University of Modena e Reggio Emilia

XXXV cycle of the International Doctorate School in
Information and Communication Technologies

Doctor of Philosophy dissertation in
Computer Engineering and Science

Securing automotive communication: from intra to inter-vehicular networks

Francesco Pollicino

Supervisor: Prof. Mirco Marchetti
Ph.D. Course Coordinator: Prof. Sonia Bergamaschi

Contents

Abstract - English

Cooperative Intelligent Transportation Systems (C-ITS) are being adopted with the aim to reduce emissions, and commuting time, and increase safety, with direct benefits on the quality of life of people. However, the increasing adoption of C-ITS solutions requires the development of new network models that can satisfy the diverse requirements of C-ITS applications. Vehicular Ad-Hoc NETworks (VANETs) is one of these technologies designed to enable communication between road infrastructure services, road users, and vehicles. Once the VANET is deployed, Vehicle-to-Vehicle (V2V) communication allows the different entities to broadcast periodic messages to announce their state (such as position, heading, and speed) with transmission ranges up to 1000 meters in ideal conditions. However, designing and implementing this complex system is a challenging task. For example, VANETs must support a highly heterogeneous environment composed of vehicles and boards produced by different manufacturers, while also complying with the strict C-ITS constraints in terms of latency of the communication and dynamic configuration of the network. Moreover, to prevent attacks on the C-ITS, with consequences on the safety of all road users, all communication must be authenticated.

This thesis focuses on the security of automotive communication, ranging from a single vehicle to a complex communication network composed of vehicles and infrastructure. The first part of this thesis targets intra-vehicle communication presenting a benchmark framework designed to compare algorithms targeting the security of the communication between microcontrollers found in modern vehicles This framework is used to compare different CAN bus detection algorithms and test their performance on two different publicly available datasets. In the second part of this work, the focus shifts toward inter-vehicle communication in three macro areas. The first area regards the study of V2X applications and their utility in realistic scenarios, including both safety-related applications for emergency vehicles and commercial applications. The second area regards the application of cryptography to secure V2X communications. This field includes the analysis of the applicability of the current cryptographic standards for V2X communication, and the proposal of a new cryptographic scheme to improve their performance. The last contribution regards the development of a new class of internal dynamic attacks to VANET and the proposal of two new approaches for detecting misbehavior and position falsification attacks.

Abstract - Italiano

L'incremento nell'adozione dei Sistemi di Trasporto Intelligenti e Cooperativi (C-ITS) ha l'obiettivo di ridurre le emissioni, i tempi di percorrenza e aumentare la sicurezza, con benefici diretti sulla qualità della vita delle persone. Tuttavia, l'implementazione di soluzioni C-ITS richiede lo sviluppo di nuovi modelli di rete in grado di soddisfare i diversi requisiti delle applicazioni C-ITS. Vehicular Ad-Hoc NETworks (VANETs) è una di queste tecnologie, progettata per consentire la comunicazione tra l'infrastruttura stradale, gli utenti della strada e i veicoli. Le VANET sono progettate per migliorare l'esperienza di guida attraverso la comunicazione diretta tra i veicoli (V2V), consentendo alle diverse entità di trasmettere il proprio stato (come posizione, direzione e velocità) tramite messaggi periodici con distanze di trasmissione fino a 1000 metri in condizioni ideali. Tuttavia, le VANET devono supportare un ambiente altamente eterogeneo composto da veicoli e dispositivi elettronici di diversi produttori, rispettando anche i vincoli di latenza della comunicazione e configurazione dinamica della rete caratteristici degli scenari C-ITS. Inoltre, per prevenire accessi non autorizzati, è anche necessario che tutte le comunicazioni siano autenticate in modo da prevenire attacchi che potrebbero avere ripercussioni sulla sicurezza di tutti gli utenti della strada.

Questa tesi si concentra sulla sicurezza delle comunicazioni automobilistiche, considerando sia la rete interna di un singolo veicolo, sia reti di comunicazione composte da diversi veicoli e l'infrastruttura. La prima parte di questa tesi riguarda le comunicazioni all'interno del veicolo, presentando un framework progettato per confrontare algoritmi mirati alla sicurezza della comunicazione tra microcontrollori presenti nei veicoli moderni. Questo framework è stato utilizzato per confrontare diversi algoritmi di rilevamento di anomalie del CAN bus e testarne le prestazioni su due diversi set di dati pubblicamente disponibili. Nella seconda parte di questo lavoro il focus si è spostato sulle comunicazioni interveicolari e comprende tre macro aree. La prima riguarda lo studio di nuove applicazioni V2X e della loro utilità in scenari realistici, includendo sia applicazioni destinate ad aumentare la sicurezza stradale, sia applicazioni destinate ad offrire servizi di pubblica utilità. La seconda riguarda l'analisi dell'applicabilità degli attuali standard crittografici, sviluppati per garantire la sicurezza delle comunicazioni veicolari e la proposta di un nuovo schema crittografico per migliorare le prestazioni degli standard attuali. L'ultimo contributo riguarda lo sviluppo di una nuova classe di attacchi dinamici interni a VANET e la proposta di due nuovi approcci atti a rilevare comportamenti anomali dei veicoli.

Chapter 1

Introduction

The increasing popularity of connected vehicles allows to enable Vehicular Ad-hoc NETworks (VANETs), which is one of the intelligent components forming the Cooperative Intelligent Transportation Systems (C-ITS) with novel services and features that can improve the driving experience through the cooperation between road-infrastructure services, road users and vehicles. In this case, the network is designed to improve the automotive driving experience through communication among roadside infrastructure, road users, and vehicles. Once a VANET is in place, Vehicle-to-Vehicle (V2V) communications allow the broadcasting of periodic messages to announce vehicle position and speed, with transmission range up to 1000 meters in ideal conditions beyond line-of-sight. The current vehicular communication is based on two primary technologies: Dedicated Short Range Communication (DSRC) and Cellular Network. The first one (DSRC) is standardized and developed by different standardization bodies that are IEEE for the US, ETSI for Europe, and ARIB for Japan; while the Cellular-V2X (C-V2X) is based on Cellular Network and is mainly diffused in China. The wide adoption of VANETs will enable a new set of applications to reduce fuel/energy consumption and increase travel comfort and safety. Examples include green wave systems, smart lighting, smart parking, improving road management, decreasing driving time, and reducing emissions. Moreover, the resulting systems can provide increased safety guarantees by integrating novel Advanced Driver-Assisted Systems (ADAS) to control the vehicle's dynamics. However, designing and implementing this complex system is a challenging task. For example, the novel communication networks must support a highly heterogeneous environment comprising many vehicles and board manufacturers, and comply with the strict C-ITS constraints in terms of small latency and dynamic configuration of the network. Moreover, all communications must be authenticated to prevent unauthorized malicious attacks. As an example, attacks could affect ADAS and change the expected behavior of the connected vehicles.

This thesis focuses on the security of automotive communications, starting from the single vehicle and extending to automotive networks composed of vehicles and the infrastructure. The first part of the thesis focuses on the intra-vehicle communications of modern vehicles, particularly on CAN, one of the most deployed networking protocols for in-vehicular communications. The second part of the thesis focuses on inter-vehicles communications proposing novel V2X applications and security countermeasures.

1.1 CAN Contributions

The contributions of this thesis concerning CAN security regard mainly the gap in the literature between the presence of different proposals and the lack of standard metrics useful to compare them. Novel solutions designed to detect anomalies in CAN communication are not compared to existing solutions or are compared with naive detection metrics to demonstrate that a more complex solution is better for the detection task. In particular, in the vast majority of cases, the authors of a scientific paper do not disclose a reference implementation, thus requiring other researchers to re-implement the proposed algorithm. More often than not, the paper only includes a high-level description of the proposed algorithm that lacks many relevant details required for implementation. Finally, several papers omit essential aspects related to the proposed algorithm's tuning and training that strongly impact their detection performance.

Chapter 3.1 proposes a benchmark framework for CAN IDS with a detailed threat model, allowing the test and the comparison of anomaly detection algorithms and the analysis and the comparison of 4 IDS for the CAN bus against a set of anomalies representing the considered threat model [155].

Chapter 3.2 proposes as a first significant contribution the empirical and unbiased comparison of eight different time-based CAN anomaly detectors over two different datasets, an original one and one already publicly available. The second contribution to the state-of-the-art is to foster the reproducibility of similar studies. All the reference implementations of the detectors considered in this study and the novel dataset used to tune and test the detection algorithm are publicly released. This contribution allows all researchers and industry practitioners to fully replicate our research results, validate the correctness of our reference implementations, assess the dataset's quality, and quickly compare a novel proposal concerning the state-of-the-art. Finally, it highlights the limits of publicly available datasets used for the experimental evaluation of existing solutions against the dataset presented at first in this work. This analysis also demonstrates that it is crucial to identify a comprehensive threat model to demonstrate how anomaly detectors are affected by attacks, such as the effects of different cycle times and injection frequency on the overall performance evaluation [133].

1.2 V2X Contributions

The contributions of this thesis concerning V2X communications and V2X security are spread in three macro areas. The first contribution regards the study of V2X applications and their utility in realistic scenarios, including both safety-related applications and utility-related applications. The second regards the analysis of the applicability of the current cryptographic standards developed to guarantee security against external attackers and the proposal of a new cryptographic scheme to improve the performance of the current standards. The last contribution regards the development of a new class of internal dynamic attacks to VANET and the proposal of two new approaches for detecting misbehavior and position falsification attacks.

Chapter 4.1 presents the design of a novel architecture for flexible car sharing and rental services, which allows people to rent their own vehicles to other people through

intermediate brokers. These brokers must operate as authorization services and enforce the requirements defined by vehicle owners when allowing users to access the vehicles. Our proposal tackles security and privacy challenges related to the possibility that the involved entities do not always behave honestly and that vehicle users could be traced for commercial purposes. Our proposal represents a solution for building an accountable delegated authorization protocol that allows vehicle owners to expose the user's potential misbehavior and guarantee the users' privacy with respect to both the vehicle owner and the authorization service. In existing systems, users must submit their information, such as their identity and driving license to the service provider when they request a car-sharing service. The service provider verifies that the customer has the right and ability to drive as a valid driver. After that, the user can utilize the car-sharing service through a service provider [127].

Chapter 4.2 proposes a novel heuristic designed to exploit V2V communication to alert non-EVs of an emergency situation, which leads to a faster response from EVs and the implementation of a V2V communication scheme designed to increase the safety of non-EVs and reduce the arrival time of EVs. The proposed solution exploits the VEINS implementation of the full dedicated short-range communications (DSRC) - wireless access in vehicular environments (WAVE) stack to simulate realistic communication between the involved actors, including obstacle shadowing and other solutions that are commonly used to better represent realistic results. Moreover, the results are based on an experimental evaluation on real cities characterized by medium and high traffic volumes. Experimental results demonstrate that V2V communications in emergency scenarios can be used to increase the safety of non-EVs and decrease the response time of EVs [132]. Furthermore, it is essential to underscore that while this work may not be directly related to cybersecurity, it strongly reinforces the significant utility of V2V communications in enhancing the quality of services available today. As such, safeguarding these systems is of utmost importance.

Chapter 5.1 proposes four main contributions to state of the art. First presents an implementation of the Elliptic Curve Qu-Vanstone (ECQV) implicit certificate scheme and the Elliptic Curve Digital Signature Algorithm (ECDSA) that is compliant with the IEEE 1609.2 standard and evaluate its deployment on automotive-grade boards [131]. To the best of our knowledge, this is the first open implementation of implicit certificates for resource-constrained devices in terms of computational power and memory. Second, it investigates the feasibility of the implicit certificate scheme in multiple C-ITS scenarios characterized by different latency constraints identified by the National Highway Traffic Safety Administration (NHTSA) for safety-critical communications between vehicles [113]. Third, it analyzes the applicability of the automotive boards against realistic scenarios. The realistic scenarios are built on different areas of the city of Modena (Italy) and simulated using real traffic data provided by the municipality. These experiments highlight the limitations imposed by the constraints of several automotive-grade embedded platforms. The last contribution is the proposal of a prioritization strategy to improve the applicability of automotive-grade boards to real traffic scenarios. The proposed strategy adopts position-based heuristics based on information included in V2V messages. The proposal's effectiveness is addressed using simulations based on the considered traffic scenarios [129, 130].

Chapter 5.2 shows that a simple combination of EdDSA and ECQV is not possi-

ble and explores the proposal of an ECQV key generation procedure to account for specific characteristics of the Edwards curves and to guarantee high security and low computational overhead. Moreover, our proposal is a modification of the EdDSA signing operation, which guarantees compatibility with the existing signature verification operation. The design allows reusing cryptographic and mathematical sub-routines of existing EdDSA libraries and does not require novel implementations of complex algorithms [68].

Chapter 6.1 presents SixPack, a novel internal dynamic attack to V2V networks that interfeer with cooperation and safety applications. With SixPack, the attacker manages to signal the false presence of an obstacle on the road, hence requiring the activation of the braking system to avoid a collision, with a direct trigger of the antilock braking system (ABS) and corresponding propagation of this information to the nearby vehicles. The SixPack attack is composed of six different phases, in which the attacker fakes a sudden brake, the activation of the ABS, and the acceleration required to rejoin the fake representation of the vehicle with the real one, thus demonstrating the inability of the current state-of-the-art anomaly detectors against dynamic attacks – namely attacks that achieve their goal by mimicking realistic vehicle behaviors. We tested the implementation of the SixPack attack using the F^2MD test framework, demonstrating the evasion capabilities of the attack. The scope of this research is to shepherd future researchers to focus on more sophisticated attack scenarios, enabling faster adoption of secure VANETs applications [128, 171].

Chapter 6.2 presents the design of two new detection systems, namely Local and Global, for use in vehicular networks. The Local method utilizes V2V communication and performs algorithms within each vehicle, while the Global method utilizes the VANET infrastructure and V2I communication, executing algorithms on each RSU. These detection solutions utilize both data from broadcasted BSMs and analysis of the physical environment for optimal detection performance [172].

Chapter 6.3 proposes Miradouros, a novel mechanism for position verification of moving vehicles. The core idea is to extend the reach of vehicle awareness with transitive sensor readings, i.e., relying on sensors from other vehicles to confirm positions. This makes each vehicle into a vantage point for observation, a miradouro, that verifies the position of vehicles with its sensors and extends BSM with a list of vehicles whose position was confirmed by sensors. Our proposal exploits existing broadcast communication to carry additional information about the positions of the vehicles and their surrounding vehicles in a decentralized way. The received positions can be tagged as trusted if confirmed by sensors, plausible, or not trusted. This classification can improve the decision-making process of safety use cases [126].

Chapter 2

Background and related work

2.1 A primer on CAN bus

The Controller Area Network (CAN) is one of the communication protocols used between the Electronic Control Units (ECU) deployed within the vehicle [73]. CAN is one of the most deployed networking protocols for in-vehicular communications due to its high resilience to electromagnetic interferences and low implementation costs. The CAN bus design enables communication between the nodes without requiring a host computer. When two different nodes start transmitting a frame simultaneously, the node with the highest priority continues sending the frame without interruption. In contrast, the other node backs off and retries transmission later. It is a broadcast-message-based communication protocol, and transmission on the CAN bus uses a bit-wise arbitration method for contention resolution.

The CAN protocol defines 4 different types of frames with different usages, but only the *data frame* is used to transmit data between ECUs. The main fields composing the CAN data frame are the *identifier (ID)*, the *data length code (DLC)*, and the *payload (data)*. The *ID* identifies the CAN data frame and the content of its *data* field. Each ECU transmits only a limited set of messages with a particular ID, while receiving ECUs use the value of the ID field to select data frames relevant for their functioning. A message with a particular ID field value is always sent by only one ECU. The ID field is also used for arbitration of the CAN messages, where lower values of this field denote messages with higher priority. In the current standard for basic CAN communication, the ID field is defined with a size of either 11 (standard format) or 29 bits (extended format). Figure 2.1 shows the structure of an extended CAN message. Note that the extra 18 bits of the *extended* format (ID #2) are encoded separately from the 11 bits of the *standard* format (ID #1) for backward compatibility.

The *DLC* field encodes the number of bytes composing the *data* field and has a size of 4 bits. Since the maximum length of the *data* field is 8 bytes, valid DLC values range from 0 to 8, while values from 9 to 15 are left unused.

The *data* field encapsulates the information that the sender ECU transmits to other ECUs on the network. The data field is variable (from 0 to 8 bytes) and usually packs several different signals. The CAN standard leaves complete freedom to car manufacturers regarding these signals' structure, number, encoding, and semantics.

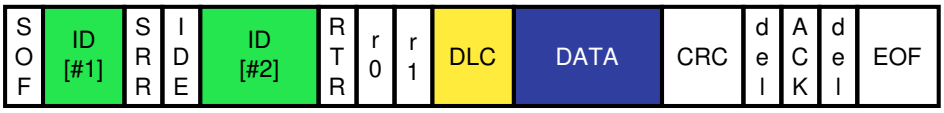


Figure 2.1: CAN data frame in the extended format

Hence, without having access to the formal specifications of the CAN messages for a particular vehicle model, the data encoded in the data field can only be interpreted as an opaque binary blob.

Between two consecutive CAN data frames, an *interframe space* is required. The *interframe space* consists of at least three consecutive recessive bits, called *interframe bits*. Following the interframe space, if a dominant bit is detected, it is considered the *start-of-frame* of the next data frame.

2.1.1 CAN bus related work

Since the introduction of microcontrollers to in-vehicle networks, the automotive domain is considered one of the most prominent examples of Cyber-Physical Systems (CPSs).

The main characteristic of an automotive CPS is the central role of the automotive in-vehicle network, in which the ECUs exchange data for their operational needs. With the increasing development of advanced driver assistance systems (ADAS), cyber-security researchers started demonstrating attacks on these features. Miller and Valasek [108, 109] demonstrated the consequences of an Internet-based, remote cyber-attack on a modern, unmodified, licensed vehicle with massive media coverage. Since then, many researchers have developed Intrusion Detection Systems (IDS) by applying concepts borrowed from classical computer networks to the in-vehicle networks [49, 85]. Some of these solutions focus on the analysis of the low-level characteristics of the ECUs [38, 93], and other solutions focus on the analysis of the in-vehicle network communications. Of this last group of solutions, security researchers have developed intrusion detection systems based on the statistical analysis of the content of the CAN bus [112, 105, 116]. In contrast, other solutions are focused on analyzing the content of the CAN data frames [100, 103, 152, 154, 162]. Detection algorithms based on the timings of the CAN messages represent one of the most analyzed group of solutions for the anomaly detection task on CAN communications, thanks to their flexibility that allows them to be deployed as a software-only solution without requiring dedicated hardware components for their functioning. The first work focused on this aspect is presented in [118], while following works are focused on either frequency [160, 111] or timing analysis [151, 74, 153]. All these works are based on the assumption that most CAN messages are sent periodically on the network within a fixed time interval; hence it is possible to exploit this feature to detect messages that do not follow the expected timing. However, these detection methods only apply to cyclic messages and cannot detect any anomaly if the attack targets a non-cyclic message.

Since most of these works only focus on a particular aspect of CAN communication, comparing novel solutions with existing literature is complex. Moreover, only a handful of current state-of-the-art implementations are publicly available, preventing the

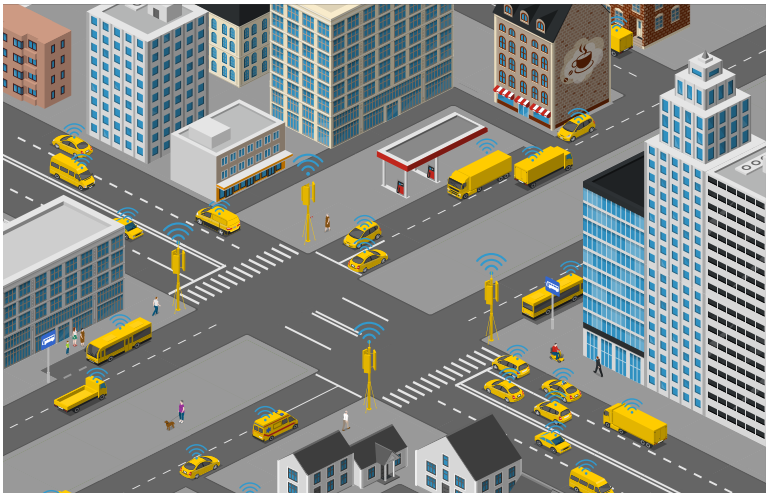


Figure 2.2: Example of C-ITS scenario

comparison of a novel solution with the existing ones.

2.2 A primer on V2X communications

Vehicular Ad-hoc Networks (VANETs) [161] are a prime example of communication between intelligent components in a Cooperative Intelligent Transportation System (C-ITS) [120]. In this case, the network is designed to improve the automotive driving experience through communication among roadside infrastructure, road users, and vehicles. VANETs are created by applying principles of mobile ad-hoc networks (MANETs) to the automotive domain, supporting heterogeneous environments and satisfying strict constraints such as high mobility, low latency, security, and dynamic network reconfigurations.

In VANETs, there is a Vehicle-to-everything (V2X) communication that includes all the communication between a vehicle and any entity that may affect, or may be affected by, the vehicle. It is a vehicular communication system that incorporates other more specific types of communication as V2I (vehicle-to-infrastructure) and V2V (vehicle-to-vehicle). An example of a C-ITS scenario is presented in Figure 2.2, where vehicles participating in the C-ITS share their information with the infrastructure through different antennas as access points. In the Figure, the V2I communication is between the Road Side Units (RSU) and the vehicles, while the V2V communication is between all the vehicles (cars, trucks, buses, ambulances, etc...). Once a VANET is in place, Vehicle-to-Vehicle (V2V) communications allow broadcasting of messages to announce the state of each vehicle, such as position and speed, with transmission range up to 1000 meters in ideal conditions, beyond line-of-sight. On the other hand, V2V communication is used to share information with RSU and notifies the driver about traffic and weather updates to keep an eye on the nearby environment.

The increasing adoption of VANETs will enable a new set of applications to reduce fuel/energy consumption and increase travel comfort and safety. Some examples are coordination between Emergency vehicles (EVs) and non-EVs, blind spot warnings, do-not-pass warnings, intersection crossing assistance, and general lane/road problems.

Typically, in these applications, each vehicle will receive messages through the VANET, using its sensors and plausibility filters to decide if there are any necessary changes to the status (speed or direction) or if it is necessary to alert the driver about some unexpected event. Other use cases can benefit from position verification, such as *platooning* [8], where the participants need to verify membership of other vehicles in the travel platoon and detect suspicious entries and exits – and *proof-of-travel* – where the past positions are used to assert correct transport of goods for vehicle inspections [140].

V2V data sharing has great potential to improve road safety, but it also raises security concerns.

External Security for a VANET is given by using cryptographic keys and algorithms to protect the exchanged messages, supported by a Public Key Infrastructure (PKI). Namely, the vehicle messages are digitally signed to assure their integrity and authenticity. Internal security is handled by various misbehavior detection systems [86] that detect incongruences in a series of position messages received by the vehicles, but they have only a local perspective. Current state-of-the-art anomaly detectors cannot detect some attacks [128, 96] and existing techniques for position verification [69, 64] are not directly applicable in VANETS due to the high mobility and speed of the nodes that frequently change the network topology.

2.2.1 V2X technologies overview

The current V2X communication is based on two primary technologies: Dedicated Short Range Communication (DSRC) and Cellular Network. The first one (DSRC) is standardized and developed by different standardization bodies that are IEEE for the US, ETSI for Europe, and ARIB for Japan; while the Cellular-V2X (C-V2X) is based on Cellular Network and is mainly diffused in China.

2.2.1.1 Dedicated Short Range Communication (DSRC)

The Dedicated Short Range Communication (DSRC) stack is currently a de-facto standard for V2V and V2I wireless communications. In the US, the With Wireless Access in Vehicular Environments (WAVE) [82] refers to the system designed to provide vehicles with direct V2X connectivity through DSRC, also referred to as the DSRC/WAVE protocol stack, while in Europe the DSRC V2X technology is referred to as ITS-G5 and it is standardized as ETSI EN 302 663 [59].

For the Physical (PHY) and Medium Access Control (MAC) layers, both the DSRC/WAVE and ITS-G5 are based on IEEE 802.11p [83]. The main difference is that the US standard the channel coordination is regulated by the IEEE 1609.4 [79] channel wireless radio operations over IEEE 802.11, while ITS-G5 employs a decentralized congestion control (DCC) method specified in TS 102 687 [63] to limit the channel load and allow high priority messages to be received with reasonable probability.

The network and transport layer standards support the Internet Protocol and the Transmission Control Protocol (TCP)/User Datagram Protocol (UDP). However, different protocols are also supported since V2X communication is mainly based on one-hop communications with packets sent directly over the air from the source to the

destination. The DSRC/WAVE supports the IEEE 1603.3 [80] that defines the WAVE short messaging protocol (WSMP). WSMP includes two types of messages that are the are the WAVE Short Message (WSM) and WAVE Service Advertisement Message (WSA). WSM messages are optimized for efficiency with a minimum overhead of 5 bytes and are used mainly to broadcast safety-critical messages. WSA provides information about one or more DSRC services offered in an area and are used for non-safety-critical applications like traffic alerts, navigation, and entertainment. The ITS-G5 supports the GeoNetworking routing protocol instead of WSMP. GeoNetworking is a routing protocol that provides packet delivery in an ad-hoc network exploiting the geographical position to facilitate sending packets to individual ITS stations or a target geographical area using broadcast or anycast (ETSI EN 302 931 []). The ITS-G5 alternative to WSM is the Basic Transport Protocol (BTS), which provides a connection-less, unreliable end-to-end packet transport similar to UDP.

At the application layer, the DSRC/WAVE stack includes 15 different types of messages [138]. For example, the *Map* message provides intersection and roadway lane geometry data for one or more locations, and the *TravelerInformationMessage* contains a variety of traffic conditions such as weather, local or regional emergencies, and speed warnings. The most used schema is the Basic Safety Message (BSM), which is designed for low latency and safety-relevant applications. The default size of each safety message is 254-byte and includes two types of data. The first part of a BSM message is mandatory and contains the core information about the vehicle (i.e., its size) and its status (i.e., speed, position, and accelerations). The second part is optional and adds a variable number of event-related data, such as notifications about the activation of safety-related subsystems within the vehicle (e.g., the activation of the ABS). On the other hand, ITS-G5 supports three types of messages that are: the Cooperative Awareness Message (CAM), which is the equivalent of the BSM; the Decentralised Environmental Notification Messages (DENM), which is an event-based message to inform about road hazards; and the Service Announcement Message (SAM) that is similar to WSA.

2.2.1.2 C-V2X

C-V2X is a promising alternative or complementary vehicular communication technology based on Cellular Network that is being developed as part of the overall Third Generation Partnership Project (3GPP) process to advance cellular systems from 4G to 5G technologies. It has been introduced in Release 14 of 3GPP [143, 144]. The primary goal of C-V2X technologies is to supersede the IEEE 802.11p/DSRC stack. We observe that C-V2X adopts communication channels with higher bandwidth with regard to DSRC in the context of V2I communications. However, the bandwidth of C-V2X compared to V2V communications is slightly better or comparable [11, 156]. Finally, we highlight that C-V2X replaces the PHY and the MAC layers and leverages all the existing standards in the applications, message/facilities, security services, and the Transport/Networking layers defined by SAE International, ETSI, and IEEE [135]. Since the works proposed in this thesis focus only on the characteristics of these higher layers, the proposed methodologies and solutions also apply to C-V2X. Due to the comparable performance in V2V communications, quantitative results based on simulations should also apply to C-V2X.

2.2.2 VANETs Security

The C-ITS security infrastructure comprises two principal components: the Public Key Infrastructure (PKI) and the Misbehavior Detection system. The PKI is responsible for the generation and the distribution of the cryptographic certificates required by the vehicles to ensure authenticity and integrity of the communications, while the Misbehavior Detection system is responsible for the evaluation of the truthfulness of the messages sent by the vehicles, reporting suspicious activities to the Misbehavior Authority. Moreover, to guarantee the required privacy for the communication between the entities, multiple pseudonyms are assigned to the vehicles to prevent malicious entities from easily associating a single pseudonym with a target vehicle [29, 30].

The cooperation between the PKI and the Misbehavior Detection system in securing VANETs communication is essential since the former component provides security guarantees against *external attackers* (as described in the IEEE 1609.2 security standard [81]), while the latter is responsible for the detection and mitigation of attacks coming from *insiders* (i.e., vehicles or infrastructure elements). An example of a Misbehavior Detection system is described in [86] and is composed by the Local Misbehavior Detection and by the Misbehavior Authority (MA). The Misbehavior Detection module runs on each C-ITS entity (i.e., vehicles and other connected objects participating in the V2X communication). It is responsible for the local analysis of the incoming traffic, generating security reports in case of suspicious activities. These reports are then sent to the MA, a global module responsible for the analysis and the security reports to identify anomalous behaviors from the vehicles participating in the V2X communication. The MA module is also responsible for the reaction upon detection of suspicious activities that could result in a certificate revocation request to the PKI for the misbehaving vehicle.

2.2.2.1 Outsider Threats

External attackers are entities that have not acquired valid credentials to participate in the communication (e.g., a valid certificate obtained by the PKI).

Availability. The availability is compromised when the continuity of the system is compromised, for example, when denial of service attacks or jamming attacks are in place. In these, a high volume of false messages is maliciously and artificially generated to flood the system to undermine the hard real-time requirements of safety applications. These attacks typically cause all the C-ITS participants to under-perform their normal functions slowing down or interrupting the receiving of the messages and the reaction time.

Integrity. The integrity of the information is compromised when an attacker can manipulate or corrupt the data without being noticed. It is accessed without authorization. Most common attacks rely on impersonation through masquerading attacks. Digital signatures, public key infrastructure, and cryptography revocation mechanisms may be employed to ensure the integrity between the sender and receiver.

Authenticity. The authenticity is compromised when an attacker can enter the network without a priori authentication phase. An attacker able to bypass the authentica-

tion mechanism can, for example, impersonate high-priority vehicles (e.g., ambulances) to obtain advantages and affect the traffic.

Confidentiality. The confidentiality is compromised when an attacker can obtain some information or link the available information to real users. An attacker can, for example, collects all the broadcast messages and track and profile the users. The privacy of all the participants can be protected using pseudonyms that hide the real identity from the other users and, simultaneously, allow authorized parties to request the user information from the issuer of the pseudonym in case of misbehavior.

Non-repudiation. This feature ensures that the source of the originating message may not deny the fact that it has generated a particular message. In legal scenarios, the ability to prove the occurrence of a specific event preventing users from denying their action is crucial. As for Confidentiality, pseudonyms allow authorized parties to link the user's real identity with the pseudonym in case of misbehavior.

Using suitable cryptography algorithms can mitigate some attacks (e.g., replay, eavesdropping, or message manipulation). However, physical attacks can only be prevented through secure hardware that prevents keys from being extracted from the vehicles or other components (e.g., RSU).

2.2.2.2 Insider Threats

An insider attack is a typology of attack in which the attacker has acquired valid cryptographic credentials from the PKI to participate in the communications. The attacker can also be classified as active or passive, malicious or faulty, and local or extended.

Active or Passive. An active attacker can forge messages by arbitrarily modifying the values of its fields and deciding if or when to broadcast a message. In the case of a passive attacker, it eavesdrops on the wireless network and collects information about the network without sending or receiving any message.

Malicious or Faulty. A malicious attacker interferes with the network to gain advantages or personal benefits. A faulty entity may send erroneous messages due to a technical fault or some circumstances of the environment (e.g., poor GPS signal).

Local or Extended. The difference between local and Extended is mainly on the attack's scope and area; it can target a single or small group of the local entity or can have a more extensive region or the entire network.

Suspicious activities can be detected by the Misbehavior architecture that could lead to a certificate revocation request to the PKI and a report to the competent authorities for the misbehaving vehicle.

2.2.2.3 Comparison of the security standards

The comparison of security specifications included in the IEEE 1609.2 and in the ETSI ITS standard is proposed in this section.

2.2.2.4 IEEE 1609.2

The IEEE 1609.2 [81] standard includes (I) the *security services* that all the devices must support, (II) the schema of the messages, and (III) the adopted cryptographic schemes and protocols.

Security services IEEE 1609.2 defines two types of security services: the WAVE Internal Security Services and the WAVE Higher Layer Security Services. The services composing the WAVE Internal Security Services are the *Secure Data Service* (SDS) and the *security management*. These two services convert Protocol Data Units (PDUs, unsecured by definition) into Secured Protocol Data Units (SPDUs) and vice-versa. The security services defined in the WAVE Higher Layer Security Services are the *Certificate Revocation List Verification Entity* (CRLVE) and the *Peer-to-Peer Certificate Distribution Entity* (P2PCDE). The former service is used to validate incoming *Certificate Revocation List* (CRL), forwarding the revocation lists to the *Security Services Management Entity* (SSME). In contrast, the latter service is used to enable Peer-to-Peer certificate distribution.

Message schemas The IEEE 1609.2 standard supports three types of SPDUs: unsecured, signed, and encrypted. Unsecured SPDUs does not provide any security guarantees. Signed SPDUs guarantee authenticity and non-repudiation and can be used for authorization mechanisms and Basic Safety Messages (which are not encrypted). Encrypted SPDUs guarantee confidentiality. It is possible to combine Signed and Encrypted SPDUs into a single SPDU through encapsulation to guarantee authenticity, non-repudiation, and confidentiality. We remark that signature verification is performed for each received message. In a V2V communication protocol, the number of messages received by any given vehicle is significantly larger than the number of messages sent within the network. Due to these characteristics, one of the most computationally expensive operations operated by vehicles is the verification of the digital signatures attached to Signed SPDUs.

Cryptographic schemes The IEEE 1609.2 standard requires adopting the Elliptic Curve Digital Signature Algorithm (ECDSA) instantiated with one out of three elliptic curves identified as *NIST-P256*, *BrainpoolP256r1* and *BrainpoolP384r1*. Moreover, IEEE 1609.2 requires a Public Key Infrastructure (PKI) to distribute public keys, where trusted *Certification Authorities* (CA) binds the identity information of communicating parties to their public keys within *certificates*. Two types of certificates are supported: traditional *certificate chains* and *implicit certificates* [33].

A certificate chain includes the public keys of the message's sender and all the intermediate CAs; hence the size of the certificate chain is proportional to the number of intermediate CAs. Verifying the certificate chain requires verifying the digital signatures attached by all the intermediate CAs up to the Root Certificate.

For implicit certificates, the IEEE 1609.2 standard specifies usage of the Elliptic Curve Qu-Vanstone (ECQV) scheme. Intuitively, an ECQV certificate does not include the sender's public key but allows a recipient to re-compute it by using the certificate of the sender and the public key of the CA, thus saving network usage. Network savings of implicit certificates can be considered negligible in most Web scenarios because communications are mostly high-bandwidth and can tolerate latency. Moreover, certificates

are only used once during the secure channels handshakes. However, vehicular networks are characterized by datagram-oriented communications that include small data packets, each including a digital signature and a certificate (especially safety-critical packets, see Section 5.1.1). Moreover, vehicular networks, especially vehicle-to-vehicle communications, are deployed on low-rate wireless networks and have tighter latency requirements. Hence, ECQV is the best fit for this scenario.

2.2.2.5 ETSI ITS Security

The ETSI ITS standard includes all the security specifications in different Technical Reports (TR) and Technical Specifications (TS). In the following is reported the list of the documents related to the security of the ITS systems with a brief description extracted mainly from the original documents.

Security services and architecture [53]. This document describes credential and identity management facilities, privacy and anonymity, integrity protection, authentication, and authorization. In particular, this document map the security services and their functional components to the ITS architecture and lays the foundation on which all other documents have been developed.

Threat, Vulnerability and Risk Analysis (TVRA) [58]. This document summarizes the results of a Threat, Vulnerability, and Risk Analysis (TVRA) of 5,9 GHz radio communications in an Intelligent Transport System (ITS). The analysis considers vehicle-to-vehicle and vehicle-to-roadside network infrastructure communications services operating in a fully deployed ITS. This document is based on the methodology described in the TVRA Technical Specification [57] (Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA)). With the Security services and architecture document, it identifies requirements for future work and lays the guidelines and principles on which all other documents have been developed.

ITS communications security architecture and security management [60]. This document specifies a security architecture for Intelligent Transport System (ITS) communications. Based upon the security services defined in [53] (Security Services and Architecture), it identifies the functional entities required to support security in an ITS environment and the relationships that exist between the entities themselves and the elements of the ITS reference architecture defined in [52] (Communications Architecture). This document also identifies the roles and locations of a range of security services to protect transmitted information and manage essential security parameters. These include identifier and certificate management, PKI processes and interfaces, and basic policies and guidelines for trust establishment.

Trust and privacy management [62]. This document specifies the trust and privacy management for Intelligent Transport System (ITS) communications. Based upon the security services defined in [53] (Security Services and Architecture) and the security architecture defined in [60] (ITS communications security architecture and security management), it identifies the trust establishment and privacy management required to support security in an ITS environment and the relationships that exist between the entities themselves and the elements of the ITS reference architecture defined in [52] (Communications Architecture). This document identifies and specifies security services for establishing and maintaining identities and cryptographic keys in

an Intelligent Transport System (ITS). Its purpose is to provide the functions upon which systems of trust and privacy can be built within an ITS.

Access control [54]. This document specifies authentication and authorization services to avoid unauthorized access to ITS services. It also specifies measures to ensure the required level of security and privacy for ITS message communication.

Confidentiality services [55]. This document specifies services to ensure that the confidentiality of information sent to and from an Intelligent Transport System (ITS) station can be maintained at a level acceptable to the station's users.

Security header and certificate formats [61]. This document specifies the secure data structure, including header and certificate formats for Intelligent Transport Systems. In addition to supporting the use of explicit certificates, this document includes the amendment for using implicit certificates. As such, this document is backward compatible for the sender of secure data structures but implies the additional implementation of implicit certificates on the receiver side to support interoperability. In particular, it is worth noting that the use of implicit certificates is not allowed in previous versions of this document. However, with the newer version, the ITS standard is aligned with the IEEE 1609.2 standard regarding certificates.

Mapping for IEEE 1609.2 [56]. This document maps all the similarities and differences of the IEEE 1609.2 [81] within the ITS communications architecture defined in [52] (Communications Architecture) to provide an implementation for a subset of the security services defined in [53] (Security Services and Architecture). This document identifies:

- Those areas where IEEE 1609.2 provides a security service defined in TS 102 731 (Security Services and Architecture).
- Those areas where IEEE 1609.2 needs to be extended or modified in a minor way to provide security services defined in TS 102 731 (Security Services and Architecture) and suitable for CAM and DENM.
- Those areas where IEEE 1609.2 does not provide a basis for a security service defined in TS 102 731 (Security Services and Architecture) and consumed by CAM and DENM.

In those cases where IEEE 1609.2 does not fully provide a required service, this document identifies the requirements for that service but does not specify that service in full. This document should therefore be seen not as a full security specification for CAM and DENM but as a subset of that specification. In particular, this document helps to implement the security requirements compatible with both standards.

2.2.2.6 ECQV implicit certificates

A certificate chain includes the public keys of the message's sender and all the intermediate CAs; hence the size of the certificate chain is proportional to the number of intermediate CAs. Verifying the certificate chain requires verifying the digital signatures attached by all the intermediate CAs up to the Root Certificate.

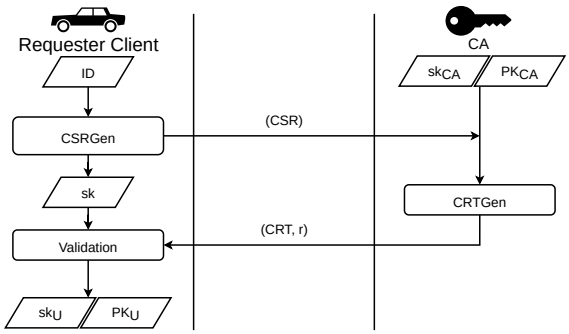


Figure 2.3: Operation flow for the generation of the ECQV certificate

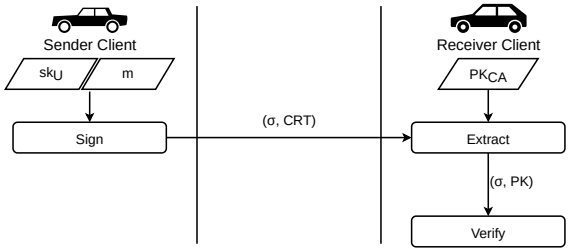


Figure 2.4: Operations flow for the ECDSA signature based on ECQV certificates

The implicit certificate scheme uses a more complex approach that leverages particular mathematical properties to bind identity information and public key *without explicitly* storing them. The most popular protocol for implicit certificates is ECQV [33]. Intuitively, an ECQV certificate does not include the public key of the sender but allows a recipient to re-compute it by using the certificate of the sender and the public key of the CA, thus saving network usage. Although implicit certificates seem very convenient thanks to their space efficiency, they have a few disadvantages that limit their adoption in common Web communications. Network savings of implicit certificates can be considered negligible in most Web scenarios because communications are mainly operated through channels exchanging large amounts of data, and certificates are only used once during the secure channels handshakes. However, vehicular networks are deployed on possibly low-rate wireless networks and have tighter latency requirements. As a result, network savings of implicit certificates might be worth the more complex management procedure and reduced flexibility.

The ECQV operations framework includes four routines: *certificate sign request*, *certificate generation*, *certificate reception*, and *public key extraction*. Combined with the *signature* and *verification* procedures of ECDSA, these operations guarantee message integrity and authenticity. The following describes the flow of the operations from the generation of an ECQV certificate to the signature verification by referring to Figures 2.3 and 2.4.

A *requester* is a client that requests a valid certificate to the CA by generating a *Certificate Sign Request* (CSR) via the *CSR generation* function ($CSRGen$). The $CSRGen$ function requires the id of the entity requesting the certificate, and produces an output composed of the CSR (that includes id and PK, which is an intermediate public key of the requester) and the intermediate private key sk. The CSR is sent to the CA and, upon identity verification, produces the corresponding certificate CRT

and a private key contribution r by using the *CRT generation* function ($CRTGen$). The $CRTGen$ function requires the CSR and the key pair of the CA sk_{CA}, PK_{CA} to produce the CRT. After receiving the CRT from the CA, the requester validates the certificate with the *CRT Validation* function ($CRTRec$). Upon verification, the requester generates the final key pair (sk_u, PK_u) by using CRT and r. The private key sk_U is used generate the signature σ of the messages with the ECDSA *signing* function ($Sign$). The client uses the private key SK_U with the ECDSA *signing* function ($Sign$) to sign messages. After the signature *sigma* is computed, the client sends the message m, the signature σ, and its own certificate, CRT, by broadcasting it to the nearby clients.

2.2.2.7 Security Credential Management System (SCMS)

The Security Credential Management System (SCMS) is a public key infrastructure (PKI) designed to secure V2X messages with privacy as the highest priority. The SCMS structure must mitigate attacks on end-users privacy from SCMS insiders and outsiders. The POC has been officially adopted as a protocol by the United States Department of Transportation (USDOT) [163] and became an industry standard for all providers of PKI-based V2X security solutions, including AUTOCRYPT.

The architecture described in [20], is similar to the European C2X PKI described in [17]. In the following, there is a description in terms of components, certificate types, and operations of the SCMS that are useful for a better comprehension of the works presented in this thesis, in particular of 4.1.

The SCMS includes a large set of components, but the focus is only on a set that is useful for comprehending the works in this thesis. For a complete reference of all the components and actors, please refer to the original work [20].

- *enrollment certification authority (ECA)*: issue enrollment certificates that the users use to authenticate against the registration authority, there may be more ECA divided, for example, by geographical region;
- *registration authority (RA)*: creates individual requests for certificates to the PCA, ensuring that revoked user cannot require new certificates and that a user cannot request more than one set of certificates for a given period,
- *pseudonym certification authority (PCA)*: issue short-term pseudonym certificates to users;
- *intermediate CA*: shield the root CA from traffic and attacks;
- *root CA* is the root at the top of the certificate chains in the SCMS.

The SCMS supports different requirements for certificate management and different type of certificates in the following three certificates types that are of interest for the contribution of the work:

- *enrollment certificates* are provided during bootstrap and are used to request message signing and/or encryption certificates;

- *pseudonym certificates* provides pseudonymity, unlinkability, and efficient revocation of a multitude of certificates;
- *identification certificates* are classic certificate that identify and entity.

In particular, the focus is on pseudonym certificates used to grant authorization when unlinkability is required. A pseudonym certificate does not include any real-world identifier, and a given user can have a set of multiple simultaneous valid pseudonym certificates that cover an interlaced period.

The following are the main operations of the SCMS that allow managing users' certificates from enrollment to (eventually) revocation.

- *Bootstrapping*: provides all the information required to communicate with the SCMS. Bootstrapping consists of two operations: initialization and enrollment. With the first one, the user obtains certificates, it needs to be able to trust received messages; with the second, the user obtains an enrollment certificate that is used to sign messages to the SCMS.
- *Pseudonym certificate provisioning*: is designed to protect the privacy of the end-users and is the most complex process in the SCMS. The privacy mechanisms included in the SCMS allow to obscure the physical location of the user to the RA and the MA; hide certificates from the RA in a way that no one can correlate the public key seed with the resulting certificates; hide receiver and certificate linkage from PCA in way that the PCA cannot correlate two certificate requests to the same user.
- *Linkage values*: are coupled to a set of pseudonyms and are used to revoke all the certificates with validity equal to or later than some time.
- *Misbehavior*: allows reporting, investigation, and eventually reaction to misbehaving users. The reaction includes revocation and blacklisting all the certificates corresponding to the linkage values of the incriminated pseudonym certificate.

Chapter 3

Securing CAN communications

3.1 A Benchmark Framework for CAN IDS

This chapter presents a benchmark framework for Controller Area Network (CAN) Intrusion Detection Systems (IDS). This framework ships with a dataset and currently supports four detection algorithms explicitly designed for the CAN bus. The dataset is composed of a set of traces gathered from a licensed, unaltered passenger vehicle used for training the detection models and by a set of traces containing simulation of the most common attack scenarios affecting modern vehicles. The detection performance of the four supported detection algorithms allows comparing their behavior against different attack scenarios, highlighting the best detection algorithms against the different attack scenarios.

3.1.1 Threat model

The threat model considered in this work is based on different attack scenarios already published by security researchers in technical reports and white papers [165, 109, 110, 88, 40, 119].

Replay Attack The replay attack is used to inject messages on the CAN bus to subvert its normal behavior by exploiting modern drive-by-wire capabilities such as *brake-by-wire* or *steer-by-wire*. The replay attack sequence is usually selected after an initial phase of reverse engineering of the values encoded in the payload, allowing the attacker to inject messages whose content is expected to command part of the vehicle dynamics. For this work, are considered two different typologies of replay attack based on the length of the injected message sequence:

1. **Single ID Replay:** A Single Message ID is injected on the normal flow of the CAN bus with a particular frequency;

2. **Sequence Replay:** A sequence of messages is injected into the normal flow of the CAN bus with a particular frequency. The length of the injected sequence is a parameter that will be inspected in the detection process.

Fuzzing Fuzz-testing is a technique *for discovering faults in software by providing unexpected input and monitoring for exceptions* [158]. Fuzz testing, usually involves (automatic) generation and sending of malformed input to software being tested in order to observe its behavior. In the context of internal automotive networks, fuzzing can involve the injection of CAN messages with malformed values in the data frame's fields. For this particular work, two different types of fuzzing techniques will be employed.

1. **ID Fuzzing:** CAN data frames with the ID and data field randomly generated. The ID field is chosen to be different from the values of IDs observed in the dataset;
2. **Payload Fuzzing:** CAN data frames with valid ID field and randomly generated data field.

Disruption Attack The disruption attack comprises all the other attacks focusing on the interruption of the normal operation of either the network or its components [42]. In the automotive scenario, a disruption attack can either target the network or its microcontrollers. For this work, it is considered the following disruption attacks:

1. **Denial-of-Service:** CAN data frames with the highest priority are injected with high frequency, thus preventing any other node from starting transmission of other messages;
2. **ECU inhibition:** messages generated by a target ECU are removed by sending the ECU in a *bus-off* state, thus preventing the target ECU from participating in the normal CAN communication.

3.1.2 Anomaly Detectors

The anomaly detectors supported by the framework are representative examples of different detection algorithms designed to analyze the different fields of CAN messages.

Message sequence algorithm The *message sequence algorithm* [104] is based on analyzing the message IDs. This algorithm analyzes the possible transitions between pairs of IDs and later uses the list of all the possible transitions for its detection purposes. This algorithm has no configuration parameter, so it is implemented as-is.

Bus entropy algorithm The *bus entropy algorithm* [106] is based on the analysis of the information entropy of the bus for detection purposes. This algorithm uses the entropy evaluated over a time window t to define the normal entropy range $[\mu_e - k\sigma_e, \mu_e + k\sigma_e,]$, where μ_e is the mean entropy of the time windows, σ_e is its standard deviation, and k is a tuning parameter. In the experimental evaluation it is used a time window $t = 0.01$ seconds, resulting in $\mu_e = 2.7225$, $\sigma_e = 0.3176$, and with a value of $k = 3$ to achieve 0 false positives in the validation process.

Hamming distance algorithm The *Hamming distance algorithm* [154] is based on the analysis of the Hamming distance of consecutive payloads of messages with the same ID for detection purposes. This algorithm evaluates the Hamming distance between consecutive payloads of messages with the same ID for the definition of the

normal model. The normal model is composed of the pair $[min, max]$ values for each message ID, representing the minimum and maximum Hamming distance between consecutive payloads of the same message ID. This normal model is later used for detection purposes. An anomaly is raised if the Hamming distance evaluated between two consecutive payloads of messages with the same ID is outside the $[min, max]$ range.

Missing message algorithm The *missing message algorithm* [153] is designed to detect missing messages from regular CAN communication. This algorithm evaluates the cycle time ct of each message ID. It later uses it for the definition of the valid *waiting time* ($ct^{ID} * k^{ID}$) that normally occurs between two consecutive messages with the same ID, where k^{ID} is a configuration parameter. The validation process on the clean dataset achieved zero false positives with a maximum value of $k^{ID} = 2$.

3.1.3 Ventus dataset

The CAN data is recorded by physically connecting a laptop to the On-Board Diagnostic (OBD-II) port with a PCAN-USB adapter by Peak System [121] and a D-Sub to OBD-II cable. The high-speed CAN bus segment exposed on the OBD-II port of the vehicle contains data related to the *powertrain*; hence it is possible to access CAN data frames exchanged by the ECUs to control the dynamic of the vehicle. The Ventus dataset is composed by the *clean* and the *infected* sections. The former is used for training and validating the detection algorithms, while the latter is used for the performance evaluation of the detection algorithms. For the generation of the *infected* dataset, a threat model is built based on the attack scenarios considered by the analyzed algorithms. The final dataset is publicly available at [125].

Clean dataset description

The clean dataset comprises 7 different CAN traces, including more than 8 million CAN messages corresponding to approximately 90 minutes of CAN traffic. The CAN traces are gathered in different driving sessions performed on different road types (urban, suburban, and highway), traffic conditions, weather conditions, and geographical areas (plain, hill, and mountain). The CAN traces include ID, DLC, and payloads of each CAN data frame associated with its relative timestamp. The clean dataset includes 51 different message IDs, each characterized by its cycle time. The cycle time of each message ID is available to car makers and their suppliers in the *DataBase for CAN (DBC)* file, which describes details of CAN communications. However, this file is kept confidential and is not publicly available. Since the cycle time of each message might influence the detection outcome, it is necessary to estimate the cycle times of the messages in the clean dataset to define multiple attack scenarios, each targeting a message with a different cycle time. The cycle time of each message is evaluated as the mean value of the inter-arrival times between two consecutive messages with the same ID and is rounded to the nearest millisecond.

Figure 3.1 shows on the y-axis the cycle time ct (expressed in milliseconds) evaluated for each message ID (depicted on the x-axis) on the clean dataset. As shown in Figure 3.1, the clean dataset is composed of 49 messages exposing a cyclic behavior, while 2 messages are identified as non-cyclic. The 49 cyclic messages can be grouped

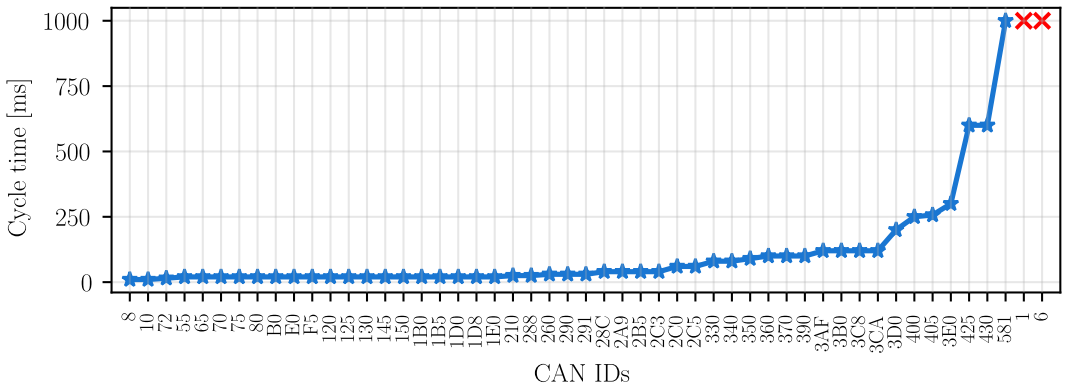


Figure 3.1: Cycle time evaluated on the Ventus clean dataset

in 17 different cycle time classes, ranging from a minimum of 10 milliseconds (IDs $0x8$ and $0x10$) to a maximum of 1 second (ID $0x581$). Note that these results are achieved on the clean dataset. The cycle time evaluated on a single trace of the clean dataset presents extremely low variance compared with the cycle time evaluated on the other traces. This confirms that the results presented in Figure 3.1 are representative of the real cycle time of the messages gathered from the same vehicle model.

Infected dataset description

The infected dataset is created by simulating the attacks described in the considered threat model (see Section 3.1.1) on the clean dataset. IDs with different probability distributions are used to simulate attacks on the clean dataset to avoid any possible bias toward unrealistic detection results due to the different frequencies of messages composing the clean dataset. For the simulation of the different attacks, 4 different message IDs are selected to represent the most frequent message (CAN ID `0x10`, cycle time of 10 milliseconds, labeled as `top`), a medium frequency message (CAN ID `0x145`, cycle time of 20 milliseconds, labeled as `mid`), a low frequency message (ID `0x210`, cycle time of 25 milliseconds, labeled as `low`), and a non-periodic message (ID `0x1`, undefined cycle time, labeled as `not`).

The infected dataset is composed of the following attack scenarios:

- **Replay attack - single ID:** set of traces in which a single valid ID is injected. This set is composed of 4 different traces, each corresponding to the injection of one of the 4 selected IDs.
- **Replay attack - valid sequence:** set of traces in which a sequence of IDs observed in the traces is injected. Different traces are generated by injecting sequences with different lengths, ranging from 2 to 10.
- **Replay attack - invalid sequence:** set of traces in which a randomly generated sequence of valid IDs is injected. Different traces are generated by injecting sequences with different lengths, ranging from 2 to 10.
- **Fuzzing - random ID:** set of traces in which a single invalid ID is injected.

- **Fuzzing - random payload:** set of traces where a single valid ID is injected with randomly generated payloads. The valid ID used for the injection of a random payload is selected using the same criteria used for the single ID replay attack.
- **Disruption - Bus DoS:** Set of traces where 100 messages with the same ID are injected each second.
- **Disruption - ECU inhibition:** Set of traces where CAN messages with a particular ID are entirely removed. This attack scenario represents an attacker that can disable a target ECU permanently. This set is composed of 4 different traces, each corresponding to removing one of the 4 selected IDs.

3.1.4 Performance evaluation

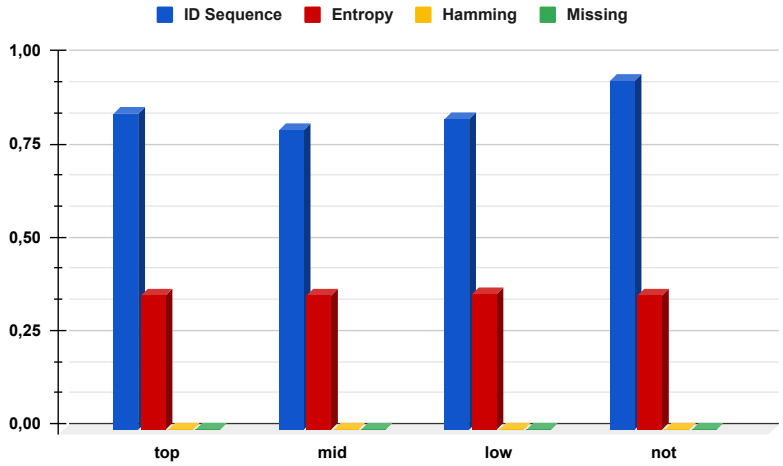
The detection performance of the algorithms supported by the framework is evaluated through $\mathcal{F}-$measure, a statistical index representing the accuracy of a test. The $\mathcal{F}-$measure is the harmonic mean between the *precision* (i.e., the number of correctly identified anomalies on the total of detected anomalies, hence including also the false positives) and the *recall* (i.e., the number of correctly identified anomalies on the total of actual anomalies, hence including the false negatives). Its value ranges from 0 to 1, where 0 denotes the inability of the detector to identify anomalies in the data, while 1 denotes the ability to identify all the anomalies and only the anomalies in the data. All the detection algorithms are implemented using the Python programming language on a computer equipped with an Intel® Core™ i7-7700HQ CPU @3.8 GHz and with 16 GB of RAM running Fedora 33 x64.

Performance against replay attacks

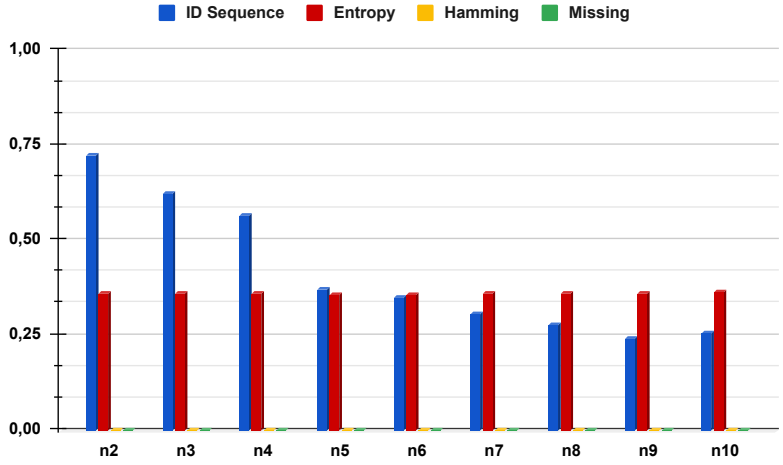
The detection results evaluated with the framework against the replay attacks are depicted in Figure 3.2, showing the $\mathcal{F}-$measure of the supported algorithms against the single ID replay attack (Figure 3.2a), valid sequence replay attack (Figure 3.2b), and invalid sequence replay attack (Figure 3.2c).

Single ID replay detection results

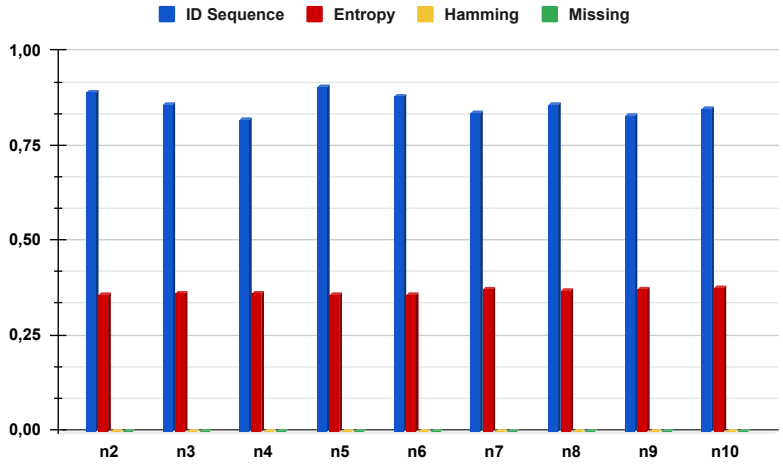
Figure 3.2a shows the detection results of the detection algorithm against the single ID replay attack. The x-axis of Figure 3.2a shows the labels representing the different replayed IDs, while the y-axis shows the $\mathcal{F}-$measure evaluated using the algorithms. The four vertical bars represents the four detection algorithms: *message sequence* (blue bar), *bus entropy* (red bar), *Hamming distance* distance (yellow bar), *missing message* (green bar). From the analysis of the results presented in Figure 3.2a we notice that the message sequence detection algorithm can constantly identify the injection of a single message ID. In contrast, the bus entropy detection algorithm only detects a limited subset of the injected messages. Moreover, these results are independent of the injected message (despite being higher in the case of injection of the `not` message). In contrast, the Hamming distance and the missing message algorithms cannot detect any anomaly.



(a) Detection performance against the single ID replay attack



(b) Detection performance against the valid sequence replay attack



(c) Detection performance against the invalid sequence replay attack

Figure 3.2: Detection results of the supported algorithms against the message replay attacks (top to down: single ID replay, valid sequence replay, invalid sequence).

Valid sequence replay detection results

Figure 3.2b shows the detection results of the detection algorithm against the valid sequence replay attack. The x-axis of Figure 3.2b shows the length of the injected sequence, while the y-axis represents the $\mathcal{F}-$measure achieved by the different detection methods against the attack scenarios. The vertical bars represent the detection algorithm's results, as already described for the previous attack scenario. From the analysis of the results presented in Figure 3.2b we notice the same trend already observed in the previous case, despite in this attack scenario, the message sequence detection algorithm performance decreases against the injection of longer valid sequences. This trend is explained by considering that with the injection of valid sequences of messages, the message sequence detection algorithm can detect an anomaly only in the transition from the regular CAN communication to the first message of the sequence or from the last message of the sequence to the regular CAN communication. Hence, by increasing the overall length of the attack, the percentage of anomalies that this detection method can detect decreases (since the internal transitions are considered valid).

Invalid sequence replay detection results

Figure 3.2c shows the detection results of the detection algorithm against the invalid sequence replay attack. The results of Figure 3.2c are shown using the same structure already described for Figure 3.2b. The results presented in Figure 3.2c show that the entropy detection algorithm is still unable to detect anomalies consistently, while the performance of the message sequence algorithm is higher compared to the previous attack scenario and is constantly higher than $\mathcal{F}-\text{measure} = 0.8$.

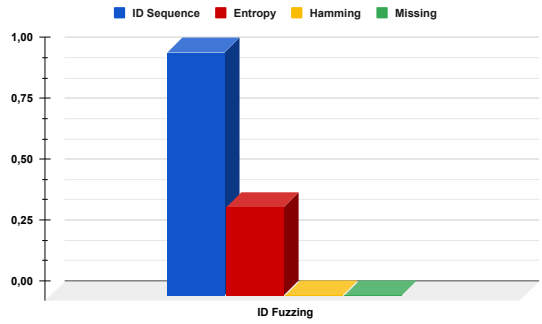
As a final comment on the detection results achieved with the framework against the replay attack scenario, the detection algorithm based on the analysis of the sequence of messages can achieve higher detection results compared to the others. At the same time, the bus entropy anomaly detector struggles to achieve detection results higher than $\mathcal{F}-\text{measure} \geq 0.5$. Moreover, the Hamming distance and the missing message detection algorithm cannot detect any anomaly against this attack scenario.

Performance against the Fuzzing

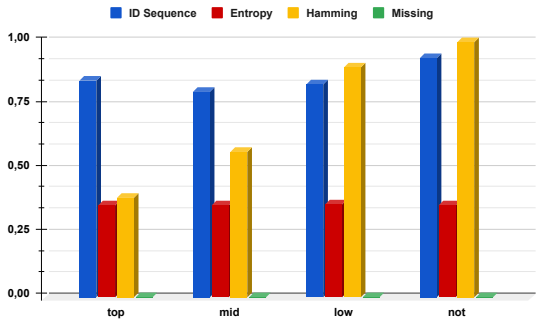
The detection results achieved with the framework against the replay attack are depicted in Figure 3.3, showing the $\mathcal{F}-$measure of the algorithms against the random ID fuzzing (Figure 3.3a) and the random payload fuzzing (Figure 3.3b).

Random ID fuzzing detection results

The results presented in Figure 3.3a show that the detection performance in this attack scenario is similar to the previous scenario, with the only difference being that the message sequence detection algorithm can achieve a fixed $\mathcal{F}-\text{measure} = 1$ in all the attack simulations. This result is easily explainable by considering that the attack is simulated using a message ID never observed in the clean dataset, while the detection model of the message sequence algorithm is composed of all the message IDs found



(a) Detection performance against the random ID fuzzing



(b) Detection performance against the random payload fuzzing

Figure 3.3: Detection results of the supported algorithms against the fuzzing (left to right: random ID fuzzing, random payload fuzzing).

in the clean dataset. With this analysis, it is clear that the message ID sequence can detect all the instances of this attack scenario with absolute precision and recall.

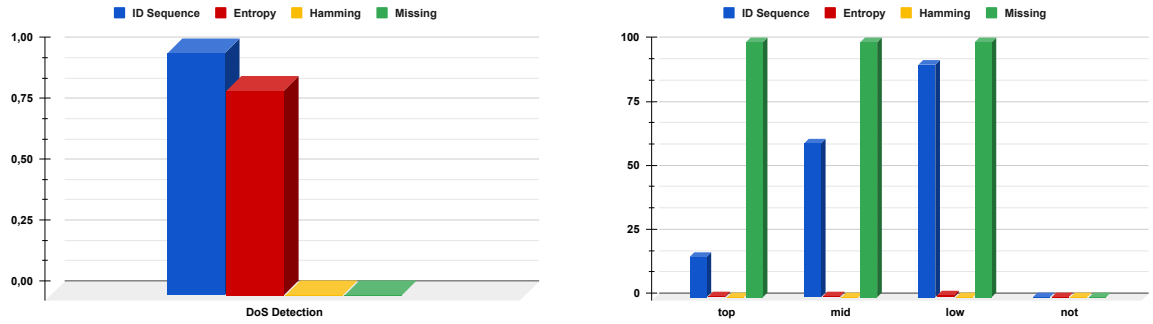
Random payload fuzzing detection results

The results achieved by the detection algorithms against the random payload fuzzing are depicted in Figure 3.3b, showing the $\mathcal{F}-$measures (y-axis) achieved with the algorithms (vertical bars) in case of fuzzing of a payload with different IDs (x-axis). The results presented in Figure 3.3b are equal to the ones presented in the case of the single ID replay attack (Figure 3.2a) for the message sequence, bus entropy, and missing message detection algorithms since the attack is simulated using identical message IDs. However, it is necessary to remark that the Hamming distance detection algorithm can detect anomalies and that its detection performance increases by decreasing the message frequency.

As a final comment on the detection performance of the algorithms against the fuzzing, the message sequence detection algorithm is still reaching high $\mathcal{F}-$measures in both attack scenarios, achieving the maximum $\mathcal{F}-$measure against the random ID fuzzing. In contrast, the bus entropy algorithm cannot detect anomalies consistently. The missing message detection algorithm is still not able to detect any anomaly. However, the Hamming distance detection algorithm can detect anomalies consistently in half of the simulated scenarios against the random payload fuzzing.

Performance against the disruption attacks

Figure 3.4 shows the detection performance of the algorithms against the two considered disruption attacks: Bus DoS (Figure 3.4a) and ECU inhibition (Figure 3.4b). The results against the bus DoS attack in Figure 3.4a shows the $\mathcal{F}-$measure (y-axis) of the detection algorithms (vertical bars), as already presented in previous scenarios. The results against the ECU inhibition attack depicted in Figure 3.4b show that for each message used for the attack simulation (x-axis), the percentage of the detected anomalies (y-axis) of the four different detection algorithms (vertical bars).



(a) Detection performance against the Denial-of-Service attack

(b) Detection performance against the ECU inhibition attack

Figure 3.4: Detection results of the supported algorithms against fuzzing (left to right: random ID fuzzing, random payload fuzzing).

Denial-of-Service detection results

From the analysis of the results depicted in Figure 3.4a, the Hamming payload distance and the missing message algorithms cannot detect any anomaly. In contrast, the message sequence and the bus entropy algorithms can achieve consistent detection results. Moreover, the message sequence detection algorithm can achieve a perfect $\mathcal{F}-$measure $= 1$ in this attack scenario since the detection model does not have any transition between the same message ID and the injection of 100 consecutive messages with the same ID. On the other hand, the bus entropy algorithm can achieve $\mathcal{F}-$measure $= 0.8$ consistently, demonstrating the ability of this detection method to detect attacks composed by the injection of multiple messages within the same time window.

ECU inhibition detection results

In the case of the ECU inhibition attack (Figure 3.4b) the best algorithm for detecting the ECU inhibition attack is the missing message algorithm, which can detect almost 100% of the missing messages. The message sequence algorithm can also detect anomalies, despite its detection performance being inconsistent throughout the different tests. However, the bus entropy and the Hamming distance payload detection algorithms cannot detect any anomaly. As a final remark, in case of the removal of the `not` message, the tested algorithms cannot detect any anomaly. This is easily explained by considering that the `not` message is a *non-cyclic* message; hence it is impossible to define this message's expected behavior. This implies that it is impossible to identify the message's cycle time; hence it is impossible to apply the missing message algorithm in this particular scenario.

3.2 Performance comparison of timing-based anomaly detectors for Controller Area Network: a reproducible study

This chapter presents an experimental evaluation of the detection performance of eight different algorithms for anomaly detection on the Controller Area Network (CAN) bus of modern vehicles based on the analysis of the timing or frequency of CAN messages. This work solves the current limitations of related scientific literature, which is based on private datasets, lacks open implementations and detailed descriptions of the detection algorithms. These drawbacks prevent the reproducibility of published results and make it impossible to compare a novel proposal against related work, thus hindering the advancement of science. We solves these issues by publicly releasing implementations, labeled datasets, and describing an unbiased experimental comparison.

3.2.1 The Algorithms

The following sections describe the time-based CAN anomaly detectors considered for the experimental evaluation presented in this work. For clarity, the names of the main parameters, variables, and attack scenarios described in the original works have been uniformed. For readability purposes, a label composed of the last name of the first author and the last two digits of the publication year is associated with each algorithm.

Otsuka14

In [118], the authors propose an anomaly detection algorithm that uses a delayed-decision cycle method to detect (and possibly prevent) spoofing attacks. The algorithm presented in *Otsuka*14 assumes that, since data frames are transmitted at a constant cycle time ct, any modification of the expected behavior of CAN communication should change its cycle time. When an ECU receives a data frame with the same CAN ID as the previous data frame and a cycle time less than $ct + \delta$ (where δ is a threshold parameter), the ECU holds the data frame until the expected time T is passed. If another message with the same ID is received in the waiting period, then the ECU can detect an ongoing attack, and the data frames are not processed. The algorithm presented in *Otsuka*14 is trained with actual CAN data and tested against the injection of two different message IDs. One exhibits a stable cycle time ct while the other exhibits a non-cyclic pattern. The results are presented through False Positive Rate (FPR) and False Negative Rate (FNR). Although the comparison of the detection performance with the previous state-of-the-art is not discussed, note that *Otsuka*14 is probably the first paper presenting a detection algorithm based on the timing analysis of CAN communications. The computational overhead of the detection algorithm *Otsuka*14 is equal to $\mathcal{O}(1)$ for each received CAN message.

Taylor15

In [160], the authors design an anomaly detection algorithm for the detection of anomalies based on inter-packet time over a sliding window. In particular, the algorithm presented in *Taylor*15 uses test values over consecutive CAN flows (defined as a sequence of CAN data frames) for its detection task. Each test value is evaluated as a $t-$test, comparing the mean time difference with its historical value (i.e., the cycle time ct of the same CAN ID). The algorithm then uses the test values to evaluate the anomaly score (defined as a logarithmic sum) over a sequence of scores to identify anomalies in the CAN communication. This algorithm is tested against the injection and the removal of CAN messages with different attack duration, ranging from 100 milliseconds to 1 second. For the message injection attack and each duration, the injected pack is inserted once, five, and ten times its normal frequency. The receiver operating characteristics (ROC) and the Area Under Curve (AUC) measure present the detection performance. The authors do not compare the algorithm with previous works but only with a One-Class Support Vector Machine classifier trained on the same data. The computational overhead of the detection algorithm *Taylor*15 is equal to $\mathcal{O}(1)$ for each received CAN message.

Cho16

The authors of [39] design and test a *Clock-based Intrusion Detection System* (CIDS) for CAN communications. This algorithm (labeled as *Cho*16) leverages the intervals of periodic in-vehicle messages for fingerprinting ECUs. Then, the fingerprints are used for constructing a baseline of ECUs' clock behaviors based on the Recursive Least Squares (RLS) algorithm. The Cumulative Sum (CUMSUM) is then computed on this baseline to detect anomalies in message timings thanks to an adaptive threshold. The *Cho*16 algorithm can detect anomalies inside a window of size W data frames and does not identify a single data frame as malicious. This algorithm is presented in two versions: the *per-message* detection and the *message-pairwise* detection. The latter version (*message-pairwise detection*) is based on the assumption that the number and identity of CAN messages generated from the same ECU are known. Note that this information can only be known a priori by either having access to the DBC or by applying modern mapping techniques such as [95]. Since it is only possible to rely on CAN log traces without having access to the DBC of the test vehicle, only the *per-message detection* version can be applied to the datasets. The *Cho*16 algorithm is tested against the injection of CAN messages (called *fabrication attack* in the original work) on both a CAN bus prototype and an actual vehicle. In the real vehicle scenario, the authors target a message with a cycle time ct equal to $20ms$. However, the original paper does not contain information about the injection frequency. Moreover, the paper does not include a performance comparison against previous work. The computational overhead of the detection algorithm *Cho*16 is equal to $\mathcal{O}(N^2)$ for each window, where N is the size of the data matrix. Note that this computational complexity is equal to the one of the *RLS algorithm* used for constructing the baseline of ECUs' clock behaviors.

Gmiden16

In [74], the authors design a lightweight intrusion detection algorithm based on the analysis of the frequencies of the CAN messages. In particular, the algorithm presented in *Gmiden*16 uses the frequency of a CAN message to detect anomalies in the CAN communication. Upon reception of a message, the algorithm compares the time difference Δ_t between the new message with the previous one sharing the same ID and generates an anomaly in case the time difference is less than half the estimated cycle time ct. In [74], the algorithm is only presented theoretically; hence there is no test of the algorithm against any attack scenario, nor its detection performance is compared with previous works. The computational overhead of the detection algorithm *Gmiden*16 is equal to $\mathcal{O}(1)$ for each received CAN message.

Song16

In [151], the authors design a detection algorithm based on the inter-arrival times of CAN messages. In particular, *Song*16 evaluates the time difference Δ_t of messages with the same CAN ID and uses the time difference Δ_t to detect two attack scenarios. The first attack scenario considered in *Song*16 is the injection attack, in which random messages are injected into the CAN bus. To detect this attack scenario, the algorithm compares the time difference Δ_t with the cycle time ct. It raises an anomaly if Δ_t is lower than half the expected cycle time ct. Note that this detection algorithm appears to be the same as the one proposed in *Gmiden*16. The second attack scenario is a Denial-of-Service attack, in which a message with a fixed ID field value is injected with a high frequency. To detect this attack scenario, the algorithm increments the value of a counter every time the Δ_t is lower than 0.2 milliseconds and raises an anomaly if the counter value is higher than a given threshold. For the test of the algorithm, they used data gathered from an actual vehicle and simulated the attacks by injecting messages for a random time window ranging from 5 to 10 seconds. In the injection attack scenario, they injected a message with twice, five, and ten times the original frequency. In the DoS attack scenario, the injection is fixed at 2000 messages per second, testing threshold values of $1, 2, 3$, and 5. The detection results are evaluated through detection accuracy, but no comparison with previous work is presented. The computational overhead of the detection algorithm *Song*16 is equal to $\mathcal{O}(1)$ for each received CAN message.

Moore17

In [111], the authors describe a frequency-based anomaly detection algorithm. The algorithm uses the time difference Δ_t between consecutive messages with the same ID value for detection purposes. In particular, *Moore*17 uses the sequence of Δ_t of each message ID for the identification of the *maximum observed error* m, which is the maximum absolute difference between the expected cycle time ct and the observed Δ_t. Upon reception of a message, the algorithm compares the Δ_t from the previous message with a threshold value defined as $ct \cdot 0.15 + m$. If three consecutive values of Δ_t are found outside the defined threshold, then an anomaly is raised. The algorithm is tested against both injection and Denial-of-Service attacks and presents the results through

True Positive Rate (TPR), False Positive Rate (FPR), and False Negative Rate (FNR). No comparison with previous work is described. The computational overhead of the detection algorithm *Moore*17 is equal to $\mathcal{O}(1)$ for each received CAN message.

Stabili19

In [153] [1], the authors designed an anomaly detection algorithm to detect missing messages from CAN communications. In particular, the algorithm presented in *Stabili*19 evaluates each CAN ID's cycle-time (ct) to build its detection model. In the detection phase, the cycle time is used with a configuration parameter k (defined in the validation process for each ID) to detect missing messages from the CAN bus. A message with a particular ID is considered missing if it is not seen on the CAN bus for at least $ct \times k$ milliseconds.

This algorithm is tested against two similar attack scenarios, the *ECU shutdown* attack (in which messages with a particular ID are removed from the CAN for a period that ranges from 10 to 120 milliseconds) and the *ECU inhibition* attack (in which all messages with a particular ID are removed from the CAN bus). The detection performance is presented through the F-measure and compared with other detection algorithms, despite only *Moore*17 being used for the comparison with time-based anomaly detection algorithms. The computational overhead of the detection algorithm *Stabili*19 is equal to $\mathcal{O}(n)$ for each received CAN message, where n is the number of message IDs found in the monitored CAN section.

Olufowobi20

In [117], the authors presented SAIDu-CANT, a specification-based intrusion detection system (IDS) using anomaly-based supervised learning. The detection model is learned in the first phase of the algorithm and requires a clean CAN data trace for learning different parameters that will be later used in the detection phase. The learned parameters are the minimum and maximum inter-arrival time of each CAN message ID $f_{i,min}, f_{i,max}$, the estimated message period $\tilde{P}_i = f_{i,min}$ and the release jitter $J_i = f_{i,max} - f_{i,min}$.

In the detection phase, the algorithm monitors the arrival time of each CAN ID and compares it against the detection model. If a message is found outside the acceptable interval defined by the specification of the detection model, it is labeled as anomalous.

The algorithm is tested against injection attacks on two different datasets. One dataset consists of CAN traces gathered from two Sedan vehicles, while the second dataset contains data from a single vehicle. Of these datasets, only the latter is publicly available[2].

The detection results are presented through True Negative (TN), True Positive (TP), False Positive (FP), False Negative (FN), accuracy, recall, precision, and F1 score. No comparison with previous work is presented, despite the authors showcasing the

[1] Acknowledgement: some of the authors of the work presented in this chapter are also the authors of *Stabili*19

[2] `https://sites.google.com/a/hksecurity.net/ocslab/Datasets/CAN-intrusion-dataset`

algorithm's detection performance against two other detection mechanisms directly described in the original work. The computational overhead of the detection algorithm *Olufowobi*20 is equal to $\mathcal{O}(1)$ for each received CAN message.

3.2.2 Dataset description

This section describes the dataset used for the training and the test of the detection algorithms. Two different datasets are used for testing and training the detection algorithms. The first dataset is gathered from the CAN bus of an unmodified, licensed 2016 Volvo $V40$ Kinetic and is called the *Ventus* dataset. The second dataset is the OTIDS [98] dataset, which is gathered from a KIA Soul.

Ventus dataset

The clean datased is the same described in Section 3.1.3, while the infected version contains some differences. The threat model used for the generation of the infected dataset is built on the attack scenarios described in the related work [118, 160, 74, 151, 111, 153]. The threat model is composed of two different attack scenarios, the *message injection* and the *message removal* attacks. The attack traces are generated in a laboratory environment for safety reasons. The laboratory setup comprises a laptop computer, a Raspberry Pi 4 board, and a CANPico [89] device. The CAN bus is implemented through a breadboard. The laptop is connected via a PEAK CAN-USB device used to record the content of the CAN bus. The Raspberry is connected to the CAN bus via a CAN shield and is used to replay the normal traces gathered directly from the vehicle. The CANPico device has an integrated CAN transceiver connected directly to the CAN bus to generate the attacks. Since all transmitting devices are connected to the CAN bus via a CAN transceiver, re-transmissions, delays, arbitration and, in general, all the low-level details that might have been affected by simulation artifacts are handled directly by the transceivers. Since the algorithms considered in this work are based on the analysis of the timings of the CAN messages, each attack scenario is replicated by targeting 10 different messages, each characterized by a different inter-arrival time. The list of the selected message IDs and their cycle-time is presented in Table 3.1.

ID	0x10	0x120	0x290	0x2B5	0x2C5	0x330	0x350	0x3D0	0x3E0	0x581
Cycle Time	10 ms	20 ms	30 ms	40 ms	60 ms	80 ms	100 ms	200 ms	300 ms	1 s

Table 3.1: Details of the IDs selected for the simulation of the attack scenarios

Message injection The message injection attack scenario is used to inject messages on the CAN bus to subvert its normal behavior by exploiting modern drive-by-wire capabilities such as automatic emergency braking, lane assist, park assist, adaptive cruise control, automatic transmission, and other similar features. The message injection scenario comprises different attacks: the *replay attack*, in which a message already seen in the CAN communication is injected at a later time on the network, *fuzzing*, in which messages with altered field values are injected to study the consequences on the system,

and the *denial-of-service*, in which messages with high priority are injected with a very high frequency to prevent the delivery of regular CAN data frames. For the aim of the analysis, the focus is not on both *fuzzing* and *denial-of-service* for the following reasons:

- in case of *fuzzing* to the ID field of the CAN data frame, the detection algorithms would fail to detect any ongoing attack since it is not possible to build the detection model for a never observed message;
- *fuzzing* on the data field of the CAN data frame (with a legit message ID) can be considered as a *replay attack* on the same message;
- the *denial-of-service* attack scenario can be analyzed by considering a *replay attack* with a high injection frequency.

Hence, for the aforementioned reasons, the focus is only on the *replay attack* injection scenario. The replay attack is conducted by injecting a targeted message on the CAN (usually selected after an initial phase of reverse engineering of the values encoded in the payload) with a particular injection frequency.

The final **injection attack** scenario considered in the experimental evaluation is composed of 50 different attack instances, in which each of the 10 selected message IDs is injected with a frequency of $1, 10, 25, 50$, and 100 messages each second. The injected messages are equally distributed inside the 1 second time window. Each attack scenario is simulated on all the traces of the clean dataset for a total of 350 CAN traces.

Removal attack The removal attack scenario is used to remove messages from the CAN bus to subvert its normal behavior, preventing the ECUs from receiving data required for their functioning. The removal attack is composed of two different attack scenarios, as already presented in [153]: the *ECU shutdown* and the *ECU inhibition* attacks. However, since the only difference between the two attacks is their duration, which does not impact the overall performance of the detection algorithms, in this work, the focus is only on the *ECU inhibition* scenario, in which target messages are entirely removed from the CAN bus.

The final **removal attack** scenario considered in the experimental evaluation is composed of 10 different attack instances, in which each of the 10 selected message IDs is wholly removed from the CAN communication. As for the previous attack scenario, this attack is also simulated on all the traces composing the clean dataset for a total of 70 CAN traces.

OTIDS dataset

The OTIDS dataset is gathered from an unmodified, licensed KIA Soul vehicle [98]. The OTIDS dataset is constructed by logging CAN traffic via the OBD-II port from an actual vehicle while performing message injection attacks. The OTDIS dataset is composed of 4 different traces, one of them representing the clean dataset (*"Attack free state"*) used for training the algorithm. In contrast, the other traces represent 3 different injection scenarios. Note that all the OTIDS dataset traces contain both data frame and remote frame messages. Since remote frames are not used in the algorithms

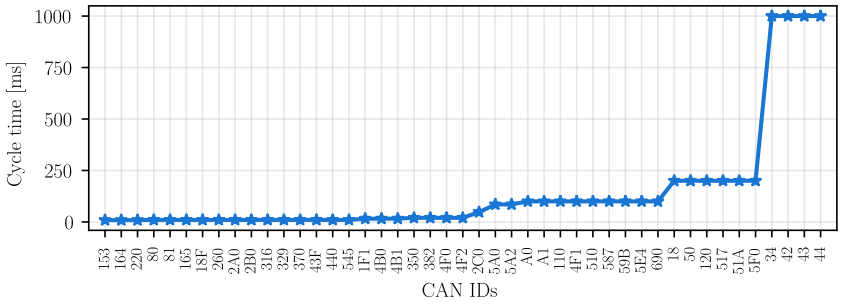


Figure 3.5: Cycle time evaluated on the OTIDS clean dataset

tested in this paper and might change the training process outcome, the OTIDS dataset traces are pre-processed to leave only CAN data frames.

Clean dataset

The clean part of the OTIDS dataset is composed of a single trace, with more than 1.4 million CAN messages corresponding to approximately 10 minutes of CAN traffic. The clean CAN trace includes the ID, DLC, and payloads of each CAN data frame associated with its relative timestamp. The clean part of the OTIDS dataset includes 45 different message IDs, each characterized by its own cycle time. As for the Ventus dataset, the cycle times of each message are extracted from the clean trace (with the same process already described in 3.1.3), and the results of the analysis are shown in Figure 3.5.

From the results shown in Figure 3.5, all the 45 IDs found in the OTIDS dataset exhibit a cyclic behavior, with a minimum of 9 milliseconds (IDs $0x153$, $0x164$, and $0x220$), to a maximum of 1 second (IDs $0x34, 0x42, 0x43$, and $0x44$). In the OTIDS dataset, there are 9 different cycle time classes.

Infected dataset

The OTIDS dataset assumes a threat model extremely different from the one considered in the Ventus dataset. Note that the OTIDS infected dataset is composed of three different types of Injections:

- **Fuzzy:** in the fuzzy scenario, messages of spoofed random CAN ID and data are injected into the CAN communication;
- **Denial-of-Service:** in the DoS attack scenario, messages with a CAN ID set to $0x0$ are injected into the CAN communication with a high frequency;
- **Impersonation:** in the impersonation attack scenario, valid messages with ID $0x164$ are removed from the network, and the attacking node injects messages with the same ID.

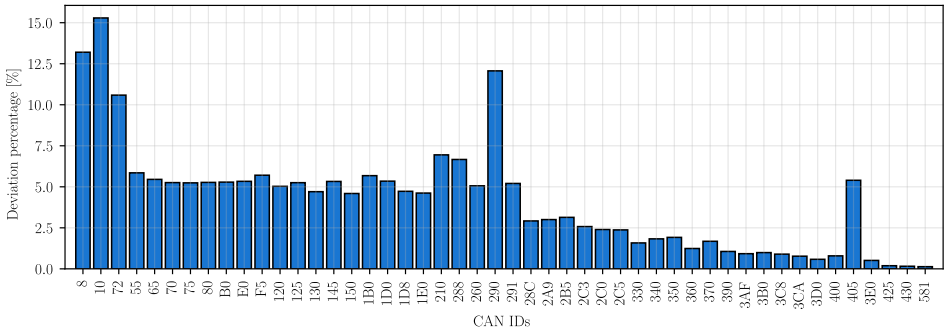


Figure 3.6: Deviation percentage on the Ventus dataset for each CAN ID

In the case of fuzzy and impersonation attacks, the injection starts after 250 seconds of normal CAN communication, while in the case of the DoS attack scenario, the attack starts at the beginning of the trace.

3.2.3 Implementations of the detection algorithms

None of the related work considered in this work is distributed together with a reference implementation. Moreover, several papers neglect many relevant details required to implement the proposed detection algorithm. This section describes the additional assumptions used to implement the detection algorithms. All assumptions are motivated and based on the maximization of the detection capabilities of the algorithms. Note that all the proposed reference implementations are publicly available at [125], thus allowing researchers to easily replicate the experiments and benchmark novel time-based detection algorithms with respect to the state-of-the-art.

Implementation of Otsuka14

For the implementation of the *Otsuka*14 detection algorithm is used the estimated cycle times (see Figures 3.1 and 3.5) as the *mean reception cycle* used in the detection process. note that the original work only considers 2 IDs, of which only 1 is cyclic. The cyclic ID has a maximum deviation with respect to the mean cycle time of 2%, while the other ID has a maximum deviation of 30%. Based on these results, the authors set a fixed detection threshold of $\delta = 5\%$ from the expected cycle time. Hence any message received outside the valid time range of $[ct^{ID} - 5\%, ct^{ID} + 5\%]$ is considered anomalous. The value of the threshold δ is defined as the threshold value that minimizes the number of false positives in the validation process. The same experimental analysis is replicated on both datasets, and the results are presented in Figure 3.6 and 3.7 for the Ventus and OTIDS datasets, respectively. The two figures show the deviation from the evaluated cycle time for each message ID. The IDs of the messages are sorted by their cycle time in ascending order (left to right). The results depicted in Figures 3.6 and 3.7 show that messages found on the CAN bus more often exhibit a higher deviation from the expected cycle time.

The value of δ that minimizes the overall number of false positives on the Ventus dataset is experimentally evaluated as $\delta = 4\%$. While testing the best value of δ to minimize the false positives on OTIDS, increasing the value of δ improves the output

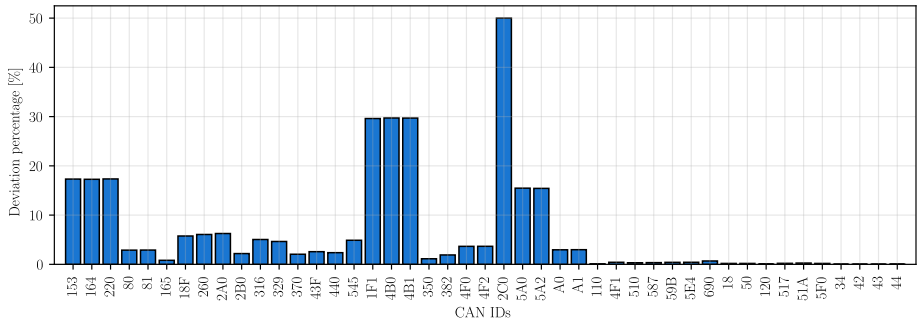


Figure 3.7: Deviation percentage on the OTIDS dataset for each CAN ID

of the validation results. However, since higher values of δ introduce a more significant time window in which messages are not considered anomalous, the value used in the experimental evaluation is fixed at $\delta = 25\%$. In the experimental evaluation presented in the next section, the *Otsuka*14 algorithm is configured with these values for its detection purposes.

Moreover, the algorithm presented in *Otsuka*14 is designed to discard the malicious CAN messages using a waiting mechanism. With this mechanism, all the messages with the same CAN ID received before $ct + \delta$ are held and discarded as soon as another message with the same ID is received after $ct+\delta$. However, this mechanism assumes that at least the first received message for each ID is legit, and its detection performance is highly affected if this assumption is violated. To this aim, the proposed implementation of *Otsuka*14 considers all the messages with the same ID received before $ct + \delta$ as a single case: in the case at least one of the held messages is an injected frame, then a single anomaly is raised. In contrast, if no one of the held messages is anomalous, only one false positive is considered. This design choice minimizes the false positive rate of this algorithm in case of high-frequency injection attacks, thus optimizing its detection results.

Implementation of Taylor15

For the implementation of *Taylor*15, the cycle times estimated on the clean datasets are used as the *historical mean* required by the algorithm for the $t-$tests. For the experimental evaluation of the detection performance of *Taylor*15, the same assumptions described in [160] are followed. Note that in *Taylor*15, the messages with a cycle time below 50ms are representative of more than 90% of the CAN IDs, while in the Ventus and the OTIDS datasets, these messages are only 61.22% and 53.33% respectively of the whole datasets (30 out of 49 and 24 out of 45 cyclic messages). This difference does not impact the detection performance of the algorithm. However, this method applies only to messages having a cycle time below 50ms; hence in both datasets, the *Taylor*15 algorithm has limited applicability compared to the original paper. Moreover, in the original work where *Taylor*15 is presented, some missing details prevent it from being directly implemented; hence some additional assumptions are also made for this algorithm.

The first additional assumption is focused on the number of sequences $\mathcal{S}_q$ used for

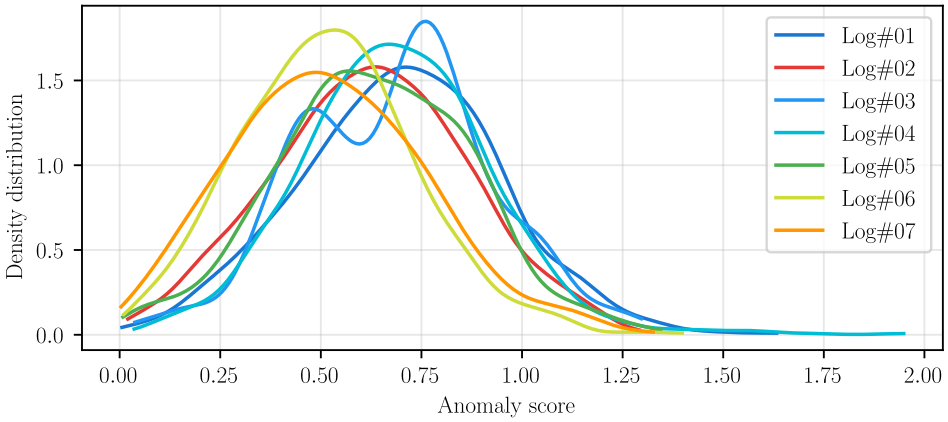


Figure 3.8: Distribution of the $A(\mathcal{S}_q)$ on the Ventus dataset

the evaluation of the anomaly score $A(\mathcal{S}_q)$. This is a crucial aspect for the evaluation of the detection performance of the algorithm since it directly impacts the detection performance. As an example, consider a scenario in which a value of $\mathcal{S}_q = 10$ is used, i.e. the anomaly score is evaluated every 10 $t-$tests. As described in [160], each item of the sequence is the value of the highest $t-$test in a window of 1 second. Hence, in the considered scenario, an anomaly score $A(\mathcal{S}_q)$ evaluated on a sequence of 10 $t-$tests implies that the algorithm can detect a single anomaly within a 10 seconds time window. This design choice would prevent a fair comparison between the detection performance of *Taylor*15 and the other algorithms that can flag many different anomalies in a similar time window. The test includes different values of $\mathcal{S}_q$ for the evaluation of the anomaly score, from a minimum of $\mathcal{S}_q = 2$ up to the whole series of t-tests. At the end of this analysis, the results show that the mean value of the $A(\mathcal{S}_q)$ is minimally affected by the number of sequences used for its evaluation. Hence, the value of $\mathcal{S}_q = 2$ is chosen to allow the detection of anomalies in the smallest possible time window.

The second additional assumption is focused on the definition of the detection threshold. In [160], this aspect is not discussed since the presented evaluation is focused on analyzing the ROC and AUC measures, which are threshold-independent. However, for the performance evaluation, it is necessary to define a threshold value to distinguish between normal and anomalous time windows. As a first step in defining the threshold value, the distribution of the anomaly scores on the clean datasets is analyzed, and the results are shown in Figure 3.8 for the Ventus dataset and Figure 3.9 for the OTIDS dataset.

From the analysis of the distribution of the anomaly scores presented in Figures 3.8 and 3.9 we design two different detection mechanisms for each dataset. The first one is based on analyzing the distribution range across the different traces and defines a threshold value higher than the maximum anomaly score evaluated in the validation process. As an example, by considering the results achieved on the Ventus dataset depicted in Figure 3.8, the anomaly scores are always below 2.0. Hence we use this value as the threshold for classifying the anomalies. In the detection process, any $A(\mathcal{S}_q) \geq 2.0$ is considered an anomaly, and all values below the threshold are considered legit. The second detection mechanism is based on the similarity of the distributions with the distribution of the normal function; hence we define a threshold using the mean value and the standard deviation of the anomaly scores, as already proposed in [36, 105].

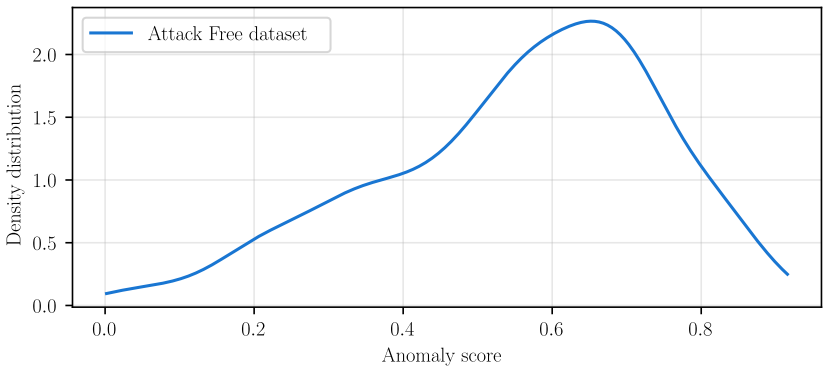


Figure 3.9: Distribution of the $A(\mathcal{S}_q)$ on the OTIDS dataset

Anomaly detection methods based on the similarity with the normal distribution often define the normal threshold as $[\mu - 3 \times \sigma,\ \mu + 3 \times \sigma]$, and consider any outside value as anomalous. However, note that only 99.68% of the distribution is covered with a value of $k = 3$; hence an anomaly detector based on this detection model introduces 0.32% of false positives. Since this last detection method introduces false positives that negatively impact the performance evaluation of this detector, the proposed implementation of *Taylor*15 uses a fixed detection threshold of 2.0 for the Ventus dataset and 1.0 on the OTIDS dataset to distinguish between valid and anomalous time windows.

Implementation of Cho16

The *Cho*16 detection algorithm uses the estimated clock skew to detect attacks to the CAN communications. The algorithm used for the clock skew estimation is described in the original work (Algorithm 1). Although the pseudo-code is well documented and described, some missing information requires additional assumptions. The most critical assumptions are related to the initialization of different parameters used to evaluate the identification error. Since the identification error is used for the detection task, it is critical to initialize these values correctly. The three most impacting parameters are the size of the window N, the initial value of the parameter P used in the `SkewUpdate` procedure, and the number of standard deviations κ. While the size of the window N only impacts the detection delay of the algorithm, the initial value of the parameter P heavily impacts the computation of the identification error, thus impacting the detection capabilities of the algorithm. In an example described in the original work, N and κ are initialized to the values of 20 and 5, respectively. However, the value used to initialize P is not disclosed. The test includes combinations of N, P, and κ on the clean datasets, aiming to minimize false positives. The final values evaluated on the Ventus dataset are $N = 10$, $P = 0.05$, and $\kappa = 0.1$, while on the OTIDS dataset the lowest number of false positives is given by the combination of $N = 5$, $P = 0.001$, and $\kappa = 2.5$. note that these values are identified after a long process of testing on the clean dataset since there is no description of how to identify the best parameters in the original work. Moreover, since the threat model considered in *Cho*16 assumes that the attack starts after 420 seconds of normal CAN communication, and the attacks on the Ventus dataset are generated from the beginning, with a modified implementation to allow *Cho*16 to evaluate the clock skew and identification errors on the clean base trace

used for attack generation. This allows *Cho*16 to learn the clock skew and identification errors on legit values, which are then used as a reference against the attack scenarios.

Implementation of Gmiden16

The description of the *Gmiden*16 detection algorithm presented in its original work is complete with all the details required for its functioning, producing a reference implementation that complies with the original design without the need for additional assumptions or design choices. In the presented version, the cycle time evaluated on the clean dataset is used as the reference inter-arrival time required by the algorithm.

Implementation of Song16

The *Song*16 detection algorithm presented in [151] is based on the assumption that CAN message exhibit a fixed inter-arrival time. However, the experimental analysis presented in 3.2.3 demonstrates that this assumption does not hold in the proposed dataset, where more frequent messages have a higher deviation from the mean value. Note that this phenomenon is well known in real CAN buses implemented in modern vehicles. Other real datasets exhibit the same behavior due to possible delays and retransmissions in CAN bus segments with relatively high usage (about 50% or higher). This is also acknowledged by producers of automotive ECUs that consider deviations up to 15% of the reference cycle time to be within the normal working parameters. While these assumptions are included in the original paper, they appear unnecessary for the proposed detection algorithms.

Note that *Song*16 includes two different algorithms targeting message injection and denial of service. The algorithm for detecting message injection appears identical to *Gmiden*16, reusing the exact implementation. The algorithm for detecting DoS attacks is clearly explained in the original paper, hence producing a reference implementation that fully complies with the description provided by the authors.

Implementation of Moore17

In the original work presenting *Moore*17, the authors assumed that each trace's first 15 seconds is unaltered, and they used the first 5 seconds to define the detection model. However, in the infected traces composing the Ventus dataset, the attacks are generated starting at the beginning of each trace, as described in Section 3.2.2. Since the Ventus dataset contains a clean dataset composed of more than 90 minutes of clean CAN traffic, the first 5 seconds of the traces composing the clean dataset for training *Moore*17 are used. However, in the OTIDS dataset, the attacks are generated after 250 seconds of regular traffic; hence the detection model is trained on the first 5 seconds of the infected trace.

Following the analysis presented in [111], is proposed a comparison of the maximum variance of each message ID cycle time with respect to the expected cycle time ct on the first 5 seconds of the traces composing the datasets. Results are presented in Figure 3.10 and 3.11 for the Ventus and OTIDS dataset, respectively.

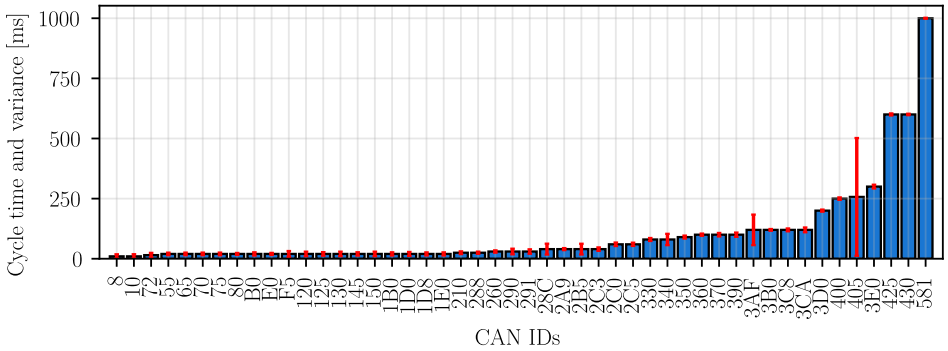


Figure 3.10: Maximum distance from average reception cycle [ms] on the Ventus dataset

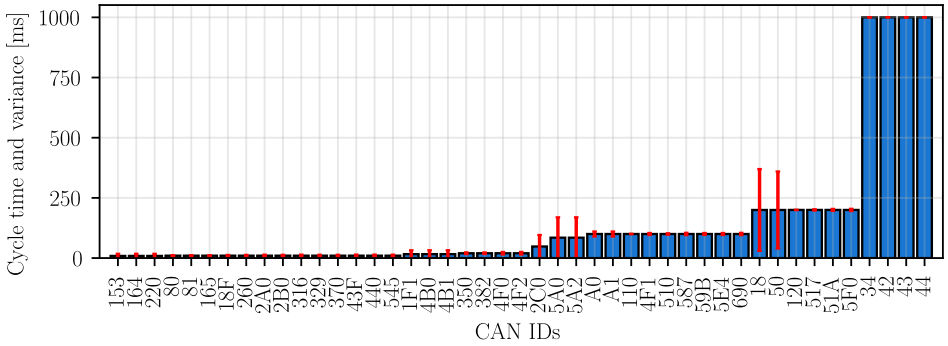


Figure 3.11: Maximum distance from average reception cycle [ms] on the OTIDS dataset

From this analysis, we notice that all the IDs exhibit a minor variance from the expected cycle time in the first 5 seconds of the traces of the clean section of the datasets, except for ID $0x405$ for the Ventus dataset and of IDs $0x18$ and $0x50$ for the OTIDS dataset. Note that these results are comparable to the ones presented in [111] on the dataset at their disposal.

Note that *Moore*17 raises an alert if the time between consecutive messages with the same ID Δ_t is lower or greater than a threshold (see Section 3.2.3 for additional details). To prevent false positives, *Moore*17 raises an anomaly only if three consecutive alerts are generated for the same ID. This design choice prevents directly comparing false positive and false negative rates with the other algorithms.

To allow performance comparisons without penalizing *Moore*17, given an anomaly, all the injected messages that generated one of the three consecutive alerts are classified as a true positive. If a legit message generated one of the alerts required for issuing an anomaly, the result is not counted as a false positive. On the other hand, if an anomaly has been raised after three alerts all generated by legit messages, that anomaly is considered a single false positive. This solution increases the overall detection performance of *Moore*17 and makes it comparable to other algorithms in terms of $\mathcal{F}-$measure.

Implementation of Stabili19

For the *Stabili*19 detection algorithm we have access to original implementation since it was proposed by the researchers of the our laboratory. The *Stabili*19 detection

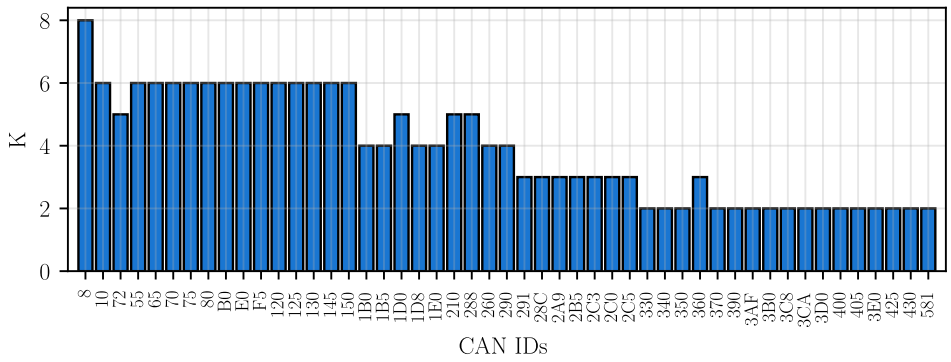


Figure 3.12: Comparison of the value of the configuration parameter k with the *ct* of each message ID on the Ventus dataset

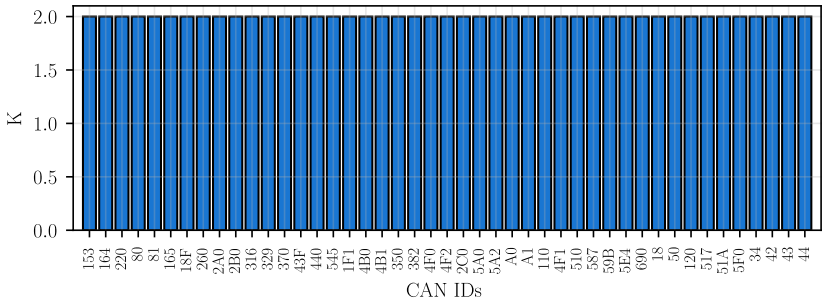


Figure 3.13: Comparison of the value of the configuration parameter k with the *ct* of each message ID on the OTIDS dataset

algorithm is based on the assumption that it is possible to create a detection model that raises 0 false positives in the validation phase. To this aim, *Stabili*19 uses a parameter k for the definition of the *valid waiting time* for each different ID. The tuning of the parameter k follows the description provided in [153] over both datasets, and the results of this analysis are presented in Figure 3.12 and 3.13 for the Ventus and OTIDS datasets, respectively.

At the end of this training process on the Ventus dataset, the minimum value of the parameter k that does not generate false positives on the clean dataset is equal to 2 (for 16 message IDs), while the maximum value is 8 (for a single message ID). On the OTIDS dataset, however, all the IDs do not generate any false positive by using a value of $k = 2$.

Moreover, focusing on the Ventus dataset and comparing the results presented in Figure 3.12 with the ones presented in Figure 3.6, we notice that messages with lower cycle time values exhibit a higher deviation from the expected cycle time and require a higher value of the parameter k to achieve 0 false positives. This interesting result is explained by considering that messages with lower cycle time (i.e., appearing on the CAN bus more frequently than the others) are more prone to delays and transmission errors over a real CAN bus that has a load comparable to or higher than 50%; hence they require higher tolerance by any detection mechanism based on the analysis of the timings.

Implementation of Olufowobi20

The *Olufowobi*20 detection algorithm is composed of the training and the detection phase, which are presented in Algorithm 1 and 2 of the original paper, respectively.

The algorithm is based on the assumption that the actual model and its parameters are unknown and unavailable since manufacturers generally do not disclose this information (specifically the precise message periods). For this reason, *Olufowobi*20 considers a detection model based on parameters derived from observations of the CAN communications. Hence, the training phase of *Olufowobi*20 is focused on the definition of the parameters required for the detection process: the estimated message period $\tilde{P}_i = f_{i,min}$ and the release jitter $J_i = f_{i,max} - f_{i,min}$.

For estimating these parameters, the transmission time C_i is required, defined as the time required to transmit a message on the CAN bus. However, note that it is impossible to identify the value of C_i by observing normal CAN communications. However, it should be learned by measuring the transmission time on the ECU transmitting a CAN message. Since it is impossible to identify the value of C_i from the clean dataset, a worst-case estimation of this parameter is used in the training process. For the worst-case estimation of C_i is used the clean dataset to reconstruct the sequence of bits composing the CAN data frame to calculate the size of the message $size(m_i)$. Than the value of C_i is defined as $C_i = max(size(m_i))/bitrate$, where $max(size(m_i))$ is the maximum reconstructed bit size of CAN messages with the same ID i and $bitrate$ is the nominal bitrate of the CAN. In the experiments, the nominal bitrate of both datasets is $500kbps$.

Another issue raised during the implementation of *Olufowobi*20 is related to the availability of the value of the specific message period P_i. Note that P_i (as described in the original work) belongs to the list of parameters not available for defining the detection model. However, in the description of the detection process (Algorithm 2 of the original work), the specific message period P_i is used as one of the algorithm's parameters. Since the specific message period P_i is different from the estimated message period $\tilde{P}_i$ (which is computed in the training phase), in the proposed implementation, P_i is defined as the cycle time ct.

Moreover, another issue raised in the implementation of the detection algorithm is that the variable k (one of the inputs of the detection function) is modified (in algorithm 2, row 7), but the modified value is used. This variable is used to evaluate the arrival time window of the next message and is crucial for the detection task. In the proposed implementation of *Olufowobi*20, the detection algorithm is modified by returning the value of k in the case.

Despite the aforementioned issues raised during the implementation of *Olufowobi*20, in its experimental evaluation, the latter modification introduced a significant downside that afflicts the detection performance of *Olufowobi*20. In particular, the novel issue is related to modifying the value of k in case a message is marked as an anomaly. In this scenario, the algorithm will update the value of k and, in case of injection attacks, all the following messages are identified as anomalies since the arrival time window used for the detection task is out of sync with the tested message, resulting in thousands of false positives. To overcome this issue, an "update protection mechanism" is introduced

on the value of k that triggers when a message is considered anomalous, reverting the value of k to its previous value.

3.2.4 Experimental evaluation

This section presents the experimental evaluation of the detection performance of the algorithms against the datasets (and their respective threat models) presented in Section 3.2.2. To compare the detection performance of the different algorithms against the described attack scenarios, the $\mathcal{F}-$measure is considered the key performance indicator. The $\mathcal{F}-$measure is the harmonic mean of the precision and the recall of the detection performance. This indicator is commonly used in comparing intrusion detection systems. It ranges from 0 to 1, where values close to 0 denote the inability to detect any anomaly, and values close to 1 denote the ability of the algorithm to detect all the anomalies with low false positives. The perfect detection algorithm exhibiting 100% precision and recall has $\mathcal{F}-$measure equal to 1.

Detection results against the Ventus dataset

Message injection detection comparison

The performance evaluation of the algorithms against the message injection attack is presented in Figure 3.14. Figure 3.14 is composed by 5 different subplots representing different attack scenarios. Top to bottom, these attack scenarios are the injection of $1, 10, 25, 50$, and 100 messages each second. The y-axis of each subplot of Figure 3.14 shows the $\mathcal{F}-$measure evaluated with the detection algorithms, while the x-axis represents the injected message ID, ordered by ascending cycle time. The 10 reference message IDs used in these experiments are the ones presented in 3.1 of Section 3.2.2, as they represent messages with different cycle times.

The detection results of each algorithm against a given combination of ID and injection frequency are represented as a boxplot. This graphical indicator is used since it summarizes in a concise representation the detection results achieved against the same attack (same ID and injection frequency) across different infected traces. For readability purposes, is represented only the main components of the boxplot: the $10_{th}, 25_{th}, 50_{th}$ (median), 75_{th}, and 90_{th} percentiles.

For example, the top subplot of Figure 3.14 refers to injection attacks performed with an injection frequency of 1 message per second. This subplot is divided into 10 "columns", each referring to the injection of a different message ID. The first of these columns refer to the injection of message ID $0x10$, includes 6 boxplots drawn in different colors, and has a slight horizontal offset to improve readability. The first boxplot (red) summarizes the detection performance of *Otsuka*14, the second (blue) of *Taylor*15, the third (dark orange) of *Cho*16, the fourth (cyan) of *Song*16 and *Gmiden*16, the fifth (green) of *Moore*17, while the last one (purple) of *Olufowobi*20. Note that *Stabili*19 is not included in this set of experiments since it is designed to detect only missing messages and cannot be applied against injection attacks. To highlight the trend related to the detection performance depending on the message cycle time, solid lines connect the median values of all the boxplots related to the same algorithm.

Note that *Taylor*15 is designed to support only messages with a cycle time lower than 50ms (see Section 3.2.3 for additional details), hence this algorithm is only applicable to the first 4 message IDs having the lower cycle times among the 10 considered in the experiments. To visually represent the inapplicability of *Taylor*15 to the last 6 IDs, a blue $\times$ is drawn instead of the boxplot.

From the analysis of the results shown in Figure 3.14, we notice different trends by either focusing on the comparison of the detection performance against the injection of the same volume of messages with different IDs ("horizontal" analysis) or by focusing on the difference between the performance against the increasing injection rate of the same message ID ("vertical" analysis). In both cases, however, note that the proposed implementation of *Taylor*15 cannot detect any anomaly in the considered attack scenarios since the injection of messages does not introduce a significant deviation from the normal anomaly score used by this algorithm.

With the "horizontal" analysis, we notice that by using *Otsuka*14 and *Song*16 (*Gmiden*16), the overall detection performance of the algorithms increases by simulating the attack on messages with higher cycle times. In contrast, the overall trend for *Cho*16 is the opposite. By focusing on the results of *Otsuka*14 and *Song*16 (*Gmiden*16) algorithms with a "vertical" analysis, we notice that *Otsuka*14 achieves higher detection performance against the injection of a low volume of messages (≤ 10 messages per second), while for injection frequency of at least 25 messages per second *Song*16 (*Gmiden*16) converge to higher $\mathcal{F}-$measure also against the injection of messages with low cycle times. Both algorithms are limited in their detection against the injection of the message with ID $0x10$ (i.e., the message with the lowest cycle time in the proposed dataset). Note also that *Song*16 (*Gmiden*16) can achieve a $\mathcal{F}-$measure of 1.0 against different injection scenarios (the top $2, 4, 6$, and 8 messages with the highest cycle times against the injection of $10, 25, 50$, and 100 messages per second, respectively), while *Otsuka*14 achieves a perfect $\mathcal{F}-$measure only in a subset of these scenarios (the top 1, and 3 messages with the highest cycle times against the injection of 50, and 100 messages per second, respectively). By focusing on *Cho*16 with a "vertical" analysis, however, we notice that the overall detection performance of the algorithm increases by increasing the injection frequency, despite having high variance across different tests.

However, the detection results achieved by *Moore*17 are entirely different and require a dedicated analysis. As presented in Figure 3.14, by comparing the detection performance of *Moore*17 against the different injection frequencies, it is impossible to identify any trend with both "horizontal" and "vertical" analysis. Note that *Moore*17 uses a false-positive prevention system which raises an anomaly only after 3 consecutive alerts, as described in Section 3.2.1. For example, consider the case of injection of a single message each second using the message with the lowest cycle time (ID $0x10$). In this scenario, the detection performance of *Moore*17 is deficient since the injected message is right after or just before another valid message, thus resulting in the generation of up to 2 alerts that are not enough to cause the generation of an anomaly. However, messages with higher cycle times have a smaller margin of error (comparing the experimental results for the definition of the parameter m discussed in Section 3.2.3); hence it is possible to detect anomalies more frequently. One could expect that by increasing the injection frequency, the overall detection performance should also increase. However, the experimental evaluation demonstrates that for high

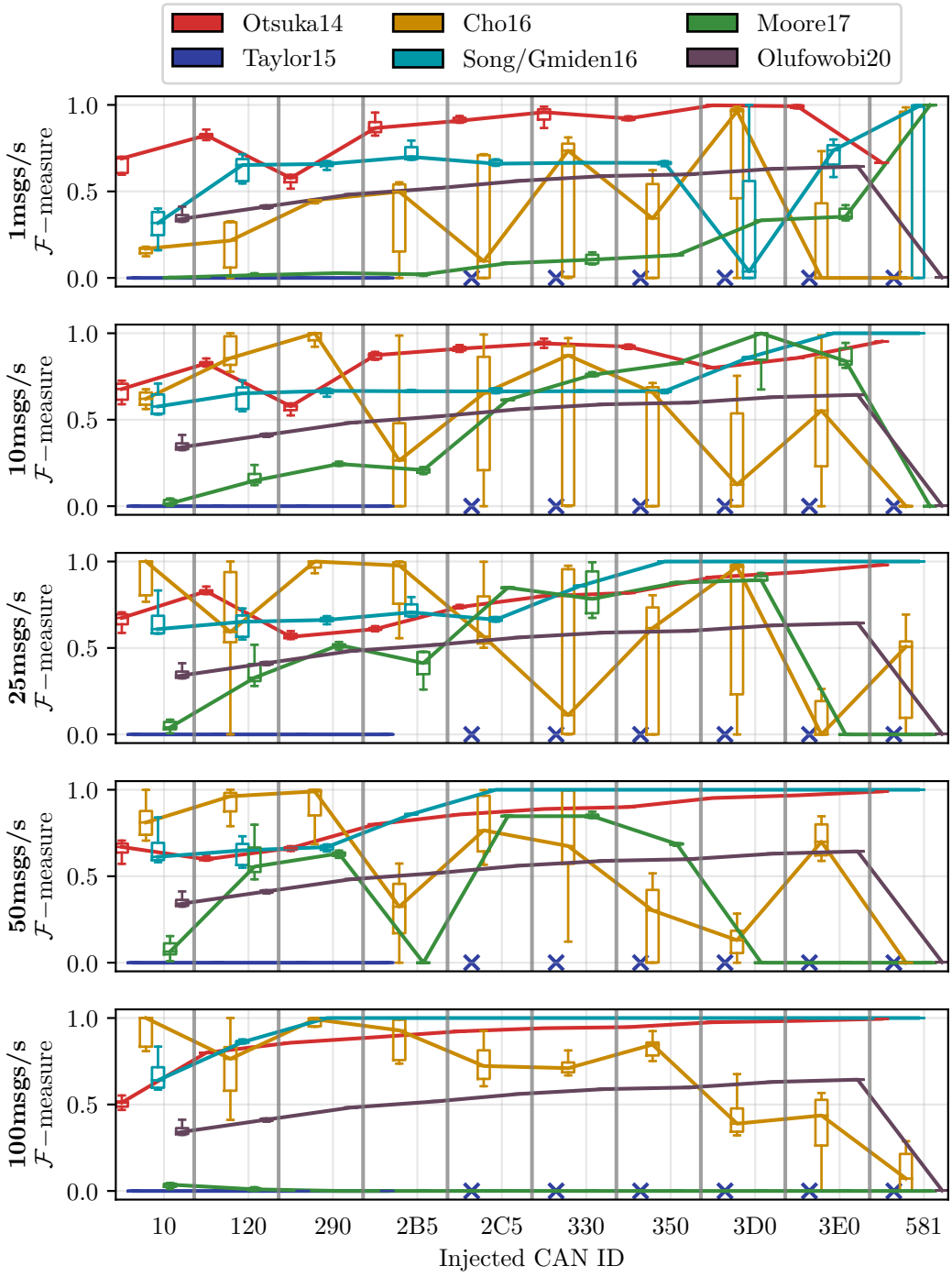


Figure 3.14: Experimental evaluation of the detection algorithms against the injection attack scenario of the Ventus dataset

enough injection frequencies *Moore*17 starts to generate an increasing number of false negatives (i.e., missed anomalies). To better explain this counter-intuitive behavior, an example is used. Consider the injection scenario of the message $0x581$, which has a $ct = 1000$ms. For this message, the value of the parameter m is 20ms, hence the time required for the identification of an alarm for this message is approximately 152ms (see Section 3.2.1 for additional details on how *Moore*17 detects anomalies). Since in the proposed dataset, the injection is simulated by equally distributing the injected messages on the interesting time window, by injecting messages with a frequency of 10 messages per second, a message is injected every 100 millisecond, thus the time required for the identification of an alert of 152ms is higher than the cycle time between two consecutive messages, increasing the number of false negatives. This explains the two different trends that can be observed by comparing the "vertical" and "horizontal" analysis. Targeting messages with higher cycle time increases the overall detection performance of *Moore*17. However, by increasing the injection frequency, it is possible to increase the detection performance against the injection of messages with lower cycle times, despite there is a huge increment of false negatives for messages whose time required for the identification of a single alert is higher than the time between two consecutive injected messages.

Finally, by analyzing the detection performance of *Olufowobi*20 "horizontally" we notice that the detection performance of the algorithm increases by increasing the cycle time of the injected message, in a trend similar to the one already observed for *Otsuka*14 and *Song*16 (*Gmiden*16). However, with the "vertical" analysis, we notice that the detection performance is not influenced by increasing the injection frequency. Note that the detection performance of *Olufowobi*20 against the injection of the message with the highest cycle time is always equal to 0.

Message removal detection comparison

To compare the detection performance of the only two algorithms supporting the detection against the message removal attack, the results are presented by means of *percentage of detected anomalies*, i.e., the number of alarms raised by the two detectors compared to the number of removed messages. Note that it is possible to use this detection metric only in case the training of the detectors is based on a zero false positives approach since it is impossible to distinguish between true positives and false positives otherwise.

The detection results of the *Taylor*15 and *Stabili*19 against the message removal attack are presented in Figure 3.15, where the percentage of detected anomalies (y-axis) for each removed message ID (x-axis) is shown.

The percentage of detected anomalies is presented using the boxplots. The detection performance of the two algorithms against this attack scenario is extremely consistent across the different missing IDs; hence it is nearly impossible to notice the different percentiles since they overlap with the median. Note that *Taylor*15 is designed to support only messages with a cycle time below 50 milliseconds; hence it is possible to use it only against the removal of the first 4 message IDs. From the analysis of the results depicted in Figure 3.15, it is clear that the proposed implementation of *Taylor*15 is not able to reliably detect anomalies also against this attack scenario since the

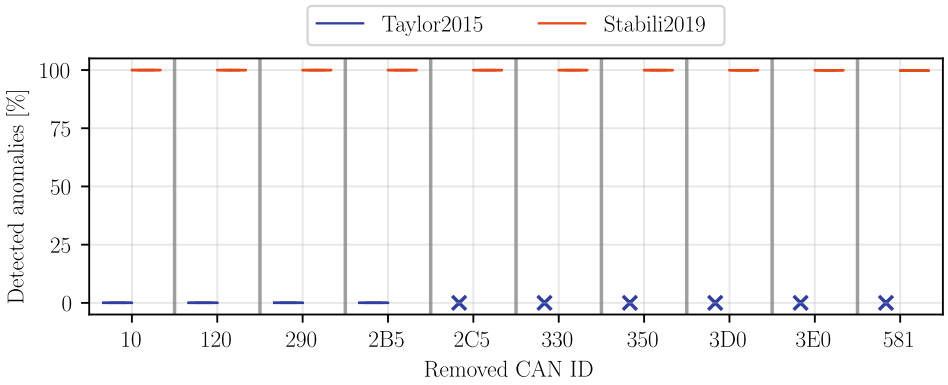


Figure 3.15: Experimental evaluation of the detection algorithms against the message removal attack scenario of the Ventus dataset

removal of messages from normal CAN communications does not introduce significant deviations in the anomaly score used by *Taylor*15. The detection performance achieved by *Stabili*19 is close to 100%, although this ideal value is never reached. This aspect has already been addressed in [153] and is related to the introduction of the *valid waiting time* required to achieve zero false positives in the validation process.

Detection results against the OTIDS dataset

This section presents the performance evaluation of the detection algorithms against the OTIDS dataset. To evaluate the detection algorithms, a labeled dataset is required, and since the OTIDS dataset is not labeled, the labels are recreated by following the description of the attacks. The three attack scenarios included in the OTIDS dataset are described as follows:

- **Fuzzing:** the fuzzing starts after 250 seconds of normal traffic, and includes both normal and injected messages with 8 different message IDs: $0x153$, $0x164$, $0x1F1$, $0x220$, $0x2C0$, $0x4B0$, $0x4B1$, and $0x5A0$.
- **Denial-of-Service attack:** the attack starts at the beginning of the trace and injects messages with ID $0x0$.
- **Impersonation attack:** the attack starts after 250 seconds of normal traffic, removes all messages with ID $0x164$, and injects messages with the same message ID mimicking its normal cycle time.

Following the attack description, note that it is not possible to distinguish between normal and injected messages in the case of the *fuzzy* attack scenario; hence it is not possible to use that particular attack scenario in the experiments with only the *DoS* and the *impersonation* attack scenarios available for performance evaluation. However, note also that in the *DoS* attack scenario, the injected message ID is not found in the training trace, thus making the detection task trivial by simply checking if there is a reference for the ID in the detection model. Hence, the only attack scenario from the OTIDS dataset that can be used for performance evaluation is the *impersonation* attack.

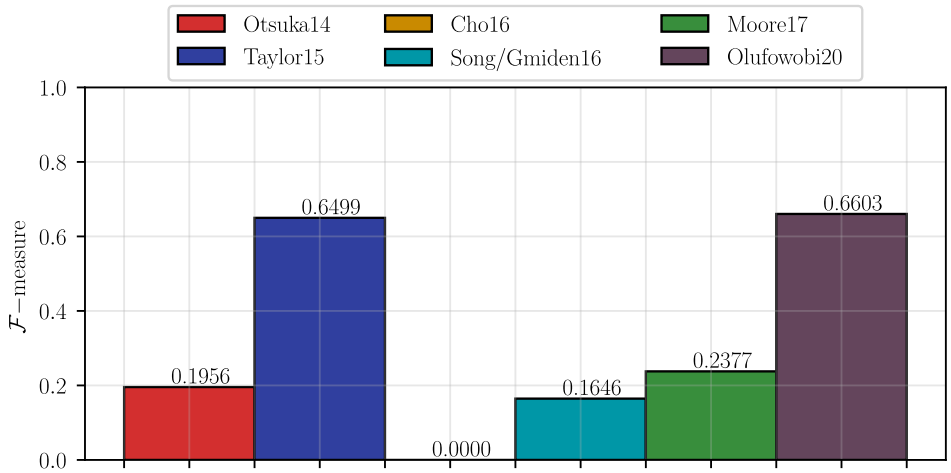


Figure 3.16: Experimental evaluation of the detection algorithms against the impersonation attack scenario of the OTIDS dataset

The comparison of the performance evaluation of the algorithms against the impersonation attack scenario of the OTIDS dataset is shown in Figure 3.16. Figure 3.16 shows, for each detection algorithm, the $\mathcal{F}-$measure evaluated on the impersonation attack scenario.

Figure 3.16 shows that in this particular attack scenario, the algorithms that can achieve the highest detection performance are *Taylor*15 and *Olufowobi*20. In contrast, the other detection techniques struggle to achieve a $\mathcal{F}-$measure close to 0.5, with *Cho*16 failing to detect a single anomaly ($\mathcal{F}-$measure equal to 0). However, note that these results are relative to a single test case. By changing the injected message or its injection frequency, the overall detection results might vary significantly, as observed previously in Section 3.2.2.

In the OTIDS impersonation attack scenario, the impersonated message ID $0x164$ is one of the most frequent messages, with a cycle time of 9 milliseconds. The impersonation attack requires the removal of the target message after 250 seconds and injecting a single message with the same message ID every 9 milliseconds, mimicking the original frequency. In these conditions, note that the impersonation attack of the OTIDS dataset is a particular scenario of masquerading attack, in which the injected CAN messages substitute the normal ones.

However, following the performance evaluation of the tested algorithms, a manual analysis of the detection results is necessary to understand the detection algorithms' behavior. The first exciting result from this analysis is that *Otsuka*14, *Song*16, and *Moore*17 raise anomalies after the start of the attack. By analyzing the beginning of the attack simulation, we notice that in the first 18 milliseconds following the start of the attack, there are 4 different messages with ID $0x164$ instead of the expected 2. This implies that in this time window, the attack scenario is closer to the message injection attack considered in the threat model of Ventus, allowing *Otsuka*14, *Song*16, and *Moore*17 to detect anomalies. The second interesting result is that both *Taylor*15 and *Olufowobi*20 raised the majority of their alarms after approximately 50 milliseconds since the start of the attack, being able to identify manipulated messages as anomalous. This implies that the evaluated detection performance of these two algorithms is to be

considered against the impersonation/masquerade attack scenario and not relative to the message injection attack. This interesting result might also cause the low detection performance of both algorithms against a real message injection attack scenario, as presented earlier in Section 3.2.4. As a final remark, it is worth noting that the detection performance of *Cho*16 is heavily affected by the values of its configuration parameter. However, it was impossible to identify any combination of values that allowed *Cho*16 to identify anomalies against the attacks on the OTIDS dataset.

Chapter 4

Applications enabled by secure V2X

4.1 Accountable and privacy-aware flexible car sharing and rental services

The transportation sector is undergoing rapid changes to reduce pollution and increase life quality in urban areas. One of the most effective approaches is flexible car rental and sharing to reduce traffic congestion and parking space issues. This chapter presents a flexible car-sharing framework where vehicle owners want to make their vehicles available for flexible rental to other users. The owners delegate the management of their vehicles to intermediate services under certain policies, such as municipalities or authorized services, which manage the due infrastructure and services that users can access. This chapter prose the investigation of the design of an accountable solution that allows vehicle owners who want to share their vehicles securely under certain usage policies to control that delegated services and users comply with the policies. While monitoring users' behavior, this approach also protects users' privacy, preventing tracking or profiling procedures by other parties. Existing approaches put high trust assumptions on users and third parties, do not consider users' privacy requirements or have limitations in terms of flexibility or applicability. Our proposal consists of an accountable protocol that extends standard delegated authorizations and integrates it with Security Credential Management Systems (SCMS) while considering the requirements and constraints of vehicular networks. The proposed approach represents a practical approach to guarantee accountability in realistic scenarios with acceptable overhead.

4.1.1 Introduction to shared mobility

For decades, large cities have become overcrowded, and vehicles and traffic congestion are destined to grow even further due to ongoing urbanization trends, also causing issues related to the shortage of parking places and, even worse, increasing local pollution and reducing the quality of life. Shared mobility systems were introduced to help solve transportation problems in urban areas. It has many benefits for the end-users, such as the lack of responsibilities of private vehicle ownership, maintenance, and insurance costs. Rather than owning one or more vehicles, the user can access a fleet of shared vehicles on an as-required basis. They also represent a flexible solution for commuters

by offering them alternatives to public transport. They also help reduce the amount of intra- and inter- cities traffic raised by personal vehicles. In particular, personal car-sharing systems, where any vehicle owner can decide to rent their own vehicle to others, represent a novel business model and an opportunity for developing innovative markets [147, 146]. The flexibility of this service also introduces challenges in terms of usability and requires developing novel infrastructures and services that traditional systems cannot satisfy [147]. Sharing and rental services based on smart vehicles, apps, and cloud services have become widespread and offer a way to build a modern personal car-sharing system. However, they introduce risks in terms of security and privacy [159].

Our proposal outlines a novel architecture for flexible car-sharing and rental services, which enables individuals to rent out their own vehicles to others through intermediate brokers. These brokers must operate as authorization services and enforce the requirements defined by vehicle owners when allowing users to access the vehicles. Our proposal tackles security and privacy challenges related to the possibility that the involved entities do not always behave honestly and that vehicle users could be traced for commercial purposes. Our proposal represents a solution for building an accountable delegated authorization protocol that allows vehicle owners to expose the user's potential misbehavior and guarantee the users' privacy with respect to both the vehicle owner and the authorization service.

In existing systems, users must submit information such as their identity and driving license to the service provider when requesting a car-sharing service. The service provider verifies that the customer has the right and ability to drive as a valid driver. After that, the user can utilize the car-sharing service through a service provider.

4.1.2 Overview on car renting solutions

Car rental systems may be deployed using two approaches depending on the characteristics and requirements of the involved scenarios: station-based and free-floating [41]. In station-based systems, the rental service reserves stall for pick-up and return and ensure the car's availability when needed. This approach is aimed at users who decide to replace ownership of a car with the use of other means (public transport, car sharing, car rental, taxi) and for those who make sporadic use of the car and rely on systematic travel. It is therefore connoted as strongly complementary to public transport and allows a gradual expansion of the service geographically. Furthermore, it also has the advantage of being suitable for medium-sized city centers. In free-floating systems, cars can only be picked up if available at the time of use, the car can be released in any place within a predefined urban perimeter, and the withdrawal takes place from the point of release of the previous user. It is aimed at users who make trips which can be carried out with other means of transport. For this reason, it is an alternative to taxis, public transport, or bicycles. It is therefore connoted as an addition to the public transport system. These two models are suitable for different types of mobility and can be considered complementary. Our proposal supports both types of models, and it is not introduced any constraints on the operational model.

The literature has analyzed the security and privacy of car rental sharing systems from multiple perspectives [159, 164, 47, 164]. The authors of [159] propose an analysis

of the security risks of keyless systems for accessing vehicles in car sharing services and propose a keyless sharing system (KSS). This work can leverage any authentication system for allowing users to access vehicles and thus can be considered complementary to existing proposals in this context. Our proposal is more related to authorization systems for car sharing proposed in [47], [164] and [97]. In [47], the authors propose a secure free-floating car-sharing system that allows users to reserve the car-sharing service by authenticating themselves using standard access tokens using a mobile device. However, the proposed scheme does not protect the users' privacy nor enable auditability. In [164], the authors analyze cybersecurity issues for Shared electric and automated mobility (SEAM), including peer-to-peer car sharing, and present security solutions regarding authentication and authorization procedures. Our proposal differs from those works because it uses a third party that manages the pseudonyms to guarantee the privacy of the end-users. The authors of [97] analyze users' privacy and propose an authentication protocol for a car sharing service that addresses privacy-preserving using a pseudonym. In particular, they use a user-centric pseudonyms management system based on Hierarchical Identity-Based Signature (HIBS). The main difference with this proposal relies on using the same security credential management system (SCMS) used in V2X communication [20] that has the advantage of reusing protocols that are emerging as the standard for vehicular infrastructures. Moreover, the security schemes used by SCMS systems are based on more established and lightweight cryptographic primitives that can be deployed in existing contexts.

Accountability for authorization protocols has also been proposed in the context of IoT for improving control of authorizations released for accessing distributed devices [65] and for improving the security of supply chains in cyber-physical environments [66]. In particular, these proposals adopt transparency logs to monitor delegated parties and compliance with authorization policies. Our proposal represents a similar approach but does not introduce transparency logs to the authorization architecture. Integration of similar approaches is left as future work.

Delegated authorizations in constrained environments

Delegated authorization frameworks allow to manage *authorizations* to access to *resources* in modern Web scenarios, which include many stakeholders and require secure and scalable solutions.

Our proposal is modeled after the OAuth2 framework [84] which is the most popular standard in Web architectures, which includes four types of entities: a *resource owner* that is authoritative for the life-cycle of the *resources*, a *resources server* that hosts the *resources*, a *client* that needs to access the *resources*, an *authorization server* that releases *security tokens* that allow *clients* to access *resources* stored in *resource servers*. Moreover, OAuth2 is an extension for Access Constrained Environments (OAuth2-ACE) [142], which consider scenarios where *resource servers* might not be typical Web services that are assumed to be always connected to the Internet and available through standard HTTPs protocols. Instead, OAuth2-ACE supports *resource servers* with unstable or no Internet connection and possibly using transport protocols with poor security guarantees.

4.1.3 Reference scenario

The scenario considers a flexible car rental scenario where people can rent their vehicles to others under certain constraints, such as adopting the vehicle only within a certain distance and within specific timings. To regulate access control procedures to their vehicles, owners subscribe to an intermediate broker to delegate it with the capability of allowing users to access vehicles through their smartphone (e.g., NFC technology in the vehicle door or by using the GPS position of the user). The broker deploys the due infrastructures and services to handle users' rental requests and release authorization material to access vehicles. Each car is equipped with a smart start system that unlocks and starts the vehicle only if the user has obtained the due authorization, possibly after completing a payment procedure [28]. Moreover, the car has sensors that collect information about the vehicle's state and position. The car is also connected to the Internet to send this information to an online service that can detect potential misbehavior.

The scenario considers a rental procedure where a user wants to rent a specific vehicle. The procedure is modeled in three phases. First, the user sends a request to the broker to access the vehicle. Second, the broker validates the request and decides whether the user should be authorized to use the vehicle. Third, if the broker decides that the user is authorized, it releases authorization material that allows the user to access and start the vehicle. If the user is not authorized, the broker denies the user access by not releasing the authorization material.

Our proposal aims to design an architecture allowing vehicle owners to monitor the behavior of brokers and users. Our proposal allows vehicle sensors to detect if a user violates the usage agreement established between the user and the broker. The bad behavior of a user allows the architecture to revoke all the authorizations released to the same user to use other vehicles. Moreover, our proposal also allows to detect if a broker authorizes users to use a vehicle under agreements that violate the usage constraints defined by vehicle owners. The bad behavior of a broker can be exposed publicly, thus affecting its reputation and acting as a deterrent. This significantly contributes to building a flexible rental ecosystem, where many brokers can act in different geographical locations. Moreover, an essential guarantee of our proposal is to protect the users' privacy against tracking by brokers and owners.

It is assumed that owners and brokers have known identities and can establish secure and mutually authenticated connections using standard Web protocols for secure communications and credential systems. It is assumed that users can adopt secure authentication protocols to access vehicles. It is assumed that vehicles can be provided with a small number of public keys used to authenticate access requests and support the related authentication protocols. It is assumed that per-owner public keys can be installed in the vehicle at purchase time and can be updated via online update systems.

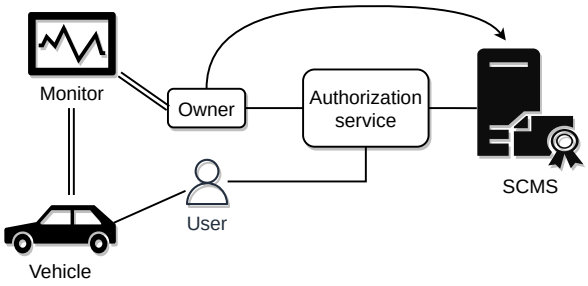


Figure 4.1: Proposed architecture

4.1.4 Proposed architecture

System model

Figure 4.1 describes the proposed architecture, which includes the *Security Credentials Management System (SCMS)* (Section 2.2.2.7) and five additional roles: *vehicle*, *owner*, *monitor*, *user*, and *authorization service*. Section 4.1.2 discusses each of the roles concerning actors of delegated authorization frameworks.

The *vehicle* represents the resource that the authorization architecture must manage. The *vehicle* can be used by authorized *users* through a dedicated service via wireless connectivity (e.g., smart vehicle door with smart card [6]). Moreover, the vehicle is provided with sensors that can detect information about its state (e.g., GPS position) and send this information to the vehicle *monitor*.

The vehicle *owner* acts as the authority of the resource and decides the *policies* to access the vehicle. It delegates the *authorization service* to decide which *users* can obtain the due authorization grants to access its *vehicle*. The *owner* can also interact with the vehicle *monitor* to control the state of the vehicle and to verify the compliance of the usage *policies*.

The vehicle *monitor* denotes a third party that collects the vehicle sensors data and allows the *owner* to access these data to validate delegation policies. Real-world examples might include the vehicle manufacturer or an insurance company that already collects such data for other purposes (e.g., remote diagnostic, emergency assistance, incidents forensics).

The *user* denotes a person who requires vehicle access. To this aim, he submits a request to the *authorization service* by providing a *pseudonym certificate* and *usage policies*. If authorized, he receives the *authorization grant* that he can use to access the *vehicle*. An honest *user* must comply with the submitted *usage policies*. Otherwise, it could be reported by the *owner* of the vehicle and possibly revoked from the rental service.

The *authorization service* acts on behalf of the vehicle owner to control the access to the vehicle. It receives the *authorization request* from the *user* and releases the *authorization grant* if the request complies with the policies defined by the owner.

The *SCMS* acts as the identity provider for *users*, releases *pseudonym certificates* and manages the revocation of users. There is no modification to its behavior to existing standard SCMS architectures (see Section 2.2.2.7).

Our proposal assumes that the *user* has already registered at the SMCS through the *bootstrapping* procedure and obtained an *enrollment certificate*, and that he can obtain multiple *pseudonym certificates* through the *pseudonym certificate provisioning* procedure. Moreover, it is assumed that the *owner* has already registered at the *authorization service* and that he can delegate authorization management to it.

The three macro-phases of the life-cycle of the architecture are:

- *authorization delegation*: the owner delegates the management of the authorization of a vehicle to the authorization service, specifying the policies that must be enforced;
- *authorization grant and vehicle access*: the user submits a request to the authorization service and releases the authorization grant if the request complies with the policies defined by the owner
- *vehicle monitor and user revocation*: the owner interacts with the vehicle monitor to control the state of the vehicle and to verify compliance with the usage policies; if the user misbehaves, he can be reported to the SCMS that can eventually revoke him from the system.

Section 4.1.5 describes the details of each of the phases and of the management of the disputes.

Threat model

- *Users* are untrustworthy or malicious; they can try to obtain illegitimate access to the vehicle or alter the information exchanged with the authorization service to lower the system's credibility. The user can try to corrupt the information exchanged with the authorization service or the vehicle to gain advantages (e.g., extend the rent period). The user can try to interfere with the vehicle to obtain information about the other user that rented the vehicle.
- *Owners* are semi-honest; they can try to learn and extract user information, such as booking preferences, travel, and frequency. Owners are unlikely to have any motivation to manipulate the protocol flow or the exchanged messages, as any such actions may result in their expulsion from the system or a damage in their reputation. As they stand to benefit from rentals, the potential loss of earnings far outweighs any gains they may derive from manipulating the protocol.
- *Monitors* are semi-honest, but they can try to learn and extract information about the users on behalf of the owners. Monitors have no interest in altering the protocol flow or the exchanged messages.
- *Vehicles* are untrusted but tamper-evident; attackers can try to interfere with the vehicle sensors to obtain advantages or learn about the behaviors of the other users. The vehicles are assumed to be equipped with Trusted Platform Modules (TPMs) to store information safely. Considering the value of the data we aim to protect, it is reasonable to conclude that it may not be worthwhile for attackers to invest in violating tamper-proof systems.

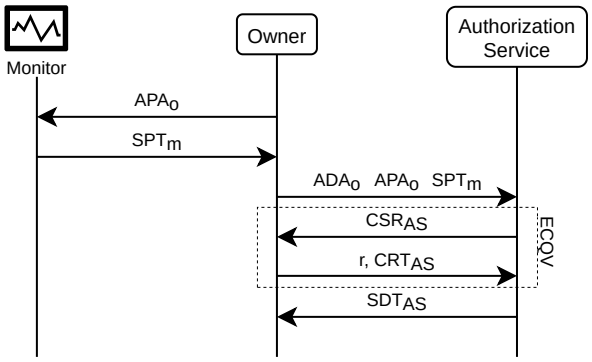


Figure 4.2: Authorization delegation

- *Authorizations services* are semi-trusted and could issue authorizations that do not comply with the policies defined by the resource owners, that is, they may illegitimately deny or release authorization to a client that does not comply with the authorization policies defined by the resource owners.
- *SCMS* is a trusted third party.

4.1.5 Details of the proposed protocol

In this section there is the description of the details of the macro-phases of the protocol: *authorization delegation* (Sections 4.1.5), *authorization grant and vehicle access* (Section 4.1.5), *vehicle monitor and user revocation* (Section 4.1.5). This section concludes by discussing how our proposal solves potential disputes among the parties (Section 4.1.5).

Authorization delegation

Figure 4.2 describes authorization delegation. The owner starts the procedure to delegate authorization management by sending an authorization policy attestation APA_o to the *monitor* that is built as follows:

$$AP_o = \langle id_o, id_v, T, AR \rangle \tag{4.1}$$

$$APA_o = \langle id_m, AP_o, \sigma_o \left(id_m, AP_o \right) \rangle \tag{4.2}$$

where id_o is the unique identifier of the owner, id_v is the unique identifier of the vehicle, T is the time range that identifies the validity of the delegation, and AR is the authorization rule decided by the owner. The *monitor* validates the contents and the digital signature, and accepts the delegation procedure by answering with a *signature policy timestamp* SPT_m built as:

$$SPT_m = \langle \mathcal{H} \left(APA_o \right), \sigma_m \left(APA_o \right) \rangle \tag{4.3}$$

The owner builds an authorization delegation attestation ADA_o as follows:

$$ADA_o = \langle AP_o, \sigma_o \left(AP_o, PK_{AS} \right) \rangle \tag{4.4}$$

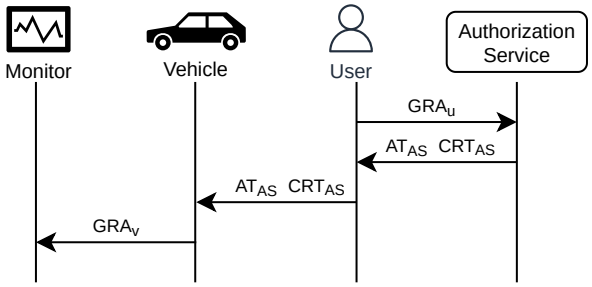


Figure 4.3: Authorization grant and vehicle access

The owner sends ADA_o, APA_o, and SPT_m to the AS, which verifies that SPT_m has been generated using APA_o and is signed by the monitor. If the AS accepts the delegation, it starts the ECQV certificate generation procedure (see Section 2.2.2.6):

- the AS generates the certificate sign request CSR_{AS} by using the owner identifier id_o, the vehicle identifier id_v and the time range T as the certificate metadata in the *csrgen* routine:

$$CSR_{AS} = screen(id_o, id_v, T); \tag{4.5}$$

- the owner generates the implicit certificate CRT_{AS} by using the certificate generation routine:

$$(CRT_{AS}, r) = crtgen(sk_o, PK_o, CSR_{AS}), \tag{4.6}$$

  where r denotes the private key contribution;

- the AS generates the key pair (sk_{AS}, PK_{AS}) by using the certificate reception routine:

$$(sk_{AS}, PK_{AS}) = crtrec(PK_o, CRT_{AS}, r). \tag{4.7}$$

Finally, the AS sends the signed delegation timestamp SDT_{AS} built as follows:

$$SDT_{AS} = \langle \mathcal{H}(ADA_o), \sigma_A S(ADA_o) \rangle \tag{4.8}$$

Authorization grant and vehicle access

Figure 4.3 describe authorization grant and vehicle access. The user requests access to a vehicle by sending a Grant Request Attestation GRA_u to the AS:

$$GRA_u = \langle pcrt_u, AP_u, \sigma_u(pcrt_u, AP_u) \rangle \tag{4.9}$$

where $pcrt_u$ is the pseudonym certificate chosen by the user to guarantee its anonymity, and AP_u is the authorization policy to which the user declares to comply when using the vehicle. First, the AS must verify the authenticity of the digital signature $\sigma_u(pcrt_u, AP_u)$ and the validity of the provided pseudonym certificate $pcrt_u$ against the public key of the SCMS PK_{scms} and the most updated version of the CRL_{scms}.

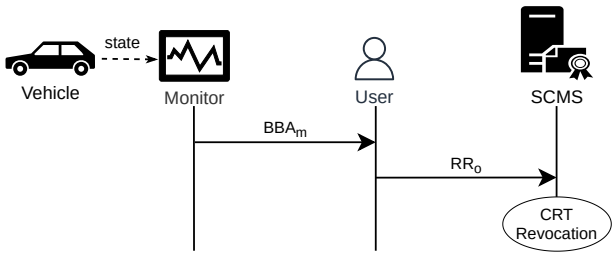


Figure 4.4: Vehicle monitor and user revocation

Second, the AS verifies that AP_u complies with the authorization policy AP_o previously released by the owner (Equation (4.1)). If both verification procedures succeed, the AS returns its implicit certificate CRT_{AS} and the access token AT_{AS} built as follows:

$$AT_{AS} = \langle pcrt_u, AP_u.T, \mathcal{H}\left(GRA_u, APA_o\right), \\ \sigma_{AS}\left(pcrt_u, AP_u.T, \mathcal{H}\left(GRA_u, APA_o\right)\right)\rangle \tag{4.10}$$

The user presents both CRT_{AS} and AT_{AS} to the vehicle. First, the vehicle uses the owner public key PK_o and CRT_{AS} as inputs of the ECQV *extract* routine (see Section 2.2.2.6) to obtain the AS public key PK_{AS}. Second, it verifies the authenticity of AT_{AS} by using PK_{AS}. Third, it verifies that the time range $AP_u.T$ is valid with regard to the current time known by the vehicle and that it is included in the time range of the implicit certificate metadata, that is $CRT_{AS}.T$. Finally, the vehicle presents the GRA_v to the monitor:

$$GRA_v = \langle pcrt_u, AP_u.T, \sigma_v\left(pcrt_u, AP_u.T\right)\rangle \tag{4.11}$$

The monitor verifies that $AP_u.T$ is included in APA_o; if it is not included, the monitor sends an alert to the owner.

Vehicle monitor and user revocation

Figure 4.4 describes vehicle monitor and user revocation. The *monitor* receives information about the state of the vehicle from its sensors in real-time and verifies compliance with the authorization policies established by the *owner* and communicated to the monitor within the APA_o data structure (Eq.(4.2)). If the vehicle is not compliant with the authorization policies, the *monitor* sends an alert BBA_m to the *owner*:

$$BBA_m = \langle id_v, id_m, pcrt_u, VS, \mathcal{H}\left(APA_o\right), \\ \sigma_m\left(id_v, id_m, pcrt_u, VS, \mathcal{H}\left(APA_o\right)\right)\rangle \tag{4.12}$$

Then the owner sends a revocation request RR_o to the SCMS, attaching BBA_m as proof of the user's misbehavior:

$$RR_o = \langle BBA_m, APA_o, \sigma_m\left(BBA_m, APA_o\right)\rangle \tag{4.13}$$

The SCMS verifies the validity of RR_o, revokes all the pseudonym certificates related to the user pseudonym $pcrt_u$, updates the CRL, and saves the RR_o.

Disputes

The *user* starts the repudiation procedure at the SMCS by sending the access token AT_{AS} and the Grant Request Attestation GRA_u (Eq. (4.9)) received by the AS (Eq. (4.10)). The SCMS validates the access token AT_{AS} against the collected revocation requests RR_o (Eq. (4.13)) by operating the following steps:

- it verifies the digital signatures included in GRA_u and AT_{AS};
- it retrieves the revocation request RR_o associated with the user: $RR_o.BBA_m.pcrt_u \stackrel{?}{=} GRA_u.pcrt_u$;
- it verifies that the received AT_{AS} is correctly bound to GRA_u and to the policy APA_o included in RR_o. That is, it computes $\mathcal{H}(GRA_u, RR_o.APA_o)$ and verifies that it is equal to the corresponding value included in AT_{AS}.

Then, the SCMS decides the outcome of the dispute by analyzing the policies included in $RR_o.APA_o$ and $GRA_u.AP_u$, and the vehicle state information $RR_o.BBA_m.VS$:

- the user revocation is invalidated if the vehicle state $RR_o.BBA_m.VS$ complies with $GRA_u.AP_u$. This scenario implies that $GRA_u.AP_u$ violates $RR_o.APA_o$ and thus that the authorization service AS (which approved $GRA_u.AP_u$) misbehaved. Thus, the authorization service AS could also be reported and possibly sanctioned;
- the user revocation is confirmed if $GRA_u.AP_u$ does not violate $RR_o.APA_o$, because this implies that the usage of the vehicle by the user violated the policies decided by the owner.

4.2 On the effectiveness of BSM communications in V2V emergency scenarios

Cooperative Intelligent Transportation Systems (C-ITS) improve driving experience and safety through secure Vehicular Ad-hoc NETworks (VANETs) that satisfy strict security and performance constraints. The use of Vehicle-to-Vehicle (V2V) communications to improve safety in emergency scenarios is already considered in the relevant standards. However, there is a lack of scientific efforts to evaluate and compare the effectiveness of these solutions. This work improves the state of the art by providing an assessment of the effectiveness of V2V communications in reducing travel time and safety–relevant events of emergency vehicles (EV). The assessment is based on realistic simulation, considering real road networks, traffic intensity, and all constraints of V2V communications.

4.2.1 Simulation of road traffic and V2V communications

The VEINS framework [149] is a simulation framework used to couple the execution of two simulators (OMNeT++ [167], a well–known network communication simulator; and SUMO [101], a road traffic simulator) to provide realistic inter–vehicular communication. VEINS supports the IEEE 802.11p, IEEE 1609.4 DSRC/WAVE [51], Physical Layer [23], Obstacle Shadowing [150] and Antenna Patterns [50] modules. The first two modules (IEEE 802.11p and IEEE 1609.4 DSRC) extend the VEINS capabilities by enabling the DSRC/WAVE stack, including Quality-of-Service channel access, the Wave Short Message (WSM) management, and periodic beacon of BSMs. The other modules simulate the propagation and attenuation of the wireless signals to recreate proper signal coverage of messages in urban environments.

SUMO includes the *Blue Light Model* [16], which is a model for emergency vehicles that allows the simulation of emergency scenarios. An *emergency scenario* is a generic scenario in which at least one vehicle is equipped with blue flashing lights and sirens. The Blue Light Model is designed as an evaluation tool that allows to define and support strategies for emergency vehicles in a safe environment before their implementation. The blue light model defines different types of safety-relevant events that might occur during the simulation of the EV:

- Emergency Braking (EB): the EV performed an emergency braking (due to a stopped preceding vehicle or a vehicle incoming from the opposite direction);
- Junction Collision (JC): the EV crashed in the middle of a junction;
- Frontal Collision (FC): the EV crashed frontally with another vehicle;
- Teleport (TP): the EV teleported in front of a column of vehicles when no routes are available (usually after a crash).

Those safety-relevant events are mostly related to the simulated environment and allow the EV to bypass the current limits of the Blue Light Model.

Note that the Blue Light Model only manages EVs; hence the behavior of non-EVs is not modified in an emergency situation. This work extends the Blue Light Model to take advantage of V2V communication and proposes heuristics for non-EVs evaluated through VEINS.

4.2.2 Improving the Blue Light Model

The SUMO simulator supports different configurations of road networks. The default configuration only simulates a single vehicle per driving lane, and lane changes are performed instantaneously. However, two models can be used to improve these scenarios, the *Sublane Model* [145] and the *Opposite Direction Driving* [4]. The *Sublane Model* is built by splitting the lane into more sub–lanes to simulate multiple vehicles driving in parallel on a single lane. This model helps simulate multiple realistic scenarios, from 2–wheeled vehicles in the same lane to vehicles overtaking a bicycle in a single lane. The *Opposite Direction Driving* model allows vehicles to overtake through the

opposite direction lane to improve their travel speed. The overtaking maneuver is only simulated in case multiple safety–relevant checks are passed (such as the presence of vehicles from the opposite direction, the duration of the maneuver, and the availability of lane space at the end of the overtaken vehicles). However, these two simulation models are incompatible; hence, deploying the Sublane model makes it impossible to perform an overtaking maneuver using the opposite lane.

The SUMO simulator supports emergency vehicles as a particular class of vehicles. EVs in SUMO are allowed to drive on reserved lanes and can initiate a right-hand overtaking in all traffic situations. If the *opposite direction driving* model is enabled, EVs can also drive on the opposite direction lane while approaching intersections and overtaking long columns of vehicles (up to 100 meters). Moreover, in case the blue light model (described in Section 6.3.2) is enabled, EVs also ignore traffic lights, perform more aggressive lane changes, and enter intersections from both correct and opposite driving lanes. This behavior is realistic in many cases but does not allow simulating overtaking using the same lane. On the other hand, by combining the *blue light model* with the *sublane model*, traffic participants are forced to create a virtual middle lane (called *rescue lane*) to allow EVs to travel safely. This combination of models allows us to simulate a typical behavior from vehicles in front of EVs in emergency scenarios. However, its applications on the simulator are only limited in case the width of the same direction travel road fits at least three vehicles, requiring at least two travel lanes for being implemented. This limitation does not represent urban scenarios characterized by roads with only one lane for travel direction since EVs can not overtake and thus are stuck behind vehicles even if the opposite travel direction lane is free.

To overcome current limitations, this work aims to improve the combination of *blue light model* and *opposite direction driving model* by using V2V communications, to reduce the arrival times of EVs in urban scenarios. This solution exploits the existing *blue light model* of SUMO to simulate the low-level behavior of EVs and implements V2V-based heuristics to improve cooperation by non-EVs. The heuristics aim to try to leave a lane as accessible as possible for the EV, forcing non-EVs to stay in the rightmost lane and slow down to facilitate overtaking and reduce the risk of safety-critical maneuvers from the EV. In our proposal, the EV broadcasts to all the other vehicles a BSM containing its position, the arrival position, and a flag value denoting its *"emergency"* state. All vehicles receiving this BSM must verify their presence in the expected EV path. At first, the vehicle receiving the BSM checks its proximity with the EV. Then, there are two different scenarios based on the typology of the road the vehicle is traveling. In case the road is composed of at least two or more lanes for travel direction, all non-EVs traveling on the road in proximity of the EV are programmed to occupy the rightmost lane to free the passage for the EV on the central virtual lane. In case non-EVs travel on a road with a single lane for travel direction, their behavior is to slow down to facilitate overtaking by the EV. Both heuristics are developed to minimize the opposite direction driving time of the EVs, reducing potential safety crashes and other safety-critical maneuvers from the EV.

	micro #1	*micro #2*	*macro*
run #1	120 (*Ti: 12*)	100 (*Ti: 11*)	250 (*Ti: 3*)
run #2	100 (*Ti: 11*)	95 (*Ti: 10*)	460 (*Ti: 5*)
run #3	90 (*Ti: 10*)	85 (*Ti: 9*)	490 (*Ti: 6*)
run #4	130 (*Ti: 14*)	140 (*Ti: 15*)	670 (*Ti: 8*)
run #5	150 (*Ti: 17*)	140 (*Ti: 16*)	680 (*Ti: 8*)

Table 4.1: Number of total vehicles involved in each traffic scenario

4.2.3 Experimental analysis

This section presents the experimental results of the heuristics presented in Section 4.2.2. Section 4.2.3 presents the setup used for the experimental evaluation. Section 4.2.3 presents the simulation results and compares them against a baseline scenario in which the novel heuristics are disabled.

Simulation setup

Multiple scenarios are simulated to assess the effectiveness of the proposed heuristics. All simulations rely on VEINS [149] activating both *Opposite Direction Driving* and *Blue Light* models described in Section 4.2.1. The scenarios are based on different sections of the LuST [43]. LuST focuses on C-ITS and telecommunications and is widely used in VANETs simulations. It includes 24 hours of synthetic mobility communication based on the realistic data validated by the VehicularLab of the University of Luxembourg. The LuST road network was modified using SUMO's *netedit* tool [3] to enable the *Opposite Direction Driving* model, hence enabling the usage of opposite traveling direction lanes for overtakes.

Three different scenarios are defined for the evaluation of the proposed heuristics, two of them being *micro–benchmarks* and one of them being a *macro–benchmark*. Micro–benchmarks are created using the LuST Nano (a subset of the LuST scenario), composed of 73 nodes, 199 edges, and extend for 21 kilometers. In the first scenario of the micro–benchmarks, the EV is forced to pass through a single–lane road, while in the second scenario, only double–lane roads are used. The scope of the micro–benchmarks is to test the effectiveness of the novel heuristics in the two opposite scenarios, demonstrating the actual improvement in terms of reduction of the arrival time of the EV by deploying the proposed heuristics. Note that the first micro–benchmark scenario is the worst case for our proposal in terms of travel time since the single–lane roads force overtaking maneuvers only using the opposite direction lane. Table 4.1 shows the numbers of simulated vehicles for each run of the two micro–benchmark scenarios. Each micro–benchmark scenario is repeated 5 times, with different traffic configurations and number of simulated vehicles. The starting and arrival point of the EV is fixed throughout all the different runs.

The scope of the macro–benchmark is to analyze the impact of the heuristic in a broader and more realistic scenario characterized by both single and multi-lane roads. The macro–benchmark is based on the LuST scenario. The starting and arrival point

	Default		V2V		
	tt [s]	SRE	tt [s]	SRE	Improvement
run #1	106	FC/TP	106	FC/TP	0%
run #2	110	FC/TP	110	FC/TP	0%
run #3	116	*none*	116	*none*	0%
run #4	117	EB	117	*none*	0%
run #5	119	EB/FC/TP	119	ES	0%

Table 4.2: Comparison of travel time and safety–relevant events in the *micro #1* scenario

of the EV are randomly selected with a minimum distance of about 2 kilometers. As for the previous scenario, each experiment was replicated 5 times, using different traffic configurations representing low, medium, and high-traffic situations. Table 4.1 summarizes the configuration used in the benchmark tests in terms of the overall number of simulated vehicles and the average number of vehicles per kilometer (traffic index Ti).

In all the simulated scenarios, the EV is configured with a maximum speed of 1.5 times higher than its current lane speed limit. The EV's length, width, and height are 6.5, 2.16, and 2.86 meters, respectively. The maximum acceleration and deceleration of the EV in normal conditions are set to $2.6m/s^2$ and $4.5m/s^2$ respectively, while in emergency conditions, the maximum deceleration speed is equal to $9m/s$, and the top speed is set to $200km/h$.

Simulation Results

This section presents the results of the evaluation process presented previously.

Micro–benchmarks. The simulation results of the micro–benchmark scenario #1 are presented in Table 4.2. The columns of Table 4.2 shows the travel time (`tt`, expressed in seconds) and the safety-relevant events (`SRE`, previously described in Section 4.2.1) for the default scenario (*blue light + opposite direction driving*) and the same scenario with V2V communication and heuristics enabled. Each row represents the results achieved with the configuration described in Table 4.1.

From the comparison of the Default and V2V results presented in Table 4.2, we notice that there are no increments in the travel time of the EV by applying V2V heuristics on the single–lane scenario. Note that this is a worst-case scenario for the proposed approach. Experiments confirm that the application of V2V communication and heuristics does not decrease the travel time of the EV. However, it positively impacts the safety-relevant events (SRE) triggered during the simulations. The number and severity of the SRE in the V2V scenario are the same in 3 configuration settings. In comparison, in the other two settings, the number and severity of the SRE activated in the V2V scenario are sensibly lower than those activated in the Default scenario. This result is achieved without increasing the travel time of the EV; hence even in this worst-case scenario, the proposed heuristics lead to a measurable improvement.

	Default		**V2V**		
	tt [s]	SRE	tt [s]	SRE	Improvement
run #1	175	JC/TP	150	*none*	14%
run #2	239	*none*	144	*none*	40%
run #3	170	EB/JC/TP	153	*none*	10%
run #4	243	*none*	158	*none*	34%
run #5	162	EB/JC/TP	157	*none*	3%

Table 4.3: Comparison of travel time and safety–relevant events in the *micro #2* scenario

		Default			**V2V**			
	Dist	tt[s]	Avg speed	SRE	tt[s]	Avg speed	SRE	Improv.
run #1	2629	197	48km/h	*none*	192	49km/h	*none*	2.54%
run #2	2979	1057	10km/h	FC/TP	194	55km/h	*none*	81.65%
run #3	2019	117	62km/h	EB	115	63km/h	*none*	1.71%
run #4	3409	198	61km/h	*none*	185	66km/h	*none*	6.57%
run #5	1900	116	59km/h	FC/TP	107	63km/h	*none*	7.76%

Table 4.4: Comparison of travel time and safety–relevant events in the *macro* scenario

Table 4.3 shows the simulation results achieved in the micro–benchmark scenario #2. The table structure is the same as described in the previous scenario.

From the comparison of the Default and V2V results presented in Table 4.3 we notice that applying heuristics developed by exploiting V2V communications can decrease the arrival time of the EV and reduce the number and severity of the SRE. In particular, the travel time of the EV by deploying V2V communication and heuristics improves in all scenarios, from a minimum improvement of 3% up to a maximum of 40%. At the same time, no SRE is observed using the proposed heuristics, hence demonstrating the effectiveness of the proposed solution in incrementing safety during emergencies.

Macro–benchmarks. The simulation results of the macro–benchmark are presented in Table 4.4. The columns of Table 4.4 shows the distance traveled by the EV (`Dist.`, expressed in meters), the travel time (`tt`, expressed in seconds), the average speed (`Avg speed`, expressed in km/h), and the safety-relevant events (`SRE`, previously described in Section 4.2.1) for the default scenario (*blue light + opposite direction driving*) and the same scenario with V2V communication and heuristics enabled. Each row represents the results achieved with the configuration described in Table 4.1.

The results represented in Table 4.4 confirm the results of Tables 4.2 and 4.3, i.e., that it is possible to decrease the arrival time of the EV and to reduce the number and severity of the SRE by applying the heuristics developed by exploiting V2V communications. The travel time of the EV by deploying V2V communication and heuristics improves in all runs, from a minimum improvement of 1.71% up to a maximum of 80%. At the same time, no SRE is observed using the proposed heuristics, hence confirming the effectiveness of the proposed solution in incrementing safety during emergencies.

Chapter 5

Securing V2X against external attackers

5.1 Hardware limitations to secure C-ITS: experimental evaluation and solutions

Cooperative Intelligent Transportation Systems (C-ITS) improve driving experience and safety through secure Vehicular Ad-hoc NETworks (VANETs) that satisfy strict security and performance constraints. Relevant standards, such as the IEEE 1609.2, prescribe network-efficient cryptographic protocols to reduce communication latencies through the Elliptic Curve Qu-Vanstone (ECQV) implicit certificate scheme and the Elliptic Curve Digital Signature Algorithm (ECDSA), that together allow to use smaller certificates and signatures. However, the literature lacks open implementations and performance evaluations for vehicular systems.

This chapter assesses the applicability of IEEE 1609.2 and of ECQV and ECDSA schemes to C-ITSs. It shows that compliance with strict latency requirements defined for C-ITS requires computational resources that many automotive-grade embedded hardware platforms do not meet. One of the outputs of this work is an open implementation of the standard ECQV scheme to benchmark its execution time on automotive-grade boards. The performance of the proposed implementation is evaluated in real road and traffic scenarios. As a final contribution, are proposed and evaluated novel heuristics to reduce the number of signatures to be verified in real C-ITS scenarios.

5.1.1 Prototype implementation and microbenchmarks

This section describes the microbenchmarks of ECDSA and ECQV on representative automotive-grade boards. Results have been obtained using The prototype implementation that supports the NIST P256 curve (*secp256r1*) as required by the IEEE 1609.2 standard. The implementation depends on the *microECC* [102] library, which provides efficient elliptic curve operations on constrained devices thanks to optimized ASM code for ARM platforms.

The implementation complies with the following best practices:

- cryptographic operations comprising secret information adopt time-constant code to avoid timing side-channels (e.g., no conditional branches, time-constant scalar point multiplication [22]);
- no variables are allocated by using dynamic memory allocation;
- random numbers are generated with a hardware TRNG when available; otherwise, software pseudo-random number generator provided by the board library;
- although the implementation supports multiple elliptic curves, the code selects only the required curve at compile time to avoid wasting storage by including useless code;
- elliptic curve points are always transferred using the compressed format defined in the standard SEC 1 [32] to reduce the network overhead. At the same time, cryptographic operations are computed using the uncompressed representation (x, y).

The source code of the prototype implementation is open for reuse and inspection by researchers and industry practitioners[1].

The testbed of the evaluation included the following automotive-grade boards:

- **A72**: based on 64-bit ARM Cortex-A72 quad-core CPU operating at 1.5GHz.
- **A53**: based on 64-bit ARM Cortex-A53 quad-core CPU operating at 1.2GHz.
- **ARM11**: based on ARM1176JZFS single-core CPU operating at 700MHz.
- **M4**: based on a 32-bit Cortex-M4 CPU operating at 80MHz, with 1MB of flash memory and 128KB of RAM.
- **M3**: based on a 32-bit ARM Cortex-M3 CPU operating at 84MHz, with 512KB of flash memory and 96KB of RAM.

These boards are similar to automotive microcontrollers produced by *STMicroelectronis* [2]. For completeness, the benchmarks include results of an **x86_64** architecture: a modern laptop with an Intel Core i7-9750H.

Note that the implementation is single-threaded. This is a typical design choice for most cryptographic algorithms that cannot be easily parallelized without the risk of introducing security vulnerabilities. This is not a limitation since low-power embedded devices are based on single-core architectures, and more powerful architectures can verify concurrently multiple signatures on different cores.

The columns of Table 5.1 represent the platforms used for the evaluation, while the rows represent the cryptographic operations of ECDSA and ECQV: *key generation*

[1] `https://weblab.ing.unimore.it/resources/uECQV.zip`

	x86_64	**A72**	**A53**	**ARM11**	**M4**	**M3**
KeyGen	0.32	1.33	2.82	14.67	109.67	147
CsrGen	0.33	1.37	2.79	14.21	109.73	148
CrtGen	0.62	2.85	5.83	30.02	233.45	317
CRTReception	0.60	2.79	5.62	29.29	231.47	313
Extract	0.34	1.44	3.03	15.38	120.13	162
Sign	0.32	1.44	3.05	15.38	118.00	161
Verify	0.35	1.58	3.36	16.57	133.47	182

Table 5.1: Timings of the operations on the *secp256r1* curve [ms]

(KeyGen), *certificate request generation (CsrGen)*, *certificate generation (CrtGen)*, *certificate reception (CRTReception)*, *extraction (Extract)*, *signature (Sign)* and *verification (Verify)*. All results are expressed in milliseconds. Note that the most powerful ARM architectures (A72 and A53) can execute all operations in a few milliseconds, while cheaper ARM architectures (ARM11) are an order of magnitude slower. Moreover, ultra-low power architectures (M4 and M3) are two orders of magnitude slower, thus requiring a few hundred milliseconds to execute each operation. Note that although the certificate generation operation (*CrtGen*) can usually be deployed on a dedicated server machine, the implemented library could generate novel certificates with low-power devices.

The applicability of the proposed implementation deployed on automotive-grade boards for securing V2V communications is analyzed using the communication requirements defined by the National Highway Traffic Safety Administration (NHTSA) [114]. Those requirements consider multiple vehicle communication scenarios and define the constraints that must be satisfied to guarantee safety. Among these constraints, the most interesting is the *allowable latency*, which is the maximum latency allowed for end-to-end communication and processing of data, and the *communication range*, which is the maximum distance for which the communication is of interest for the receiver. The report identifies 8 high-priority and safety-critical scenarios and, for each of them, defines the associated allowable latency and communication range:

- Pre-Crash Sensing: $20ms$ and $\sim 50m$;
- Traffic Signal Violation Warning: $100ms$ and $\sim 250m$;
- Curve Speed Warning: $1s$ and $\sim 200m$;
- Emergency Electronic Brake Light: $100ms$ and $\sim 300m$;
- Cooperative Forward Collision Warning: $100ms$ and $\sim 150m$;
- Left Turn Assistant: $100ms$ and $\sim 300m$;
- Lane changing Warning: $100ms$ and $\sim 150m$;
- Stop Sign Movement Assistance: $100ms$ and $\sim 300m$;

The timings required to compute all the due cryptographic operations for *sending* and *receiving* an authenticated message are evaluated on the automotive-grade boards. Three device roles are defined: **full sender (FS)**, **direct sender (DS)**, and **receiver (R)**. The device operating as *FS* generates a novel certificate for each communication, as used in the Butterfly protocol [148]. In contrast, a *DS* device requests a valid certificate offline and uses the same certificate for multiple communications. The operations required by a *FS* are *CsrGen*, *CRTReception*, and *Sign*, while the only required operation for a *DS* is *Sign*. The *R* devices must extract the public key from the certificate and verify the validity of the signature, thus requiring the execution of *Extract* and *Verify*. Table 5.2 shows the timings required for the cryptographic operations executed by each role.

	Full Sender	**Direct Sender**	**Receiver**
x86_64	1.25	0.32	0.69
A72	5.61	1.44	3.02
A53	11.46	3.05	6.39
ARM11	58.88	15.38	31.94
M4	459.20	118.00	253.60
M3	622.00	161.00	344.00

Table 5.2: Timings for the cryptographic operations of the roles of the device on different platforms [ms]

Table 5.3 summarizes the applicability of the considered automotive-grade boards for secure V2V communications in the scenarios outlined by the NHTSA. The rows of Tables 5.3 represent the three most strict allowable latencies ($20ms$, $100ms$, and $1000ms$), while the columns represent the different platforms and roles. The *FS* and *DS* sub-columns are used to present the applicability of each board as a sender device with regard to sending rates, and the *R* sub-column shows the maximum number of messages that the platform can validate as a receiver within the allowed latency. Please note that the actual number of received messages depends on the number of vehicles within the communication range; hence it might exceed the validation capability of a board. Section 5.1.2 investigate the capabilities of the boards in real traffic scenarios.

In the following are summarized the main results. With an allowable latency of 20 ms, it is possible to deploy the $x86\_64$, the $A72$ and the $A53$ boards in the *FS* role, while in the *DS* role, it is also possible to deploy the $ARM11$ board. When using the boards as *R* role, only the $x86\_64$, $A72$, and $A53$ systems can verify the signatures of 29, 6 and 3 incoming messages within the allowable latency. In safety-critical applications with allowable latency of 100 ms, it is possible to use the $x86\_64$, $A72$, $A53$, and $ARM11$ boards in all roles, while the $M4$ nor $M3$ boards cannot be deployed for any role. The maximum number of messages that the boards are able to verify within the $100ms$ allowable latency in the *R* role are 144, 33, 15, and 3 for the $x86\_64$, $A72$, $A53$, and $ARM11$ boards, respectively. Moreover, it is essential to highlight that both the $M4$ and $M3$ boards cannot send any message nor verify any signature within allowable latencies of 20 and 100 ms due to their limited computational power. Finally, with an allowable latency of 1000 ms, it is possible to deploy all the boards for all the roles, but the $M4$ and $M3$ boards can only verify 3 and 2 signatures, respectively. Finally, the

	x86_64			A72			A53			ARM11			M4			M3		
	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R
20 ms	✓	✓	29	✓	✓	6	✓	✓	3	×	✓	×	×	×	×	×	×	×
100 ms	✓	✓	144	✓	✓	33	✓	✓	15	✓	✓	3	×	×	×	×	×	×
1000 ms	✓	✓	1449	✓	✓	331	✓	✓	156	✓	✓	31	✓	✓	3	✓	✓	2

Table 5.3: Analysis of the constraints of the boards. Applicability for both sender roles and maximum number of received messages for the receiver role

$M4$ and $M3$ boards can still be deployed for non-safety-critical scenarios with higher allowable latencies, as demonstrated in [129].

5.1.2 Simulation in realistic traffic scenarios

This section assesses the applicability of the automotive-grade boards to realistic traffic scenarios in simulated environments.

Section 5.1.2 describes the characteristics of the traffic scenarios, Section 5.1.2 reports the results of the simulations, and Section 5.1.2 analyzes the applicability of the boards. In Section 5.1.2, the results of the simulations are adopted to assess the advantages of implicit certificates over certificate chains in V2V networks.

Realistic traffic scenarios

We collected traffic data from the city of Modena for multiple road and traffic scenarios, and we used that data to recreate realistic simulations. Each scenario is characterized by multiple parameters, including the average number of vehicles per hour, the distribution of the road vehicles per hour, and the average number of vehicles in a kilometer for different roads. For clarity of exposure, this section describes the traffic scenarios by the average number of vehicles per kilometer (traffic index Ti) during rush hour in descending order.

- **Roundabout (Grapes) [$Ti = 27.33$]**: represents the roundabout located in the southwest part of the city, nearby the Department of Engineering "Enzo Ferrari" of the University of Modena and Reggio Emilia. The Grapes scenario connects 2 segments of a 6-lanes expressway (two road segments in each direction that become three in the proximity of the roundabout) with one segment of urban road and one segment of extra-urban road, for a total of 12 road kilometers. The number of simulated vehicles during normal traffic and rush hour are 167 and 328, respectively.
- **Highway (H-Way) [$Ti = 14.41$]**: represents the highway junction located between the "Campogalliano" and the "Modena Nord" highway toll booths. The Highway scenario includes two highway segments connected with a T-junction. The first highway segment is a 4-lanes highway (two lanes for each direction), while the second is a 6-lanes highway (three lanes for each direction). Vehicles can only enter or exit the highway through an entry or exit lanes. The total road

length for this scenario is 54 kilometers. The number of simulated vehicles during regular traffic and rush hour are 131 and 778, respectively.

- **University campus (Campus) [**$Ti = 6.58$**]**: represents the area surrounding the Department of Engineering "Enzo Ferrari", located in the southwest part of the city, near the central city hospital. This scenario is composed of multiple urban roads connected through plain intersections. The Campus scenario includes a residential area, a shopping center, the hospital, and the engineering campus, for a total of 52 kilometers of roads. The number of simulated vehicles during regular traffic and rush hour are 159 and 342, respectively.
- **Modena Automotive Smart Area (MASA) [**$Ti = 4.24$**]**: represents the Modena Automotive Smart Area [1], located behind the central railway station of Modena. The MASA scenario is a roughly rectangular residential area, connecting 4 perimeter urban roads through roundabouts for a total of 34 kilometers. The number of simulated vehicles during regular traffic and rush hour are 77 and 144, respectively.

Simulations results

The simulation is powered by VEINS [149]. Veins is an open source framework for vehicular network simulations based on OMNeT++ [167] for the simulation of networks and SUMO [101] (Simulation of Urban MObility) for the simulation of road traffic. The simulation adopt the IEEE 802.11p, IEEE 1609.4 DSRC/WAVE [51], Physical Layer [23], Obstacle Shadowing [150] and Antenna Patterns [50] modules. The first two modules (IEEE 802.11p and IEEE 1609.4 DSRC) extend the VEINS capabilities by enabling the DSRC/WAVE stack, including Quality-of-Service channel access, the Wave Short Message (WSM) management, and periodic beaconing of BSMs. For details on the protocols, please refer to Chapter 2.2. The other modules simulate the propagation and attenuation of the wireless signals to recreate proper signal coverage of messages in urban environments.

The simulations aim to provide meaningful insights about the number of messages received by all the vehicles in a particular road scenario; hence the four scenarios are based on the city of Modena. Each scenario is used twice in the simulations, once to simulate the communication between vehicles during rush hour in a high traffic condition (*Rush*) and once to simulate the communications during normal traffic conditions (*Normal*).

The simulated environment is recreated using the road and polygonal maps of the buildings available at Open Street Map [157]. The vehicles simulated in the scenarios are programmed to send beacon messages with a frequency of $10Hz$ (i.e., one message each 100 milliseconds), as recommended by the SAE J2945-201712 [139] standard. The four scenarios were simulated for five minutes for both *Rush* and *Normal* traffic conditions. Results are summarized in Table 5.4, including the total number of vehicles and messages exchanged in the simulated VANETs.

In the following, there is an analysis of the results of the simulations that are of interest concerning the specifications included in the NHTSA report for 20 ms and 100 ms safety-critical messages, respectively (see Section 5.1.1).

	Highway		**MASA**		**Campus**		**Grapes**	
	Rush	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
# cars	778	131	144	77	342	159	328	167
# messages	21,227,443	830,210	1,106,882	262,669	4,645,905	980,121	15,462,204	3,427,977

Table 5.4: Overview of the simulation results on the different scenarios in both rush hour and normal traffic conditions

20 ms allowable latency:

The analysis results on the number of messages received by the vehicles in the realistic scenarios with allowable latency of 20 ms and a distance of 50 m are presented in Table 5.5. The columns represent the simulated areas with different traffic conditions, and the rows represent the maximum, minimum, and average (rounded to the lowest integer) number of messages received by a single vehicle within the allowable latency. The table shows that the maximum number of messages received by the vehicles ranges from 2 to 5 and 5 to 12 in *Normal* and *Rush* traffic conditions. The minimum number of messages in all scenarios and conditions is 1.

	Grapes		**Highway**		**Campus**		**MASA**	
	Rush	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
# max of msgs	12	5	6	2	5	3	5	2
# min of msgs	1	1	1	1	1	1	1	1
# avg of msgs	4	2	2	1	2	1	2	1

Table 5.5: Results of the simulations with allowable latency of 20 ms and a distance of 50 m

100 ms allowable latency

Table 5.6 presents the analysis results of the number of messages received by the vehicles in the realistic scenarios with allowable latency of 100 ms and within all distances. The table shows that the maximum number of messages received by the vehicles ranges from 27 to 83 and 61 to 175 in *Normal* and *Rush* traffic conditions. The minimum number of messages in all scenarios and conditions is 1.

	Grapes		**Highway**		**Campus**		**MASA**	
	Rush	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
# max of msgs	175	83	89	38	69	34	61	27
# min of msgs	1	1	1	1	1	1	1	1
# avg of msgs	85	42	33	12	25	12	20	9

Table 5.6: Results of the simulations with a maximum allowable latency of $100ms$ (all distances)

Applicability of the automotive-grade boards

The applicability of the automotive-grade boards is determined by an analytical evaluation of the time required by each board to verify the authenticity of the messages received by a vehicle in each traffic scenario. To this aim, the maximum number of messages received by a vehicle is multiplied by the time required by the boards to extract the implicit certificate and verify the digital signature (see the *Receiver* timings in Table 5.2 of Section 5.1.1).

Table 5.7 shows the results for 20 ms allowable latency. The rows include the boards, and the columns include the traffic scenarios. It is important to highlight in gray the cells of the table that refer to the boards that do not satisfy the latency requirement. The results show that no automotive-grade board can verify all the received messages within 20 ms in the worst-case traffic scenario (*Rush*) by using a single core. Only the *A*72 board can satisfy all *Normal* traffic scenarios. Moreover, the reference $x86\_64$ architecture can verify the signatures of all the received messages in all scenarios.

	Grapes		**Highway**		**Campus**		**MASA**	
	Rush	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
x86_64	8	4	4	2	4	2	4	2
A72	36	15	18	6	15	9	15	6
A53	77	32	38	13	32	19	32	13
ARM11	383	159	191	64	160	96	160	64
m4	3043	1268	1521	507	1268	761	1268	507
m3	4128	1720	2064	688	1720	1032	1720	688

Table 5.7: time [*ms*] required by each board to verify all the signatures with a maximum allowable latency of 20ms using a single core.

Table 5.8 shows the results for 100 ms allowable latency. The table omits the boards with timings greater than 4000 ms. The results show that no automotive-grade board can verify all the received messages within 20 ms in the worst-case traffic scenario (*Rush*) by using a single core. Only the *A*72 board can satisfy all *Normal* traffic scenarios. Moreover, the $x86\_64$ reference architecture can verify the signatures of all the received messages in all scenarios. The results show that no automotive-grade board can verify all the received messages in both traffic scenarios. Only the $x86\_64$ reference architecture can verify all the signatures in all scenarios, except for the *Grapes* scenario in the *Rush* traffic condition.

The results highlight that even modern vehicles fail to verify all safety-related messages in real urban scenarios. As a result, a few safety messages will either be ignored entirely or accepted without verification. Both solutions are unacceptable since they expose drivers and road users to safety risks and cyberattacks. However, it is essential to note that improved results could be achieved if we consider the concurrent verification of digital signatures by multiple cores of a single board. To this aim, it is possible to operate an approximated analytical evaluation by dividing the timings reported in Tables 5.7 and 5.8 by the number of cores of the considered board.

	Grapes		**Highway**		**Campus**		**MASA**	
	Rush	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
x86_64	121	57	61	26	48	23	42	19
A72	529	251	269	115	208	103	184	82
A53	1118	530	569	243	441	217	390	173
ARM11	5590	2651	2843	1214	2214	1086	1948	862

Table 5.8: time [ms] required by each board to verify all the signatures with a maximum allowable latency of 100 milliseconds using a single core

Comparison with certificate chains

The following is a comparison of the effectiveness of implicit certificates with respect to explicit certificate chains.

The analysis of the sizes of implicit and explicit certificates for ECDSA instantiated over a 256 bit curve (*secp256r1*), that is the recommendation of IEEE 1609.2 standards for vehicular communications. Each BSM message transmitted by using the DSRC stack over V2V networks is authenticated with implicit certificates and includes 64 bytes for the ECDSA digital signature, $93 - 180$ bytes for the implicit certificate (the size depends on the attached metadata and padding), and a minimum of 254 bytes for the payload (see Section 2.2.1.1). Hence, the average size of an entire message ranges from 410 to 500 bytes.

Since an ECDSA digital signature size is 64 bytes, a message authenticated with explicit certificates (with a certificate chain with no intermediate CAs) is about 64 bytes longer (the actual size may vary depending on padding). The average size of an entire message ranges from 474 to 565 bytes. Considering these estimations, using ECQV implicit certificates reduces network usage from 10% to 15%.

The NHTSA report [113] defines variable data rates and distance ranges for V2V communications, which nominally range from 3 Mb/s to 27 Mb/s and from 300 m to 1000 m. The maximum data rate suggested by the NHTSA report for the control channel related to safety applications is 6 Mb/s, while the 27 Mb/s data rate should only be used for bursts. Increasing the data rate from 6 Mb/s to 27 Mb/s can cause higher packet losses and reduced communication ranges. Considering a bandwidth of 6 Mb/s, a latency of 100 ms, an average packet size of 460 bytes for ECQV implicit certificates and 520 bytes for explicit certificates, the maximum number of messages supported by the network is 163 and 144 messages respectively.

The validation of an explicit certificate requires two digital signature verification operations: the first to validate the signature of the explicit certificate and the second to validate the signature of the BSM. On the other hand, the ECQV implicit certificate scheme requires only one signature verification, that is, to validate the signature of the received BSM. However, the receiver must first extract the sender's public key by using the ECQV *public key extraction* operation. An analytic estimation based on the timings of the operations obtained by using the proposed implementation on the automotive-grade boards (see Table 5.1) lets us conclude that implicit certificates are comparable to explicit certificates in terms of computational costs.

The analysis confirms that using implicit certificates, as indicated by the IEEE 1609.2 standard, is advantageous for reducing network overhead (and increasing the throughput of wireless vehicular networks).

5.1.3 Prioritization strategy based on senders position

The results of the applicability evaluation in realistic scenarios (Section 5.1.2) show that the computational constraints of the boards may prevent verification of some messages within safety-critical timings. This section proposes a scheduling strategy to identify a subset of messages that should be verified with a higher priority. The strategy adopts heuristics based on the relative positions of the vehicles. Upon reception of a message, the receiving vehicle extracts the geographical coordinates of the sender. If the coordinates are within a specific area surrounding the receiver, they are verified and analyzed before the other messages. Two rationales guide this approach. First, messages sent by vehicles relatively far from the receiver do not convey meaningful information for immediate reaction; hence it is safe to ignore their content, thus decreasing the number of signature validations. Second, messages sent by vehicles that are equally distanced from the receiver may have different importance depending on their relative positions. In the following, there is an analysis of the impact of the proposed strategy in the scenarios considered in Section 5.1.2 by considering three different areas: circular, elliptical, and forward-facing. It is considered the same scenarios of the simulation presented in Section 5.1.2 and analyzes the number of messages sent within the areas. For each area and scenario, the board's average time to analyze all the prioritized messages is evaluated.

Circular area heuristic

This heuristic considers a distance-only approach where the receiving vehicle controls whether the sender vehicles are within a circle with a radius r. The values of the radius r in the analysis are $150, 250$, and 300 meters, which are the distance requirements proposed by the NHTSA report for the $100ms$ latency. The results for a radius of 1000 meters, the typical maximum communication range of DSRC, are reported as a comparison. Table 5.9 shows the average number of messages sent within the circle of radius r. The columns represent the scenarios used in the evaluation, and the rows represent the values of the radius r. The table shows that, on average, the 26%, 59%, and 66% of the messages are sent by vehicles within a maximum distance of 150, 250, and 300 meters. All messages are within the maximum distance of 1000 meters.

Table 5.10 shows the timings required for each board to verify message authenticity within the circular area within $100ms$. Cells of the table with a gray background identify the scenarios that cannot be deployed due to high timings. It is possible to deploy the $A72$ board to satisfy almost all scenarios with a radius of 150 meters (the only exception is Grapes with heavy traffic conditions). Note that the $A53$ platform can only satisfy a minimal number of scenarios, and the $ARM11$ is not able to satisfy any scenario. For this reason, the results omit the $m4$ and $m3$ boards that have even worse performance.

		Grapes		**Highway**		**Campus**		**MASA**	
		Rush	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
100 ms	$\leq 150m$	54	23	21	8	18	9	19	8
	$\leq 250m$	113	52	48	20	40	20	39	18
	$\leq 300m$	125	59	55	22	47	23	44	20
	$\leq 1000m$	175	83	89	38	69	34	61	27

Table 5.9: Number of messages in different ranges according to the distance of the sender vehicle using the circular area

		Grapes		**Highway**		**Campus**		**MASA**	
	r	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
x86_64	$150m$	37	16	14	6	12	6	13	6
	$250m$	78	36	33	14	28	14	27	12
	$300m$	86	41	38	15	32	16	30	14
	$1000m$	121	57	61	26	48	23	42	19
A72	$150m$	163	69	63	24	54	27	57	24
	$250m$	341	157	145	60	121	60	118	54
	$300m$	378	178	166	66	142	69	133	60
	$1000m$	529	251	269	115	208	103	184	82
A53	$150m$	345	147	134	51	115	58	121	51
	$250m$	722	332	307	128	256	128	249	115
	$300m$	799	377	351	141	300	147	281	128
	$1000m$	1118	530	569	243	441	217	390	173
ARM11	$150m$	1725	735	671	256	575	287	607	256
	$250m$	3609	1661	1533	639	1278	639	1246	575
	$300m$	3993	1884	1757	703	1501	735	1405	639
	$1000m$	5590	2651	2843	1214	2204	1086	1948	862

Table 5.10: Timing $[ms]$ of the automotive-grade boards with a circular heuristic for different radius values with a latency of $100ms$.

Elliptical area heuristic

The elliptical area considers the surroundings of the vehicle by building an ellipse centered on the vehicle. This type of area gives more priority to vehicles that are behind or in front of the receiver, and reduces the priority of side vehicles. This heuristic is useful in particular in all those scenarios that do not require information from the side vehicles, as the Pre-Crash Sensing in one-way road (e.g. Highway), Lane Change Warning that provides a warning to the driver if an intended lane change may cause a crash with nearby vehicles, Cooperative forward collision warning that is designed to aid the driver in avoiding or mitigating collisions with the rear-end of vehicles, Emergency Electronic Brake light application that sends a message to other vehicles following behind, and Left Turn Assistant with regard to vehicles coming from the opposite direction.

The elliptical area is modelled by using the equation $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$, where a denotes the side distance and b denotes the distance from the front and rear vehicles. The parameter a is fixed, with a value of 25 meters. Instead, the parameter b includes $150, 250, 300$ and 1000 meters in case of $100ms$ allowed latency, and a single value of b of 50 meters for the $20ms$ allowed latency. As a result, a total of five different ellipses is considered.

Table 5.11 shows the average number of messages sent within the different ellipses. The columns represent the different scenarios and the rows represent the values of the two parameters a and b. The table shows that the number of messages received within $20ms$ allowed latency ranges from 6% to 10% for the different scenarios, and on average the 17%, 18% and 20% of messages are received within $100ms$ for values of b equal to $150m$, $250m$ and $300m$, respectively.

Table 5.12 shows the timings required for each board to verify messages authenticity within $20ms$ and $100ms$ allowed latencies. Note that it is possible to deploy the $A72$ and $A53$ boards to satisfy almost all scenarios within $20ms$ allowed latency (the only exception is Grapes with heavy traffic conditions). Instead, $ARM11$ and less powerful boards are not able to satisfy any scenario within $20ms$ allowed latency. For $100ms$ allowed latency, the Grapes scenario with heavy traffic conditions still cannot be satisfied by any board (with the exception of the reference $x86\_64$ architecture) and the Grapes scenario with normal traffic conditions can be satisfied only by the $A72$ board. The $ARM11$ board can satisfy only the Highway with normal traffic conditions. Less performing boards (that are omitted from the table) cannot satisfy any scenarios.

Forward-facing area heuristic

The forward-facing area considers a circular arc in front of the vehicle. This type of area helps to improve all the scenarios in which priority should be given to front vehicles like the Pre-Crash Sensing, Left Turn Assistant, and Stop Sign Movement.

The angular aperture of the circular arc is denoted as α, for which it is considered values of 90, 135, and 180 degrees. Moreover, it is considered a radius of the arc equal to 50 and 300 meters for the $20ms$ and $100ms$ allowed latencies, respectively. These values are compliant with recommendations by the NHTSA for the scenarios considered.

a	b	**Grapes** *Rush*	**Grapes** *Normal*	**Highway** *Rush*	**Highway** *Normal*	**Campus** *Rush*	**Campus** *Normal*	**MASA** *Rush*	**MASA** *Normal*
20 ms									
$25m$	$50m$	8	3	3	1	3	2	3	2
100 ms									
$25m$	$150m$	40	14	7	2	9	4	11	4
$25m$	$250m$	43	17	8	3	10	5	13	6
$25m$	$300m$	45	18	8	3	11	5	14	6
$25m$	$1000m$	51	22	9	3	12	6	17	7

Table 5.11: Number of messages originated within an elliptical area centered in the receiving vehicle

	a	b	**Grapes** *Rush*	**Grapes** *Normal*	**Highway** *Rush*	**Highway** *Normal*	**Campus** *Rush*	**Campus** *Normal*	**MASA** *Rush*	**MASA** *Normal*
20 ms										
x86_64	$25m$	$50m$	6	2	2	1	2	1	2	1
A72	$25m$	$50m$	24	9	9	3	9	6	9	6
A53	$25m$	$50m$	51	19	19	6	19	13	19	13
ARM11	$25m$	$50m$	256	96	96	32	96	64	96	64
100 ms										
x86_64	$25m$	$150m$	28	10	5	1	6	3	8	3
	$25m$	$250m$	30	12	6	2	7	3	9	4
	$25m$	$300m$	31	12	6	2	8	3	10	4
	$25m$	$1000m$	35	15	6	2	8	4	12	5
A72	$25m$	$150m$	121	42	21	6	27	12	33	12
	$25m$	$250m$	130	51	24	9	30	15	39	18
	$25m$	$300m$	136	54	24	9	33	15	42	18
	$25m$	$1000m$	154	66	27	9	36	18	51	21
A53	$25m$	$150m$	256	89	45	13	58	26	70	26
	$25m$	$250m$	275	109	51	19	64	32	83	38
	$25m$	$300m$	288	115	51	19	70	32	89	38
	$25m$	$1000m$	326	141	58	19	77	38	109	45
ARM11	$25m$	$150m$	1278	447	224	64	287	128	351	128
	$25m$	$250m$	1373	543	256	96	319	160	415	192
	$25m$	$300m$	1437	575	256	96	351	160	447	192
	$25m$	$1000m$	1629	703	278	96	383	192	543	224

Table 5.12: Timing [ms] of the automotive-grade boards with an elliptical heuristic for different parameters.

Table 5.13 shows the average number of messages sent within the different forward-facing areas. The columns represent the different scenarios, and the rows represent the parameter values α. The table shows that the average number of received messages are 3%, 4% and 6% for $20ms$ allowed latency, and 6%, 8% and 36% for $100ms$ allowed latency for 90, 135 and 180 degrees, respectively.

Table 5.14 shows the timings required for each board to verify messages' authenticity within $20ms$ and $100ms$ allowed latencies. It is possible to deploy the $A72$ and $A53$ boards to satisfy almost all scenarios within $20ms$ allowed latency. Instead, $ARM11$ and less powerful boards cannot satisfy any scenario within $20ms$ allowed latency. For $100ms$ allowed latency, the $A72$ board satisfies almost all scenarios. The $A53$ board satisfies all scenarios for α equal to 90 and 135 degrees, but it can only satisfy a few scenarios for α equal to 180 degrees. The $ARM11$ board can satisfy only shallow traffic scenarios.

		Grapes		**Highway**		**Campus**		**MASA**	
	α	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
20 ms	90°	1	1	1	1	1	1	1	1
	135°	1	1	1	1	1	1	1	1
	180°	5	2	2	1	2	1	2	1
100 ms	90°	13	6	4	1	6	3	3	1
	135°	15	7	6	2	6	3	5	2
	180°	69	31	24	11	24	12	22	12

Table 5.13: Number of messages in different ranges according to the distance of the sender vehicle using the Forward-facing Area

Evaluation of the prioritization strategy

The results demonstrate that prioritizing messages based on the sender's position is a viable solution to validate all relevant safety messages in actual traffic conditions by adopting general-purpose automotive-grade boards. By using the circular area heuristic described in Section 5.1.3 it is possible to confirm that the distance-prioritization scheme implicitly specified in the NHTSA requirements helps to reduce the number of signatures by up to 75% with a distance of $150m$. However, this heuristic is insufficient with greater distances (e.g., $300m$) and heavy traffic scenarios, as shown in Table 5.10. By introducing more complex heuristics like the elliptical area described in Section 5.1.3 it is possible to improve the applicability of the boards enabling low power boards like the $A53$ to be suitable in most scenarios as shown in Table 5.12. Finally, with the Forward-facing area heuristic 5.1.3, the applicability of the boards can be further improved by reducing by up to 90% the number of the messages in some specific scenarios (see Table 5.14).

A question worth analyzing is whether the proposed prioritization strategy affects the security guarantees of the protocol. In particular, this work shows that the strategy does not improve nor decrease security guarantees. Analyzing the security guarantees concerning the IEEE 1609.2 standard and the NHTSA specifications, we notice that

		Grapes		**Highway**		**Campus**		**MASA**	
	α	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*	*Rush*	*Normal*
20 ms and 50 m									
x86_64	90°	1	1	1	1	1	1	1	1
	135°	1	1	1	1	1	1	1	1
	180°	4	2	1	1	2	1	2	1
A72	90°	3	3	3	3	3	3	3	3
	135°	4	3	3	3	3	3	3	3
	180°	15	7	6	3	7	4	7	3
A53	90°	7	6	6	6	6	6	6	6
	135°	9	6	6	6	6	6	6	6
	180°	33	15	13	6	15	9	14	6
ARM11	90°	37	32	32	32	32	32	32	32
	135°	44	32	32	32	32	32	32	32
	180°	164	74	66	32	75	45	71	32
100 ms and 300 m									
x86_64	90°	9	4	3	1	4	2	2	1
	135°	10	5	4	1	4	2	3	1
	180°	48	21	16	8	17	8	15	8
A72	90°	40	18	12	3	18	9	10	4
	135°	45	22	18	5	18	9	15	6
	180°	208	93	71	34	72	36	68	37
A53	90°	85	37	25	7	37	18	21	8
	135°	95	47	39	10	37	18	32	12
	180°	440	197	151	72	153	77	143	78
ARM11	90°	424	186	127	33	186	91	106	40
	135°	473	233	195	51	187	92	158	60
	180°	2201	982	754	361	765	386	716	390

Table 5.14: Timing [ms] of the automotive-grade boards with a forward-facing heuristic for different angular apertures.

the proposed strategy does not conflict with the safety requirements specified by the NHTSA (Section 5.1.1) and can be potentially integrated with other prioritization strategies based on the classes of messages. Moreover, the proposed strategy requires that all messages' authenticity be verified using the ECQV scheme as specified by the IEEE 1609.2 standard.

Note that the proposed strategy is not meant to defend receiving vehicles against Denial of Service (DoS) attacks by adversaries within the receiver's communication range. First, attackers could transmit any number of messages to saturate the wireless communication channel and thus completely jam the wireless network [5]. This type of attack is outside the scope of this proposal because it regards the physical layer of the communication protocols stack. Second, attackers could send messages with fake sender positions, either with legitimate or illegitimate digital signatures. The first case is outside the scope of this work because it considers legitimate senders only to send legitimate information. The effects of false information sent by legitimate vehicles are further analyzed in [128]. In the second case, messages with illegitimate digital signatures are discarded by receivers. However, the proposed prioritization strategy is based on senders' positions before verifying message authenticity. A potential attack could envision sending a high number of messages with fake sender positions that satisfy the heuristic that could saturate the verification throughput of the receiver, thus possibly causing a Denial of Service. Despite this attack, the proposed strategy does not introduce security issues. First, since the prioritization heuristics are modeled after the NHTSA heuristics, the messages that are discarded due to the attacks also have a lower priority for safety recommendations. Second, an adversary with the same capabilities (e.g., sending a message within the communication range of a specific vehicle) can operate an even more powerful DoS attack by simply jamming the physical layer of the wireless network, as mentioned above.

5.2 Implicit certificates for Edwards Curve Digital Signature Algorithms

Elliptic Curve Qu-Vanstone (ECQV) implicit certificates represent space-efficient alternatives to well-known certificate chains, but they are typically only used in combination with the Elliptic Curve Digital Signature Algorithm (ECDSA). Although the Edwards-curve Digital Signatures Algorithm (EdDSA) is a popular alternative thanks to its high performance and security, its adoption in combination with ECQV has not been investigated yet. This chapter analyzes how to combine ECQV and EdDSA, and shows that a straightforward adoption of the original schemes is not possible, and propose a few modifications to their designs to allow an efficient and secure composition. The resulting scheme complies with original EdDSA validators but requires small modifications to EdDSA secret keys and to ECQV key generation. This work includes a formal security proof for the proposed composition, and an experimental evaluation that shows better performance with regard to standard compositions of ECQV and ECDSA.

5.2.1 Overview on Implicit Certificates

Implicit certificates have been first proposed for signature schemes based on the RSA trapdoor [72] and denoted as *self-certified public keys*. However, as their main benefit with regard to certificate chains is space-efficiency, research focused on implicit certificates for Elliptic Curve Cryptography (ECC). The first implicit certificate scheme for ECC is the Optimal Mail Certificates (OMC) [123], whose design was focused on postal revenue collection and included a digital signature scheme with partial message recovery, the Elliptic Curve Pintsov-Vanstone Scheme (ECPVS). Although the security of OMC with ECPVS has been proved [26], the combination of OMC with other signature schemes might introduce security vulnerabilities [25, 26]. More recent proposals regarding implicit certificates include applications for efficient secure communications in VANETs [10], involving extensions to OMC with key-prefixing technique previously reported in [37] to prevent known attacks [25] and tight integration with ECIES for also guaranteeing message confidentiality. Our approach differs from [10] by complying as much as possible with EdDSA, which is widely known and implemented for many architecture, thus allowing adoption of existing optimized implementations to achieve high performance with little effort. A different approach for space-efficient message authentication could involve adopting BLS digital signatures [18], which produce digital signatures with size comparable to ECPVS. However, they are based on so-called pairing cryptography, which involves stronger security assumptions and requires higher computational resources.

The most popular composition of ECQV is with NIST ECDSA, whose security has been proved in [25]. Although ECDSA is a widespread standard, it also has a few drawbacks including higher memory and computational requirements with regard to EdDSA. Moreover, many real-world vulnerabilities showed high difficulty in providing secure timing side-channel resistant implementations [7, 13, 70, 115, 168, 169]. Investigating the combination of ECQV and EdDSA seems worth effort as EdDSA improves over ECDSA both in terms of security and performance [15, 137]. The complete addition law of the underlying Edwards curves [14] allow easier time-constant implementations, and potential implementation flaws are mitigated by additional design choices, such as key clamping (see Section 5.2.2). EdDSA is already widespread in many popular protocols and applications, and proved to be convenient for constrained devices [35, 67, 78]. Formal security proofs exist for the combination of Schnorr-based implicit certificates with Schnorr signatures that apply key-prefixing during signing [37]. As ECQV and EdDSA represent specifications of these classes of schemes, the existing proofs also apply to our proposal. Although this work focuses most on cryptographic engineering strategies, to further improve confidence in the proposed combination of ECQV and EdDSA it also proposes a novel security proof that follows the rationale of the original proof proposed for the combination of ECQV and ECDSA [25].

Finally, it is important to observe that although this work focuses on the original design of EdDSA [15], our proposal can also comply with *hedged* variants for fault attack resilience [107] and with alternative point encoding strategies [75, 76] (see Section 5.2.5).

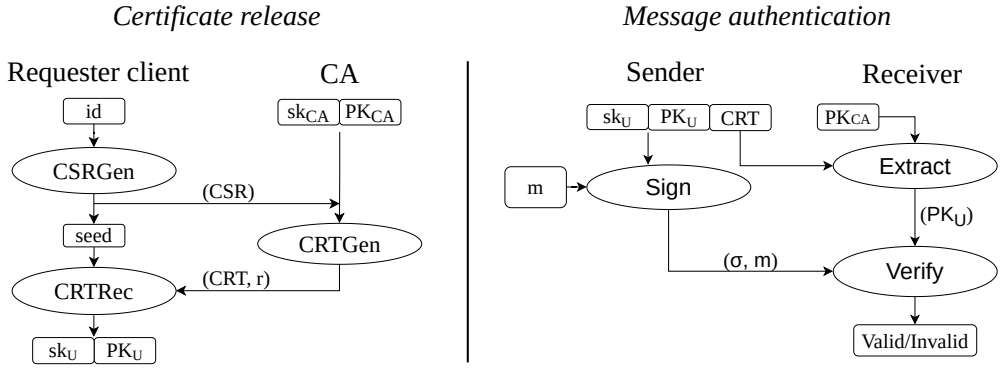


Figure 5.1: Operation flows for release and usage of ECQV implicit certificates

5.2.2 Base knowledge and notations

ECQV framework. Figure 5.1 shows the main data and routines of the ECQV framework combined with a generic digital signature scheme: on the left the release of an implicit certificate, on the right the usage of the certificate for message authentication. Certificate release requires a *requester client* to generate a *certificate sign request* (*CSR*) through a *CSR generation* routine (*CSRGen*), that takes as input identity information (and possibly other metadata) for which the certificate is requested (*id*). The routine returns the temporary private key *sk*, the *CSR* that includes *id* and a temporary public key (*PK*). The client sends the *CSR* to the CA. The CA can validate the *id* field within the *CSR* based on the application context. If the CA accepts the *CSR*, it produces the certificate (*CRT*) and a *private contribution data* (r) by using the *certificate generation* routine (*CRTGen*). The client uses *sk*, *CRT* and r to generate a user key pair (sk_U, PK_U) through the *certificate reception* routine (*CRTRec*). To send an authenticated message, a *sender signs* message m by using its private key sk_U and obtains the signature σ. Then, it sends m, σ and *CRT* to the receiver. The receiver recomputes PK_U by using the *public key extraction* routine (*Extract*), and verifies authenticity of m by using σ and PK_U as inputs of the *verify* routine.

ECQV trade-offs. While certificate chains require sending both the public key and the certificate of the CA (including a digital signature and metadata) over the communication channel, ECQV implicit certificates only require sending the implicit certificate. The size of ECQV implicit certificates is equal to the public key size plus the size of metadata. Hence, implicit certificates reduce network overhead by avoiding sending an amount of data equal to the size of a signature. However, ECQV schemes have two limitations. First, since computing the key pair of the client (sk_U, PK_U) requires a contribution of the CA, an ECQV scheme must be specifically designed for a given digital signature scheme. Second, the receiver cannot verify the authenticity of the implicit certificate by using the *certificate reception* routine, but can only assume that the implicit certificate is authentic after verifying authenticity of a signed message. Although these disadvantages limit applicability of implicit certificates to Web protocols, their smaller network overhead make them the preferred choice in network-constrained environments.

EdDSA notations and conventions. The main symbols used throughout the

Symbol	Description
$\|E\|$	Order of the cyclic group formed by the (twisted) Edwards curve
q	Order of the field $\mathbb{Z}_q$ that underlies the Edwards curve
B	Generator of the prime order subgroup used for EdDSA signatures
$\langle B \rangle$	Prime order subgroup generated by B
ℓ	Order of the cryptographic subgroup $\langle B \rangle$
c	Base 2 logarithm of the cofactor of the cryptographic subgroup
t	Size of scalars expressed in bits minus two
b	Size of coordinate and scalars encoding expressed in bits
b_8	Size of coordinate and scalars encoding expressed in bytes
$\mathcal{O}$	Neutral element of the group formed by the Edwards curve
$\underline{P}$	Byte encoding of elliptic curve point P
$\mathcal{H}_{2b}(\cdot)$	Hash function $\{0,1\}^* \rightarrow \{0,1\}^{2b}$
$\mathcal{H}_{\ell}(\cdot)$	Hash function $\{0,1\}^* \rightarrow \mathbb{Z}_{\ell}$

Table 5.15: Notations for Edwards curves and EdDSA parameters

manuscript for EdDSA are summarized in Table 5.15. All parameters can be found in [15] and [137], but this thesis might adopt different notations to avoid conflicts with parameters of ECQV scheme. An EdDSA scheme is based on a *twisted Edwards curve* E, which is defined over a finite field $\mathbb{Z}_q$ of prime order q [14]. The curve forms a cyclic group of non-prime order $|E|$, with neutral element $\mathcal{O}$, that includes a subgroup of prime order ℓ generated by a base point B. We denote as $\langle B \rangle$ the subgroup generated by B, and as c the base 2 logarithm of the cofactor of the subgroup (that is, $|E| = 2^c\ell$). We also denote as $t = [\lceil log_2(2^c\ell) \rceil - 2]$ the size of the scalars expressed in bits minus two. Scalars and curve coordinates are encoded by using variables of size $b = 8\left\lceil \frac{\lceil log_2(q) \rceil + 1}{8} \right\rceil$-bit. We also adopt the size expressed in bytes denoted as $b_8 = \frac{b}{8}$. Given elliptic curve point P, we denote as $\underline{P}$ its compressed byte encoding representation. We denote as $\mathcal{H}_{2b}(\cdot)$ a cryptographic hash function that accepts a variable length byte string as input and outputs a digest of size $2b$-bit. We also denote as $\mathcal{H}_{\ell}(\cdot)$ a hash function based on $\mathcal{H}_{2b}(\cdot)$ that outputs an integer in $\mathbb{Z}_{\ell}$. This thesis proposes examples for a specific instance of EdDSA for the 128-bit security level, that is Ed25519 based on the edwards25519 curve [15, 137]. Values of parameters for this scheme used throughout the thesis include $b = 256, b_8 = 32, t = 254, c = 3, q = (2^{255} - 19), \ell = (2^{252} + 27742317777372353535851937790883648493)$. Our proposal can be easily instantiated with a higher security level such as Ed448, defined by the same standards for a 224-bit security level.

EdDSA routines. We describe *sign* and *verify* routines of EdDSA [15, 137]. Algorithm 1 shows the *sign* routine, which takes the secret s_U as input, that is a random bit string of size b-bit, and outputs signature σ. A peculiar design choice of EdDSA is to use a random bit string s_U as input secret key instead of a valid scalar that can be used for elliptic curve operations. The *sign* routine derives cryptographic material from s_U internally by using a hash function (Line 2) and an internal *key derivation* routine (Line 3) before computing scalar multiplication (Line 6). The discussion of the details of *key derivation* is below. This approach introduces advantages in many scenarios, but it prevents integration between EdDSA and ECQV certificates, because in ECQV the client must compute the secret key by using contributions from the CA. As a result, the *sign* routine of EdDSA must be redesigned (as detailed in Section 5.2.3). Finally, we observe that signatures generated with the proposed variant are compliant with the verification routine of the original EdDSA scheme and thus it is not cessary to discuss the details of verification. For clarity, Algorithm 2 reports it with the notations used

Algorithm 1 EdDSA message signing algorithm

1: **function** SIGN(m, s_U)
2: $h = \mathcal{H}_{2b}(s_U)$
3: $sk_U, PK_U = \text{KEYDERIVATION}(h[:b_8])$
4: $rs = h[b_8:]$
5: $n = \mathcal{H}_\ell(rs \parallel m)$
6: $R = n \cdot B$
7: $s = [n + sk_U \cdot \mathcal{H}_\ell(\underline{R} \parallel \underline{PK_U} \parallel m)] \bmod \ell$
8: $\sigma := (R, s)$
9: **return** σ

Algorithm 2 EdDSA signature verification algorithm

1: **function** VERIFY(m, σ, PK_U)
2: $R, s \Leftarrow \sigma$
3: **if** $\neg(1 < s < \ell)$ **then**
4: **return** *InvalidSignature*
5: $h = \mathcal{H}_\ell(\underline{R} \parallel \underline{PK_U} \parallel m)$
6: $S = 2^c \cdot s \cdot B$
7: $S' = 2^c \cdot h \cdot PK_U + 2^c \cdot R$
8: **if** $S = S'$ **then**
9: **return** *ValidSignature*
10: **else**
11: **return** *InvalidSignature*

in this thesis. As multiple variants of verification have been observed in real world usage [21, 35], the considered routine is compliant with RFC 8032 [137] and with very popular implementations, including that provided by the original paper [15] and by the Libsodium library [99].

EdDSA key derivation and clamping. The *key derivation* routine (Algorithm 3) generates a key pair (sk, PK) from a random bit string k of size b-bit. The routine modifies k by using the *clamp* routine (Line 2) to compute a secure secret scalar sk. The secret scalar sk is used to compute the public key PK through a scalar point multiplication with the generator B. The output is the key pair (sk, PK). The *clamp* routine avoids flawed secret scalars that could affect the security of the scheme using only bit manipulation operations, comprising:

① clearing the most significant $(b-t-1)$ bits to zero, to let all secret scalars having the same bit length and to improve performance [19];

② setting the bit at position 2^t to one, to increase resistance against timing side-channel attacks in presence of non time constant implementations [46];

③ clearing the c less significant bits to zero, to allow using the same EdDSA secret scalars with other protocols that could be vulnerable to small-subgroup attacks (typically, the X25519 secure key exchange protocol for which the clamping procedure was first proposed [12]).

As a result, EdDSA secret scalars are the elements of the field $\mathbb{Z}_{2^c\ell}$ that satisfy equation $2^t + \sum_{i=c}^{i=(t-1)} 2^i h_i$, where h_i denotes the i-th bit of a bit string $h \in \{0,1\}^b$. Although *clamping* has many practical benefits, it also introduces challenges for integrating EdDSA with deterministic key derivation schemes [91] and for formally proving EdDSA security [21]. This chapter shows that *clamping* also introduces challenges to integrate ECQV schemes with EdDSA. Section 5.2.5 discusses how our scheme complies with clamping.

Algorithm 3 EdDSA key derivation algorithm

1: **function** KeyDerivation(k)
2: $sk = \text{Clamp}(k)$
3: $PK = sk \cdot B$
4: **return** (sk, PK)

Algorithm 4 Certification Authority Key Generation

1: **function** CAKeygen( )
2: $seed \xleftarrow{\$} \{0,1\}^b$
3: $sk_{CA}, PK_{CA} = \text{KeyDerivation}(seed)$
4: **return** (sk_{CA}, PK_{CA})

5.2.3 Proposed scheme

This thesis describe the proposed scheme for combining ECQV and EdDSA by complying with the operations framework described in Section 5.2.2, including *setup* and *certificate sign request* (Section 5.2.3), *certificate generation* (Section 5.2.3), *public key extraction* (Section 5.2.3), *certificate reception* (Section 5.2.3) and *message signing* (Section 5.2.3). In this chapter we do not describe *verify* because it is unmodified with regard to the original EdDSA routine (see Section 5.2.2, Algorithm 2). This section also omits details regarding point compression and decompression for transferring elliptic curve coordinates and we assume that received points belong to the Edwards curve E (see Section 5.2.8). However, we do not assume that received points also belong to $\langle B \rangle$. These assumptions are the same of the original EdDSA design (see Section 5.2.2).

Setup and Certificate Sign Request Generation

The setup of the scheme requires generating the key pair of the Certification Authority (sk_{CA}, PK_{CA}) by generating a scalar and computing the corresponding public key through the EdDSA *key generation* routine (see Algorithm 4). We assume that the public key of the CA is known and trusted by all parties. We assume that each client certificate can be associated with unique identity information and possibly additional metadata (e.g., validity period), denoted as *id*, for which the certification authority acts as root of trust.

The *certificate sign request generation* routine (Algorithm 5) is used by a client to generate a *certificate sign request* (*CSR*) for requesting a valid certificate from the CA. The client must first generate a random bit string of size b-bit, that we denote as *seed*. Then, it expands the *seed* through $\mathcal{H}_{2b}(\cdot)$ and computes a secret scalar *sk* from the first b_8 bytes of the expanded seed (Line 3). We introduce a novel clamping function denoted as $ClampMSB$ that manipulates the most significant bits of the secret scalar as devised by the operations ① and ② of the EdDSA clamping routine (see Section 5.2.2). Then, the public key is computed by multiplying the clamped secret scalar by the base point B. We remind that this key pair represents intermediate information that will be combined with additional cryptographic material returned by the certification authority (see Section 5.2.2). The second half of the expanded seed will be used by the *certificate reception* routine to generate the actual user key pair. The *CSR* is built as the tuple (PK, id). The routine outputs the *CSR*, that the client

Algorithm 5 Certificate sign request generation

1: **function** CSRGen(id)
2: $seed \xleftarrow{\$} \{0,1\}^b$
3: $sk = \mathcal{H}_{2b}(seed)[:b_8]$
4: ClampMSB(sk)
5: $PK = sk \cdot B$
6: $CSR := (PK, id)$
7: **return** $(seed, CSR)$

Algorithm 6 Certificate generation

1: **function** CertificateGen(sk_{CA}, PK_{CA}, CSR)
2: $id, PK \Leftarrow CSR$
3: **do**
4: $seed' \xleftarrow{\$} \{0,1\}^b$
5: $sk', PK' =$ KeyDerivation($seed'$)
6: $P = 2^c \cdot PK + PK'$
7: $CRT := (P, id)$
8: $x = \mathcal{H}_\ell(\underline{P} \parallel id)$
9: **while** $(x \cdot P + PK_{CA}) = \mathcal{O}$
10: $r = (x \cdot sk' + sk_{CA}) \bmod \ell$
11: **return** (CRT, r)

sends to the CA for the *certificate generation* routine, and the secret data *seed*, that the client keeps for later use in the *certificate reception* routine.

We observe that our routine modifies that of the original ECQV scheme [31] by incorporating key clamping operations that are usually adopted during sign operations in the EdDSA scheme. This design choice allows to use a randomly generated *seed* without the risk of incurring in timing side-channel attacks (see Section 5.2.2). Moreover, clamping only the most significant bits during the *CSRGen* routine allows to generate keys compatible with the original EdDSA secret scalars and to design a more robust procedure (see Section 5.2.5 for a detailed discussion).

Certificate generation

The *certificate generation* routine is used by the CA to generate a certificate for a client. The routine takes as inputs the key pair of the CA (sk_{CA}, PK_{CA}) and the *CSR* of a client, and produces as outputs the certificate *CRT* and the private contribution data r. The rationale is the same as the standard ECQV algorithm: to produce a key with homomorphic properties that bind the client certificate to the CA public key. At the same time, the routine prevents the CA to know the secret key of the client. However, our routine modifies that of the original ECQV scheme to handle points of the Edwards curve that do not belong to $\langle B \rangle$.

We describe the routine by referring to Algorithm 6. The CA unpacks *id* and *PK* from the *CSR* received from the client. To produce a valid certificate, the CA must generate a random key pair that is valid for the input *CSR* through an iterative approach. First, it generates a tentative key pair (sk', PK') (Lines 4 and 5). Then, it computes a tentative implicit certificate for the client that we denote as P (Line 6). The algorithm deviates from the original ECQV procedure because it multiples the point *PK* extracted from the *CSR* by the cofactor 2^c. This implies that even if $PK \notin \langle B \rangle$, the resulting point $(2^c \cdot PK) \in \langle B \rangle$.

Algorithm 7 Public key extraction

1: **function** EXTRACT(PK_{CA}, CRT)
2: $P, id \Leftarrow CRT$
3: $x = \mathcal{H}_\ell(\underline{P} \parallel id)$
4: $PK_U = x \cdot P + PK_{CA}$
5: **if** $PK_U = \mathcal{O}$ **then**
6: **return** *InvalidCertificate*
7: **return** PK_U

The implicit certificate P is valid if, when combined with id, it generates a scalar x through hash function $\mathcal{H}_\ell(\cdot)$ such that $(x \cdot P + PK_{CA}) \neq \mathcal{O}$ (Line 9). As we show in the security proof (Appendix 5.2.7), generating a temporary key pair that satisfies $(x \cdot P + PK_{CA}) = \mathcal{O}$ would cause a severe vulnerability. Although the probability of generating such a key is negligible, we maintain the original approach of ECQV and completely rule out such possibility by using this iterative approach. As the result of the *certificate generation* routine, the CA returns the certificate CRT and the private contribution data r, that will be used to compute the actual key pair by the client with the *certificate reception* routine, and to compute the public key of the client by other participants with the *public key extraction* routine.

Public key extraction

Public key extraction (Algorithm 7) takes as input the public key of the CA PK_{CA} (that we assume as known by all parties) and the sender's certificate CRT to compute the sender's public key PK_U. As for *certificate generation*, from a high-level perspective this routine does not modify that of the original ECQV scheme we observe that the operation does not prevent $P \notin \langle B \rangle$ and thus PK_U could also be a point that does not belong to the prime order subgroup $PK_U \notin \langle B \rangle$. However, this is not an issue because the original EdDSA *verify* routine already assumes that public keys could not belong to $\langle B \rangle$ (see Section 5.2.2).

Certificate reception

Certificate reception (Algorithm 8) takes as inputs the CA public key PK_{CA}, the *seed* generated with the *CSRGen* routine, the certificate CRT and the private contribution data r returned by the *CRTGen* routine of the CA. It outputs the key pair (s_U, PK_U) that can be used by the client to sign messages.

The routine requires the client to compute the digest h of the *seed* and to extract two b-bit strings: the first one ($h[:b_8]$) allows to compute the scalar sk (Lines 5-6) as described for *CSRGen*; the second one ($h[b_8:]$) allows to compute the secret string rs (Line 7). Then, the client computes a temporary secret scalar sk'_U by using the scalar sk and the private contribution data r received from the CA (Line 8). This computation differs from the standard ECQV procedure because it multiplies scalar sk by the cofactor 2^c to comply with the analogous operation computed by the CA on the corresponding public key PK during the *CRTGen* routine (Algorithm 6, Line 6). The client picks a value δ, which is a positive integer that is smaller than the cofactor 2^c, through function *ClampCoeff* (Line 9). The secret scalar sk_U is computed by adding value $(\delta \cdot \ell)$ to sk'_U (Line 10) such that $2^{t+1} > (sk'_U + \delta \cdot \ell) \geq 2^t$. This computation

Algorithm 8 Certificate reception

1: **function** CRTRec($PK_{CA}, seed, CRT, r$)
2: $P, id \Leftarrow CRT$
3: $x = \mathcal{H}_\ell\,(\underline{P} \parallel id)$
4: $h = \mathcal{H}_{2b}\,(seed)$
5: $sk = h[:b_8]$
6: ClampMSB(sk)
7: $rs = h[b_8:]$
8: $sk'_U = (r + 2^c \cdot x \cdot sk) mod\, \ell$
9: δ = ClampCoeff(sk'_U)
10: $sk_U = sk'_U + \delta \cdot \ell$
11: $PK_U = sk_U \cdot B$
12: PK'_U = Extract(PK_{CA}, CRT)
13: **if** $PK_U \neq PK'_U$ **then**
14: **return** *InvalidCertificate*
15: $s_U \Leftarrow (sk_U, rs)$
16: **return** (s_U, PK_U)

Algorithm 9 Message signing

1: **function** Sign(m, s_U, PK_U)
2: $sk_U, rs \Leftarrow s_U$
3: $n = \mathcal{H}_\ell\,(rs \parallel m)$
4: $R = n \cdot B$
5: $s = \left[n + sk_U \cdot \mathcal{H}_\ell\left(\underline{R} \parallel \underline{PK_U} \parallel m\right)\right] mod\, \ell$
6: $\sigma := (R, s)$
7: **return** σ

allows to comply with constraints ① and ② of key clamping (see Section 5.2.2) without affecting correctness of signing operations ($sk'_U \equiv sk_U mod\, \ell$). The implementation of *ClampCoeff* depends on the curve used to instantiate the scheme (edwards25519 or edwards448). For clarity we postpone details to Section 5.2.5. The client verifies that sk_U is valid with regard to the implicit certificate released by the CA by verifying that the corresponding public key $PK_U = sk_U \cdot B$ (Line 11) is equal to the public key PK'_U computed through the *public key extraction* routine by using as inputs the public key of the CA PK_{CA} and the certificate CRT (Line 13). We observe that this comparison also validates that the value of P included in CRT belongs to $\langle B \rangle$, because PK_U is computed through a point scalar multiplication on the base point B. Finally, the private key s_U is built as the tuple (sk_U, rs), and the *extract* routine terminates by returning the key pair (s_U, PK_U).

Message signing

The *sign* routine (Algorithm 9) requires as inputs the message m and the key pair (s_U, PK_U). For ease of presentation, we also show the public key PK_U, that can be re-computed when needed by using s_U.

The function differs from the original EdDSA scheme because the private key s_U is a $2b$-bit string, instead of the original b-bit string (denoted as rs in the original scheme, see Algorithm 1). This approach is similar to a proposal for EdDSA key derivation [91], which denotes the $2b$-bit secret key as the *extended secret key*. The additional data included in the private key avoids the computation of the scalar sk_U through key derivation operations that would break the mathematical properties of the

ECQV scheme. Instead, the proposed routine obtains the scalar sk_U and the secret string rs by extracting them from s_U (Line 2). The secret string rs is used to derive the nonce n (Line 3).

By looking at the whole operations flow including both ECQV and the proposed *sign* routine, we observe that the scheme maintains the same approach of the original EdDSA scheme, because the derivation operation used to compute string rs and scalar sk_U has been moved from the *sign* routine to the *CSRGen* routine (Section 5.2.3 and Algorithm 5). We highlight that the random string rs must not be generated for each signature because it is a persistent value computed during the *certificate sign request generation* routine (Algorithm 5). As a result, the scheme maintains the same security guarantees of the original EdDSA scheme, but requires a larger secret key ($2b$-bit instead of b-bit).

A potential variant of the scheme could use b-bit strings as secret keys by adopting designs that are similar to those adopted for deterministic nonce derivation in *deterministic ECDSA* [134] and for hierarchical deterministic EdDSA key derivation [34], which derive the nonce from the same random data used as secret scalar within the scheme. We estimate that such a variant would achieve a different trade-off in terms of security assumptions and performance, however we omit details for space constraints.

5.2.4 Generic Group Model with passive adversaries

We describe the generic group model with passive adversaries, considered by [25] to prove security of the combination of ECQV and ECDSA, by using our notations.

An adversarial algorithm working in the generic group model has access to a (generic group) oracle. At any time, the oracle maintains a state which is a list of pairs $((A_1, z_1), \ldots, (A_i, z_i))$. The value of the index i is equal to the number of queries made to the oracle so far ($i = 0$ corresponds to an empty list). We denote as z_i an element of the cyclic group $\mathbb{Z}_\ell$, and as A_i the bit string representation of the corresponding element of the cyclic group $\langle B \rangle$, such that $A_i = z_i \cdot B$. A query to the generic group oracle can require execution of two types of routines: *Subtract* and *Push* (Algorithm 10). The *subtract* routine requires no arguments and can only be invoked when $i \geq 2$. The *push* routine can be invoked at any index value and requires A_{i+1} as argument.

The model defines that an element can be either *independent* or *dependent*. An element A_i is *independent* if it is the output of the *push* routine and if $A_i \neq A_j, \forall j < i$, while it is *dependent* if it is the output of the *subtract* routine or a *push* of a previous element.

For any element A_i, we define an integer sequence $c(A_i)$ as follows:

- if A_i is independent and is the k^{th} independent element ranked by index i, then $c(A_i)$ is the sequence that is all 0 except for a 1 in position k;
- if A_i is dependent and is obtained from the *Push* routine with $A_i = A_j$ for $j < i$, then $c(A_i) = c(A_j)$;
- if A_i is dependent and is obtained from the *Subtract* routine, then $c(A_i) = c(A_{i-1}) - c(A_{i-2})$.

Algorithm 10 Generic group model subtract and push routines

```
1: function SUBTRACT( )
2:     if i ≤ 2 then
3:         return
4:     i = i + 1
5:     z_{i+1} = z_i − z_{i−1} mod ℓ
6:     if z_{i+1} = z_j for some 1 ≤ j ≤ i then
7:         A_{i+1} = A_j
8:     else
9:         A_{i+1} ←$ {A_ℤ ∉ [A_1, ..., A_n]}
10: function PUSH(A_{i+1})
11:     if A_{i+1} = A_j for some 1 ≤ j ≤ i then
12:         z_{i+1} = z_j
13:     else
14:         z_{i+1} ←$ {ℤ_ℓ ∉ [z_1, ..., z_i]}
```

Finally, a *coincidence* occurs at index i, if for some $j < i$, we have that $c(A_i) \neq c(A_j)$ and $A_i = A_j$. The probability that a coincidence occurs is at most i^{-2}, so if the adversary working in the generic group model is efficient, then it has negligible chance of finding a collision [24].

5.2.5 Compliance with EdDSA secret scalars

We analyze how secret scalar sk_U computed by the proposed scheme relates to the three key clamping operations of EdDSA (see Section 5.2.2): ① clearing the $(b-t-1)$ most significant bits, ② setting bit at position 2^t to one, ③ clearing the c less significant bits.

First, we observe that the key generation procedure of the original ECQV scheme computes a random secret scalar that belongs to $\mathbb{Z}_\ell$, which complies with operation ① of clamping, but does not comply with operations ② and ③. Our approach allows to comply with ① and ②, but does not allow to comply with ③. we describe the design of our approach and discuss the implications. We modify the original scheme to also comply with ②, but our proposal is not able to also satisfy ③. In the following, we first discuss our modification, then we discuss the reason of this design choice and its impact on security and on real-world usage.

As anticipated in the previous Section 5.2.3 (Algorithm 8), our proposal first computes a temporary secret scalar $sk'_U \in \mathbb{Z}_\ell$ (Line 8), then it computes the secret scalar that is returned to the client as $sk_U = sk'_U + \delta \cdot \ell$ (Line 10), where δ is a small positive integer smaller than 2^c chosen by function *ClampCoeff* (Line 9), which does not affect correctness or security of cryptographic operations because $sk'_U \equiv sk_U \, mod \, \ell$. The rationale of this approach is to chose a value of δ which sets bit at position 2^t to one without setting any bits within the $(b-t-1)$ most significant bits (that is, $2^{t+1} > (sk'_U + \delta \cdot \ell) \geq 2^t$, where $sk'_U \in \mathbb{Z}_\ell$). To this aim, the implementation of *ClampCoeff* must consider the parameters of curve used to instantiate the scheme. we propose candidate implementations to pick δ for edwards25519 and for edwards448. For edwards25519, δ can be a constant value arbitrarily chosen within $\{4, 5, 6\}$, regardless of the value of $sk'_U \in \mathbb{Z}_\ell$. Instead, for edwards448 the value of δ must be chosen

according to the value of sk'_U: $\delta = 2$ if $sk'_U < 2^{445}$, otherwise $\delta = 1$. As this operation represents a decision based on secret information, it must be implemented by using a constant-time algebraic operation to avoid potential timing side-channel attacks.

As already discussed in Section 5.2.2, ① and ② are important for EdDSA to make it more resilient to timing side-channel attacks. Moreover, our practical experience with existing implementations of EdDSA also showed that satisfying ① and ② is mandatory to preserve compatibility with cryptographic subroutines (in particular, with scalar point multiplication). Thus, the proposed approach is important for minimizing implementation efforts, even in presence of secure time-constant implementations which would be secure even without complying with clamping. Instead, probably due to the intrinsic design of the algorithms, we did not find any compliance issue even without complying to ③. We remind that such non-compliance does not introduce any security issue for EdDSA *per se*. However, we remark that the secret scalars computed through our scheme *must not be used* for other schemes that are vulnerable to small-subgroup attacks, such as X25519. The same keys could be used by modifying these schemes to also validate that elliptic curve points received from untrusted parties belong to $\langle B \rangle$ (at the expense of computational costs penalties). Otherwise, we also observe that our proposal could fit well recent EdDSA variants that do not adopt clamping, such as schemes based on so-called *Decaf* [75] and *Ristretto* [76] encoding techniques. Going into further details is out of the scope of this thesis and we leave it as future work.

5.2.6 Security analysis of the EdDSA and ECQV variants

We discuss how the proposed modifications may impact security guarantees with regard to the original EdDSA and ECQV schemes by considering security threats analyzed by the original discussions [15, 137] and additional threats emerged afterwards [27, 136], which include: *non-malleability* and *unforgeability*, resistance to *weak keys*, generation of *secure nonces*, resistance to *side-channel* attacks.

Unforgeability and non-malleability. The original EdDSA scheme guarantees both unforgeability and malleability by relying on the hardness of the ECDL problem over $\langle B \rangle$ and on one-way hash functions. The proposed scheme modifies EdDSA by changing how the secret scalar and the nonce are computed. However, the other operations that guarantee unforgeability and malleability have not been modified. Thus, the proposed scheme guarantees unforgeability and malleability as long as it is able to correctly compute the secret scalar and the nonce. Below we discuss whether the proposed scheme could compute flawed secret scalars or nonces that give advantages to adversaries. In Appendix 5.2.7 we also show that the structure of the secret scalar, that is a combination of the implicit certificate (P, id) and of the CA public key PK_{CA}, does not give advantages to the attacker.

Resistance to weak keys. The capability of generating strong keys is critical for any cryptographic scheme. EdDSA uses a secret random string s_U as secret key, and uses hash function $\mathcal{H}_{2b}(\cdot)$ and *key derivation* to compute secret scalar sk and secret string rs used to compute the nonce (see Section 5.2.2, Algorithms 3 and 1). To ensure a strong secret scalar, EdDSA also applies the clamping procedure during signing operations. The security of the keys computed by the proposed scheme must be analyzed

by considering the whole operations workflow required by implicit certificate architectures. When using ECQV, the only routine where a client has to generate a secret key on its own is during *CSR generation*, as in the key generation procedure of the original EdDSA scheme. Since all the other operations do not require entropy sources, our proposal complies with EdDSA derivation strategies. The proposed signature routine cannot derive the secret scalar from a random string because it is computed by the *certificate reception* operation (Algorithm 9). Thus, it cannot be modified without affecting the properties that bind it to the cryptographic material of the CA. On the other hand, the contribution given by the CA in the *certificate generation* routine can increase the security of the secret key used in signature operations. also observe that intermediate secret scalars used throughout the proposed scheme comply with clamping procedures of EdDSA. However, as we already discussed in depth in Section 5.2.5, the secret scalars generated by our procedure cannot be used with other schemes that could be affected by small-subgroup attacks.

Pseudo-random generation of the nonce. The security of the scheme is compromised if the procedure computes biased nonces or if the attacker is able to compute the nonce associated to a signature (or a portion of it). EdDSA uses a deterministic procedure to compute secure pseudo-random nonces by using the hash function (Algorithm 1). As a result, the nonce computed for different messages is always unique and cannot be computed without the secret key. The scheme proposed in this thesis maintains the deterministic approach of EdDSA as it still uses a hash function to compute nonce. However, the secret key includes two values: the seed of the PRF: rs, and the secret scalar sk_U (Algorithm 9). The security of the scheme is the same of the original EdDSA scheme because the two values allow to handle the pseudo-random generation of the nonce and $ECDL$ operations independently. The disadvantage is that the size of the secret key is doubled. A similar design choice has been proposed for handling deterministic hierarchical key derivation of EdDSA keys [91]. Our proposal could also be easily instantiated by using a secret key of the same size of EdDSA by deriving the nonce from the secret scalar through a PRF, similarly to proposals for deterministic ECDSA variants [134] and to variants of the above-mentioned key derivation scheme [34]. For space constraints, we omit details of a such a variant.

Side-channel attacks can be operated by adversaries that have privileged access to the devices (also called *gray-box* access [141]) that operates computations on confidential data, such as secret keys. For signature schemes, the critical operation is the computation of a signature. Side-channel attacks include *profiling attacks* [71], that try to infer information from the *power consumption* [94] or the *timing responses* [27], and *fault attacks* [136], which are able to disrupt the normal behavior of a device. For profiling attacks, it is important that operations are implemented by using time-constant algorithms. Our design natively supports time-constant point addition operations, as the original EdDSA. Moreover, it supports key clamping features that make schemes resilient against timing side-channel attacks even in presence of possibly flawed non time-constant implementations [46] (although time-constant implementation are still recommended, as we discuss in Section 5.2.8). In the context of fault attacks, the original EdDSA scheme has known vulnerabilities [136], that can be mitigated by reintroducing random data within computation of the nonce [107] or through hardware countermeasures [87]. Although we follow the original design, our proposal can also easily comply with [107]. As a result, the proposed scheme is as resistant against this

class of attacks as EdDSA.

5.2.7 Security analysis of the combination of EdDSA and ECQV

We propose a novel security analysis showing that the proposed combination of ECQV and EdDSA is secure while considering the same security assumptions adopted by the original security proof for the composition of ECQV and ECDSA [25]. First, we propose a preliminary analysis based on corner cases and known attacks to implicit certificate schemes. This analysis motivates a few design choices of our proposal (Section 5.2.7). Second, we prove security against a passive adversary that tries to forge valid signatures without interacting with legitimate users (Section 5.2.7). This analysis assumes existence of random oracles and an ideal view of the cryptographic group as devised by the generic group model [24] (see Appendix 5.2.4).

Systematic analysis of corner case conditions

We identify corner case conditions that could allow solving the verification routine of EdDSA (see Section 5.2.2, Algorithm 2), and analyze whether these conditions could be exploited by an adversary. We represent the verification procedure through the following equation:

$$s \cdot B = \mathcal{H}_\ell \left(\underline{R} \parallel \underline{PK_U} \parallel m \right) \cdot PK_U + R \tag{5.1}$$

For simplicity we omit the multiplication of both members by 2^c because we analyze security by using a generic group model.

By combining EdDSA and ECQV with our proposal, the sender public key PK_U is computed from the implicit certificate P and from the CA public key PK_{CA}. The point R is computed as $\mathcal{H}_\ell (rs \parallel m) \cdot B$, so Equation (5.1) is expanded as follows:

$$s \cdot B = \mathcal{H}_\ell \left(\underline{R} \parallel \underline{PK_U} \parallel m \right) \cdot [\mathcal{H}_\ell (\underline{P} \parallel id) \cdot P + PK_{CA}] + \mathcal{H}_\ell (rs \parallel m) \cdot B \tag{5.2}$$

We adopt a black-box notation for hash functions by introducing values x, y and z:

$$x = \mathcal{H}_\ell (\underline{P} \parallel id) \tag{5.3}$$

$$z = \mathcal{H}_\ell (rs \parallel m) \tag{5.4}$$

$$y = \mathcal{H}_\ell \left(\underline{R} \parallel \underline{PK_U} \parallel m \right) = \mathcal{H}_\ell \left(\underline{(z \cdot B)} \parallel \underline{(x \cdot P + PK_{CA})} \parallel m \right) \tag{5.5}$$

By substituting x, y and z within Equation (5.2) and by separating known variables PK_{CA} and B from target variables P and s we obtain:

$$s \cdot B - x \cdot y \cdot P = y \cdot PK_{CA} + z \cdot B \tag{5.6}$$

To solve Equation (5.6) we analyze three corner case conditions:

$$\begin{cases} x \cdot P = -PK_{CA}, & s = z, y \neq 0 \\ x \cdot y \cdot P = -y \cdot PK_{CA} + (s - x) \cdot B, & z = x, y \neq 0 \\ s \cdot B = y \cdot PK_{CA} + z \cdot B, & x = 0, y \neq 0 \end{cases} \tag{5.7}$$

The first condition $s = z, y \neq 0$, which can be easily satisfied by legitimate owners of secret keys, would allow a user that is able to obtain P such that $x \cdot P = -PK_{CA}$ to generate valid signatures for any message. Since $x = \mathcal{H}_\ell(\underline{P} \parallel id)$, the probability of finding P such that $x \cdot P = -PK_{CA}$ is negligible. Moreover, the possibility of generating a valid implicit certificate that satisfies this equation by a client is completely ruled out because such a value of P is rejected during the *extract* routine (see Algorithm 7, Line 6). Moreover, no legitimate users obtain such a value of P because the CA prevents it during the *certificate generation* routine (see Algorithm 6, Line 9). We observe that this verification routine is also implemented by the standard combination of ECQV and ECDSA schemes, and does not represent an additional security condition of our proposal.

The second condition, where $z = x$, can be easily satisfied by any adversary by computing $z = \mathcal{H}_\ell(\underline{P} \parallel id)$ instead of computing $z = \mathcal{H}_\ell(rs \parallel m)$. We observe that the resulting equation is worth further analysis. In particular, if the adversary selects $s = x$ the equation is further simplified to $x \cdot y \cdot P = -y \cdot PK_{CA}$. By dividing both members by y we obtain $x \cdot P = -PK_{CA}$. As a result, the corner case condition falls back to the first condition that we already analyzed, and thus can be completely ruled out by using the same verification operations.

We remark that this corner case condition is somehow similar to a peculiar attack to the combination of ECQV and ECDSA schemes [25], where the adversary selects a message $m = (\underline{P} \parallel id)$ to produce a collision between the inputs of different hash functions during verification. As a result, when using ECQV and ECDSA the message $m = (\underline{P} \parallel id)$ is not allowed and must be rejected by verifiers. In the proposed scheme, no non-empty values are able to produce duplicate inputs within x, y and z. The inputs of $y = \mathcal{H}_\ell(rs \parallel m)$ and $z = \mathcal{H}_\ell\left(\underline{R} \parallel \underline{PK_U} \parallel m\right)$ are designed as in the EdDSA scheme for which no such values are known to exist. Moreover, we observe that the input of z includes $R = y \cdot B$ (Equation (5.4)), and thus producing the same inputs for y and z is unfeasible for any values. The value of y is also computed by using x as input (Equation (5.5)), thus implying that the probability of finding equal inputs for x and y is negligible.

The third condition, where we consider $x = 0, y \neq 0$, allows an adversary to obtain a simplified equation that however does not give any advantage. The resulting equation that must be solved is $s = y \cdot sk_{CA} + z$, which requires knowledge of the secret key of the certification authority. This condition could provide advantages if multiple values of (P, id) were found that satisfy $x = \mathcal{H}_\ell(\underline{P} \parallel id) = 0$, because the same values (R, s) would be valid signatures for multiple identities. However, this would represent a collision within the adopted hash function. Thus, the third corner case condition does not require additional verification operations.

Security analysis against passive adversaries in the generic group model

We now proceed by analyzing values of P that could solve Equation (5.2) by considering a passive adversary in the combined random oracle and generic group models, as considered by security proofs on the combination of ECQV and ECDSA [24]. For an overview of the generic group model refer to Appendix 5.2.4.

We model points B and PK_{CA}, which are public values of the scheme that we assume are correctly distributed to verifiers, as independent points. Without loss of generality, we take PK_{CA} and B to be the first and the second independent points respectively, that is, $c(PK_{CA}) = (1, 0, 0, \ldots)$ and $c(B) = (0, 1, 0, \ldots)$. We consider that points P and R are dependent points, that is, points that the adversary can select by invoking the generic group oracle by using any independent point or other dependent points. Thus, we define P and R as linear combinations of the points PK_{CA} and B as $c(P) = (d, e, \ldots)$ and $c(R) = (f, g, \ldots)$. We remark the dependencies of x and y with regard to dependent and independent points:

$$x = \mathcal{H}_\ell \left(\underline{c(P)} \parallel id \right) = X(d, e, id), \tag{5.8}$$

$$y = \mathcal{H}_\ell \left(\underline{c(R)} \parallel \underline{[x \cdot c(P) + c(PK_{CA})]} \parallel m \right) = Y(f, g, x, d, e, m), \tag{5.9}$$

where functions $X(\cdot)$ and $Y(\cdot)$ model random oracles and are used to remark the input variables on which x and y depend.

By combining EdDSA and ECQV with our proposal, the sender public key PK_U is computed from the implicit certificate P and from the CA public key PK_{CA}. With regard to Equation (5.2), where we assumed any type of adversary (even legitimate signers), we do not substitute point R with $\mathcal{H}_\ell (rs \parallel m) \cdot B$ because we assume that the adversary could choose any value of R. Thus, Equation (5.1) is expanded as follows:

$$s \cdot B = \mathcal{H}_\ell \left(\underline{R} \parallel \underline{PK_U} \parallel m \right) \cdot [\mathcal{H}_\ell (\underline{P} \parallel id) \cdot P + PK_{CA}] + R \tag{5.10}$$

We express Equation (5.10) through linear combinations of PK_{CA}, B, P and R:

$$s \cdot c(B) - x \cdot y \cdot c(P) = y \cdot c(PK_{CA}) + c(R) \tag{5.11}$$

We collect on the left the members that multiply independent points $c(B)$ and $c(PK_{CA})$, and on the right the members that multiply dependent points $c(P)$ and $c(R)$, and express the equation that represents the linear combinations through vectors as follows:

$$(-y, s) = x \cdot y \cdot (d, e) + (f, g) \tag{5.12}$$

From Equation (5.12) we obtain a system of two equations:

$$\begin{cases} x \cdot y \cdot d + f + y = 0 \\ x \cdot y \cdot e + g - s = 0 \end{cases} \tag{5.13}$$

We solve the first equation of (5.13) by identifying two cases:

$$\begin{cases} f = 0, & d \cdot x + 1 = 0 \\ y = -f \cdot (d \cdot x + 1)^{-1}, & d \cdot x + 1 \neq 0 \end{cases} \tag{5.14}$$

- the first case $d \cdot x + 1 = 0$ allows a valid solution in $f = 0$, but requires finding d such that $x = -d^{-1}$. However, the value of x is already bound to d by the non-linear relation $X(d, e, id)$ (Equation (5.8)). In the random oracle model, any modifications to d would cause a random and unpredictable modification to x. Thus, the probability for an adversary of finding d that satisfies both the equations $d \cdot x + 1 = 0$ and $x = X(d, e, id)$ is negligible. Note that we already observed this condition in the first and in the second corner cases of the previous Section 5.2.7 (Equation 5.7);

	Cost	Comp.	Dec.
Compression	$ExpZq$		
Decompression	$ExpZq$		
CSRGen	$SMulB^*$	✓	
CRTGen	$SMulB^* + SMul$	✓	✓
CRTRec	$SMulB^* + SMul$		✓
Sign	$SMulB^*$	✓	
Extract	$SMul$		✓
Verify	$SMulB + SMul$		✓

Table 5.16: Asymptotic costs of the algorithms

- the second case $d \cdot x + 1 \neq 0$ binds y to the values of f, d and x. However, we observe that the value of y is already bound to all these variables by the non-linear relation $Y(f, g, x, d, e, m)$ (Equation (5.9)), and does not have any additional unbound variables. In the random oracle model, any modifications to f, d or x would cause a random and unpredictable modification to y. Thus, the probability for an adversary of finding f, d, x that satisfy both the equations $x \cdot y \cdot d + f + y = 0$ and $y = Y(f, g, x, d, e, m)$ is negligible.

Although a solution for the second equation of (5.13) exists, which is $x \cdot y \cdot e + g - s = 0$, the unfeasibility of solving the first equation is sufficient to prevent attacks from passive adversaries in the combined generic group and random oracle models.

5.2.8 Computational costs

To represent elliptic curve points there are various alternatives, and depending on the chosen coordinate system we can obtain different space-time trade-offs. Some representations allow to implement faster mathematical operations, others allow to reduce memory consumption [77]. As implicit certificates are meant to minimize network usage, compressed coordinates to transfer elliptic curve points seem the best choice. Senders *compress* points before transferring them, and receivers *decompress* points before their use. Decompression includes validating that coordinates are expressed through canonical values, and that the point lies on the Edwards curve E and that it is not within a small number of weak values [12, 21]. However, decompression does not include validating membership of the point to the subgroup $\langle B \rangle$. The costs of compression and decompression depend on the coordinate representation system. We assume using typical implementations of EdDSA schemes that use projective and extended projective coordinates.

We analyze the asymptotic computational costs of all the routines of the scheme, comprising *certificate signing request generation (CSRGen), certificate generation (CRTGen), public key extraction (Extract), certificate reception (CRTRec), sign generation (Sign)* and *sign verification (Verify)*. We describe the results of the analysis by referring to Table 5.16. The first and second columns show the operation and its associated asymptotic cost. The third and fourth columns show where the compression and decompression operations are used. In the analysis, we consider that the public key of the CA is cached by clients that must operate *Verify*, *CRTRec*, and *Extract*. For this reason, we do not include the cost of decompression of the CA public key.

We consider costs of operations implemented as described by the original paper [15] and the IETF standard [137]. Operations that have a significant impact on performance are inversion, exponentiation and square root in $\mathbb{Z}_q$. However, inversion and square root in $\mathbb{Z}_q$ can be implemented by using a faster exponentiation in $\mathbb{Z}_q$ [15]. We distinguish scalar point multiplication with the fixed base point B (*SMulB*), that could be optimized with pre-computation tables [124], and with variable base points (*SMul*). We denote operations that must be implemented by using side-channel resistant algorithms by adding the symbol * to the operation. We observe that all scalar point multiplications with base point B require an implementation that is side-channel resistant (*SMulB**), except for the one executed in *Verify* (*SMulB*). Instead, no scalar point multiplications with variable base points require side-channel resistance because they always operate on public data (*Extract* and *Verify*). Other operations, such as sum, subtraction and product in $\mathbb{Z}_q$, are omitted because they have little or negligible costs. Finally, *point equivalence verification* is also negligible because it is implemented without converting points to affine coordinates, thus requiring four products in $\mathbb{Z}_q$.

All costs except decompression can be easily obtained by observing each routine. Although decompression requires an inversion and a square root in $\mathbb{Z}_q$, in Edwards curves it is possible to combine and optimize them such that it "takes just a few multiplications more than a single exponentiation" [15]. Thus, we consider that decompression requires an exponentiation in $\mathbb{Z}_q$ (*ExpZq*).

We observe that the asymptotic costs of the proposed schemes in terms of field elements and elliptic curve operations are the same of the original ECQV and EdDSA schemes. The analysis shows that the most expensive operations are *CRTGen*, *CRTRec* and *Verify*, that require two scalar point multiplications each, plus compression and/or decompression. However, their actual performance may differ due to different requirements in terms of time constant algorithms. Finally, we observe that the high cost of the *CRTGen* operation is not an issue for most realistic scenarios, because it is operated less frequently and by the CA that is typically provided with higher computational capabilities.

5.2.9 Experimental evaluation

The testbed includes five architectures: an Intel Core i7-8650U (`x86_64`), a 1.5GHz 64-bit ARM Cortex-A72 CPU (`A72`), a 700MHz ARM11 CPU (`ARM11`) based on, a 168MHz 32-bit Cortex-M4 CPU (`M4`), a 16MHz 8-bit Atmega2560 AVR MCU (`Amega2560`). We implement the scheme for edwards25519 by leveraging optimized subroutines of existing libraries. The implementation for `Amega2560` includes ASM code proposed by Düll et al [48], while the other implementations use internals of the Libsodium library [99], which adopt assembly optimizations for all these platforms. All implementations run as single-threaded code. Implementations of `x86_64`, `A72` and `ARM11` run on Linux platforms, while those of `M4` and `Amega2560` run over the Embedded ABI. For Linux platforms, timings have been computed as the average of measures over batch executions based on large test vectors to reduce biases. Finally, measures do not include random data generation required by *CSRGen* and *CRTGen* routines.

Results are shown in Table 5.17. We observe that for the `Atmega2560` platform, the results are consistent with the asymptotic analysis proposed in Section 5.2.8 (Ta-

	x86_64	A72	ARM11	M4	Atmega2560
Comp.	0.0042	0.0247	0.207	0.737	67.4
Decomp.	0.0055	0.0267	0.222	0.781	179.3
CSRGen	0.019	0.117	1.38	4.77	1799
CRTGen	0.074	0.446	4.98	17.4	3885
CRTRec	0.079	0.470	5.34	18.2	4247
Sign	0.021	0.116	1.51	4.94	1939
Extract	0.059	0.372	3.68	14.1	1985
Verify	0.057	0.355	3.53	12.0	4106

Table 5.17: Timings of the proposed scheme instantiated for edwards25519 [*ms*]

ble 5.16). The other platforms (`x86_64`, `A72`, `ARM11`, and `M4`) show results that, at a first glance, seem partially inconsistent with the asymptotic analysis: *extract* takes longer than *Verify* although it requires the computation of an additional scalar point multiplication with variable base point. This is probably due to optimizations of the Libsodium library to improve verification of digital signatures, where the two scalar point multiplications are computed through a single invocation of an optimized non-constant time routine. This operation is faster than the single routine for single scalar point multiplication on a variable base point used by *extract*, that could also be optimized if needed.

When comparing our results with similar evaluations on the standard ECQV and ECDSA schemes [129] for `A72`, `ARM11` and `M4` (see Appendix 5.2.9), We observe that timings of the proposed scheme are four to ten times lower in all cryptographic operations comparable operations. We cannot compare *CSRGen* and *CRTGen* as their testbed include random data generation within measured timings. As we are not aware of implementations of ECQV and ECDSA that are able to operate on `Atmega2560` or similar hardware, the proposed approach seems the first able to bring implicit certificates on such a constrained platform.

In Table 5.18 we include performance results for ECQV and ECDSA schemes presented by [129] to ease comparison with our proposal (Section 5.2.9). While `A72` and `ARM11` architectures of [129] operate at the same frequencies of ours, `x86_64*` is an i7-9750H instead of an i7-8650U (the former should be slightly faster) and `M4*` operates at 80MHz instead of 168MHz. Moreover, timings of `CSRGen*`, `CRTGen*` and `Sign*` also include random data generation.

	secp256r1				secp256k1			
	x86_64*	A72	ARM11	M4*	x86_64*	A72	ARM11	M4*
CSRGen*	0.33	1.37	14.21	109.73	0.22	1.13	12.16	84.98
CRTGen*	0.62	2.85	30.02	233.45	0.48	2.35	26.15	186.20
CRTRec	0.60	2.79	29.29	231.47	0.46	2.26	25.72	184.27
Sign*	0.32	1.44	15.38	118.00	0.25	1.21	13.15	93.31
Extract	0.34	1.44	15.38	120.13	0.24	1.19	13.63	97.68
Verify	0.35	1.58	16.57	133.47	0.24	1.13	13.53	97.97

Table 5.18: Timings [*ms*] of ECQV+ECDSA based on *secp256r1* and *secp256k1* proposed by [129]

Chapter 6

Securing V2X against internal attackers

6.1 SixPack: Abusing ABS to avoid Misbehavior detection in VANETs

This chapter presents *SixPack*, a cyber attack on VANET communications that can go undetected by the current state-of-the-art anomaly detectors. The SixPack attack is a dynamic attack conducted by an insider attacker who modifies the content of the Basic Safety Messages to pretend a sudden activation of the braking system with the consequent activation of the Anti-lock Braking System and create a fake representation of the vehicle. The attacker then rejoins the fake representation of the vehicle with the real one, avoiding the current state-of-the-art anomaly detectors. This chapter experimental evaluates the evasion capabilities of the SixPack attack using the F^2MD test framework on the LuST and LuSTMini city scenarios, demonstrating the ability of the attacker to generate a high percentage of false positives that prevent the attack from being detected consistently.

6.1.1 The SixPack attack

The attack presented in this chapter is classified as an *insider attacker* following the classification presented in [86]. An insider attack is a typology of attack in which the attacker has acquired valid cryptographic credentials from the PKI to participate in the communications. The attacker is also considered active, i.e., being able to craft BSMs by arbitrarily modifying the values of its fields.

Attack dynamics

The SixPack attack presented in this work is designed as an evolution of the *position falsification* attacks described in [166], in which the vehicle falsifies its position by sending BMSs containing maliciously forged values of the speed, acceleration, heading, and other meaningful sensor values. As an example, in the *Eventual Stop* attack

presented in [166], the attacker sets the speed value of the BSM to zero and updates its position consequently to simulate the sudden stop of the vehicle. The other vehicles participating in the communication will see a vehicle stopped in the middle of the road although not physically there; thus, the detection of this attack can be easily done by comparing the content of the malicious messages with the mapping of the surrounding of the vehicle by using its advanced driver assistance system (ADAS) sensors.

The rationale behind the SixPack attack is that it can leverage modern ADAS' countermeasures. In fact, upon reception of a BSM with the ABS flag enabled, modern ADAS activate the braking system to avoid collision with the preceding vehicle as a primary safety countermeasure, even though the vehicle sending the message with the ABS flag enabled is not actually braking. This allows the SixPack attack to evade detection of nearby ADAS by creating a false representation of the vehicle position (although not being in a static position like in the eventual stop attack) end eventually rejoin the false representation of the vehicle with the real one, thus limiting the time window during which the nearby vehicles can detect the discrepancy between the messages and the actual vehicle position. It is important to remark that the attack can be repeated periodically according to the scope of the attacker, who might try to increase the traffic in the surrounding areas by creating traffic jams, creating congestion in the surrounding areas and possibly causing crashes.

Detailed attack description

The SixPack attack is composed of six different phases, described as follows:

- **Start**: this is the first phase of the attack, in which the attacker sends legit BSMs corresponding to the real measurements of the vehicle sensors without manipulating their content. From this phase, the attacker can either remain in the *start* phase or enter the next phase of the attack (*init*).
- **Init**: this is the initial phase of the attack and is used as a transitory phase between the normal vehicle operation and the attack itself. In the *init* phase, the attacker records the values of the vehicle sensors (including current time, speed, position, and other data relevant to the attack). After the attacker has gathered the data required for the first part of the attack, the *FakeBrake* phase is entered. However, recording the vehicle's state is not interrupted since it is necessary for the fourth phase of the attack (*recovery*).
- **FakeBrake**: in this phase, the attack is triggered by sending to the nearby vehicles a maliciously forged BSM in which the vehicle is faking a sudden braking with the consequent activation of the ABS. This can be achieved by manipulating the *brake system status* field of the BSM. Besides claiming the activation of the ABS, the attacker will produce BSMs whose values are consistent with the dynamic of a real braking value. Hence, fake values for the vehicle speed, position, and acceleration are computed to comply with the fundamental laws of physics. As an example, the deceleration required to bring the vehicle to a full stop in the desired distance is computed through the formula

$$deceleration = \frac{(s_f^2 - s_i^2)}{2\delta}$$

where s_f is the final vehicle speed, s_i is the initial speed, and δ is the distance. As soon as the desired final speed of the vehicle is reached, the *recovery* phase is initiated.

- **Recovery**: in this phase, the attacker simulates the acceleration required by the fake vehicle representation to rejoin with the position of the real vehicle. This phase of the attack requires the data recorded during the previous phases (*init* and *fakebrake*) to recreate the path traveled by the vehicle. The accelerating vehicle's acceleration and speed are computed using a proportional–integral–derivative (PID) controller. A PID controller is a control loop mechanism employing feedback that is also widely used in autonomous driving applications [92, 170] to recreate the realistic behavior of the vehicle. A generic PID controller is described as
$$u(t) = K_p\delta(t) + K_i \int_0^t \delta(t')dt' + K_d\frac{\delta(t)}{dt}$$
where K_p, K_i, and K_d are the proportional, integral, and derivative coefficients. The next phase of the attack (*rejoin*) is entered when the position of the fake representation of the vehicle is close to the real one.

- **Rejoin**: in this phase, the fake representation of the vehicle is rejoined with its real representation; hence the attacker computes values of the BSMs data (acceleration, speed, position, etc.) in order to smoothly close the gap between the fake and the real representation of the vehicle. The attacker then resumes sending legit BSMs instead of fake ones, thus effectively substituting the actual vehicle with its fake representation.

- **Stop**: in this phase of the attack, the attacker stops recording the real BSM values needed for the attack and re-enters the *start* phase.

The SixPack attack presented in this chapter differentiates from the attacks already presented in the literature (see Section 4.1.1) because this is the first attack that tries to mimic the dynamic behavior of a legitimate vehicle.

6.1.2 Experimental analysis of the SixPack attack

This section presents the experimental evaluation of the SixPack attack presented in Section 6.1.1. Section 6.1.2 presents the setup used for the experimental demonstration, including the description of the simulation software and the parameters used in the simulation process. In Section 6.1.2, the simulation results are depicted, demonstrating the ability of the SixPack attack to evade current state-of-the-art detectors.

Simulation setup

z To provide an accurate analysis of the effectiveness of the SixPack attack in realistic scenarios, the attack is implemented using the VEINS simulator [149] (described in 4.2.1) and extending the F^2MD test framework presented in [86] accordingly, thus allowing us to simulate different misbehavior use cases. VEINS is an open-source,

customizable framework for the simulation of vehicular network communications. The VEINS framework is based on the OMNeT++ [167] (a well-known network communication simulator) and on the SUMO [101] frameworks (a road traffic simulator). It is used to couple the execution of the two simulators to provide realistic inter-vehicular communication. The F^2MD [86] test framework is used to evaluate the effectiveness of the SixPack attack. This framework tests attack scenarios against a set of selected detection algorithms. Following the definition of the different types of attacks presented in [86], the SixPack attack is part of the *insider attacks* category; hence any detection method designed to use the PKI is inapplicable.

The F^2MD framework is designed to include all the detection algorithms in two categories: the *Local Detection* and the *Global misbehavior detection* modules. The former module groups all the detection methods designed to run on a single vehicle (i.e., using external messages and the data produced by the vehicle itself) and includes the plausibility and consistency checks. Upon detecting an anomaly, the *local misbehavior application* sends a detailed report to the MA. The Global misbehavior detection module implements the MA and is responsible for investigating and revoking the misbehaving entity.

For evaluating the proposed SixPack attack, the F^2MD framework is configured to use *ExperiCheck* as the plausibility check. At the same time, the local detection is implemented using both the *ThresholdApp* and the *BehavioralApp*. The plausibility check is a list of comparisons based on the analysis of the following values contained in the BSMs:

- **range plausibility**: checks if the position of the sender is inside the radio coverage;
- **position plausibility**: checks if the position of the sender is on the road pavement;
- **speed plausibility**: checks if the speed of the sender is lower than a pre-defined threshold;
- **position consistency**: checks if the position values of two consecutive BSMs sent from the same vehicle are consistent;
- **speed consistency**: checks if the speed values of two consecutive BSMs sent from the same vehicle are consistent;
- **position speed consistency**: checks if the speed and position values of two consecutive BSMs sent from the same vehicle are consistent;
- **beacon frequency**: checks the frequency of the BSMs;
- **position heading consistency**: checks if the heading of two consecutive BSMs sent from the same vehicles is consistent;
- **intersection check**: checks if two or more BSMs sent by different vehicles have overlapping positions;

- **sudden appearance**: checks if a vehicle is suddenly appeared within radio coverage range.
- **Kalman filter tracking**: checks if values of the fields of the BSMs are inside a set of Kalman filter predicted values.

The consequences of the SixPack attack and the detection capabilities of the algorithms included in the F^2MD framework are evaluated using the Luxembourg SUMO Traffic (LuST) [43] scenario and its reduced version LuSTMini. The LuST scenario includes 24 hours of synthetic mobility communication based on the realistic data validated by the VehicularLab of the University of Luxembourg. The LuST scenario is extended for more than 155 square kilometers, while the total length of the roads is above 930 kilometers. For the experimental evaluation, both scenarios are configured using the recommended parameters except for the *maximum plausible range* of the receiving transmission, which is increased from 420 to 800 meters. In each scenario, the attacker is configured with a probability of starting the attack equal to 50%, 10%, and 1%.

Detection results against the SixPack attack

Tables 6.1 and 6.2 presents the detection results of the *ExperiCheck* (plausibility check), and the *Threshold App* and *Behavioral App* (misbehaving detection) against the SixPack attack simulated on the LuST and LuSTMini scenarios, respectively. The results are presented using false positive and recall rates because these two indexes allow us to demonstrate the ability of the SixPack attack to go undetected by the current state-of-the-art detection algorithms. In contrast, other synthetic indexes (such as the *F-measure* that measures the test accuracy) do not allow this analysis to be an aggregate of multiple indexes. Both Tables show the false positive rate and the recall rate of the detection algorithms (columns) against the SixPack attack simulated with a probability of 50%, 10%, and 1%.

	ExperiCheck		**Threshold App**		**Behavioral App**	
Atk Prob	*FP rate [%]*	*Recall rate [%]*	*FP rate [%]*	*Recall rate [%]*	*FP rate [%]*	*Recall rate [%]*
50%	7.99%	20.62%	41.23%	60.19%	40.77%	59.26%
10%	7.89%	19.46%	48.60%	67.82%	47.70%	66.67%
1%	7.75%	2.70%	48.03%	55.56%	47.56%	66.67%

Table 6.1: Detection results of the SixPack attack in the LuST scenario

	ExperiCheck		**Threshold App**		**Behavioral App**	
Atk Prob	*FP rate [%]*	*Recall rate [%]*	*FP rate [%]*	*Recall rate [%]*	*FP rate [%]*	*Recall rate [%]*
0.5	0.15%	1.40%	6.90%	12.79%	6.90%	11.63%
0.1	0.27%	0.44%	6.41%	16.67%	6.41%	11.11%
0.01	0.34%	0.00%	9.88%	0.00%	9.88%	0.00%

Table 6.2: Detection results of the SixPack attack in the LuSTMini scenario

From the analysis of the results evaluated in the LuST and LuSTMini scenarios (Tables 6.1 and 6.2), we notice that the best detection performances are achieved by

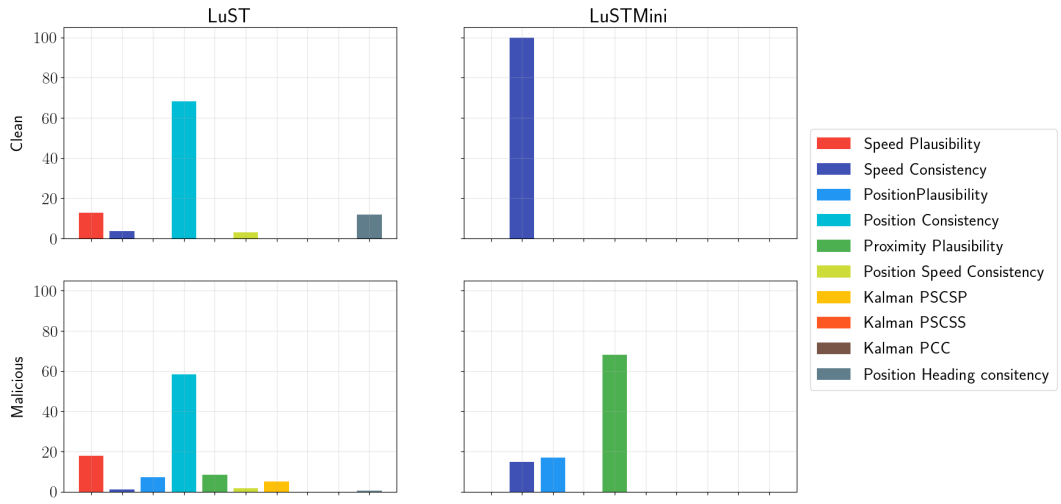


Figure 6.1: Trigger analysis of the False Positives and Recall of the detection algorithms on both LuST and LuSTMini simulated scenarios

using the misbehaving detection algorithms in both scenarios. At the same time, the plausibility check has relatively low values of False Positive and Recall rates in both scenarios. From comparing the results in the two scenarios, it is clear that the LuST scenario detectors can achieve higher detection of the SixPack attack compared to the LuSTMini scenario. However, the SixPack attack on any scenario can consistently avoid detection (more than 30% of the times), and its execution also results in a high percentage of false positives, demonstrating its detection evasion capabilities. We also remark that the attack probability does not particularly affect the detection capabilities of the three different detection methods, despite the logical consequence that lower attack probabilities (i.e., lesser triggers of the attack in the simulation) result in lower detection capabilities of the algorithms.

Analysis of the detection results

Following the detection results presented in the previous section, this section presents the analysis of the plausibility check of the detection algorithms that detected the attack. In Figure 6.1 the results of this analysis are depicted, comparing the percentage (y-axis) of triggers generating false positives (in the *clean* scenario) with the triggers that can detect the ongoing attack (in the *malicious* scenario). The left part of Figure 6.1 presents the results of this analysis achieved on the LuST scenario, while the right part presents the analysis evaluated on the LuSTMini scenario.

As presented in Figure 6.1, the results of this analysis demonstrated that in the LuST scenario, characterized by a higher number of simulated vehicles and comprising more roads than the LuSTMini scenario, the most triggered detector is the *position consistency check* in both clean and malicious scenarios (with 68.3% and 58.27% of the overall number of triggers, respectively), while the second most triggered detector is the *speed plausibility check* (12.83% and 17.86% of the triggers in the clean and

malicious scenarios, respectively). Another interesting result is that both the *proximity plausibility* and the *position plausibility* checks are responsible for a negligent amount of false positives in the clean scenario (0.09% and 0%) while they are responsible for the trigger of the detection algorithm for the 8.51% and 7.18% of the times, respectively.

A different scenario is presented from the analysis of the triggers in the LuSTMini simulation, in which the 100% of the false positives are triggered by the *speed consistency* check, while the detection of the attack is triggered entirely by the *proximity plausibility* check (68.09%), the *position plausibility* check (17.02%), and the *speed consistency* check (14.89%).

In conclusion, if the SixPack attack is conducted in a scenario in which the current state-of-the-art detectors are deployed, it would be unnoticed by most detectors, being able to trigger a significant amount of false positives while being detected approximately the same times. Moreover, the two detectors that have demonstrated higher resilience to the SixPack attack (despite not being able to detect it consistently) are the *proximity plausibility* and the *position plausibility* checks.

6.1.3 Limitations of the SixPack Attack

This section analyzes SixPack and the F^2MD detection framework to understand the current limitations of the attack. Section 6.1.3 proposes an analysis of how F^2MD is able to detect it, and Section 6.1.3 outlines the improvements required to increase SixPack's evasion capabilities.

Trigger analysis

The original SixPack work was tested against an older version of the F^2MD framework, demonstrating its evasion capabilities. However, since the F^2MD framework has been updated, the same experiments were replicated to demonstrate the improvement in the detection framework, and the results are summarized in Table 6.3. In particular, the two right-most columns of Table 6.3 shows the percentage of triggers of the original SixPack attack on the old and new version of F^2MD, while the rows represent different triggers. From the comparison presented in Table 6.3 we notice that the improved version of F^2MD can trigger a higher number of checks against the same SixPack attack compared to its previous version. Moreover, despite the trigger percentages against the *position plausibility* and *speed consistency* are reduced, the trigger of the *proximity plausibility* trigger increases from 66.7% to 71.6%.

This section analyzes the three most triggered checks on the latest version of the F^2MD framework against the original SixPack attack: the *"proximity plausibility"* check (71.58%), the *"position plausibility"* check (11.72%), and the *"speed consistency"* check (7.32%).

Proximity plausibility From the analysis of the trigger of the *proximity plausibility* check, we observe that the F^2MD framework raises them in both *recovery* and *rejoin* phases. In particular, we observe that while the attacker is sending malicious BSM with a fake GPS position to rejoin the digital representation of the vehicle with its real one, it does not consider the presence of other C-ITS entities in the close surroundings

trigger name	F^2MD *Original*	F^2MD *Updated*
Proximity plausibility	66.7%	71.6%
Position plausibility	18.6%	11.7%
Speed consistency	15.8%	7.3%
Kalman PSCSP	0%	6.1%
Range plausibility	0%	5.0%
Position speed consistency	0%	3.7%

Table 6.3: Comparison of the original and updated version of the F^2MD framework against the original SixPack attack.

of the fake position. It may happen that the fake representation of the vehicle becomes too close to other vehicles, thus triggering this check. Moreover, during this analysis, we observe that the proximity plausibility check is always triggered on the second message coming from the C-ITS entities whose GPS position is close to the position sent by another vehicle. Hence, a malicious C-ITS entity could exploit this behavior by sending a BSM whose GPS values are close to the one of a real vehicle just before the legit BSM is sent by the legit vehicle. A similar attack would cause F^2MD to blame the legit vehicle, instead of the attacker, with consequences depending on the reaction mechanism triggered by F^2MD detections.

Position plausibility The *position plausibility* check is triggered by the F^2MD framework when the position specified in a BSM is not realistic. This check includes multiple tests that compare the GPS coordinates of each BSM with the position of obstacles and other objects on the road and check the vehicle's speed if its GPS coordinates locate the vehicle outside of the road (vehicles outside the road should be parked or moving slowly). The SixPack triggers this check mostly during the *recovery* and *rejoin* phases on curvy roads. The root cause lies in how SixPack interpolates the GPS data logged during the attack to compute fake GPS positions. This interpolation only considers two consecutive GPS coordinates, thus introducing artifacts during the recovery and rejoin phases which are sometimes identified by F^2MD as violations of position plausibility.

Speed consistency The *speed consistency* check is triggered if a vehicle accelerates or decelerates too abruptly, to the point of not being compatible with basic laws of physics (we recall that BSMs are sent every 100 ms). This check compares the speed value extracted from two consecutive BSMs sent from the same vehicle against two thresholds hardcoded in the framework: the *max plausible acceleration* and the *max plausible deceleration*. Moreover, the change of the vehicle position is also compared with the speed difference between two consecutive BSMs, resulting in additional triggers in case the expected position is different from the one encoded in the following messages. While SixPack is active, we notice that the majority of the triggers of the *speed consistency* are generated in the *recovery* and *rejoin* phases, with a trigger percentage very close to the one of the *position plausibility* check, due to the same reasons.

Limits of the SixPack attack

The current limitations of the SixPack attack can be summarized as follows:

- SixPack does not consider nearby C-ITS entities while sending BSM messages in both *recovery* and *rejoin* phases, possibly triggering the *proximity plausibility* check;
- the interpolation used to compute GPS coordinates in both *recovery* and *rejoin* phases is naïve and sometimes generates GPS coordinate outside of the road pavements, possibly triggering the *position plausibility* check;
- the speed difference between two consecutive BSMs during both *recovery* and *rejoin* phases can be higher than the detection thresholds, possibly triggering the *speed consistency* check.

To improve the SixPack attack, both the *recovery* and *rejoin* phases are improved. To avoid triggering the *proximity plausibility* check, SixPack $v2$ implements a *proximity avoidance* mechanism that prevents sending BSMs with GPS coordinates near other C-ITS entities. To implement this protection mechanism, SixPack $v2$ saves all the incoming BSM messages from other C-ITS entities to extract their coordinates. The second main improvement of the SixPack $v2$ *recovery* and *rejoin* phases is an advanced interpolation algorithm that adjusts the path according to the road and the behavior of the vehicle, thus preventing the triggering of *position plausibility* checks due to GPS coordinates outside of the road.

To reduce the trigger of the *speed consistency* check, it is necessary to limit the maximum acceleration and deceleration values calculated by the detection framework by sending consecutive BSMs with speed values very close to each other. However, this solution introduces two significant disadvantages that it is necessary to consider. The first issue is the limitation of the maximum acceleration/deceleration value. By limiting these values the duration of *recovery* and *rejoin* are increased. The second issue is that this detection mechanism relies on the value of two parameters, the *max plausible acceleration* and the *max plausible deceleration* and that any modification of these values has a direct impact on the trigger of *speed consistency*. Since the SixPack attack is designed to avoid detection by mimicking the realistic behavior of a vehicle instead of generating messages that comply with the detection parameters of the F^2MD framework, any direct countermeasure was implemented to reduce the trigger of the *speed consistency* check. However, the improvements made to address the other two checks allow SixPack $v2$ to decrease the trigger of this check, further increasing its detection evasion capabilities.

6.1.4 The Enhanced SixPack V2

This section presents the enhanced version of the SixPack attack (from now SixPack $v1$), SixPack $v2$. The improvements of SixPack $v2$ only affect the *FakeBrake*, *Recovery*, and *Rejoin* phases.

As stated in Section 6.1.3, one of the issues of SixPack $v1$ is a simplistic interpolation algorithm. SixPack $v2$ boasts an improved interpolation algorithm outlined below. We consider an interpolation factor t that is a real number.

Given two points A and B of coordinates (x_a, y_a) and (x_b, y_b), where A is the starting position and B is the final position, Formula 6.1a evaluates the coordinates of the interpolated point C.

$$C = A + t(B - A) \tag{6.1a}$$
$$x_c = x_a + t(x_b - x_a) \tag{6.1b}$$
$$y_c = y_a + t(y_b - y_a) \tag{6.1c}$$

In particular, the coordinates (x_c, y_c) of C are computed with Formula 6.1b and Formula 6.1c. This allows us to evaluate the interpolation between two points using the same interpolation factor. The interpolation factor t is computed using the following mapping formula:

$$t = t_{min} + \frac{(s - s_{min})(t_{max} - t_{min})}{(s_{max} - s_{min})} \tag{6.2}$$

where s is the initial value that we want to map (with $s \in [s_{min}, s_{max}]$) and $[t_{min}, t_{max}]$ is the reference range of the interpolation factor t.

FakeBrake v2

In this phase, the attacker simulates a sudden braking with the direct consequence of the ABS activation, modifying the `BrakeSystemStatus` field of the outgoing BSM. Compared to the original version of SixPack, SixPack $v2$ starts the linear interpolation to evaluate the realistic position of the fake BSM with the start of the attack. In this phase, the speed of the vehicle s is mapped using Formula 6.2 in the range $[0, 1]$ to evaluate the *starting interpolation factor* t_{start} that is used for the next phases, with $[s_{min}, s_{max}]$ being equal to $[0, 45]$ m/s, which are the minimum and maximum speed values for the vehicle in the simulator.

Recovery v2

In this phase, the attack starts to send malicious BSMs to mimic a realistic acceleration of the vehicle from the position where the attack is triggered. In the first step of the recovery phase, the attacker uses the interpolation formula to evaluate the fake position of the vehicle, where A is the position where the attack is triggered, and B is the next GPS position recorded by the antenna. The interpolation factor t is the starting interpolation factor t_{start}. After this step, the next fake positions are computed starting from the original GPS location recorded by the antenna. In each iteration, the

Algorithm 11 Recovery Phase

```
1:  function RECOVERY( )
2:      speed = PID.CALCULATE(J, K)
3:      t_inc = MAPPING(speed)
4:      if direction is straight then
5:          t_start += t_inc
6:      else
7:          t_start −= t_inc
8:      new_position = INTERPOLATE(J, K, t_start)
9:      while new_position is BSMs proximity do
10:         t_start += ADJUST(t_inc)
11:         new_position = INTERPOLATE(J, K, t_start)
12:     return new_position
```

attack uses two consecutive GPS readings as the starting and end position (J, K) for the evaluation of the speed of the vehicle. The speed of the vehicle is computed with a *PID controller* [92, 170], and then mapped using Formula 6.2 into an *incremental interpolation value* t_{inc}, where $[t_{min}, t_{max}]$ are equal to $[0.05, 0.2]$, and $[s_{min}, s_{max}]$ is still equal to $[0, 45]$. Then, SixPack applies the interpolation equations described by the Formula 6.1b and Formula 6.1c using the previous fake position $C : t - 1$ as the starting position A and the real next position K as the destination point B. However, the interpolation factor is evaluated as the sum of the previous interpolation factor and the incremental interpolation value in case the vehicle is in a straight direction. Otherwise, the incremental interpolation value is subtracted from the previous interpolation factor. These modifications to the recovery phase of the original SixPack attack allow evading the trigger of the *position plausibility* check.

Moreover, before sending a fake BSM, the coordinates are adjusted by modifying the mapping range of the *incremental interpolation value* in case the resulting coordinates are in the proximity of another C-ITS entity, thus preventing the triggering of the *proximity plausibility* check.

The operations just described theoretically for the *Recovery* phase are presented in Algorithm 11 where the *mapping function* of the third line refer to the Formula 6.2 in which the s parameter represent the vehicle speed calculated with the *PID controller*, while instead the *interpolate function* refer to the Formula 6.1a where J and K correspond to A and B, respectively. Finally, the *adjust function* represent the checks applied to the $new_{position}$ to implement the *proximity avoidance* mechanism described earlier.

Rejoin v2

In the rejoin phase, the same modifications already presented for the *Recovery v2* phase are applied with minor modifications. In this phase, the *incremental interpolation value* is fixed and always added to the previous *interpolation factor* to speed up the rejoin process. Moreover, the final position B used for the evaluation of the interpolated

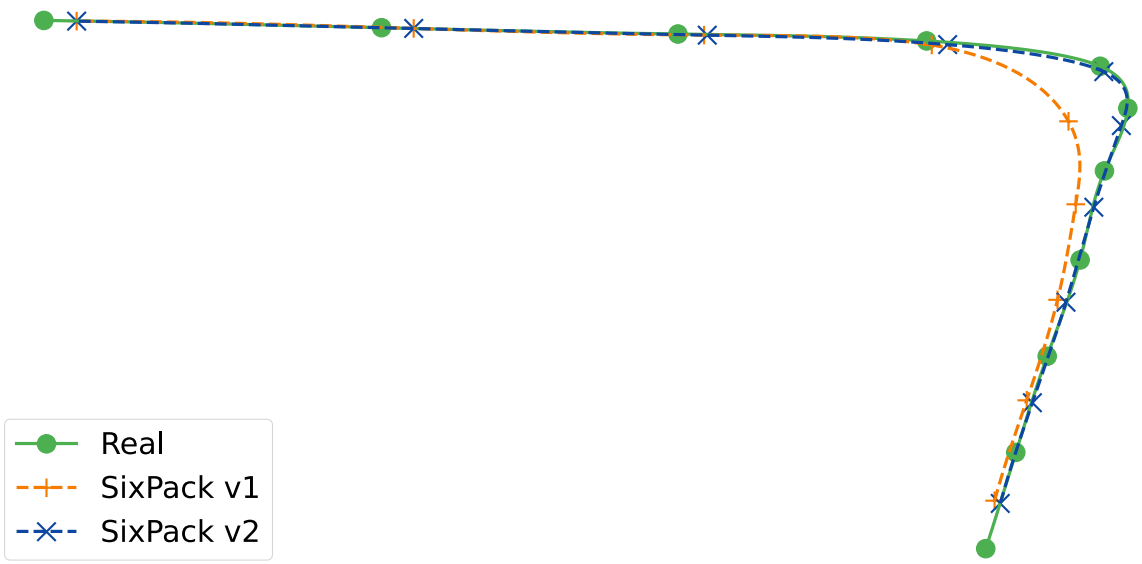


Figure 6.2: Real path vs. SixPack $v1$ vs. SixPack $v2$

position C is the actual position of the vehicle and not the last GPS coordinates observed by the antenna.

The final result of the improvements applied to SixPack $v2$ is shown in Figure 6.2, where both versions of SixPack are applied to the same scenario. The path represented in Figure 6.2 shows the *recovery* phase of the algorithms, where both versions of SixPack are generating fake GPS positions to mimic the behavior of the real vehicle. The three lines depicted in Figure 6.2 represent the real path of the vehicle (green line), and the two versions of SixPack (v1 depicted in orange, v2 depicted in blue). Figure 2 clearly shows that the improvements of SixPack v2 allow the generation of more plausible malicious positions, that are comparable with the real path followed by the vehicle. Against the same scenario however, the original version of SixPack struggles to create realistic GPS position, triggering the activation of plausibility checks of F^2MD.

6.1.5 Experimental evaluation of SixPack v2

This section compares the evasion capabilities of SixPack $v1$ and SixPack $v2$. The simulation parameters and setup are presented in Section 6.1.5, while the detection performance of F^2MD against the two SixPack versions are compared in Section 6.1.5.

Simulation setup

For comparing the two versions of the SixPack attacks, the same simulation setup already presented in [128] is used. The setup is based on the VEINS simulator [149] and the F^2MD test framework [86]. VEINS is an open source framework for the simulation of vehicular network communications based on the *OMNet++* [167] and the *SUMO* [101] frameworks. We configured the F^2MD framework to use only the *ExperiCheck* as a plausibility check for the detection since previous work has proven that it is the only one capable of detecting malicious messages [128]. The detection results

of the F^2MD framework are tested on the Luxembourg SUMO Traffic minified (LuST-Mini) scenario [44] released by the VehicularLab of the University of Luxembourg. The configuration parameters used for the experimental evaluation of the F^2MD framework against both SixPack versions are the ones recommended for the LuSTMini scenario.

We repeated the test 5 times to avoid possible biases related to single simulations.

For the experimental evaluation of the detection performance of the F^2MD scenario, the results are presented by means of *Precision*, *Recall*, and $\mathcal{F}-$measure.

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.4}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.5}$$

Precision which is described by the Formula 6.3 measures the ratio of correctly identified anomalies over all the detected anomalies, while the *Recall* metric described by the Formula 6.4 measures the ratio of correctly identified anomalies over the overall amount of real anomalies. Being TP, FP, and FN, the number of true positives (non-legit messages identified as anomalies), false positives (legit messages identified as anomalies), and false negatives (non-legit messages identified as normal). The $\mathcal{F}-$measure of Formula 6.5 is a statistical index representing the accuracy of the test and corresponds to the harmonic mean of the precision and recall. All three indexes range between 0 and 1, with values close to 0 denoting poor detection results and values close to 1 representing good detection results.

Detection results against the SixPack attacks

This section presents the detection results of the F^2MD framework against the original and the improved version of SixPack, labeled as SixPack (original) and SixPack v2 (improved). While the detection results of the F^2MD framework against the original version of SixPack have already been presented in [128], the F^2MD framework has been updated since thus, a re-evaluation is necessary for the sake of completeness.

Table 6.4 shows the detection results of the F^2MD framework against the SixPack. The columns of Table 6.4 shows the true positive rate (*TP rate*), false positive rate (*FP rate*), false negative rate (*FN rate*), and $\mathcal{F}-$measure achieved by the F^2MD framework against the SixPack v1 attack. The first 5 rows represent the results of different tests, while the last row shows the average across the 5 simulations.

From the analysis of Table 6.4 we notice that the detection performance of the F^2MD framework against the SixPack $v1$ attack is limited, achieving a maximum of $\mathcal{F}-$measure equal to 0.5, with an average of 0.4497. However, the results achieved by the current version of the F^2MD framework against SixPack are significantly higher than those achieved by the previous version analyzed in [128].

#	*TP rate [%]*	*FP rate [%]*	*FN rate [%]*	$\mathcal{F}-measure$
1	3.2609	7.6087	2.1739	0.4
2	3.8043	7.0652	1.6304	0.4667
3	4.3478	9.7826	1.0870	0.4444
4	4.8913	9.2391	0.5435	0.5
5	3.8043	8.1522	1.6304	0.4375
avg	4.0217	8.3696	1.413	0.4497

Table 6.4: Detection results of the F^2MD framework against the SixPack $v1$ attack

#	*TP rate [%]*	*FP rate [%]*	*FN rate [%]*	$\mathcal{F}-measure$
1	2.1739	8.6975	3.2609	0.2667
2	1.087	7.6087	4.3478	0.1538
3	2.1739	8.1522	3.2609	0.2759
4	2.7174	8.1522	2.7174	0.3333
5	1.6304	8.1522	3.8043	0.2143
avg	1.9565	8.1522	3.4783	0.2488

Table 6.5: Detection results of the F^2MD framework against the SixPack $v2$ attack

Table 6.5 shows the detection results of the F^2MD framework against the SixPack $v2$. Table 6.5 is organized in the same way as Table 6.4.

From the comparison of Table 6.5 and Table 6.4 we notice that the detection results of F^2MD against the SixPack $v2$ attack are significantly lower. The improvements described in Section 6.1.4 are effective in increasing the evasion capabilities of SixPack $v2$, as demonstrate by the overall lower *true positive rate* and higher *false negative rate*. We also remark that these improvements do not affect the *false positive rate* of the F^2MD framework, being almost the same against both attack versions. Moreover, by comparing the average $\mathcal{F}-$measure achieved by framework against SixPack and SixPack $v2$, we notice that SixPack $v2$ is undetected more frequently, with an average $\mathcal{F}-$measure of only 0.2448 compared to the baseline of 0.4497 against SixPack.

6.2 Local and Global detection of internal false position attacks in V2X communications

This chapter describes the design of a new detection system, namely *Local* and *Global* detection. The proposal's rationale is to use the data collected from reality through vehicle sensors to verify the abstract information provided by analyzing the messages received from other vehicles and mark each message as plausible or not plausible. The *Local* detection method is executed on each vehicle. In contrast, the *Global* is applied to each RSU, requiring a dedicated VANETs infrastructure to collect and analyze the

surrounding data.

6.2.1 Detection Design

This section describes the design of the new detection system. Section 6.2.1 introduce the detection solutions, Section 6.2.1 analyzes the *Local detection* method, and Section 6.2.1 analyzes the *Global detection* method.

Detection procedure

The *Local* method is entirely based on V2V communication, and all the algorithms are executed inside each vehicle. In contrast, the *Global* method uses all the VANETs infrastructure, taking advantage of the V2I communications and the algorithms executed on each RSU. The proposed detection solutions combine and compare the abstract information provided by the analysis of the received BSMs exchanged over the VANETs network and the information obtained from the analysis of the surrounding physical reality. Since it is not the primary purpose of this work, the surrounding physical reality that can be achieved through *sensor fusion*, *data fusion* and *artificial intelligence* algorithms is simulated by making available to each *detector node* (vehicle or RSU, depending on the detection method) the position of the nearby vehicles. To better describe the design of the detection method, Figure 6.3 proposes a diagram representing the entire data flow of the detection procedure.

Both detection methods start whenever a BSM arrives. When a message is received, its position is extracted from the location information obtaining the GPS coordinates of the sending C-ITS entity. After this step, starting from the extracted GPS coordinates, the received message is discarded if the reported position results outside the perception area covered by the surrounding physical reality, as shown by the orange box in Figure 6.3, due to the uselessness of the information because the data cannot be compared with the proximity physical reality. Otherwise, the detection phase continues by retrieving information from the *sensor fusion* process, which is simulated and represented by the cloud box in Figure 6.3. After obtaining the abstract information from the BSM analysis and the physical information from the proximity reality analysis, it is possible to combine the actual vehicle's position with the untrusted information of the V2X communications. Due to the inaccuracy of the reality perception data, it is necessary to consider a small approximation error and thus consider as correct a not-perfect match between the position reported by the BSM and the actual real position of the surrounding vehicle. Hence, by taking into account, the approximation error just described, the position extracted from the received BSM message is compared with the vehicle's actual position in proximity of the *detector node*. The received BSM is classified as genuine (the green box in Figure 6.3) if the abstract information respects the physical reality data. Otherwise, the received BSM message is marked as malicious and undergoes investigation processes (the red box in Figure 6.3).

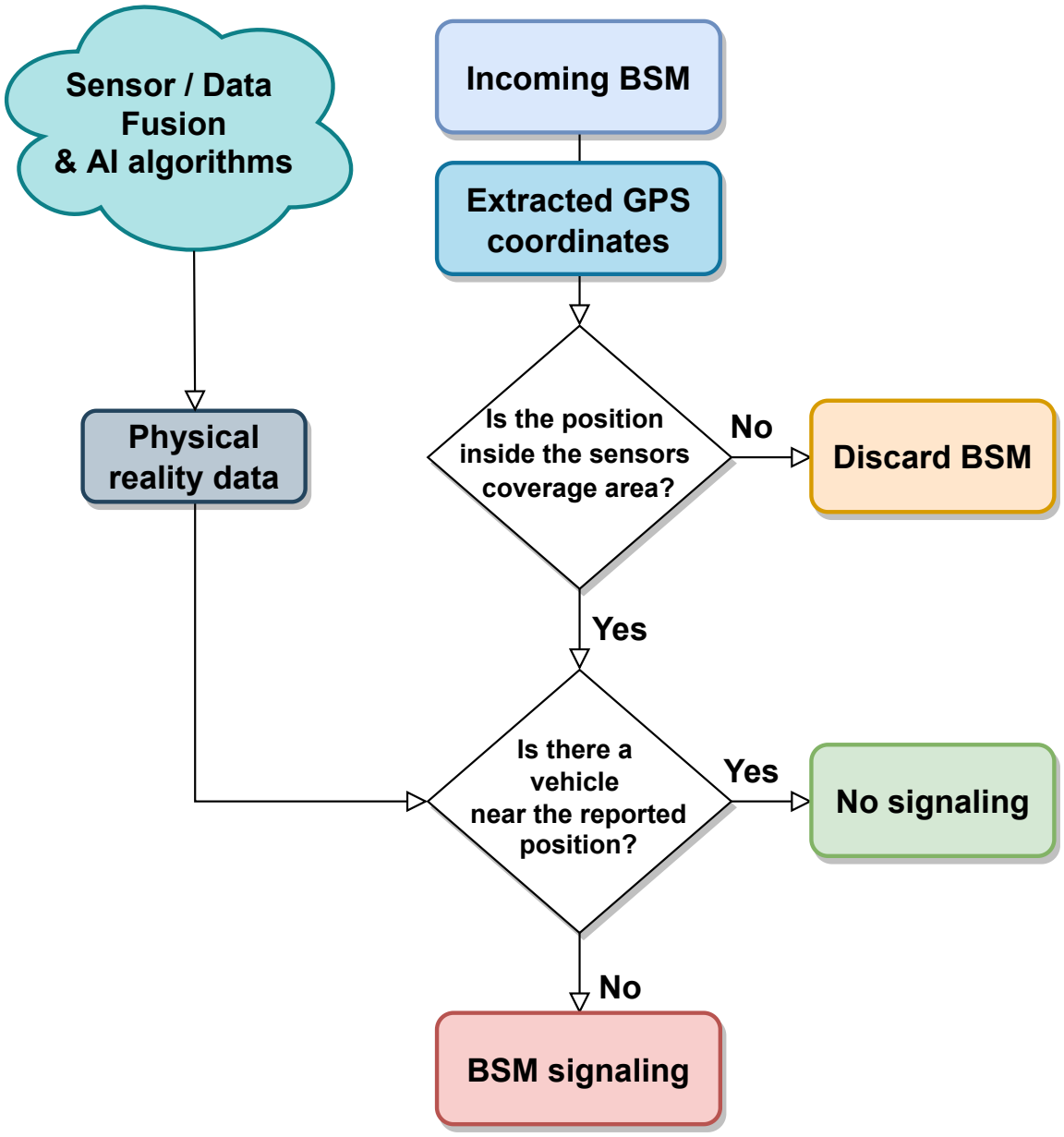


Figure 6.3: Detection procedure

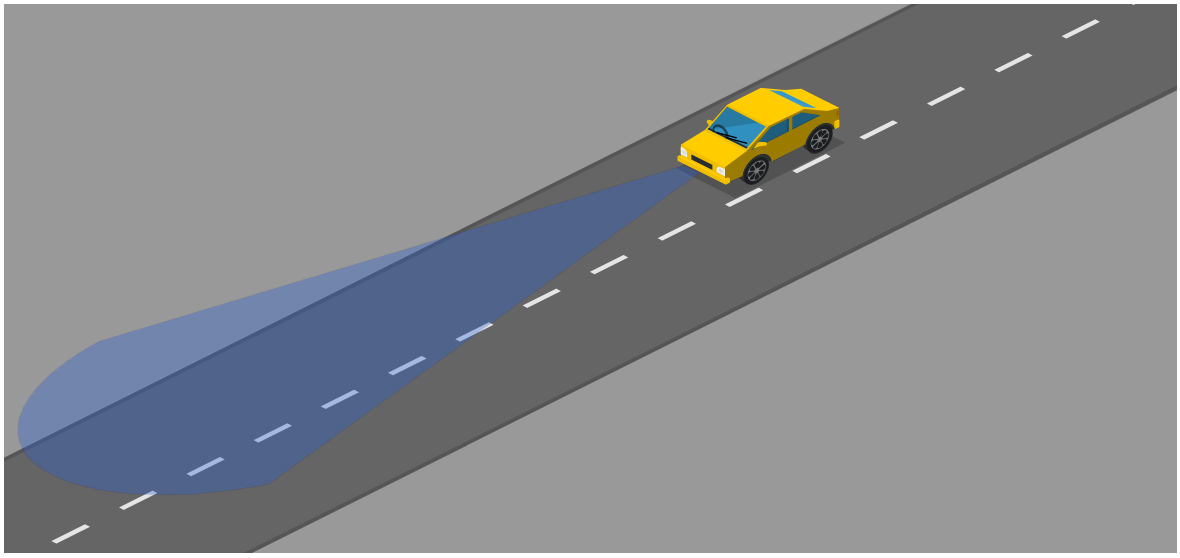


Figure 6.4: Conic shaped sensors coverage area

Local detection

The *Local* detection method, which is implemented locally on every vehicle using the on-board sensors, however, provides a more limited area of detection compared to the one offered by the *Global* detection while at the same time ensuring a less expensive investment by not requiring an ad-hoc infrastructure.

The design of *Local* detection includes two distinct variants based on different vehicles typology simulating different sensors to obtain information about the nearby vehicles. Nowadays, the automotive sector offers two types of vehicles: modern vehicles equipped with mandatory basic sensors and advanced vehicles designed to provide autonomous driving solutions. Modern vehicles use the primary *ADAS* (Advanced Driver Assistance Systems) sensors present onboard [45] which are primarily located in front of the vehicles and assist the driver by creating an automatic system that improves driving safety by reducing the reaction time required in dangerous situations. The *Local* detection method is designed to simulate the anterior sensors, such as cameras and radar, creating a frontal area of reality perception with a conical shape shown in Figure 6.4, where the blue area represents the sensors simulation while the vehicle is reported in yellow.

On the other hand, with regard to future advanced vehicles will be equipped with a more significant number of sensors. Hence, the perception area of the physical reality area is represented with an ellipsoidal shape around the vehicle, simulating most of the available sensors. The ellipsoidal shape allows to create a perception area of almost 360 *degrees* around the vehicle as shown in Figure 6.5 where, as for the *Cone* variant, the vehicle is presented in yellow, while the blue area describes the perception area.

Global detection

The *Global* detection method relies on several Road Side Units (RSUs) strategically located on the edges of the roads in the VANETs networks architecture and by using

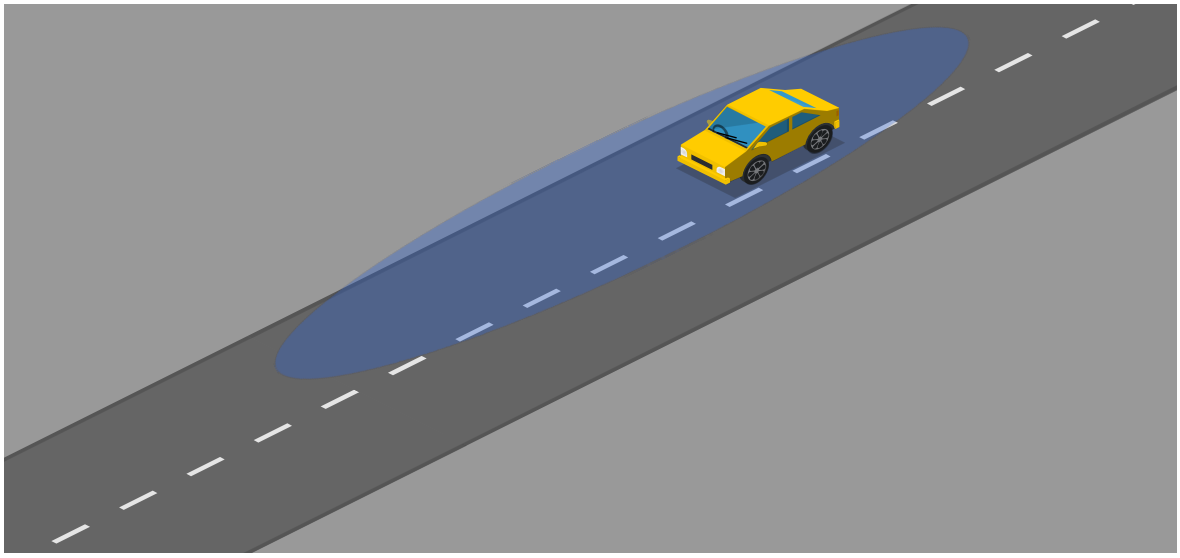


Figure 6.5: Ellipsoidal shaped sensors coverage area

technologies able to obtain information from the surrounding physical reality. Among these technologies *Intelligent Traffic Lights* (ITLs) or road cameras are considered employed for traffic analysis or traffic violations. Subsequently, through the applications of data fusion processes and artificial intelligence algorithms, it is possible to obtain information about the actual position of vehicles on road pavements.

The *Global* detection method needs to consider a more complex and expensive infrastructure composed of several cameras capable of analyzing the road traffic and providing accurate information about the actual vehicles' positions. The process of extracting information from the raw data provided by the cameras was simulated, as for the *Local* method, by making available to the RSUs, all the actual vehicles' positions in a large circular area centered on the RSU location as shown in Figure 6.6. Figure 6.6 represent a road intersection in which an RSU is placed in the center of the road intersection, simulating the behavior of an ITL, showing the simulated sensor coverage area in light blue.

The *Global* detection method is applied internally to each RSU which analyzes the BSM messages received from the nearby vehicles through V2I communications. Then, the necessary data are extracted and compared with the information obtained from the analysis of the surrounding physical reality.

The *Global* method, while requiring a dedicated infrastructure with higher investment, provides a significant advantage over the *Local* one since it is implemented directly within the VANETs infrastructure. The malicious behavior signaled will not have to undergo additional analysis processes to verify if the information is truthful.

6.2.2 Detection Implementation

This section presents an accurate analysis of the implementation of the *Local* and *Global* detection methods. The detection procedure is composed of 5 different phases: *Init*, *Abstract*, *Physical*, *Comparison* and *Reporting*.

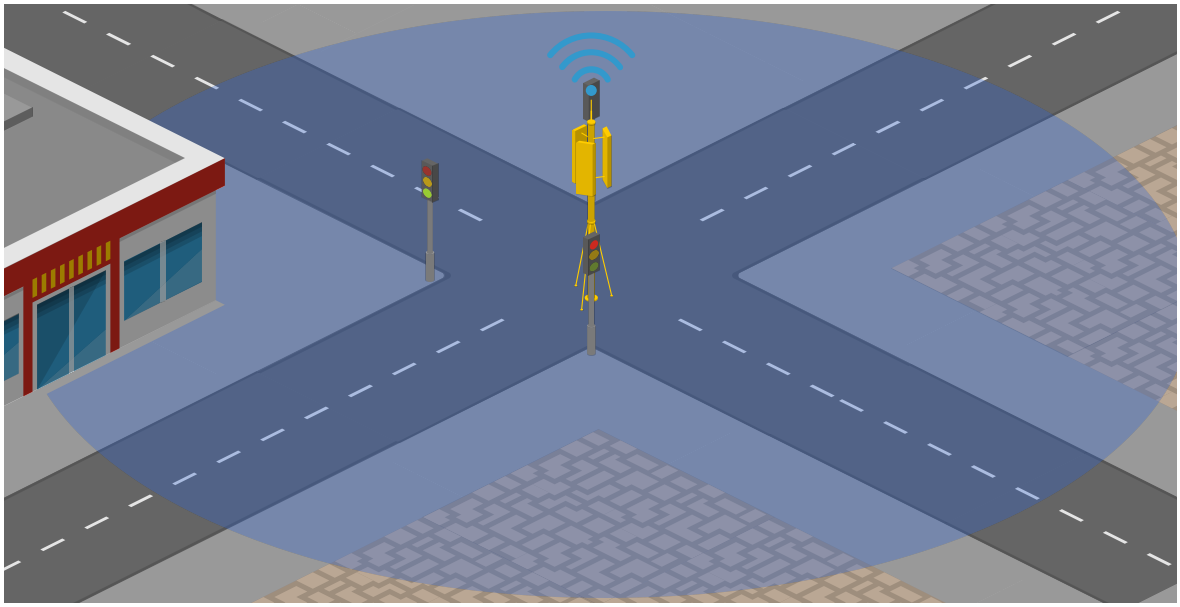


Figure 6.6: Circular sensors coverage area

For the simulation of the detection phases, the perception area of the physical reality is represented by an ad-hoc vector of size equal to the number of simulated vehicles. The vector contains for each location a data structure in which stores, at regular time intervals, the actual vehicle status, which the following three parameters can summarize:

- `position`: The actual real vehicle location expressed in GPS coordinates.
- `timestamp`: A real value representing the simulation time, expressed in seconds, which indicates the time of the last updating procedure.
- `pseudonym`: An integer value that allows to identify vehicles within the simulation and consequently enables to overwrite of previous data with the new one during the updating procedure. The identifier is also actually used in V2X communication, which is usually referred to as a `TemporaryID` in [90], and it is used to ensure confidentiality of communications within VANETs networks [9].

Evaluation phase

This first phase of the detection is carried out essentially for efficiency due to the significant vector size containing all the simulated vehicles' statuses. The output of this phase is a reduced size version of the vector extracted by calculating every distance between the position of the *detector node* (vehicle or the RSU, depending on the detection method considered) which received the message and the position of each simulated vehicle contained in the ad-hoc data structure. The distance is compared with a fixed threshold; if it is lower, the three fields representing the information about the vehicle status are copied into the new vector. Otherwise, the information is considered irrelevant for future analysis. The reduced version of the vector will subsequently represent an approximation of the surrounding reality perception for the *detector node*, on

which the detection algorithms will perform, allowing to dramatically reduce the time required for the analysis of all the positions of the surrounding reality.

Abstract phase

This phase analyzes the information extracted by the incoming BSMs and aims to verify if the GPS position of the received message resides within the perception area of the *detector node*. If the actual BSM is in the proximity of the current *detector node*, the detection procedure continues with the following phases. Otherwise, the detection ends since comparing abstract information with the surrounding reality is impossible.

Depending on the detection method, this process is achieved by implementing different mathematical formulas. The analysis is divided by considering the two variants of the *Local* detection (**Local - Cone** and **Local - Ellipse**) and the *Global* detection method (**Global - Circle**).

Local - Cone

The *Cone* variant involves modern vehicles equipped only with frontal sensors; hence it is designed and implemented with a frontal conical surface as reality perception area. Nevertheless, the conical surface has been treated as a circular sector, thus allowing to take advantage of the *polar coordinate system* theory and the following formulas:

$$\alpha_{start} = \alpha - \frac{\beta}{2} \tag{6.6a}$$

$$\alpha_{end} = \alpha + \frac{\beta}{2} \tag{6.6b}$$

$$\rho = \sqrt[2]{(x_p - x_c)^2 + (y_p - y_c)^2} \tag{6.7}$$

$$\theta = \arctan\left[\frac{(y_p - y_c)}{(x_p - x_c)}\right] \tag{6.8}$$

$$\theta \geqslant \alpha_{start}, \tag{6.9a}$$

$$\theta \leqslant \alpha_{end}, \tag{6.9b}$$

$$\rho \leqslant r. \tag{6.9c}$$

The implementation of the *Cone* variant uses 20 *degrees* for the β angle to simulate the amplitude of circular sector. The α angle, which represents the orientation of the detector vehicle, is necessary to effectively create a circular sector in front of the detector vehicle. Hence, the perception area is created by subtracting and adding to the angle α the half of the amplitude angle β as described by Formula 6.6a and

Formula 6.6b, thus creating two boundary angles. The coordinates extracted from the message in *radial* (Formula 6.7) and *angular* (Formula 6.8) coordinates belonging to the *polar coordinate system*, are converted to correctly estimate the GPS location of the received BSM. In Formula 6.7 and Formula 6.8 the coordinates (x_c, y_c) corresponds to the center of the circular segment, which in order to represent better the area covered by the sensors, has been considered as the central position of the detector vehicle. Instead, the coordinates (x_p, y_p) represent the coordinates of the position extracted from the received message. Then, to verify if the coordinates (x_p, y_p) are within the circular sector it is necessary to apply the constrains of Formula 6.9a and Formula 6.9b to verify if the *angular* value θ reside between the boundary angles α_{start} and α_{end}. Lastly, using the Formula 6.9c, it is possible to check if the **radial** value ρ is less than the radius r of the perception area which fixed to 35 *meters*.

Local - Ellipse

The *Ellipse* variant instead involves future vehicles equipped with several sensors across the entire vehicle surface, thus creating a 360 degree of awareness around the detector vehicle. The surrounding reality knowledge for the detector vehicle is implemented as an area of reality perception with an ellipsoid shape whose center corresponds approximately to the front of the vehicle, as we observe from Figure 6.5.

Mathematically, to verify if a point resides within an orientated ellipse is used the following formula:

$$\begin{aligned} &\frac{[\cos(\alpha)(x_p - x_c) + \sin(\alpha)(y_p - y_c)]^2}{a^2} + \\ &\frac{[\sin(\alpha)(x_p - x_c) - \cos(\alpha)(y_p - y_c)]^2}{b^2} \leq 1 \end{aligned} \tag{6.10}$$

where the point (x_c, y_c) represents the center of the ellipse, while the point (x_p, y_p) represents the GPS position extracted from the received BSM, and α is the orientation angle of the detector vehicle which must be taken into account to verify the message position correctly. Lastly, a and b represent the *length* and *height* of the ellipse, which are set to 35 *meters* and 3 *meters*, respectively.

Global - Circle

The *Global* detection method simulates *Intelligent Traffic Lights* (ITLs) or traffic cameras to detect misbehavior by comparing the information provided by the cameras and the information extracted by the V2I communications. Since the perception area is a complete circle, to evaluate if the position of the BSM message resides within a circular area, we use the inequality of the following formula:

$$(x_p - x_c)^2 + (y_p - y_c)^2 < r^2 \tag{6.11}$$

where the coordinates (x_c, y_c) represents the center of the circle corresponding to the *ITLs* position, while the (x_p, y_p) coordinates represent the position extracted from the received BSM message, and lastly r represents the radius of the circle which is set at 100 *meters*.

Physical phase

This phase analyzes the surrounding perception area to allow subsequent comparison between the abstract and the physical data. After ensuring that the location of the BSM resides within the perception area of the *detector node* using the formulas described in the previous phase depending on the detection method considered, we analyze the positions of all surrounding vehicles. To determinate if the vehicles' position are actually in proximity of the *detector node* it is necessary to examine the reduced size vector created in the *Init* phase checking which vehicles actually reside within the perception area of the *detector node*: *Ellipse*, *Cone* or *Circle*. This step is achieved through an iterative process that allows us to individually analyze each vector component by evaluating the `position` field of the data structure, which contains the actual GPS coordinates of the vehicle. Each step of the iterative process, depending on the detection method considered (*Local* or *Global*), verifies if the actual position of the current vehicle is actually in the perception area of the *detector node* using the formulas described in the previous phase.

Local - Cone

For the *Cone* variant of the *Local* method the same boundary angles α_{start} and α_{end} are used and the real vehicle location is converted into the *polar coordinate system* using Formula 6.7 and Formula 6.8 obtaining the θ and the ρ value. In this phase for the Formula 6.7 and Formula 6.8 the (x_p, y_p) coordinates represent real position of the the vehicle considered in the actual step of the iterative process, while (x_c, y_c) remain the *detector node* position. After these steps, using the same constraints employed in the previous phase, it is possible to check if the actual vehicle position resides within the perception area of the *detector node*. Formula 6.9a and Formula 6.9b check if the angle θ actually resides between the two limit angles α_{start} and α_{end} and Formula 6.9c verify if the the *radial* value ρ is less then the fixed radius r.

Local - Ellipse

The *Ellipse* variant of the *Local* method uses the Formula 6.10 where all the parameters remain the same respect to the *Abstract phase*, except for the (x_p, y_p) coordinates, which is now considered the actual vehicle position extracted from the reduced size vector.

Global - Circle

In the *Global* method, to check if the real position of the vehicle considered in the actual step of the iterative process resides in the perception area of a *ITL*, it is necessary to verify the inequality of Formula 6.11. In Formula 6.11, unlike before, the point (x_c, y_c) represent the real vehicle position of the vehicle corresponding to the `position` field extracted from the reduced size vector created in the *Init phase* and instead (x_p, y_p) stays the same as for the previous phase, thus the GPS position of the *ITL*.

Comparison

This phase compares the abstract data provided by the received messages with the physical data to identify potential misbehavior in the VANETs communications. After evaluating the position of the incoming BSM in the *Abstract* phase and the position of

the near vehicles in the *Physical* phase, it is now possible to merge and compare both data sources and verify possible inconsistent data between the information broadcasted with the BSMs messages and the reality information obtained from the surrounding physical analysis.

Always within the iterative process started in the previous phase, the algorithm checks if the real location of the actual vehicle coincides with the GPS coordinates extracted from the received BMS, taking into account an approximation error to avoid increasing the false positives rate. In fact, also in this phase to implement the approximation error, the Formula 6.11 is used in which (x_c, y_c) is the position of the received BSM, and it is modified at each step of the iterative process the (x_p, y_p) coordinates by assigning the GPS location of the vehicles and using a radius r of 2 *meters*. In addition to these observations about the approximation error, it is also necessary to compare the `timestamp` field of the data structure and the `timestamp` of the received BSM to ensure that the `position` data are updated with respect of the received message. Through this iterative process, we analyze the position of each real vehicle in the proximity of the *detector node*, and if at least the position of one real vehicle corresponds to the position of the received BSM, the detection procedure ends since the BSM is genuine; otherwise, the *Reporting* phase begins.

Reporting

This last phase is aimed at signaling misbehavior. If any real vehicle location can be associated with the received BSM, the message is marked as malicious and must be reported. Two different vectors are created to effectively store all the necessary data to evaluate the proposed detection methods' performance subsequently. A vector of *True positives* and a vector of *False positives*, for each detection variants (*Local* (*ellipse* and *cone*) and *Global*). The *TP* vector stores all the `pseudonym` of the received BSMs which are *correctly* classified as malicious, while instead the *FP* vector saves all the genuine `pseudonym` which are *wrongly* classified as malicious. When a BMS is classified as an attacking one, a built-in function provided by the F^2MD framework is used to check the actual BSM class (genuine or malicious), and depending on the results, it adds `pseudonym` value to the *TP* vector or the *FP* vector.

This process of storing the `pseudonyms` allows at the end of the simulation to obtain six vectors (two for each detection variant) whose size represents the total number of BSM correctly classified (*TP*) and the total number of wrongly classified (*FP*). Then from these vectors, eliminating duplicate pseudonyms makes it possible to obtain the actual number of malicious and genuine vehicles.

6.2.3 Experimental Analysis of the Local and Global Detection

This section compares the proposed detection variants with the detection algorithms offered by the F^2MD framework [86] (which are based only on message content analysis) in order to demonstrate the effectiveness of the proposed detection algorithms against the new version of the SixPack attack (described in Chapter 6.1). Section 6.2.3 presents the simulated environment used for the analysis of the proposed detection solutions against the new version of the SixPack attack.Section 6.2.3 and Section 6.2.3

present the results of both detection algorithms under various scenarios and traffic conditions.

Simulation setup

To provide an accurate analysis of the capabilities of the proposed detection methods against the SixPack $v2$ attack on a realistic scenario, the same simulation setup already presented in Section 6.1.5 is used. The F^2MD framework is extended in order to include the new version of the SixPack attack, presented in Section 6.1.4, and the proposed detection algorithms, presented in Section 6.2.2, allowing to simulate different misbehavior use cases. The simulation setup includes the VEINS simulator [149] which relies on the OMNeT++ [167] network communication simulator and on SUMO [101] frameworks which is a road traffic simulator.

For an accurate evaluation of the *Local* and the *Global* detection methods, the improved version of the SixPack attack, namely SixPack $v2$, in the *LuSTMini* scenario [44] is used. The *LuSTMini* scenario is characterized by about 84 Km of roads, 232 intersections of which 43 are equipped with traffic lights. To recreate realistic traffic situations in the simulated environment, the average number of simulated vehicles is estimated considering the length of the scenario, measured in *kilometers*. According to [130], the ratio between the number of vehicles and the scenario length is defined as the *Traffic Index (Ti)*. Hence, with the Ti index, it is possible to perform several simulations recreating realistic congestion situations, considering a Ti of approximately 5 as regular traffic conditions corresponding to more than 500 vehicles and a Ti of 15, which instead represents the roads congestion situations during rush hour corresponding to about 1300 vehicles.

Local detection results analysis

This section presents the detection results of both the *Local* detection variants compared with the results obtained by the F^2MD framework against the SixPack $v2$ attack.

To properly evaluate the *Local* variants, it is necessary to create a specific file compatible with the SUMO simulator containing all vehicles' paths and, for each attacking vehicle, add a genuine vehicle behind the malicious one with the same path set. This expedient is necessary to allow the simulation of the vehicle sensors to perceive the attacking vehicle in both the *Local* variants: *Ellipse* and *Cone*.

Table 6.6 presents the detection results of 5 different tests conducted in the *LuSTMini* scenario with a Ti equal to 6 due to the vehicle coupling. The columns of Table 6.6 shows the true positive rate (*TP rate*), false positive rate (*FP rate*), false negative rate (*FN rate*), and $\mathcal{F}-$measure achieved against the SixPack $v2$ attack by both the *Local* detection variants (*ELLIPSE* and *CONE*) and the F^2MD framework. The first 5 rows represent the results of different tests, while the last row shows the average across the 5 simulations.

From the analysis of Table 6.6 we observe that the F^2MD framework, as mention before in Section 6.1.5, is not capable of properly identify sophisticated false position attacks, such as the SixPack $v2$ attack, obtaining on average a $\mathcal{F}-$measure of about

	#	TP rate [%]	FP rate [%]	FN rate [%]	$\mathcal{F}-$measure
ELLIPSE	1	5.0607	0.0000	2.8340	0.7812
	2	4.6559	0.0000	3.2389	0.7419
	3	5.2632	0.0000	2.6316	0.8000
	4	5.0607	0.0000	2.8340	0.7812
	5	5.4656	0.0000	2.4291	0.8182
	avg.	*5.1012*	*0.0000*	*2.7935*	*0.7845*
CONE	1	7.0850	0.0000	0.8097	0.9459
	2	7.6923	0.0000	0.2024	0.9870
	3	7.2874	0.0000	0.6073	0.9600
	4	7.2874	0.0000	0.6073	0.9600
	5	7.2874	0.0000	0.6073	0.9600
	avg.	*7.3279*	*0.0000*	*0.5668*	*0.9626*
F^{2}MD	1	4.2510	5.2632	3.6437	0.4884
	2	4.8583	5.2632	3.0364	0.5393
	3	4.8583	5.2632	3.0364	0.5393
	4	4.6559	5.0607	3.2389	0.5287
	5	4.6559	5.4656	3.2389	0.5169
	avg.	*4.6559*	*5.2632*	*3.2389*	*0.5225*

Table 6.6: Detection comparison between both *Local* variants and the F^2MD framework against the SixPack $v2$ attack with regular traffic

0.46. The *Local* detection variants, instead, by combining the abstract information and the data of the reality perception, are capable of identifying almost all the attacking vehicles reaching the on average a $\mathcal{F}-$measure value of more than 0.9 with the *CONE* variant and capable of zeroing the false positive rate (*FP rate*) in both variants.

Table 6.7 instead presents the detection results of 5 different tests conducted in the *LuSTMini* scenario with a *Ti* of about 15. The columns of Table 6.7 are organized in the same way as for the previous Table 6.6.

Comparing the results of the analysis of Table 6.7 respect to Table 6.6 we observe a slight decrease in performance for all the detection method: *ELLIPSE*, *CONE* and F^2MD. Despite this general degradation in performance, the $\mathcal{F}-$measure of the *Local* detection variants remains higher than the F^2MD framework value managing to keep the *FP rate* equal to 0.

From Table 6.7 and Table 6.6 it is possible observe a slight difference in performance analyzing the true positive (*TP rate*) column comparing the *CONE* and the *ELLIPSE* variants. This difference is justifiable because both the implementations of the ellipsoidal surface and the conical surface consider a length of 35 *meters* even if the frontal conical variant is more forward projected than the other one, as can be seen by looking at the Figure 6.4 and Figure 6.5 respectively.

	#	TP rate [%]	FP rate [%]	FN rate [%]	$\mathcal{F}-$measure
ELLIPSE	1	5.7543	0.0000	3.5770	0.7629
	2	5.7543	0.0000	3.5770	0.7629
	3	5.5988	0.0000	3.7325	0.7500
	4	5.7543	0.0000	3.5770	0.7629
	5	5.7543	0.0000	3.5770	0.7629
	avg.	*5.7232*	*0.0000*	*3.6081*	*0.7603*
CONE	1	7.7760	0.0000	1.5552	0.9091
	2	7.9316	0.0000	1.3997	0.9189
	3	7.7760	0.0000	1.5552	0.9091
	4	7.9316	0.0000	1.3997	0.9189
	5	7.9316	0.0000	1.3997	0.9189
	avg.	*7.8694*	*0.0000*	*1.4619*	*0.9150*
F^{2}MD	1	4.6656	7.4650	4.6656	0.4348
	2	6.0653	7.3095	3.2659	0.5342
	3	4.6656	6.3764	4.6656	0.4580
	4	4.5101	7.1540	4.8212	0.4296
	5	4.5101	6.3764	4.8212	0.4462
	avg.	*4.8834*	*6.9362*	*4.4479*	*0.4606*

Table 6.7: Detection comparison between both *Local* variants and the F^2MD framework against the SixPack $v2$ attack during rush hour

Global detection results analysis

This section compares the results of the *Global* detection method, which take advantage of all VANETs infrastructure, and the results obtained by the F^2MD framework to identify the improved version of the SixPack attack. To properly simulate an urban scenario equipped with several *ITLs*, different RSU are strategically placed in all the 43 intersections of the *LuSTMini* scenario with a sensor coverage area of 100 *meters* simulating a future city scenario. Unlike the *Local* method, to properly evaluate the performance of the *Global* method, any adjustments on the path of the attacking and genuine vehicles were necessary. Hence, tools offered by SUMO simulator are used to create a route file containing all the vehicles' paths which are randomly decided.

Table 6.8 presents the detection results of 5 different tests conducted in the *LuSTMini* scenario with a *Ti* equal to 5. The columns of Table 6.8 are arranged as the tables of the previous section regarding the *Local* detection performance. Table 6.8 compare the results achieved by the *Global* method, namely *RSU* in the Table and the results of the F^2MD framework against the SixPack $v2$ attack. The first 5 rows represent the results of different tests, while the last row shows the average across the 5 simulations.

Analyzing the results of Table 6.8 we state that the *Global* detection method achieves higher performance in terms of $\mathcal{F}-$measure with a difference of about 0.47 against the SixPack $v2$ attack. This significant difference is due to the presence of a high number of false positives for the F^2MD as can be seen from the *FP rate* column reaching a

	#	TP rate [%]	FP rate [%]	FN rate [%]	$\mathcal{F}-$measure
RSU	1	3.1390	0.0000	2.0179	0.7568
	2	2.9148	0.0000	2.2422	0.7222
	3	3.1390	0.0000	2.0179	0.7568
	4	3.1390	0.0000	2.0179	0.7568
	5	3.3632	0.0000	1.7937	0.7895
	avg.	*3.1390*	*0.0000*	*2.0179*	*0.7564*
F^{2}MD	1	2.2422	8.0717	2.9148	0.2899
	2	2.0179	8.0717	3.1390	0.2647
	3	2.4664	8.2960	2.6906	0.3099
	4	2.0179	7.3991	3.1390	0.2769
	5	2.0179	8.0717	3.1390	0.2647
	avg.	*2.1525*	*7.9821*	*3.0045*	*0.2812*

Table 6.8: Detection comparison between the *Global* method and the F^2MD framework against the SixPack $v2$ attack with regular traffic

maximum value of 8.29 and at the same time to a low number of true positives (*TP rate*) which instead reach the average value of 2.15.

Table 6.9 instead present the results of the *Global* detection and F^2MD framework against the improved $v2$ version of the SixPack attack. In particular, the Table 6.9 shows the detection capabilities in the *LuSTMini* scenario with a *Traffic Index (Ti)* of about 15 corresponding to congestion situation. The tests consist of 5 runs to avoid possible biases displayed in 5 different rows.

	#	TP rate [%]	FP rate [%]	FN rate [%]	$\mathcal{F}-$measure
RSU	1	3.6585	0.0000	1.3937	0.8400
	2	3.3101	0.0000	1.7422	0.7917
	3	3.1359	0.0000	1.9164	0.7660
	4	2.9617	0.0000	2.0906	0.7391
	5	2.9617	0.0000	2.0906	0.7391
	avg.	*3.2056*	*0.0000*	*1.8467*	*0.7752*
F^{2}MD	1	1.9164	8.0139	3.1359	0.2558
	2	1.3937	8.0139	3.6585	0.1928
	3	2.7875	7.8397	2.2648	0.3556
	4	2.0906	7.8397	2.9617	0.2791
	5	1.3937	7.3171	3.6585	0.2025
	avg.	*1.9164*	*7.8049*	*3.1359*	*0.2571*

Table 6.9: Detection comparison between the *Global* method and the F^2MD framework against the SixPack $v2$ attack during congestion traffic

From the comparison of the Table 6.9 and the Table 6.8 we observe a further decrease

in performance for the F^2MD framework, while instead the *Global* detection method increase its performance of about 0.02 analyzing the $\mathcal{F}-$measure column. This performance gain is due to the increase in the number of vehicles (attacking and genuine ones), consequently raising the probability of an attacking vehicle being in the proximity of road intersections equipped by RSUs during the attack phases.

6.3 Miradouros: decentralized position verification for moving vehicles

Modern cars are equipped with sensors that can detect other moving vehicles and obstacles on the road. However, their range is usually limited to line-of-sight, and their accuracy is also limited. To provide information beyond the sensor range, each vehicle broadcasts Basic Safety Messages (BSMs) with its position and speed. For increased road awareness, it would be best if the position information could be confirmed by multiple vehicles in different positions (*redundancy*), using their on-board sensors for verification (*diversity*), and excluding position and speed errors (*plausibility*). This work presents *Miradouros*, a decentralized solution that uses multiple vantage points to provide more trust in moving vehicle position data. It extends broadcast messages with sensor verification and plausibility filtering. It processes a *stream* of data from nearby vehicles for short periods to achieve the safety benefits without the privacy risks of long-term data retention. Our proposal was evaluated with detailed simulations with different levels of traffic and misbehavior. It provides good detection results with only a limited increase in network and computing resources.

6.3.1 Proposed solution

This section describes the architecture, data structures, and algorithms of *Miradouros*.

Decentralized Architecture

In the proposed architecture, each vehicle runs its instance of the solution, relying on position messages received through the VANET, obstacle detection made by on-board sensors (e.g., cameras, radars, and lidars), and position plausibility rules, akin to misbehavior detection, considering the current and past positions. Each vehicle periodically broadcasts a BSM, e.g., every 100 ms as recommended in the IEEE 1609 WAVE standard. The BSM includes all information related to the current state of the sending vehicle, including the position and speed. It is assumed that each vehicle adds additional information relative to the nearby vehicles' position, as described in Section 6.3.1.

Each element does its own computations and does not rely on the computations made by others. As such, there is no centralized source of *truth* as each vehicle broadcasts its own signals, and the positions are computed by taking into account the history and messages received during the given period.

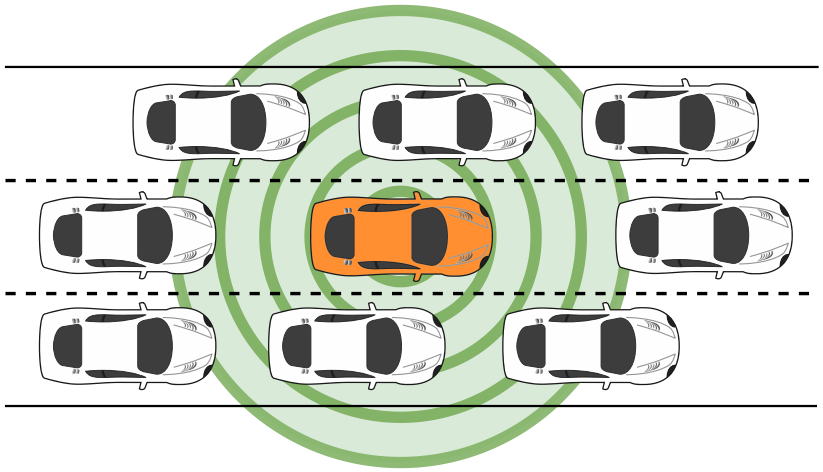


Figure 6.7: Sensor range for vehicle moving on a road with three lanes.

Threat model

Miradouros helps detect misreported positions that malfunctioning location systems may cause. However, Miradouros also considers malicious adversaries. The adversary is interested in attacking the position data protected by *Miradouros*; in particular, it aims to convince the other participants of the VANET that it is at a position different from its actual position, probably to hide some traffic violation. Considering a VANET architecture composed of all the security countermeasures described in Section 2.2.2.2, we classify the adversary as an *insider* attacker following the classification presented in [86]. An insider attack is a typology of attack in which the attacker has acquired valid cryptographic credentials from the PKI to participate in the communications. The attacker is also considered to be *active*, able to craft messages by arbitrarily modifying the values of its fields like the dynamic attacker described in [128].

Regarding the *privacy* of road drivers, the system is designed to not store persistent information. The messages and perceptions are processed in a short period, relevant for safety applications, but no long-term information is retained.

Algorithm

The rationale of the proposed solution is that each vehicle should broadcast additional information about surrounding vehicles (illustrated in Figure 6.7). In that way, each receiver vehicle can collect more information about the environment and use this information to validate the vehicles' position better.

Before describing the algorithm, it is helpful to introduce the two most crucial used data structures: the *Surrounding Vehicle List* (SVL) and *Observed vehicles Table* (ObsTable). Both SVL and ObsTable contain the same category of information (e.g., identifier, position, timestamp). However, the concrete type of information included is a parameter of the proposed algorithm. It can be adjusted to include more information at the expense of the size and network overhead (as discussed in Section 4.2.3 and 6.3.2). The SVL is populated by each sender vehicle and included in the broadcast BSMs, while each receiver manages the ObsTable. Each vehicle is both a `sender` and

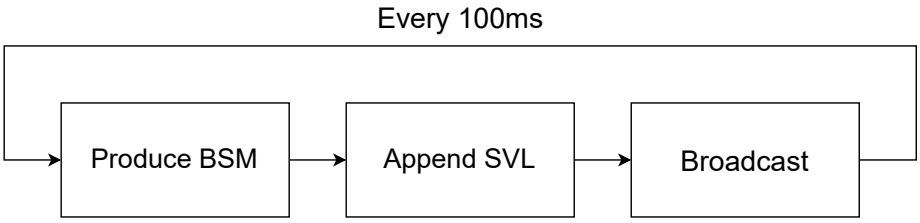


Figure 6.8: The flowchart of the `sender` role.

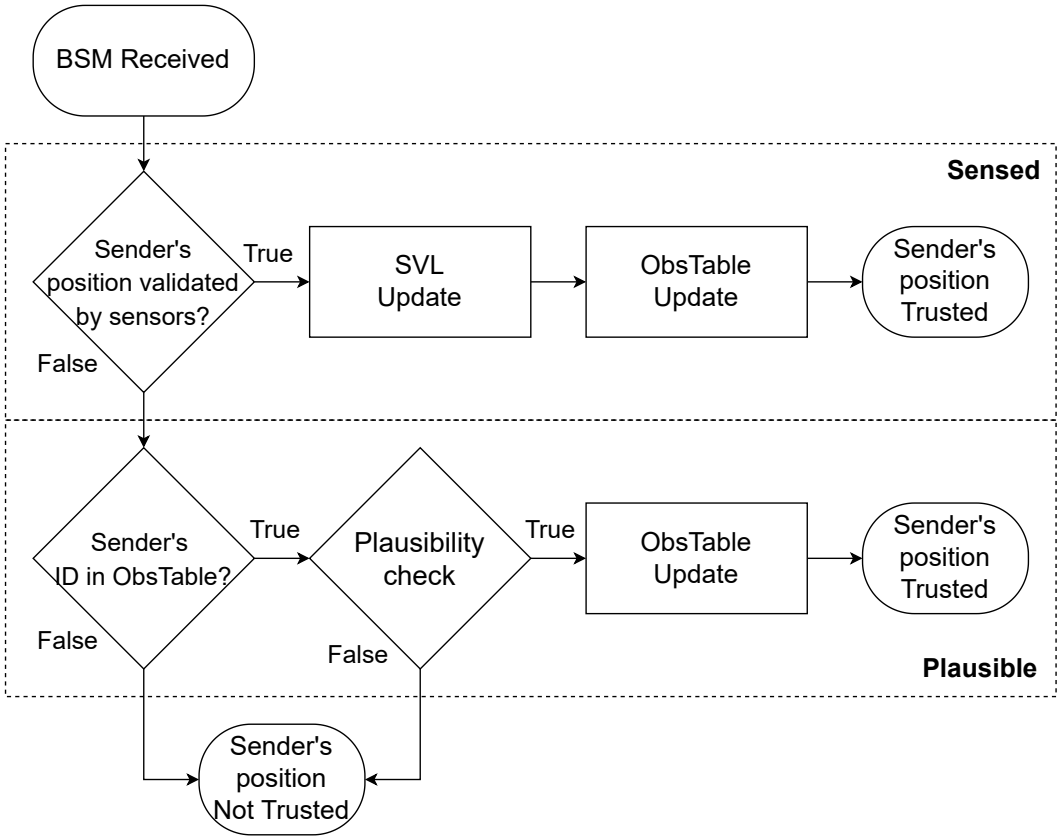


Figure 6.9: The flowchart of the `receiver` role.

a `receiver`, and the two roles are processed simultaneously, so each vehicle processes and manages, at the same time, its SVL and ObsTable data.

The `sender` role is illustrated in Figure 6.8. Every cycle, each vehicle produces a standard BSM, appends the SVL information about nearby vehicles, and broadcasts the message. The SVL is cleared at the end of each cycle (i.e., after each message is broadcast) and populated in the `receiver` role.

The `receiver` role is represented in Figure 6.9. All the depicted steps are executed for each received BSM. This procedure allows to tag the sender's position included in each BSM as *trusted* or *not trusted*.

When a vehicle receives a BSM, it should first try to validate the position of the sender using its own sensors (e.g., camera, radar, lidar). If the vehicle's sensors can confirm the position of the sender, the receiver includes the information of the sender in the SVL (`SVL Update`), updates the ObsTable (`ObsTable Update`), and tags the

Figure 6.10: Traffic example: vehicle X overtook C in a curve, crossing a solid line, and is endangering both B and A.

position of the sender as *trusted*. In this case, the BSM sender's position is considered as *sensed*.

If the sender's position included in the received BSM cannot be verified using the vehicle's sensors, it can be validated in the next step. The receiver should check if the *identity* of the sender is included in the ObsTable, and if yes, use one or more plausibility checks to validate the sender's position. If the sender's position is validated, the ObsTable is updated, and the position of the sender is tagged as *trusted*. In this case, the BSM sender's position is considered as *plausible*.

Finally, if the sender's position included in the received BSM cannot be validated, the BSM is tagged as *not trusted* but still *received*.

Operation

The main difference between a *sensed* and *plausible* position is that only the sensed positions are included in the SVL that will be added to the next BSM, i.e., only sensor confirmations are communicated in the SVL.

Let us consider the traffic example represented in Figure 6.10 and the operation steps of *Miradouros* shown in Table 6.10. In this example, the vehicle X is doing a dangerous overtaking on a road where the maneuver is prohibited due to a curve and poor visibility. Also, X is an adversary, misreporting its position to hide the traffic

Time	ObsTable State		Messages/Perceptions
T_{n-1}	Sensed:		BSM(B,SVL(A,C,X!))
	Plausible:		BSM(X)
	Received:	B, X, C	BSM(C,SVL(X!,B))
T_n	Sensed:		BSM(B,SVL(A,C,X!))
	Plausible:	B, X, C	BSM(X)
	Received:		BSM(C,SVL(X!,B))
T_{n+1}	Sensed:	B	Sensed(B)
	Plausible:	X!, C	BSM(B,SVL(A,C,X!))
	Received:		BSM(X)
			BSM(C,SVL(X!,B))

Table 6.10: Traffic example processing steps for vehicle A. *Sensed* is more trusted than *Plausible* that is more trusted than *Received*. X! represents the actual position of X.

violation.

Table 6.10 shows a representation of the operation of *Miradouros* in vehicle A. At the time T_{n-1}, the vehicle A became aware of the position of vehicles B, X, and C, thanks only to V2V communications. However, the position of these vehicles is not yet validated. In fact, the position of X is not correct. At this time, the contents of the SVL from both B and C are still being ignored because B and C are out of the sensors' range.

At time T_n, vehicle A receives updated positions. Since the changes are physically consistent with their previous position, the new positions are *plausible*. At this time, the position of B is still not confirmed by A's sensors.

At time T_{n+1}, A can now sense B. Since the position of B is validated, A can now accept the SVL contents of B and update its own ObsTable with $X!$ and C received within the BSM sent by B. This is critically important, as the position of $X!$ is now updated, and the dangerous overtaking situation is detected.

6.3.2 Experimental Results

This section presents the assessment of *Miradouros*, based on extensive simulations.

Simulation Setup

A subsection of the LuST scenario [43] is used for the simulation. It includes 84 km of roads with different levels of traffic (300, 400, 500, 700, and 1000 vehicles), and it is considered a 5% level of misbehavior in simulation scenarios. In particular, misbehavior is the set of possible faulty behaviors described in [122] in which a faulty or misbehaving vehicle broadcasts incorrect positions. In the simulation, the misbehaving vehicle broadcasts wrong or random positions from the playground after a random time for a random period. The simulation is run by VEINS [149], which is an open-source framework for vehicular network simulations based on OMNET++ [167] for the simulation of networks, and on SUMO [101] for the simulation of the road traffic.

The simulation adopts the IEEE 802.11p, IEEE 1609.4 DSRC/WAVE [51] module to extend the VEINS capabilities to enable the DSRC/WAVE stack, Quality-of-Service channel access, Wave Short Message (WSM) management and periodic beaconing of BSMs. Other used modules are: Physical Layer [23], Obstacle Shadowing [150], and Antenna Patterns [50] modules to simulate the propagation and attenuation of the wireless signals to recreate proper signal coverage of messages in urban environments. The vehicles simulated in the scenarios are programmed to send beacon messages every 100 ms, as recommended by the SAE J2945-201712 standard [138]. For the simulation of the vehicle's sensors, it is supposed that a vehicle can percept the surrounding environment in a range of 100 meters, in all directions, with an accuracy of 80%, so the receiver can *validate* the position of the BSM sender with its sensors.

As described in Section 6.3.1, the proposed solution can be tuned with different parameters. This work considers the following configuration. The lifetime of all entries in the ObsTable is set to 20 cycles, so an entry in the table is considered expired if not updated for more than 2 seconds. A consistency verification based only on the position of the vehicles is used as plausibility check. Two positions are considered plausible if, given a maximum speed *maxspeed* and the *timespan*, the distance between the two points can be traveled by a vehicle with speed *maxspeed*. In the simulation, the vehicles' maximum speed is 55 meters/second (approximately 200 Km/hour). Again, it is essential to note that the accuracy of the sensors, the expiration time of the ObsTable, and the plausibility check algorithm are only parameters of the proposal and can be tuned for more specific applications.

Simulation Results

Run	#V	#MV	Sensed	ObsTable Hit	ObsTable Miss	TN		FN	THFP	Sent SVL Size	Recv SVL Size
						MTM	Plausibility				
1	300	5% (15)	35%	21%	44%	97%	3%	0%	20%	2.23	3.17
2	400	5% (20)	37%	23%	40%	95%	3%	2%	23%	2.45	3.63
3	500	5% (25)	36%	26%	38%	96%	4%	0%	25%	3.68	4.70
4	700	5% (35)	36%	31%	33%	95%	5%	0%	29%	5.20	6.80
5	1000	5% (50)	36%	33%	31%	96%	3%	1%	30%	7.13	8

Table 6.11: Results for simulations with a duration of 60 seconds (600 BSM cycles).

The simulations were run and produced 60 seconds of data, summarized in Table 6.11. The rows report different `Runs`, and the columns report the average results for each vehicle. Each `Run` indicates the number of vehicles (`#V`), the number of malicious vehicles (`#MV`), the average percentage of sensed positions (`Sensed`), the average percentage of ObsTable hit of each vehicle (`ObsTable Hit`), and the average percentage of ObsTable miss of each vehicle (`ObsTable Miss`). There is an ObsTable hit if the sender of a received BSM is already in the table and a miss if it is not.

The results show that there is more than 50% of probability (`Sensed` + `ObsTable Hit`) that a received BSM can be validated and that probability increases in scenarios with more vehicles. The probability that the sensors validate a BSM is stable in all the different densities (`#V`).

The columns True Negative (`TN`) represent the percentage of BSMs correctly tagged as *not trusted*. It is considered as `TN` both a malicious BSM with the sender ID that

is not included in the table (Malicious ObsTable Miss `MTM`) and a BSM with a sender id included in the table but tagged as non-plausible. The column False Negative (`FN`) represents the percentage of BSMs that are tagged as *trusted* but that are malicious, and ObsTable Hit False Positive (`THFP`) the percentage of BSMs with table hit tagged as *not trusted* but that are genuine. Finally the last two columns, `Sent SVL Size` and `Recv SVL size` report the average size of the SVL for sender and receiver.

The results show that there is about 100% detection rate of the malicious BSMs in all scenarios, while the rate of `THFP` increases in more dense scenarios. A message is classified as THFP by the plausibility check, so improved plausibility algorithms will help to reduce the false positives.

The ObsTable Miss messages are tagged as *not trusted* because it is impossible to have enough evidence to classify these BSMs as *trusted*. This classification is not considered a disadvantage because it is preferred to have high reliability of the results instead of a high recall, i.e., it is preferred not to tag a malicious message as trusted at the cost of losing some genuine messages. Since the cycles are short and the BSMs are repeated, the relevant positions will eventually be detected.

Table 6.12 further investigates if the messages included in the ObsTable Miss column of Table 6.11 can be useful in some scenarios. In particular, for the analysis, the communication requirements defined by the NHTSA [114] are considered, which considers multiple vehicle communication scenarios and defines the constraints that must be satisfied to guarantee safety. The report identifies 8 high-priority and safety-critical scenarios, and for each of them defines the allowable *latency* and communication *range*, respectively:

- Pre-Crash Sensing: 20 ms, 50 m;
- Traffic Signal Violation Warning: 100 ms, 250 m;
- Curve Speed Warning: 1000 ms, 200 m;
- Emergency Electronic Brake Light: 100 ms, 300 m;
- Cooperative Forward Collision Warning: 100 ms, 150 m;
- Left Turn Assistant: 100 ms, 300 m;
- Lane changing Warning: 100 ms, 150 m;
- Stop Sign Movement Assistance: 100 ms, 300 m.

Following the setup proposed in this work (c.f. Section 4.2.3), distances less than 100 m are considered managed by the vehicle's sensors. Table 6.12 reports for each `Run` the percentage of ObsTable Miss with different distances between the sender and the receiver.

The percentage of table miss in the short-medium distances (100-150 m and 150-200 m) decreases with an increase of the number of vehicles, showing that our proposal works better in scenarios with more dense traffic. In medium-long distances (200-250 m and 250-300 m), there is no strong correlation between the percentage of table misses

Run	#V	ObsTable Miss			
		100-150 m	150-200 m	200-250 m	250-300 m
1	300	13%	20%	19%	11%
2	400	13%	20%	15%	10%
3	500	9%	17%	17%	13%
4	700	8%	15%	22%	10%
5	1000	7%	14%	19%	14%

Table 6.12: Analysis of the distances of ObsTable Miss.

and the number of vehicles. In fact, the value of table miss for the 200-250 m distance range is included between 15% and 22%, while for the 250-300 m distance, between 10% and 14%. For all the scenarios, there is about a 0% of table miss for distances < 100 m because these distances are validated mainly by the vehicles' sensors, while there is more than a 45% of table miss for messages with positions over 300m in all the scenarios. Since most of the table misses are for medium and long distances (> 200 m), an application that relies on the trustness of the positions can delay the decision of non-safety critical maneuvers to the following cycles.

Applicability of the proposed solution

Based on the results reported in [130] and specifications of the NHTSA report [114], the average size of a signed BSM that follows the security recommendation of the IEEE 1609.2 standard [81] is 460 bytes, and with a network bandwidth of 6 MiB/s the maximum number of messages supported by the network is 163 messages. The proposed solution requires each vehicle to add additional information about the positions of the surrounding vehicles, causing an increment in the size of the BSMs and an extra overhead on the entire network. In particular, it is considered that each row of the SVL and the ObsTable contains an ID, a *position*, and a *timestamp*. Let us consider that the reported position is expressed using standard GPS coordinates using one word of 32 bit for each field. We assume that the overhead is 64 bit (or 8 bytes) for each coordinate included in the BSM, 4 bytes for the identifier, and other 4 bytes for the timestamp.

As described in Section 6.3.1, each vehicle should include the position of a surrounding vehicle in a BSM only if this position was validated by its sensors. In a worst-case scenario, like the one depicted in Figure 6.7, we suppose that the maximum number of positions that a vehicle can include is 8, which gives an overhead of (8+4+4)*8 = 128 bytes, so a BSM of size 590 bytes and a reduction of network capacity from 163 to 127 messages in term of the size of the SVL, that is enough to support the scenarios proposed in [130]. However, from the results in Table 6.11, we see that, on average, the size of the SVL depends on the traffic density, with a value from 2 and 3 for the less dense scenario to 7 and 8 for the more dense scenario for the sent and received SVL, respectively. These results show sustainable network overhead in all scenarios.

The size of the ObsTable that each vehicle should use depends on the number of received BSMs and the expiry time of each row. The time required for a lookup is

$O(1)$, on average, assuming the use of hashtables.

Chapter 7

Conclusions

Vehicular Ad-Hoc Networks (VANETs) have the potential to facilitate and enable a broad range of cooperative and safety applications. In this thesis, we propose two examples of V2X technologies applications. The first one opens the door to new paradigms of commercial use of road vehicles, while the latter demonstrates how V2V communication can enhance the quality of emergency services. However, it is essential to recognize that while these technologies can enhance the quality of road services, they also pose a significant risk of exploitation by malicious actors. Such exploitation can lead to the endangerment of the safety of all participants. Therefore, it is necessary to incorporate appropriate security mechanisms to mitigate such risks and ensure the safe and effective operation of VANETs.

The aim of this thesis is to improve the security of future smart vehicle networks, laying its foundations in a defense-in-depth approach. We consider the security of individual vehicles as well as the security of an entire network built by many vehicles including both internal and external attackers. The research and solutions presented in this thesis are designed to address the unique constraints posed by cyber-physical systems in the real world. These constraints include prioritizing safety as a critical factor, adhering to international standards, and working with legacy systems that have been widely adopted. By considering these constraints, our proposed solutions strive to ensure that they are practical, feasible, and effective in real-world scenarios. The solutions proposed in this thesis aim to balance security requirements with the operational needs and constraints of cyber-physical systems.

Starting from the security of the single vehicle, this thesis focuses on the CAN protocol. Due to its wide adoption, not limited to, road vehicles, the CAN protocol attracted the attention of researchers and malicious actors. In particular, the legacy constraints of CAN imposed by the car manufacturer pushed the research toward the development of anomaly detection solutions that can detect ongoing attacks without changing the overall architecture of the vehicles. The main limitation of state-of-the-art solutions regard the difficulty of the reproducibility of the results and the lack of open implementations. To close the literature gap between the presence of different proposals and the lack of standard metrics useful to compare CAN detection algorithms, this thesis proposes a framework for comparing anomaly detection algorithms designed for CAN communications. This framework compares the algorithms' detection performance against the same attack scenarios, representing known attacks on the CAN bus.

Moreover, we contributed to the state-of-the-art by surveying and implementing eight different detection algorithms based on CAN message timing analysis; publicly releasing their reference implementations; and testing the implemented algorithms against two different datasets, to present a detailed comparison of the detection performance of the analyzed detection algorithms against the same threat model and using the same detection metrics ($\mathcal{F}-$measure and detection rate). The novel dataset used for the experimental evaluation [125], composed of more than 90 minutes of training data and more than 400 CAN traces containing different labeled attacks, is publicly available to advance current solutions. With this work, we tried to establish a fair, transparent, open-source baseline that all researchers and industry practitioners can use to compare novel and existing solutions, leaving the door open to improvements regarding the coverage of the attacks, data, tested detectors, and other improvements.

In the second part of the thesis, we move outside of the vehicle and consider Networks composed of many vehicles. Before focussing on security we provided a background on V2X communications that includes a depth insight into the two more promising V2X technologies, which are DRSC and C-V2X. Then we considered the existing standards, proposing a complete comparison of the US and EU standards for V2X communications security, listing all the relevant documents and giving an overview of each content. In particular, the study of the standards is combined with the idea of proposing solutions that are feasible in the real world and can help improve the state-of-the-art.

Starting from the standards we presented an experimental evaluation of ECQV performance on automotive-grade boards for their application in VANETs communications. First, we implemented the ECQV and ECDSA cryptographic protocols to test the performance of four different automotive-grade boards with different roles in terms of the maximum number of messages that these platforms can verify in a given time window. Then we estimated the highest possible workload each board can sustain and demonstrates their applicability in standard and safety-critical scenarios identified by the NHTSA in V2V communications. Finally, we included an experimental evaluation of the applicability of the boards in realistic scenarios, representing different portions of a real city (Modena, Italy) characterized by different traffic conditions. These experiments allow us to define several reference workloads that are representative of the number of messages that a single car has to receive and validate in an urban scenario. These workloads demonstrate that even powerful boards are not able to verify the signatures of all the incoming messages within the maximum allowed time. To mitigate this issue and improve the applicability of constrained hardware platforms to the V2V context, we proposed and evaluated different heuristics to prioritize signature validation and a new scheme that combines ECQV implicit certificates with EdDSA. In particular, the keys generated with the proposed ECQV variant guarantee the same security level as those typically used in the standard EdDSA scheme, both theoretically and practically. The proposed scheme is designed to provide the highest possible compatibility with current schemes and their implementations and is fully compliant with EdDSA validators. Moreover, the required modifications can be implemented with little effort starting from existing implementations of EdDSA, without relying on a new large codebase. Performance results show that our proposal is sensibly faster than implementations based on ECQV and ECDSA and can be executed on highly constrained devices, including the ones that are equipped with modern vehicles.

Finally, we analyzed the security capabilities of automotive networks against internal attackers, and in this scope, we developed SixPack, a novel attack to VANETs in which a vehicle generates malicious vehicle-2-vehicle (V2V) messages that mimic a sharp brake that triggers the ABS and propagates this incorrect information to the nearby vehicles. With SixPack we showed that state-of-the-art detection algorithms are not able to detect advanced dynamic attacks, so we proposed different novel detection solutions. The core idea of our solutions is to use data collected from different sources to improve the quality of the information extracted only by the analysis of network traffics, and so improve the detection capabilities. The first two detection solutions to prevent sophisticated internal attacks on VANETs, namely *Local* and *Global*, are capable of detecting false position attacks. The first method, defined *Local* since algorithms implemented are executed within each vehicle, in two different variants depending on the type of the vehicle considered. The Cone variant simulates a modern vehicle equipped with the main mandatory sensors employed in the ADAS systems and simulates an area of perception with a conical shape in front of the vehicle. With the Ellipse variant instead, we simulate a future vehicle capable of providing autonomous driving solutions, thus in possession of several sensors. The second method instead acts globally by exploiting the whole VANETs networks infrastructure simulating the behavior of Intelligent Traffic Lights (ITLs) or road cameras able to analyze the surrounding physical reality. These two detection methods based on V2V and V2I communications, respectively, are capable through the combination of the abstract information coming from the analysis of BSMs exchanged over the VANETs network and information obtained from the analysis of the surrounding physical reality to accurately identify stealthy and malicious behaviors in the VANETs communications. The last solution, namely Miradouros, is a decentralized proposal for validating the position of moving vehicles in VANETs. It is a distributed algorithm that fuses the perception of the environment that each vehicle can construct thanks to its sensors with the information collected from all the received BSMs broadcast by each nearby vehicle. A trusted position is either verified by sensors or filtered by plausibility tests using previous positions. These three detection solutions, *Local*, *Global*, and Miradouros, showed that using data coming from different sources is a promising approach that helps with the detection of also sophisticated attacks.

Overall, this thesis proposes practical and effective security solutions for intra- and inter-Vehicular networks that balance security requirements with the operational needs and constraints of cyber-physical systems, addressing individual vehicle security, V2X communication security, and security against internal attackers.

Appendix A

List of pubblications

- SixPack v2: enhancing SixPack to avoid last generation misbehavior detectors in VANETs. G. G. Zoccoli. F. Pollicino, D. Stabili, M. Marchetti. 2022 IEEE 21th International Symposium on Network Computing and Applications (NCA), 2022.
- On the effectiveness of BSM communications in V2V emergency scenarios. F, Pollicino, D. Stabili, M. Marchetti. 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), 2022
- Accountable and privacy-aware flexible car sharing and rental services. F. Pollicino, L. Ferretti, D. Stabili, M. Marchetti. 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA), 2021
- Hardware limitations to secure C-ITS: experimental evaluation and solutions. F. Pollicino, D. Stabili, L. Ferretti, M. Marchetti. IEEE Transactions on Vehicular Technology, 2021
- A Benchmark Framework for CAN IDS. D. Stabili, F. Pollicino, A. Rota. ITASEC, 2021
- SixPack: Abusing ABS to avoid Misbehavior detection in VANETs. F. Pollicino, D. Stabili, G. Bella, M. Marchetti. 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), 2021
- An experimental analysis of ECQV implicit certificates performance in VANETs. F. Pollicino, D. Stabili, L. Ferretti, M. Marchetti. 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), 2020
- Performance comparison of timing-based anomaly detectors for Controller Area Network: a reproducible study. F. Pollicino, D. Stabili, M. Marchetti. *Under review.
- Miradouros: decentralized position verification for moving vehicles. F. Pollicino, S. Eisa, P. Rosa, M. L. Pardal, M. Marchetti. *Under review.
- Implicit certificates for Edwards Curve Digital Signature Algorithms. L. Ferretti, F. Pollicino, M. Andreolini, M. Marchetti. *Under review.

- Local and Global detection of internal false position attacks in V2X communications. G. G. Zoccoli. F. Pollicino, D. Stabili, M. Marchetti. *Under review.

Bibliography

[1] MASA: Modena Automotive Smart Area. https://www.automotivesmartarea.it/?lang=en, Last visited Jan. 2023.

[2] STMicroelectronis website. https://www.st.com/content/st_com/en.html, Last visited Jan. 2023.

[3] SUMO netconvert documentation. https://sumo.dlr.de/docs/netconvert.html, Last visited Jan. 2023.

[4] SUMO Opposite Direction Driving. https://sumo.dlr.de/docs/Simulation/OppositeDirectionDriving.html, Last visited Jan. 2023.

[5] Mohammed Saeed Al-Kahtani. Survey on security attacks in vehicular ad hoc networks (vanets). In *2012 6th international conference on signal processing and communication systems*. IEEE, 2012.

[6] JAJ Alsayaydeh, AWY Khang, WA Indra, Hossain AKM Zakir, V Shkarupylo, S Saravanan, and JB Pusppanathan. Development of vehicle door security using smart tag and fingerprint system. *International Journal of Engineering and Advanced Technology*, 2019.

[7] Diego F Aranha, Pierre-Alain Fouque, Benoit Gérard, J Kammerer, Mehdi Tibouchi, and J Zapalowicz. GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In *Proc. IACR Int'l Conf. ASIACRYPT*, 2014.

[8] Mikael Asplund. Model-based membership verification in vehicular platoons. In *2015 IEEE International Conference on Dependable Systems and Networks Workshops*, pages 125–132, 2015.

[9] Messaoud Babaghayou, Nabila Labraoui, Ado Adamou Abba Ari, Nasreddine Lagraa, and Mohamed Amine Ferrag. Pseudonym change-based privacy-preserving schemes in vehicular ad-hoc networks: A survey. *Journal of Information Security and Applications*, 2020.

[10] Paulo SLM Barreto, Marcos A Simplicio Jr, Jefferson E Ricardini, and Harsh Kupwade Patil. Schnorr-based implicit certification: improving the security and efficiency of vehicular communications. *IEEE Transactions on Computers*, 2020.

[11] Alessandro Bazzi, Giammarco Cecchini, Michele Menarini, Barbara M Masini, and Alberto Zanella. Survey and perspectives of vehicular Wi-Fi versus sidelink cellular-V2X in the 5G era. *Future Internet*, 11(6):122–142, 2019.

[12] Daniel J Bernstein. Curve25519: new Diffie-Hellman speed records. In *Int'l Work. Public Key Cryptography*, pages 207–228. Springer, 2006.

[13] Daniel J Bernstein. Why EdDSA held up better than ECDSA against Minerva. `https://blog.cr.yp.to/20191024-eddsa.html`, Last visited Jan. 2023.

[14] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In *International Conference on Cryptology in Africa*, pages 389–405. Springer, 2008.

[15] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and B Yang. High-Speed High-Security Signatures. In *Proc. Int'l Conf. CHES*, 2011.

[16] Laura Bieker-Walz, Michael Behrisch, and Marek Junghans. Analysis of the traffic behavior of emergency vehicles in a microscopic traffic simulation. *EPiC Series in Engineering*, 2018.

[17] Norbert Bißmeyer, Hagen Stübing, Elmar Schoch, Stefan Götz, Jan Peter Stotz, and Brigitte Lonc. A generic public key infrastructure for securing car-to-x communication. In *18th ITS World Congress, Orlando, USA*, volume 14, 2011.

[18] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4), 2004.

[19] Joppe W Bos, Craig Costello, Patrick Longa, and Michael Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. *Journal of Cryptographic Engineering*, 6(4):259–286, 2016.

[20] Benedikt Brecht and Thorsten Hehn. A security credential management system for v2x communications. In *Connected Vehicles*. Springer, 2019.

[21] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of ed25519: theory and practice. *IEEE Security & Privacy*, 2021.

[22] Eric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In *Proc. Int'l Work. public key cryptography*, Feb. 2002.

[23] Fabian Bronner and Christoph Sommer. Efficient multi-channel simulation of wireless communications. In *2018 IEEE Vehicular Networking Conference (VNC)*, 2018.

[24] Daniel RL Brown. Generic groups, collision resistance, and ECDSA. *Designs, Codes and Cryptography*, 35(1):119–152, 2005.

[25] Daniel RL Brown, Matthew J Campagna, and Scott A Vanstone. Security of ecqv-certified ecdsa against passive adversaries. *IACR Cryptol. ePrint Arch.*, 2009:620, 2009.

[26] Daniel RL Brown, Robert Gallant, and Scott A Vanstone. Provably secure implicit certificate schemes. In *International Conference on Financial Cryptography*, pages 156–165. Springer, 2001.

[27] Billy Bob Brumley and Risto M Hakala. Cache-timing template attacks. In *Proc. IACR Int'l Conf. ASIACRYPT*, 2009.

[28] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudié, Majid Sobhani, and Ahmad-Reza Sadeghi. Smart keys for cyber-cars: Secure smartphone-based nfc-enabled car immobilizer. In *Proceedings of the third ACM conference on Data and application security and privacy*, 2013.

[29] Levente Buttyán, Tamás Holczer, and István Vajda. On the effectiveness of changing pseudonyms to provide location privacy in vanets. In *European Workshop on Security in Ad-hoc and Sensor Networks*. Springer, 2007.

[30] Giorgio Calandriello, Panos Papadimitratos, Jean-Pierre Hubaux, and Antonio Lioy. Efficient and robust pseudonymous authentication in vanet. In *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, 2007.

[31] M Campagna. Standards for EfficientCryptography 4 (SEC4). Ver. 1.0, certicom research, 2013.

[32] Certicom Research. Standards for EfficientCryptography 1. SEC 1, 2009.

[33] Certicom Research. Sec 4: Elliptic curve qu-vanstone implicit certificate scheme, standards for efficient cryptography group. version 1.0. 2013.

[34] Chain. Chain Key Derivation. `https://github.com/chain/chain/blob/chainkd-dh/docs/protocol/specifications/chainkd.md`, Last visited Jan. 2023.

[35] Konstantinos Chalkias, François Garillot, and Valeria Nikolaenko. Taming the many EdDSAs. In *International Conference on Research in Security Standardisation*, pages 67–90. Springer, 2020.

[36] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Trans. on Knowledge and Data Engineering*, 24(5), 2012.

[37] Zhaohui Cheng and Liqun Chen. Certificateless public key signature schemes from standard algorithms. In *International Conference on Information Security Practice and Experience*, pages 179–197. Springer, 2018.

[38] K. T. Cho and K. G. Shin. Viden: Attacker identification on in-vehicle networks. arXiv 1109.1123, 2017.

[39] K.T. Cho and K. G. Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *USENIX Security Symposium*, 2016.

[40] Kyong-Tak Cho and Kang G. Shin. Error handling of in-vehicle networks makes them vulnerable. pages 1044–1055, 2016.

[41] Francesco Ciari, Benno Bock, and Michael Balmer. Modeling station-based and free-floating carsharing demand: Test case study for berlin. *Transportation Research Record*, 2014.

[42] CISA - Cybersecurity and Infrastructure Security Agency. Understanding denial-of-service attacks. `https://us-cert.cisa.gov/ncas/tips/ST04-015`, 2019.

[43] Lara Codeca, Raphaël Frank, and Thomas Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2015.

[44] Lara Codecá, Raphaël Frank, Sébastien Faye, and Thomas Engel. Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017.

[45] European Commission, Directorate-General for Mobility, Transport, J Scholliers, R Janse, M Tarkiainen, M Modijefsky, A Silla, and G Born. *Study on the feasibility, costs and benefits of retrofitting advanced driver assistance to improve road safety : executive summary*. Publications Office, 2020.

[46] D. J. Bernstein. [Cfrg] Does the Curve25519 Montgomery ladder always work? `https://mailarchive.ietf.org/arch/msg/cfrg/pt2bt3fGQbNF8qdEcorp-rJSJrc/`, Last visited Jan. 2023.

[47] Alexandra Dmitrienko and Christian Plappert. Secure free-floating car sharing for offline cars. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 2017.

[48] Michael Düll, Björn Haase, Gesine Hinterwälder, Michael Hutter, Christof Paar, Ana Helena Sánchez, and Peter Schwabe. High-speed Curve25519 on 8-bit, 16-bit and 32-bit microcontrollers. *Designs, Codes and Cryptography*, 77(2), 2015.

[49] G. Dupont, J. den Hartog, S. Etalle, and A. Lekidis. A survey of network intrusion detection systems for controller area network. In *2019 IEEE Int'l Conf. on Vehicular Electronics and Safety*, Sep. 2019.

[50] David Eckhoff, Alexander Brummer, and Christoph Sommer. On the impact of antenna patterns on vanet simulation. In *2016 IEEE Vehicular Networking Conference (VNC)*, 2016.

[51] David Eckhoff, Christoph Sommer, and Falko Dressler. On the necessity of accurate IEEE 802.11 p models for IVC protocol simulation. In *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2012.

[52] ETSI. Communications architecture. EN 302 665 V1.1.1, 2010.

[53] ETSI. Security services and architecture. TS 102 731 V1.1.1, 2010.

[54] ETSI. Access control. ETSI TS 102 942 V1.1.1, 2012.

[55] ETSI. Confidentiality services. ETSI TS 102 943 V1.1.1, 2012.

[56] ETSI. Mapping for IEEE 1602.2. ETSI TS 102 867 V1.1.1, 2012.

[57] ETSI. Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA). ETSI TS 102 165-1, 2017.

[58] ETSI. Threat, Vulnerability and Risk Analysis (TVRA). ETSI TR 102 893 V1.2.1, 2017.

[59] ETSI. ITS-G5 access layer specification for intelligent transport systems operating in the 5 ghz frequency band. EN 302 663, 2019.

[60] ETSI. Its communications security architecture and security management, release 2. ETSI TS 102 940 V2.1.1, 2021.

[61] ETSI. Security header and certificate formats, Release 2. ETSI TS 103 097 V2.1.1, 2021.

[62] ETSI. Trust and privacy management. ETSI TS 102 941 v1.4.1, 2021.

[63] TS ETSI. 102 687 v1. 1.1: Intelligent transport systems (its); decentralized congestion control mechanisms for intelligent transport systems operating in the 5 ghz range; access layer part. *Access layer part*, 2011.

[64] Joao Ferreira and Miguel L Pardal. Witness-based location proofs for mobile devices. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–4. IEEE, 2018.

[65] L. Ferretti, F. Longo, M. Colajanni, G. Merlino, and N. Tapas. Authorization transparency for accountable access to iot services. In *Proc. Third IEEE Int'l Cong. Internet of Things*, pages 91–99, July 2019.

[66] Luca Ferretti, Francesco Longo, Giovanni Merlino, Michele Colajanni, Antonio Puliafito, and Nachiket Tapas. Verifiable and auditable authorizations for smart industries and industrial internet-of-things. *Journal of Information Security and Applications*, 59:102848, 2021.

[67] Luca Ferretti, Mirco Marchetti, and Michele Colajanni. Fog-based Secure Communications for Low-power IoT Devices. *ACM Trans. Internet Technologies*, 19(2), March 2019.

[68] Luca Ferretti, Francesco Pollicino, Mauro Andreolini, and Mirco Marchetti. Implicit certificates for edwards curve digital signature algorithms. In *Currently under review*, 2023.

[69] Sébastien Gambs, Moussa Traoré, Matthieu Roy, and Marc-Olivier Killijian. Props: A privacy-preserving location proof system. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 2014, 10 2014.

[70] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In *Proc. 2016 ACM SIGSAC Conf. CCS*.

[71] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In *Proc. Int'l Work. CHES*, 2006.

[72] Marc Girault. Self-certified public keys. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 490–497. Springer, 1991.

[73] Robert Bosch GmbH. Can specification version 2.0. Tech. rep., Bosh, 1991.

[74] M. Gmiden, M. H. Gmiden, and H. Trabelsi. An intrusion detection method for securing in-vehicle can bus. In *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 176–180, 2016.

[75] Mike Hamburg. Decaf: Eliminating cofactors through point compression. In *Annual Cryptology Conference*, pages 705–723. Springer, 2015.

[76] Mike Hamburg, Henry de Valence, Isis Lovecruft, and Tony Arcieri. The Ristretto Group. `https://ristretto.group/ristretto.html`, Last visited Jan. 2023.

[77] Huseyin Hisil, Kenneth K Wong, Gary Carter, and Ed Dawson. Twisted Edwards curves revisited. In *Proc. IACR Int'l Conf. ASIACRYPT*, 2008.

[78] IANIX. Things that use Ed25519. `https://ianix.com/pub/ed25519-deployment.html`, Last visited Jan. 2023.

[79] IEEE. IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Multichannel Operation. Std. 1609.4-2010.

[80] IEEE. Standard for Wireless Access in Vehicular Environments (WAVE)–Networking Services. Std. 1609.3-2016.

[81] IEEE. Standard for Wireless Access in Vehicular Environments (WAVE)–Security Services for Applications and Management Messages. Std. 1609.2a-2017.

[82] IEEE. Guide for wireless Access in vehicular environments (WAVE) architecture. *IEEE: Piscataway, NJ, USA*, page 1609, 2013.

[83] IEEE 802.11 Working Group and others. Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments. *IEEE Std*, 802(11), 2010.

[84] IETF. RFC 6749: The OAuth 2.0 Authorization Framework, Oct. 2012.

[85] H. J. Jo and W. Cho2016Fingerprinting. A survey of attacks on controller area networks and corresponding countermeasures. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–19, 2021.

[86] Joseph Kamel, Mohammad Ansari, Jonathan Petit, Arnaud Kaiser, Ines Ben Jemaa, and Pascal Urien. Simulation framework for misbehavior detection in vehicular networks. *IEEE Transactions on Vehicular Technology*, 2020.

[87] Duško Karaklajić, Jörn-Marc Schmidt, and Ingrid Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Trans. Very Large Scale Integration Systems*, 21(12), 2013.

[88] Keen Security Lab. of Tencent. Car hacking research: Remote attack tesla motors. Tech. rep., 2016.

[89] Ken Tindell. Canis Automotive Labs - The CANPico Board.

[90] John B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[91] Dmitry Khovratovich and Jason Law. Bip32-ed25519: Hierarchical deterministic keys over a non-linear keyspace. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 27–31. IEEE, 2017.

[92] Dongwook Kim, Juyoung Kang, and Kyoungsu Yi. Control strategy for high-speed autonomous driving in structured road. In *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011.

[93] M. Kneib and C. Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *Proc. 2018 ACM SIGSAC Conf. on Computer and Communications Security*. ACM, 2018.

[94] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proc. IACR Annual Int'l Cryptology Conf. (CRYPTO)*, 1999.

[95] Sekar Kulandaivel, Tushar Goyal, Arnav Kumar Agrawal, and Vyas Sekar. Canvas: Fast and inexpensive automotive network mapping. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 389–405, Santa Clara, CA, August 2019. USENIX Association.

[96] Jan Lastinec and Mario Keszeli. Analysis of realistic attack scenarios in vehicle ad-hoc networks. In *7th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE, 2019.

[97] Maryline Laurent, Jean Leneutre, Sophie Chabridon, and Imane Laaouane. Authenticated and privacy-preserving consent management in the internet of things. *Procedia Computer Science*, 2019.

[98] H. Lee, S. H. Jeong, and H. K. Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, volume 00, pages 57–5709, Aug 2017.

[99] Libsodium. libsodium. https://libsodium.org, Last visited Jan. 2023.

[100] C. Ling. An Algorithm for Detection of Malicious Messages on CAN Buses. In *Conf. Innovative Trends in Computer Science*, 2012.

[101] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie WieBner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

[102] Ken MacKay. MicroECC. https://github.com/kmackay/micro-ecc, Last visited Jan. 2023.

[103] M. Marchetti and D. Stabili. Anomaly detection of can bus messages through analysis of id sequences. In *IEEE Proc. Intelligent Vehicles Symp.*, June 2017.

[104] M. Marchetti and D. Stabili. Anomaly detection of can bus messages through analysis of id sequences. June 2017.

[105] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *IEEE 2nd Int'l Forum Research and Technologies for Society and Industry Leveraging a better tomorrow*, Sept 2016.

[106] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. Sept 2016.

[107] John Preuß Mattsson, Erik Thormarker, and Sini Ruohomaa. Deterministic ECDSA and EdDSA Signatures with Additional Randomness. https://datatracker.ietf.org/doc/html/draft-mattsson-cfrg-det-sigs-with-noise-04, Last visited Jan. 2023.

[108] C. Miller and C. Valasek. Adventures in automotive networks and control units. https://www.ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf, 2014.

[109] C. Miller and C. Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. http://illmatics.com/RemoteCarHacking.pdf, 2015.

[110] C. Miller and C. Valasek. CAN Message Injection – OG Dynamite Edition. http://illmatics.com/canmessageinjection.pdf, 2016.

[111] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell. Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks. In *CISRC '17 Proc. 12th Annual Conf. on Cyber and Information Security Research*, 2017.

[112] M. Müter and N. Asaj. Entropy-based anomaly detection for in-vehicle networks. In *IEEE Proc. Intelligent Vehicles Symp.*, 2011.

[113] National Highway Traffic Safety Administration. Vehicle safety communication projet – task 3 final report. DOT HS 809 859, March 2005.

[114] National Highway Traffic Safety Administration. Vehicle safety communication projet – final report. DOT HS 810 591, Apr. 2006.

[115] Phong Q Nguyen and Igor E Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, codes and cryptography*, 30(2), 2003.

[116] N. Nowdehi, W. Aoudi, M. Almgren, and T. Olovsson. Casad: Can-aware stealthy-attack detection for in-vehicle networks, 2019.

[117] Habeeb Olufowobi, Clinton Young, Joseph Zambreno, and Gedare Bloom. Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing. *IEEE Transactions on Vehicular Technology*, 69(2):1484–1494, 2020.

[118] S. Otsuka, T. Ishigooka, Y. Oishi, and K. Sasazawa. Can security: Cost-effective intrusion detection for real-time control systems. In *SAE 2014 World Congress and Exhibition*, page 11, 2014.

[119] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero. A stealth, selective, link-layer denial-of-service attack against automotive networks. 2017.

[120] Anand Paul, Naveen Chilamkurti, Alfred Daniel, and Seungmin Rho. Chapter 2 - intelligent transportation systems. In *Intelligent Vehicular Networks and Communications*. Elsevier, 2017.

[121] PEAK System. Pcan-usb. Technical report, 2015.

[122] Jonathan Petit and Raashid Ansari. V2x validation tool. *BlackHat 2018*, 2018.

[123] Leon A Pintsov and Scott A Vanstone. Postal revenue collection in the digital age. In *International Conference on Financial Cryptography*, pages 105–120. Springer, 2000.

[124] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing*, 9(2):230–250, 1980.

[125] F. Pollicino, D. Stabili, and M. Marchetti. Material used for the submission at ACM TCPS - Special Issue. password: PSM_TCPS.

[126] Francesco Pollicino, Samih Eisa, Pedro Rosa, Miguel Pardal, and Mirco Marchetti. Miradouros: decentralized position verification for moving vehicles. In *Currently under review.*

[127] Francesco Pollicino, Luca Ferretti, Dario Stabili, and Mirco Marchetti. Accountable and privacy-aware flexible car sharing and rental services. In *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*, pages 1–7. IEEE, 2021.

[128] Francesco Pollicino, Dario Stabili, Giampaolo Bella, and Mirco Marchetti. Sixpack: Abusing abs to avoid misbehavior detection in vanets. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–6. IEEE, 2021.

[129] Francesco Pollicino, Dario Stabili, Luca Ferretti, and Mirco Marchetti. An experimental analysis of ecqv implicit certificates performance in vanets. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6. IEEE, 2020.

[130] Francesco Pollicino, Dario Stabili, Luca Ferretti, and Mirco Marchetti. Hardware limitations to secure c-its: experimental evaluation and solutions. *IEEE Transactions on Vehicular Technology*, 70(12):12946–12959, 2021.

[131] Francesco Pollicino, Dario Stabili, Luca Ferretti, and Mirco Marchetti. Implementation of the ECVQ implicit certificate scheme for low-power devices. `https://weblab.ing.unimore.it/resources/uECQV.zip`, Last visited Jan. 2023.

[132] Francesco Pollicino, Dario Stabili, and Mirco Marchetti. On the effectiveness of bsm communications in v2v emergency scenarios. In *2022 IEEE 95rd Vehicular Technology Conference (VTC2022-Spring)*, pages 1–6. IEEE, 2021.

[133] Francesco Pollicino, Dario Stabili, and Mirco Marchetti. Performance comparison of timing-based anomaly detectors for controller area network: a reproducible study. *Currently under review*, 2033.

[134] T. Pornin. RFC6979: Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). Rfc, Aug. 2013.

[135] Qualcomm. ITS Stack 80-PE732-64 REV A. `https://www.qualcomm.com/media/documents/files/c-v2x-its-stack.pdf`, Last visited Jan. 2023.

[136] Yolan Romailler and Sylvain Pelissier. Practical fault attack against the Ed25519 and EdDSA signature schemes. In *IEEE Work. Fault Diagnosis and Tolerance in Cryptography*, 2017.

[137] I. Liusvaara S. Josefsson. RFC8032: Edwards-Curve Digital Signature Algorithm (EdDSA). RFC, Jan. 2017.

[138] SAE International. Dedicated short range communications (dsrc) message set dictionary. *SAE International*, 2016.

[139] SAE International. J2945: Vulnerable Road User Safety Message Minimum Performance Requirements. Technical report, 2017.

[140] Henrique F Santos, Rui L Claro, Leonardo S Rocha, and Miguel L Pardal. Stop: A location spoofing resistant vehicle inspection system. In *International Conference on Ad-Hoc Networks and Wireless*, pages 100–113. Springer, 2020.

[141] Pascal Sasdrich, Amir Moradi, and Tim Güneysu. White-box cryptography in the gray box. In *Proc. 2016 Int'l Conf. Fast Software Encryption*.

[142] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig. Authentication and authorization for constrained environments (ace) using the oauth 2.0 framework (ace-oauth). Internet-draft, Dec. Dec. 2018.

[143] NXP Semiconductors. 3GPP TS 36.213, v 14.2.0. `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2427`, Last visited Jan. 2023.

[144] NXP Semiconductors. 3GPP TS 36.300, v 14.4.0. `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2430`, Last visited Jan. 2023.

[145] Marc Semrau and Jakob Erdmann. Simulation framework for testing adas in chinese traffic situations. *SUMO 2016–Traffic, Mobility, and Logistics*, 2016.

[146] Susan A Shaheen and Adam P Cohen. Carsharing and personal vehicle services: worldwide market developments and emerging trends. *International journal of sustainable transportation*, 2013.

[147] Susan A Shaheen, Mark A Mallery, and Karla J Kingsley. Personal vehicle sharing services in north america. *Research in Transportation Business & Management*, 2012.

[148] M. A. Simplicio, E. L. Cominetti, H. K. Patil, J. E. Ricardini, and M. V. M. Silva. The unified butterfly effect: Efficient security credential management system for vehicular communications. In *Proc. 2018 IEEE Vehicular Networking Conf.*, Dec 2018.

[149] C. Sommer, R. German, and F. Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, 2011.

[150] Christoph Sommer, David Eckhoff, Reinhard German, and Falko Dressler. A computationally inexpensive empirical model of ieee 802.11 p radio shadowing in urban environments. In *Eighth international conference on wireless on-demand network systems and services*.

[151] H.M. Song, H. R. Kim, and H. K. Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *2016 International Conference on Information Networking (ICOIN)*, pages 63–68, 2016.

[152] D. Stabili, L. Ferretti, M. Andreolini, and M. Marchetti. Daga: Detecting attacks to in-vehicle networks via n-gram analysis. arXiv 0000.0000, 2021.

[153] D. Stabili and M. Marchetti. Detection of missing can messages through inter-arrival time analysis. In *2019 IEEE 90th Vehicular Technology Conf.*, Sep. 2019.

[154] D. Stabili, M. Marchetti, and M. Colajanni. Detecting attacks to internal vehicle networks through hamming distance. In *AEIT Int'l Annual Conf.*, Sept 2017.

[155] Dario Stabili, Francesco Pollicino, and Alessio Rota. A benchmark framework for can ids. In *ITASEC*, 2021.

[156] Carlos Renato Storck and Fátima Duarte-Figueiredo. A survey of 5g technology evolution, standards, and infrastructure associated with vehicle-to-everything communications by internet of vehicles. *IEEE Access*, 8:117593–117614, 2020.

[157] SUMO. OSMWebWizard. `https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html`, Last visited Jan. 2023.

[158] M. Sutton, A. Greene, and P. Amini. *Fuzzing: Brute Force Vulnerability Discovery*. Addison-Wesley Professional, 2007.

[159] Iraklis Symeonidis, Mustafa A Mustafa, and Bart Preneel. Keyless car sharing system: A security and privacy analysis. In *IEEE Int'l Smart Cities Conf*, 2016.

[160] A. Taylor, N. Japkowicz, and S. Leblanc. Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49, 2015.

[161] C.K. K Toh. Ad hoc wireless networks: Protocols and systems. Technical report, USA, 2001.

[162] A. Tomlinson, J Bryans, and S. A. Shaikh. Using internal context to detect automotive controller area network attacks. *Computers and Electrical Engineering*, 91:107048, 2021.

[163] U.S. Department of Transportation-National Highway Traffic Safety Administration. Federal Motor Vehicle Safety Standards; V2V Communications. `https://www.federalregister.gov/documents/2017/01/12/2016-31059/federal-motor-vehicle-safety-standards-v2v-communications`, Last visited Jan. 2023.

[164] Binod Vaidya and Hussein T Mouftah. Security for shared electric and automated mobility services in smart cities. *IEEE Security & Privacy*, 2020.

[165] C. Valasek and C. Miller. Car hacking for poories. `https://ioactive.com/pdfs/IOActive_Car_Hacking_Poories.pdf`, 2014.

[166] Rens W van der Heijden, Thomas Lukaseder, and Frank Kargl. Veremi: A dataset for comparable evaluation of misbehavior detection in vanets. In *International Conference on Security and Privacy in Communication Systems*, 2018.

[167] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and ....

[168] Yuval Yarom and Naomi Benger. Recovering OpenSSL ECDSA Nonces Using the FLUSH+ RELOAD Cache Side-channel Attack. Iacr cryptology eprint archive report 2014/140.

[169] Yubico. Security Advisory 2019-06-13 – Reduced initial randomness on FIPS keys. `https://www.yubico.com/support/security-advisories/ysa-2019-02/`, Last visited Jan. 2023.

[170] Pan Zhao, Jiajia Chen, Yan Song, Xiang Tao, Tiejuan Xu, and Tao Mei. Design of a control system for an autonomous vehicle based on adaptive-pid. *International Journal of Advanced Robotic Systems*.

[171] Giovanni Gambigliani Zoccoli, Francesco Pollicino, Dario Stabili, and Mirco Marchetti. Sixpack v2: enhancing sixpack to avoid last generation misbehavior detectors in vanets. In *2022 IEEE 21th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2022.

[172] Giovanni Gambigliani Zoccoli, Francesco Pollicino, Dario Stabili, and Mirco Marchetti. Local and global detection of internal false position attacks in v2x communications. In *Currently under review*, 2023.