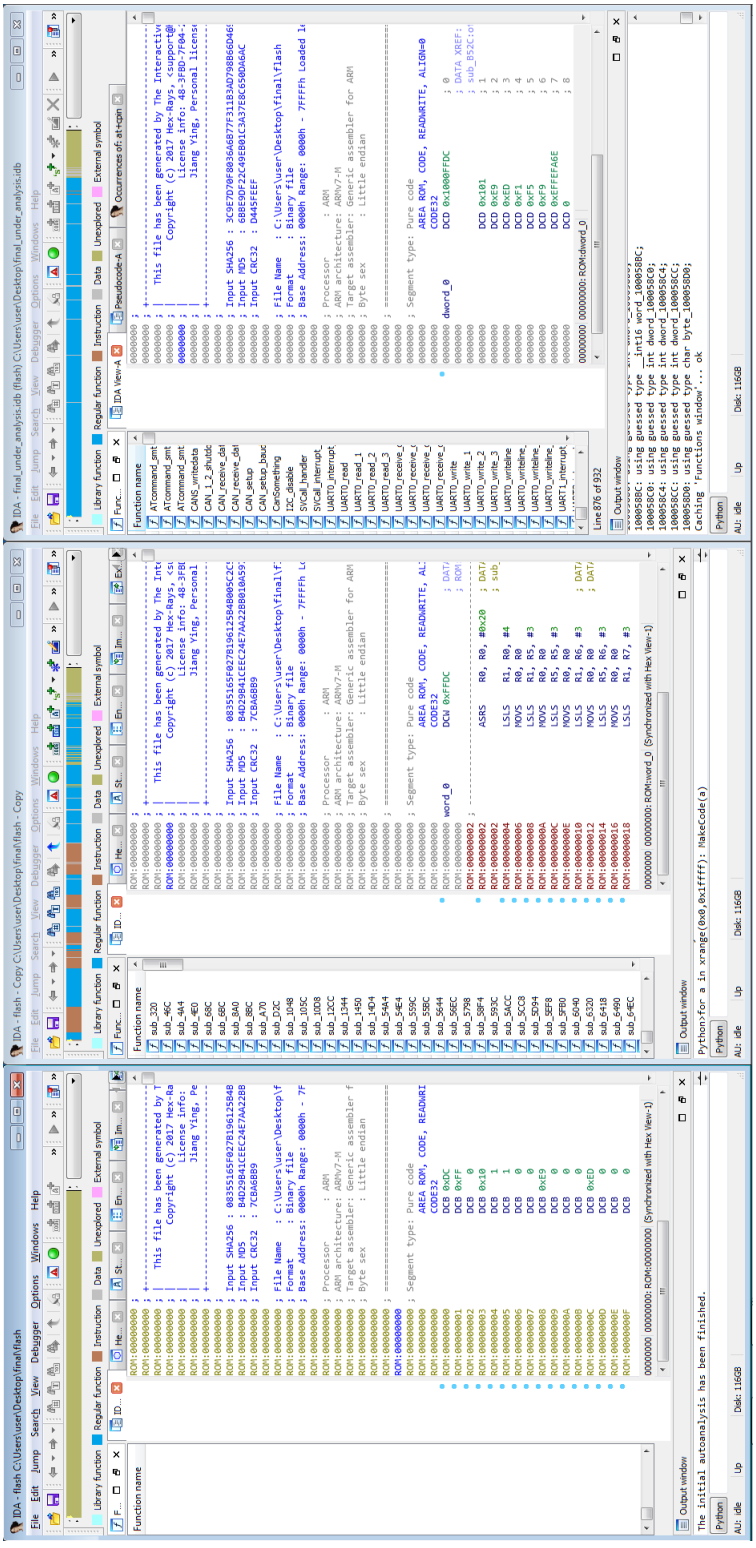

Figure 5.9

stores them in a circular buffer from which they will be handled and used.

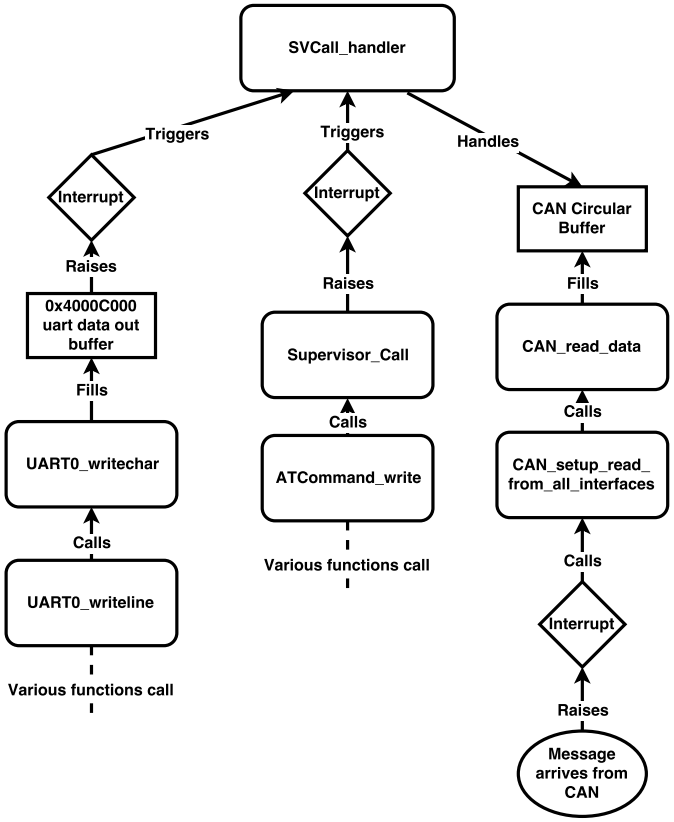

Figure 5.10: Flow of part of the analyzed functions. All lead to SVCall_Handler.

```
1  while ( MEMORY[0x4000C014] & 1 )
2  {
3    while ( 1 )
4    {
5      nextreadcounter = (readcounter + 1) & 0x1F;
6      uart0readBuffer[(unsigned __int8)readcounter] = ↩
           MEMORY[0x4000C000];
7      readcounter = nextreadcounter;
8      if ( nextreadcounter != v2 )
9        break;
10     ++word_10005E6C;
11     if ( !(MEMORY[0x4000C014] & 1) )
12       goto LABEL_9;
13   }
14 }
```

Listing 5.1: section of the interrupt handler

We repeated the same process for all the addresses that were indicated in the documentation relative to the UART and CAN interfaces and in the opposite direction from the AT messages we found as strings. We ended up creating a map of which functions were called from which that is represented in5.10, but as is visible we always ended up in a function we called SVCall_handler, which is a 1,217 lines function composed of one enormous switch case used to handle supervisor calls (calls to functions that require high privileges to be executed i.e. writes and reads to external interfaces). As it was not in the scope of the project to completely reverse the code and we estimated that the time required to completely understand that function would have approximately been a month, we moved on to the second macro section, which were string functions.

Relatively to the string functions we searched for small functions and analyzed them to understand how they were written and where they were called. We found a multitude of them, including strcpy5.2, printf, strcmp, memcpy and memset.

```
1  void __fastcall strcpy(_BYTE *dest, unsigned __int8 *source)
2  {
3    _BYTE *v2; // r3
4    int v3; // t1
5    v2 = dest;
6    do
7    {
8      v3 = *source++;
9      *v2++ = v3;
10   }
11   while ( v3 );
12 }
```

Listing 5.2: strcpy as coded in the board

On the hardware side of the analysis, although we were able to dump the majority of the areas of data on the microcontroller is hard to say that it is a flaw. JTAG is used for debugging and it is often necessary to the producer to install new firmware and check for errors in the code not only before sending

out the device but also if it breaks. Therefore it would be excessive to say that it is a bad habit, although it surely does not help on the security side.

We can hardly provide considerations on the security of the overall code, but in relation to string functions we can at least highlight one positive and one negative points: on the positive side the functions seem well written in the sense that there are no intrinsic flaws in them, and also the printf function is always called with both parameters (the actual"printed"element anticipated by the string defining the kind of data expected and their length). On the negative side instead there is no strncpy, but only strcpy, where the difference is that strncpy defines the length of the copied data first, while strcpy does not.

As a last consideration relative to the whole code, thanks also to openOCD which enabled us to look, step by step, at which line of memory was being executed we can say that the whole microcontroller is heavily reliant on interrupts. After the first boot which lasts up to thirty seconds, the code stalls between the same five instructions in an endless loop until an interrupt is raised either by the external interfaces of the internal clock, after which the board returns to the loop.

5.6 INTERACTIONS THROUGH THE EXTERNAL INTERFACES

Regarding the interaction with CAN there are two ways to connect the ECU to it: either directly connecting the CAN-H and CAN-L wires to the network or using the second, smaller device provided along the first, which can be connected to the ECU and to the CAN.

To understand better what was the second device exactly for we tried to dump its code too, but sadly it was not possible: we tried to find an accessible physical port through which we could connect with a JTAG debugger but the only available set of pins was not used as a JTAG interface. We then tried to read the code of the main chip on the device, even if it was scratched off5.11, and discovered that

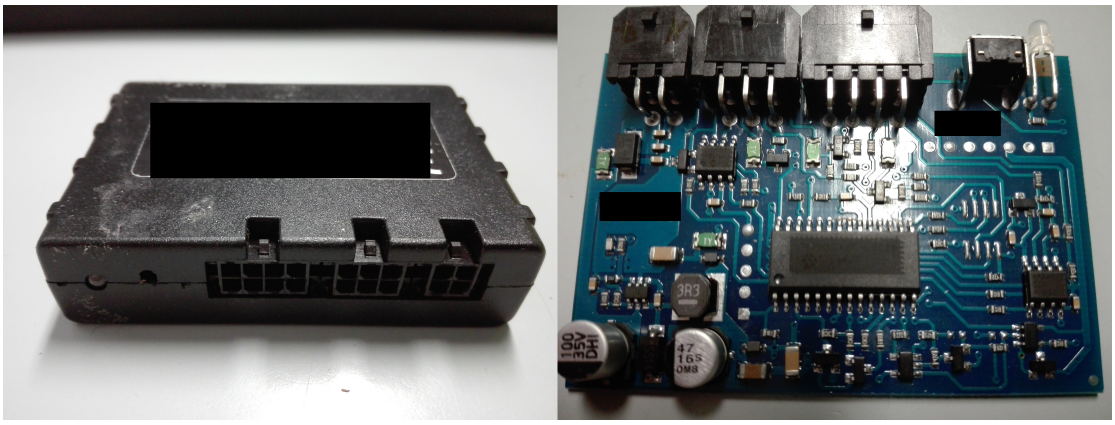

Figure 5.11: The smaller unit with and without cover. In the image on the right it is visible the removal of the code of the middle chip.

it was a PIC microcontroller similar to the one used in our ELM327 diagnostic device for OBD-II, and through it made the reasonable assumption that it does the exact same process and then relays the information to the main ECU.

The interaction with the CAN bus is made through OBD-II PIDs, which are codes used to request information on the status of the vehicle through the OBD-II port. Being diagnostic packets they are relayed by the main gateway of the car to the CAN bus. Each PID requests for a specific information, engine Repetitions Per Minute (RPM), Vehicle speed, engine oil temperature and so on. These data are then transmitted to the board which in turn sends them to the servers of *Company* that display them to the user.

The use of multiple devices to connect to the internal networks of the car is a strong security measure: for an attacker to manage to communicate with the bus he needs first to exploit the ECU connected to the cellular network and through that exploit the other one, reaching the CAN. Even if it has already been proved possible in[5] it still adds a consistent layer of security.

To analyze the security of the cellular connection we checked for the available communications of the modem, which are only GSM and GPRS. It does not support any data connection protocol higher than Second Generation (2G). We therefore opted to use a NUAND

Figure 5.12

BladeRF to generate a local cellular network, substitute the SIM card on the board with a custom one and try to connect to it and read through the USB interface of the BladeRF the data that the board was trying to send to *Company* servers. Sadly it was not feasible: as visible in image5.12, taken by the dump of the flash memory of the board, the only cellular providers provided accepted by the board are TIM, Vodafone and Wind: when we tried to use a custom SIM the board never managed to connect to our SDR.

Using only 2G connections enables many attacks on the cellular connections: 2G protocols do not provide any authentication requirements by the radio cell which connects the device to the network, easily enabling many kinds of attacks like Man In The Middle (MITM)

6

CONCLUSIONS

The automotive field is constantly developing, specially in the direction of V2X and autonomous driving. We here proposed the already known solutions for the already known flaws, but in few years the expected new development will bring new issues and new solutions on the security side. Up until now a complete survey has never been done but as in other fields there should be a constant update of documents like this one to constantly track which issues have been solved and which have been born, to help researchers in focusing on the most urgent requirements of the environment.
Generally speaking the security of the V2X environment is being taken into consideration through the design of the systems, but in the other sections of the field there are still many problems that should be tackled as soon as possible, as for instance finding the right authentication protocol for the CAN or CAN-FD protocols.
As of the main issues relatively to the on-board networks of the vehicle derive from the lack of preventively applied security in the design of the vehicle, we will focus on the creation of a system that helps the development of the on-board networks tackling their security side from various points of view.

This thesis has presented a survey of the whole automotive security environment, presenting a threat model for the current automotive system through which it highlighted the security requirements which have to be fulfilled in order to make the environment safer. These security requirements have been obtained through an analysis of the on-board networks and their connections with the outside environment and threats.

The thesis then described the currently proposed solutions for each of the threats highlighted in the threat model explaining how and if they are able to fulfill the security requirements. The described solutions cover the whole environment, from the on-board network protocols and new network structures to the new solutions already being developed for the V2X communication.

As the security of the internal structure of the single ECU is not considered in the survey due to the uncorrelation with the automotive environment and due to its correlation with traditional hardware and software security fields, the thesis proceeds to describe a specific case study of a currently on the market ECU of which a security assessment has been made which focused on the hardware, software and external connections of the device.

BIBLIOGRAPHY

[1] *OICA statistics*. http://www.oica.net/category/production-statistics/2017-statistics/ (cit. on p. 1).

[2] *The Carsharing Telematics Market*. Tech. rep. Berg, 2015 (cit. on p. 3).

[3] C. et al. "Comprehensive Experimental Security Analysis of Automotive Attack Surfaces." In: (2011) (cit. on pp. 4, 13, 16).

[4] V. Miller. *A Survey of Remote Automotive Attack Surfaces*. Tech. rep. 2014 (cit. on p. 4).

[5] Miller and Valasek. "Remote Exploitation of an Unaltered Passenger Vehicle." In: (2015) (cit. on pp. 4, 17, 61).

[6] *ISO 11898-2 (Controller Area Network)*. https://www.iso.org/standard/33423.html (cit. on p. 8).

[7] *IEEE 1609 DSRC (Dedicated Short Range Communication) standard*. https://standards.ieee.org/develop/wg/1609.html (cit. on p. 11).

[8] http://www.etsi.org/deliver/etsi_gts/07/0707/05.00.00_60/gsmts_0707v050000p.pdf (cit. on p. 12).

[9] http://www.etsi.org/deliver/etsi_gts/07/0705/05.00.00_60/gsmts_0705v050000p.pdf (cit. on p. 12).

[10] H. et al. "Security requirements for automotive on-board networks." In: (2009) (cit. on p. 13).

[11] *My Car My Data*. http://www.mycarmydata.eu/. Allgemeiner Deutscher Automobil-Club (cit. on p. 16).

[12] G. et al. "Lock It and Still Lose It - On the (In)Security of Automotive Remote Keyless Entry Systems." In: (2016) (cit. on p. 17).

[13] S. C. Aurelien Francillon Boris Danev. "Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars." In: (2010) (cit. on p. 17).

[14] Palanca. "A Stealth, Selective, Link-layer Denial-of-Service Attack Against Automotive Networks." In: (2016) (cit. on pp. 18, 22, 35).

[15] M. et al. "A Method of Preventing Unauthorized Data Transmission in Controller Area Network." In: (2012) (cit. on p. 22).

[16] P. et al. "The TESLA Broadcast Authentication Protocol." In: (2002) (cit. on p. 23).

[17] V. H. et al. "CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN Bus." In: (2011) (cit. on p. 24).

[18] G. et al. "LiBrA-CAN: a Lightweight Broadcast Authentication protocol for Controller Area Networks." In: (2012) (cit. on p. 24).

[19] H. F. A. Hazem. "LCAP - A Lightweight CAN Authentication Protocol for Securing In-Vehicle Networks." In: (2012) (cit. on p. 25).

[20] K. et al. "CaCAN - Centralized AuthenticationSystem in CAN." In: (2014) (cit. on p. 25).

[21] S. S. Q. Wang. "VeCure: A Practical Security Framework to Protect the CAN Bus of Vehicles." In: (2014) (cit. on p. 26).

[22] F. G. A. Radu. "LeiA: A Lightweight Authentication Protocol for CAN." In: (2016) (cit. on p. 27).

[23] *Automotive Gateways: Bridge and Gateway from FlexRay/CAN/LIN to AVB Networks* (cit. on p. 29).

[24] https://www.evita-project.org/ (cit. on p. 30).

[25] M. et Stabili. "Anomaly detection of CAN bus messages through analysis of ID sequences." In: (2017) (cit. on p. 32).

[26] W. et al. “Automotive Electrical and Electronic Architecture Security via Distributed In-Vehicle Traffic Monitoring.” In: (2017) (cit. on p. 33).

[27] K. et Kang. “Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security.” In: (2016) (cit. on p. 33).

[28] S. et Obermaisser. “Secure Automotive Gateway - Secure Communication for Future Cars.” In: (2014) (cit. on p. 33).

[29] W. et al. “A Practical Security Architecture for In-Vehicle CAN-FD.” In: (2016) (cit. on p. 36).

[30] *ISO 26262-1:2011 (Road Vehicles Functional Safety)*. https://www.iso.org/standard/43464.html (cit. on p. 36).

[31] Matzol. “Ethernet in Automotive Networks.” In: (2011) (cit. on p. 37).

[32] B. Corporation. “OPEN Alliance BroadR-Reach (OABR) Physical Layer Transceiver Specification For Automotive Applications.” In: (2014) (cit. on pp. 37, 38).

[33] B. et al. “A Security Credential Management System for V2X Communications.” In: (2017) (cit. on p. 39).

[34] https://www.nxp.com (cit. on p. 48).

[35] http://simcomm2m.com/En/ (cit. on p. 48).

[36] http://www.cypress.com/ (cit. on p. 50).

[37] http://www.grandideastudio.com/jtagulator/ (cit. on p. 53).